Denis Zlobin

Audio Design in Mid-Core Mobile Games

Master's Thesis

Thesis Supervisor

Antti Ikonen

Sound in New Media
Media Lab Helsinki
School of Arts, Design and Architecture
Aalto University
2018

Author Denis Zlobin

Title of thesis Audio Design in Mid-Core Mobile Games

Department Department of Media

Degree programme Master's Degree programme in Sound in New Media

Year 2018 **Number of pages** 63 **Language** English

Abstract

This thesis explores practical aspects of audio design in mid-core games – a massive segment of the modern mobile game market. Despite there is no shortage of professional literature about game audio in general, most of it describes either purely technical or very general matters, taking video games as undivided whole. Games, however, can be broken up into distinct categories, based on genres, platforms, audiences and other criteria. This is the direction I take with my research, focusing on a specific type of games, developed for mobile devices and targeted towards a well-defined audience.

The goal of my thesis was to find out if mid-core mobile games require special approach to audio design and what this approach would be. The first two chapters focus on critical aspects of mobile game audio and audio design principles of mid-core games. Two latter chapters describe practical matters of audio design for an actual video game, starting with design principles and ending with project-specific challenges and creative choices.

The results show that audio design for mid-core mobile games has a few special aspects on top of existing principles of mobile game audio. These aspects result from the design patterns shared by majority of mid-core games developed for mobile platforms.

Keywords Audio design, sound design, game audio, interactive audio, mobile game, mobile game audio

Acknowledgements

I want to thank:

My thesis advisor and supervisor Antti Ikonen not only for guiding me throughout the process of creating this thesis, but also for supporting and inspiring me during my studies.

Laura Laakso, Teemu Leinonen, Vili Viitaniemi and Aaron J. Brodkowitz for their valuable feedback on the thesis.

The Company and everyone involved in development of the Game for making this work possible. Special thanks to my colleagues from the audio department for being such a wonderful team.

All members of GRIA and FINGAP game audio communities for sharing their knowledge and experience. Their advices prevented me from making hundreds of mistakes on my way to becoming a game audio professional.

All Media Lab community for my time in Aalto University. Coming here turned out to be one of the most important decisions of my life and I feel honored to spend those three years surrounded with such friendly and talented people.

My parents for having faith in me and always supporting everything I do.

My beloved wife Julia for her love and patience.

Table of contents

Introduction

Modern mobile game market is dominated by mid-core games. According to the website Deconstructor of Fun, 14 of the 15 highest revenue mobile games of 2017 belong to the mid-core sector (Das-Gupta, Katkoff and Payton, 2018). Sometimes described as the missing piece between long-established casual and hardcore segments, mid-core is often called the new trend, and many game developers have shifted their focus towards this sector of the market.

The goal of this work is to define special qualities of mid-core mobile games in terms of audio design. Do these games require a special approach to the audio creation process? And if so, how different will it be compared to somewhat established principles of mobile game audio in general?

I approached these questions in practice while working on a massive mobile game for Android and iOS as a full-time audio designer. The case study of this project became the main focus for my thesis. Problems and challenges we faced while working on this game are likely to be relevant to other audio designers and researchers, who are interested in the booming segment of mid-core games. The thesis demonstrates a set of technical and creative decisions taken during development of a big-scale mobile project and describes the theory behind them.

Additionally, I research design principles behind modern mobile mid-core games and speculate on their effect on audio design process. Game audio is rarely observed from this perspective, since taxonomy of casual, hardcore and mid-core mostly relates to types of the audience and games, crafted for the specific audience, but not to the game genres themselves (Graft, 2013; Lien, 2013). However, this taxonomy is tightly connected to

different contexts and patterns of playing video games, and study of those might be able to provide some insights for game audio designers.

Due to agreements with my employer I'm not allowed to mention any names of people, businesses or products related to the project behind this thesis. Instead the game will be referred to as the Game, the developer company – as the Company, the parenting franchise as the Franchise and the party that owns the Franchise as the Third party.

The thesis consists of four main chapters, list of necessary definitions, introduction, conclusion and a list of references. The first chapter is focused on mobile game audio and describes the most common challenges and constraints for sound designers working on these media. The second chapter explores the concept of mid-core in general and from an audio design perspective. The third chapter describes general principles behind the Game audio design, and the fourth focuses on problems and their solutions. The conclusion provides a summary of the main ideas in this thesis and adds some closing notes about the topic.

Definitions

2D sound – sound that is sent directly to the audio output. 2D sounds are not tracked by the audio listener and not positioned in virtual space.

3D sound – sound, emitted by audio source in the game world and received by the audio listener. 3D sounds are positioned in virtual space, so the player is able to locate them.

Audio asset, sound asset – individual audio file, used in the game.

Audio budget, audio memory budget – (in this thesis) the maximum amount of memory for audio content in a game.

Audio engine, audio middleware – software toolkits for handling audio in video games and other interactive media. Audio middleware offers a variety of tools for real-time audio generation and processing and reduces the need for code support for audio designers.

Audio listener – an object or a component representing a microphone in the game world. Audio listener is necessary for positioning of 3D sounds.

Core loop – key repetitive activities that define gameplay. For example, "hitting enemies and avoiding being hit" or "collecting resources to develop an army".

DSP – digital sound processing

LU, LUFS – loudness units, loudness units relative to full-scale. Loudness standard, introduced in EBU R128. Loudness unit is equal to one decibel.

Nyquist rate – the lowest possible sample rate for sampling the signal without aliasing.

PvE – Player versus environment. Mode of gameplay where players fight against computer-controlled entities.

PvP – Player versus player. Mode of multiplayer gameplay where players are engaged in a direct conflict with each other.

RTPC – real time parameter control in Wwise. A curve that defines relation between a parameter or a variable in a game and a property in Wwise.

Sound event – a set of rules and commands for playing and processing specific audio assets or controlling properties in Wwise. The simplest example of a sound event is a command to play or stop a sound. The concept of sound event in other audio middleware might be different.

Sound bank – a container with sound events and sound assets that is getting loaded to the RAM.

Unity – popular cross-platform video game engine developed by Unity Technologies.

Voice – a single audio stream, a "unit" of polyphony.

Wwise – industry standard audio middleware, developed by Audiokinetic Inc. Consists of sound engine and authoring tool.

1. Challenges of mobile game audio

Mobile game audio holds a special place in the domain of game audio design. Limited processing power and storage capacity of mobile devices create a feeling of déjà vu for audio professionals who worked on video games in late 90's and early 2000's. Hardware limitations they had to overcome working on video games in those days are back again, but on a different platform (Horowitz and Looney, 2014, p.196). On the other hand, design principles of modern mobile games don't have much in common with the ones existing 20 years ago for PC and console game development (Luban, 2011; Clark, 2014, p.7).

Thus, modern mobile game development can benefit from a special approach to audio design. This approach should be focused on two key areas. First, technical limitations of current mobile platforms require developers to heavily optimize audio content. Second, most of modern mobile games are service-based, so audio designers should understand their game design principles. Thus, technical constraints and design aspects can define mobile sound design in numerous ways. In this chapter I will describe the most prominent challenges of mobile game audio and the most common ways of dealing with them.

1.1 Technical constraints

Mobile game audio design is very dependent on hardware constraints. The first of those is audio budget – the amount of memory that can be allocated to the sound assets. Developers typically discuss audio memory at the initial stages of development. The size

of the memory budget can significantly vary from 1-5 megabytes in simple puzzle games to over 100 megabytes in larger and more complicated projects which involve various entities and complex dynamic sound systems. It can depend on the game genre, amount of content, design principles and even marketing strategy. For example, many developers are trying to keep the overall size of the build under 150 megabytes (developer.apple.com, 2017) because Apple's AppStore and Google's Play Market recommend users to download larger applications through Wi-Fi connection. This usually means that overall audio budget will not exceed 5-15 megabytes. At the same time, minimal amount of audio content is defined by game design. Therefore, if a game requires a lot of audio, but has small memory budget for it, sound assets will need to be heavily compressed with loss of sound quality.

Another technical limitation comes via the maximum amount of RAM that can be allocated for music and sound effects. Modern game engines operate with sound banks – sets of sound events and assets, simultaneously loaded to the RAM. The overall size of sound banks that can be loaded into operative memory of the device at the same time is usually strictly limited. It is possible to partly overcome this limitation by streaming some of the heavier assets like music from the drive, but this technique needs to be applied carefully due to possible synchronization problems (Stevens and Raybould, 2011, pp.34-42).

The third constraint comes from the processing power of the mobile device and CPU that can be allocated for audio processing. It defines possibility to use real-time DSP effects, such as filters, compressors and reverb. Other than that, it influences the amount of sounds that can be played at the same time and data compression format, since CPU needs to decode all audio in real time. (Jean, 2017)

The mount of simultaneously played sounds is additionally affected by physical voice limitation. Different audio chips can play different amount of sounds at the same time. Their capacity may vary from <20 on low-end Android devices to ~60 on the newest generation iPhones and iPads (Jean, 2017). If at any given moment game engine tries to play more sounds that device might handle, some content would be cut off due to voice starvation or will be virtualized. To prevent this from happening we need to research audio channel capacity of target devices and prioritize some groups of sounds over others.

The final limitation comes from the hardware itself – the size of a mobile device speaker. It is only able to correctly reproduce frequencies within a relatively narrow working range, and likely to resonate on some of the frequencies even within it. The range is device-specific, and manufacturers usually don't disclose actual characteristics. Some literature describes average range as 100 Hz – 4 kHz (Long 2012, p.12), however multiple amateur tests on the Internet indicate that for modern devices it is closer to 300 Hz – 8 kHz (Pons, 2014; Imgur.com, 2016). Small speaker size also requires sound designers to reduce dynamic range and increase the average volume of the audio to -16-18 LUFS (ASWG-R001, 2013, p.6), compared to -23 LUFS, used in console games and broadcasting (R 128, 2014, p.4).

To summarize, I will briefly mention the main technical constraints and those aspects of game audio design they affect in the table below.

Memory	Amount and quality of content in the game
RAM	Amount and quality of content in one scene
CPU	Real-time processing capabilities, compression algorithm, amount of sounds playing at the same time
Physical voice limit	Amount of sounds playing at the same time
Speaker size	Frequency and dynamic range of the mix

The aspects mentioned above set very high requirements to the process of sound system design, including sound assets, events and banks management. The first stage of audio pre-production typically starts with an audio design document. This document might contain:

- Description of typical in-game actions and entities that would require sound effects;
- Description of categories of sound effects;
- Target amount of sound assets per certain kind of sound effect;
- List of sound banks and their description;
- Technical specifications of sound assets, including number of channels (mono/stereo), sample rate and bit depth of the asset;
- Physical voices and mixing buses routing information;
- DSP effects settings and description;

- Project-specific information on sound system configuration and sound design principles.

Insufficient planning or ignorance of technical limitations often leads to the situation when some audio content and systems end up being dropped out or oversimplified. Obviously, such decisions heavily affect aesthetic qualities of the game. In practice, this kind of problems are often revealed at the end of production, when required serious fixes might be not feasible due to time constraints.

But some problems can still be solved with proper content optimization. For memory and RAM there is a widespread practice of looking for a compromise between file size and audio quality. Sound designers can apply different sampling rates to categories of assets, setting custom Nyquist rates for different types of audio. Most important and distinct sounds can be sampled at 44.1 or 48 kHz, while 11 kHz might be enough for distant ambience and room tones. Additionally, the size of audio files can be optimized by cutting unnecessary tails and converting stereo files into mono (Stevens and Raybould, 2011, pp.34-42).

Outside of that, the overall audio size can still be reduced with a more "granular" approach, the power of randomness and real-time processing. For example, we can get rid of long ambience loops by replacing them with shorter ones and adding some randomized accent sounds on top of them to make the ambience sound unique and non-repetitive. Instead of using a pre-recorded 2-minute loop of forest ambience with animal and bird sounds, we can use a 15-second loop of clean forest ambience and randomly play various bird and animal sounds on top of it. Instead of creating 20 variations of the same sound effect, we can split it into several layers with a few randomized variations and allow the system to generate different sounds on the fly. This approach not only helps to

reduce the overall size of sound banks, but also makes content less repetitive. The downside of this method is that it takes at least twice more physical channels than a simple audio loop.

CPU and physical channel optimization, on the other hand, is often based on reducing layers and amount of real-time sound processing. Five layers of randomized assets per sound effect can create lots of variations, but at the same time they will take at least five physical channels per game object, and significantly more CPU power to decode. CPU load can be also be reduced by switching to a less expensive conversion format, but results would come at the expense of quality and/or file size. (Jean, 2017)

In general, memory and CPU optimization strategies tend to clash with each other and require tough decisions and compromises. This creates additional pressure on sound designers who need to oversee the entire project and understand all possible technical challenges from the very beginning of development. Fortunately, audio middleware like Wwise (Audiokinetc), FMOD Studio (Firelight Technologies), Fabric (Tazman-Audio) and ADX2 (CRI Middleware) offer a diverse set of tools to deal with these challenges through their authoring tools (Freeman, 2018).

Hardware speaker problems stand separately from those mentioned above and, in many cases, can be resolved at the stage of mixing the game. At the same time, it still makes sense to avoid using problematic frequencies and frequency ranges for important sound effects when designing audio assets.

1.2. Design aspects

To understand the creative challenges of mobile game audio we need to take a deeper look into their design principles. Most successful mobile games are designed under service-based, free-to-play models and meant to be played with short <10-minute sessions over months and years. (Clark, 2014, pp. 2-15) This means that players will experience a repetitive core gameplay loop every time they play the game. The key role of sound in this game design paradigm is to enhance and deepen the gameplay experience without standing out too much and becoming annoying even after a significant amount of play sessions.

One of the most important parts of any free-to-play game is the first launch experience. Since the players haven't invested any money to get a copy of the game, they might be less motivated to play it for the long term (Roseboom, 2016). Thus, the role of the first launch increases dramatically in comparison with premium games, where players make a decision to play the game before buying it. The role of audio here is to support the first launch sequence, emphasizing and fulfilling everything that happens on the screen.

Free-to-play game developers earn money through microtransactions – small purchases players make to have a better gameplay experience. Thus, in-game purchases should feel especially rewarding, in both visual and audio aspects. The tricky thing here, again, is to maintain a balance between being rewarding and being irritating. Most dedicated players will hear reward sounds hundreds of times, and even then, the experience should remain pleasant for them.

In his 2013 Game Developers Conference presentation, Nicolas Thomas pointed out that modern casual, social and mobile game design is basically "Fighting back against the mute button" (Thomas, 2013). It is very common to hear that the majority of players turn

the sound of their mobile devices off while playing the game. This statement is still hard to prove wrong due to lack of research (Lofgren, 2017), but it always helps to understand basic circumstances when the game is going to be played. I.e. simple puzzle games designed to kill some time in public transport will require very different approaches than competitive multiplayer games that need a good internet connection, longer play sessions, and lots of attention from the player. But even with the simplest "time killers", it should be important to maintain high quality standards in every aspect of production.

These principles were established mostly for casual games that used to dominate the Western mobile game market for years. And while the casual market is still big and strong, recent years indicate the rise of mid-core games, targeted to different audiences and designed with a different approach. The next chapter explores unique characteristics of mobile mid-core and investigates how they can possibly influence audio design.

2. Characteristics of mid-core mobile games audio design

According to the website Deconstructor of Fun, 14 of the 15 highest revenue mobile games of 2017 belong to the mid-core sector (Das-Gupta, Katkoff and Payton, 2018). Often described as the missing piece between long-established casual and hardcore segments, mid-core is often called the new trend, and many game developers are changing their focus towards this segment of the market. This chapter tries to define what a mid-core game is, what kind of people compose the mid-core demographic, and what is special about mid-core games in terms of audio design.

2.1 What is mid-core and how do people play it

The term “mid-core” has two interconnected meanings. One of them stands for audience of certain type of games and another – for games, crafted for this kind of audience. In 2013, shortly after the term became popular, famous video game portal Gamasutra posted an article “What the hell does 'mid-core' mean anyway?” where they collected opinions from industry professionals as well as regular people who follow them on Twitter. They got a diverse selection of opinions. Some people were implying that mid-core is a purely marketing term to present casual games to hardcore audience. Others were saying it is a name for hardcore games played on more “casual” devices like smartphones and tablets. Majority, however, defined mid-core games as games, that offer more complexity and challenge than casual games while staying easily accessible to a broader audience. (Graft, 2013)

The same article cites Tony Goodman from PeopleFun (cited in Graft, 2013) who divides games audience by lifestyle patterns:

Hardcore	Arranges daily schedule around gaming
Mid-core	Arranges gaming around daily schedule
Casual	Entertains self with games when time presents itself

This division doesn't precisely correlate with how games themselves are classified – at least because there are people who play hardcore games, arranging play sessions around their daily schedule, or when time presents itself. At the same time, it provides some ideas about what kind of people play mid-core games and their level of engagement. Mid-core players care about games, like this type of leisure, but don't spend too much time with them.

The author of Polygon, Tracey Lien, mentions three key factors for the rise of mid-core in her article "Core gamers, mobile games and the origins of the mid-core audience". The first of them is demand from the side of the hardcore audience – people who grew up playing games, and either don't have time for multiple hour-sessions anymore or just want something to play on their mobile devices. The second is maturation of casual players who previously played a lot of simple mobile games and now look for more challenging, engaging and sophisticated experiences. The third factor is technology advancement that made mobile devices more powerful and capable of running more complex games. If we think that hardcore games are mostly designed for consoles and PC,

while casuals dominate smartphones and browsers, then arguably the key target platform for the mid-core segment is tablets (Lien, 2013).

The exact location of mid-core on the spectrum from casual to hardcore is still debatable. However, Deconstructor of Fun places it closer to the hardcore edge, saying "we position mid-core as a lighter, more accessible take on a hardcore theme or a genre" (Das-Gupta, Katkoff and Payton, 2018). For the scope of this thesis I'll stick with the similar concept of mobile mid-core, defining it as deep, complex and immersive games, designed for shorter sessions and meant to be played on (or fully adopted for) mobile devices.

Mid-core mobile games exist within free-to-play games as a service paradigm, following trends of mobile game market in general. At the same time, it is important to understand, that mobile mid-core is not a game genre, but a variety of different genres appealing to certain audience and fitting their playstyle. Borders between casual and mid-core or between mid-core and hardcore games are vague and somewhat fluid (Warman and Fowler, 2012). However, modern mobile mid-core games share some characteristics or common grounds. Michail Katkoff mentions some of them in his series of posts "Mid-Core Success" (2014):

1. **Dual loop**. Core loop of mid-core games consists of two loops. The shorter one – i.e. collection of resources – may last for less than a minute, but still contributes towards player's progression. The second one contains "main" part of the gameplay – i.e. battle – and takes some minutes to play. Players are tempted to open the game and perform the shorter loop even when they don't have time to play through entire core loop – i.e. on the go.

2. **Short sessions**. Mid-core inherits short duration of sessions from casual games, and this factor is crucial for accessibility: players should be able to run the game in various contexts. Short sessions become possible when the main gameplay loop (second part of the dual loop) lasts less than five minutes. As mentioned above, long sessions are also not uncommon – they consist of several short loops played one after another.
3. **Metagame**. Metagame is a part of the game outside of the core gameplay. It is the area where players manage their resources, create strategies, interact with each other and set goals for further progression (Clark, 2014, pp. 54-55, 95-98). Katkoff calls metagame the most distinctive element of mid-core games. This element enables longer play sessions, adding gameplay depth between repetitions of the core loop (Katkoff, 2014a).

Additionally, Katkoff points out that the majority of successful mid-core games have some social or competitive component (Katkoff, 2014b). Despite that there is some room for exceptions, it can be seen as another characteristic.

Finally, outside of a game components perspective, it is worth to point out that mid-core games generally have much more in-game content in comparison to casual games. This is a logical consequence of more complex mechanics and higher level of immersion mid-core games offer to the players.

To summarize, mid-core mobile games follow free-to-play games as a service model and tend to share three crucial features: dual loop, short core loop and extensive metagame. In addition to that, absolute majority of mid-core games features social and/or competitive component and tend to me more content-heavy and complex in comparison with casual games. Some examples of successful games that have all of these

features are Marvel: Contest of Champions (2014), Clash of Clans (2012), Clash Royale (2016), Star Wars: Galaxy of Heroes (2015) and Hearthstone: Heroes of Warcraft (2014).

Combination of short core loop and extensive metagame creates a wide variety of situations mid-core games can be played. “Mobile Gaming Cross-Market Analysis” report by InMobi specifies that people often play mobile games while commuting, at the office or work, at home and specifically while watching TV (InMobi, 2014). The last context might not work for mid-core games that tend to require more concentration to the player, but three other modes perfectly fit the mid-core paradigm. At the same time 2013 demographic breakdown of Clash of Clans shows that the game is usually played during commutes, at washroom brakes, in living room, in bed and with friends (Mason, 2013). This data indicates how flexible mid-core games are in terms of gaming contexts and time they are usually played.

2.2 Mid-core games from audio design perspective

Mid-core is a broad term that includes a variety of genres with different, sometimes incompatible approaches to sound design. However, characteristics, mentioned in the previous section, indicate existence of some common ground in their game design principles. These characteristics, combined with understanding of how and when these kinds of games are played, can help us define the key aspects of mid-core games audio design. To do that I will look at the key points from the previous section from audio designer’s point of view.

The structure of a dual loop implicates that the shorter part happens almost every time the game is launched. During this loop player can collect resources, set strategic

goals or perform other quick actions. Users will repeat these actions every session, or sometimes even multiple times per session, so their sonic feedback requires careful treatment. For example, if the first loop is focused on resource collection, it makes sense to design key sounds to feel rewarding, but not too impactful – to leave some room for the contrast with bigger rewards.

Short sessions indicate significant amount of repetition in the core gameplay. It means that the players are going to interact with the same entities over and over again, getting the same kind of visual and sonic feedback from these interactions. It is important to make sure that players don't grow tired of this feedback. Common practices include creating sufficient amount of variations for frequently repeating sounds, introducing controlled randomization for pitch and volume and some elements of generative audio. These practices might be tricky regarding technical limitations of mobile platforms, but some practical solutions will be described in the last chapter of this thesis.

Additionally, sound designers need to get rid of other factors that might cause irritation by ensuring that audio content always feels natural: it should be coherent, properly mixed and always in sync with animations and visual effects. Again, this is not a mid-core specific requirement, but a basic set of recommendations to improve any game audio (Scolastici and Nolte, 2013, p. 103).

Extensive metagame is important for long-term player retention. Typically, players will be engaged in the metagame through a varied set of user interface elements. They are expected to spend a lot of time in different menus, operating in-game currency, upgrading various entities, changing settings of the scene, etc. Touch screens of mobile devices are unable to provide haptic feedback for this kind of interaction, so sound becomes a tool for improving responsiveness and overall feeling from using the interface

(Noonan, 2013, pp.5, 6, 9; Daw, 2017). And since significant part of gameplay session requires interaction with UI elements, they should feel natural, responsive and satisfying to press while staying subtle and non-irritating. Failure to meet those criteria makes interaction annoying to the player, who can get tired and press the mute button. Obviously, not only mid-core games benefit from good and well-crafted UI sound design, but for them special attention to user interface sounds becomes an actual requirement.

To go further we need to understand main functions of audio in video games. Mobile game developers mostly see audio being complementary to the visuals (Collins, 2008, pp.78-79). In casual games sound is perceived as a purely aesthetic component that enhances the immersion (Kassabian, 2015) and forms sonic identity as a part of a brand (Thomas, 2013). Despite some examples of non-casual mobile games where sound is essential for the gameplay (Papa Sangre, 2010; Lost in Harmony, 2016) they are considered experimental or niche products, while many mobile game developers design their products to be fully playable with no audio at all (Scolastici and Nolte, 2013, p. 61).

In console and PC games audio has some extra functions. In his Master thesis "Informant diegesis in Videogames" Bjørn Jacobsen points at the difference between sounds that support the narrative and emotion and sounds that provide players with information they might use in the game (Jacobsen, 2016). The article "Informative Sound Design in Video Games" by Patrick Ng and Keith Nesbitt (Ng and Nesbitt, 2013) adds extra support for that statement, opposing immersive approach, where audio complements visual information, to informative approach when audio itself provides useful information to the player.

Mid-core games that feature deep, often competitive gameplay have all chances to adopt informative approach from the bigger platforms. We know that some portion of the

mid-core audience grew up playing games. These people are familiar with game audio aesthetics and perceive it as a natural part of gameplay experience. They don't just expect sound to be present, but also rely on its immersive and informative functions (Jørgensen, 2008). Strong competitive aspects in popular mid-core games create extra demand for useful information from highly engaged players.

On the other hand, we don't want to make the no-audio experience unplayable or noticeably limited because it will affect accessibility. There are players who don't want to listen to any audio in a mobile game and there are situations when playing on a mobile device with sound is unacceptable or impolite (Heubel, 2015). If players think that having audio on is necessary for gameplay experience, they won't run the game in these situations. Since we don't want to limit variety of gameplay contexts, it is generally better not to provide valuable information exclusively with audio and limit situations when audio information gives the player any advantage in competitive play. Some examples of informative audio design in a mid-core mobile game will be given in the next chapter.

Finally, gameplay complexity and a bigger amount of in-game content should set higher standards in terms of audio production values, use of audio middleware, dynamic audio system design and audio design in general. This is, however, a vague generalization based on plain logic and common sense. It doesn't describe real situation in a very diverse market of mobile games, influenced by plethora of factors.

Mid-core mobile games inherit many principles of casual mobile audio design and mobile game audio design in general. Most of them come from hardware limitations, described in the previous chapter. Others a tied to variety of contexts and situations to play the game. However, there are some aspects of mid-core game design set their own requirements towards audio. Apart from increased complexity and amount of content,

mid-core games need more detailed approach to UI and metagame-related sound design and can benefit from informative audio. Use of the latter, though, should be somewhat limited in a way it doesn't affect ease of playing the game in different contexts.

3. Design principles behind the Game audio

The Game is a massive project with hundreds of sonic entities and thousands of individual audio assets. It has a bigger scale than the majority of projects on the market for the same platforms and this scale itself brings many different challenges. In addition to that, the game is based on a Franchise with a strong sonic identity and aesthetic principles that create extra constraints for audio design process. Outside of that there are technical and design limitations of mobile platforms that were explained above. In this chapter I will go through the key principles of the Game audio design and explain creative choices we made and challenges we had to deal with.

3.1 Project overview

The Game is a card battle game developed for iOS and Android platforms. It features both PvE (Player vs. Environment) and PvP (Player vs. Player) modes and requires players to build their card decks and use their cards to play through pre-designed levels or compete each other.

The Game features around 100 cards, each representing a unit, a group of units, a structure or a magic spell on the battlefield. Player doesn't directly control units, but decides when and where to spawn them and when to activate unit's special power if it is available. Different cards cost different amount of energy, which is a resource that restores with time during the battle. Cards can be upgraded outside of the battles with special items and soft currency. Items and currency are earned as rewards for PvP matches, completed PvE levels or from opening card packs.

PvE mode has 60 levels that can be replayed multiple times for new rewards. PvP mode doesn't limit the amount of possible matches, but caps the amount of rewards that players can get in a certain timeframe. Outside of core gameplay modes there is meta-game, where players can update their decks, upgrade cards, join teams and make purchases in the ingame store.

The Game belongs to a bigger media franchise and follows its aesthetic guidelines – both visually and sonically. The Game uses 2d-graphics with non-realistic, cartoonish visual style, and requires the same, often schematic, approach to sound design. For example, certain sound effects appear oversimplified, and most of the characters don't make any sounds when they walk.

The Game was developed with Unity engine and Audiokinetic Wwise was used to handle game audio.

3.2 Team structure

The audio department of the Company consists of 3 people: lead sound designer, who oversaw the project from both technological and creative perspectives, one senior sound designer and myself as a junior sound designer. All three of us have worked on the Game, but the other two team members were shared between different projects. I joined the Company's audio team when some part of audio work had already been done and worked solely on the Game until its release. A few months before the release, when most of the content was finished, I became the only audio designer actively working on the Game, while the rest of the team switched to other projects. At the time of publication of this thesis, I'm responsible for the Game post-launch audio support. While we didn't have

a dedicated audio programmer for the project, one of the gameplay programmers was assisting us with the tasks that required code support.

Our audio department oversaw most of the aspects of the Game audio design process except of voice-over and music production. Those exceptions were made because music and voices are a crucial part of sonic identity of the Franchise, so they were provided by the third party. Music design, final voice over editing and implementation were done on our side. In addition to that we had an opportunity to reuse sound assets from previous media products developed under same franchise.

Working as a relatively small team, we had freedom to choose our own tools like digital audio workstations and plugins for audio asset creation and processing. However, we followed a specific naming convention for asset creation and implementation guidelines. Most aspects of technical audio design were clearly documented and made available for entire team working on a project.

The implementation process was organized in a way that audio design team was working almost exclusively in Wwise without changing anything in code or Unity editor. Naming convention helped us partly automate integration of the most standard audio events. Some other audio events, which were mostly connected to in-game special effects and superpowers, could be manually integrated via custom-made web-based tool. Integration of the rest of audio events was handled by gameplay programmer.

3.3. Aesthetic and design principles

The Game is set in a fictional world where characters play some sort of live-action role-playing games about imaginary worlds, inspired by popular culture. The game has four settings or themes representing different kinds of fictional settings.

The Game characters have various roles, wear different costumes and use hand-crafted props to role-play popular fictional characters belonging to four different settings. Some elements of the game they play, however, are not physically represented in the Game world and only happen inside characters' imagination. Magical spells, curses and blessings are the most common example of these elements.

We decided to represent this duality by taking two different creative approaches to sound effects design. When designing real world sound effects, we tried to make them simple, raw and realistic. Imaginary sound effects, on the other side, were designed very magical, otherworldly and over-the-top. For example, a wooden stick that represents some legendary magical sword of a mighty warrior should still sound like a wooden stick when it hits an object. But a healing ray from a fictional magician, being an imaginary object, should sound as fairy and magical as characters would imagine it. The same rule applied to UI sounds: physical objects like buttons sound very realistic, while extraordinary events with lots of visual effects have sparkling, saturated sound events.

This decision lead us to few straightforward design rules. "Real-world" sounds are mostly done in mono, with very little or no reverb and other effects, and attached to existing objects as 3d-sounds. "Magical" sounds on the other hand are often stereo, 2d and have some extensive post-processing.

Music and voice-over assets were provided by the third-party, so we had little control over their production. Those assets were coherent with aesthetic style of the

Franchise from the beginning, and we didn't need to worry about them outside of the implementation process. More information about handling of music and voice assets is provided further in corresponding sections of this chapter.

From the design perspective, we wanted to make sound informative and useful to the player, while being pleasant and immersive. Our main principles were:

1. **Distinction**. Every character or spell should be easily identifiable by sound. We considered, that user can play the game in various situations, including commutes and public places. We wanted to make sure that players can get information about what happens on the battlefield even if they look away from the screen to check surroundings.
2. **Locatibility**. Sound position should reflect location of the source in stereo field. If the players from the previous example wear headphones, they should be able to understand not just what happened, but also where it happened on the battlefield.
3. **Informativeness**. In certain cases, audio can give player extra information about what is going to happen in the scene before any visual indication. For example, in PvE the player can sometimes hear that enemy unit is spawned outside of visible area before that unit can be seen on the screen. At the same time, we didn't want to go too far in this aspect. Regarding the variety of situations the game can be played, we didn't want to provide players who play with audio on with any information that would give them significant advantage other the players who chose to play without sound.
4. **Fun**. We simply wanted to win our battle with the mute button by making audio as pleasant and entertaining as possible.

Locatibility might need some extra explanation. We placed most of the sound effects in stereo field as 3D sounds to make them locateable. At the same time, we understood, that in many cases the Game audio would be played through mobile device speaker, so we put extra effort to make the mix mono-compatible. We used a relatively narrow stereo field of approximately 60 degrees to reflect how players would get visual information, playing with a tablet or a smartphone. Our actual steps to achieve these goals are described in chapter 4.

3.4. Technical principles

As I mentioned before, The Game is a massive project with thousands of individual audio assets and overall audio budget of 100 Mb. Mobile devices can't offer much resources to handle audio, so sound design process for the game of that scale should start with defining a proper way to manage assets.

Our approach was to spread audio assets into many small sound banks for fundamental entities, such as units, special effects or menu scenes. With this idea in mind we ended up having over 100 sound banks. Size-wise each of those was within the range of 50 to 700 kilobytes. Thus, we were able to have a precise control over RAM, loading necessary sound banks when they were needed. For example, at the start of PvE level we know in advance what units players will use and what kind of enemies will be present, so we can load corresponding sound banks together with other game assets.

Some units had similar abilities, so we wanted them to share certain sounds – usually ones that correspond to different status effects. Those were included to a special sound bank that is loaded at start of any level or PvP match. As a rule of thumb, we

decided that sound effect would be included into this sound bank when it is used by more than 2 entities, including spells. The only sound bank that is loaded at every scene (including menu scenes) contains basic UI sounds that are shared among most of the scenes.

Big audio files like music and main ambience loop were streamed from the drive. Streaming audio is a widespread practice in modern game development that helps optimize usage of RAM. The downside of this method is that streaming introduces some delay to audio playback, that might cause synchronization issues. Wwise has an option to create a buffer – small fraction of audio file that is stored in RAM to compensate the latency. We used this feature for streaming music to avoid synchronization problems. Ambiences didn't require zero-latency playback, and we decided to stream them entirely from the drive.

For physical channel limitations we had to follow existing mobile audio guidelines and keep a reasonable amount of simultaneously playing sound effects. Audiokinetic recommends targeting below 40 channels for lower-end mobile devices (Jean, 2017) so we decided to lock maximum sound instances level at 32. Despite the game not being meant to be played on low-end mobile devices we found out that massive battles with over 10 units on the same screen turned out to sound chaotic and cacophonous. This kind of limitation allowed us to clear the overall mix and save some CPU bandwidth for real-time processing.

Following 32 physical channels rule, we decided to limit the amount of sound instances per action or event. Audiokinetic Wwise offers sound designers a lot of freedom with its blend and random containers workflow, allowing to produce many variations of one sound effect by layering sounds from different randomized sets. There is, however,

always a compromise, because more layers take more physical channels and CPU resources. After running some tests, we decided that none of the unit's actions should take more than 3 physical channels: one for voice-over and two for layers of a sound effect. This measure alone brought us to the situation when the limit of 32 channels was breached in less than 5% of cases.

However, with such a hard physical voice limit we still had to make sure that the most important sounds, such as voice lines or music, will never get cut or virtualized. To do that we specified priorities for different sound effects categories, having music, UI and voice-overs as high-priority, attack- and status-related sounds as medium priority and movement sounds and weapon swings as low-priorities. The low-prio category included the quietest sounds that would not be audible in a heavy-loaded battle scene anyway.

In addition, we wanted to virtualize sounds that are too quiet to be audible, but still exist in the scene and use CPU and physical channels. We achieved that by setting a rule to virtualize every sound below -60 dBFS. This value might look too low for a mobile game with a relatively small dynamic range, but we had to keep it that way for players who play the game with headphones.

The big memory budget combined with the relatively simplistic style of the Franchise helped us achieve most of our goals without making many compromises. One of those was a limited number of layers per sound effect. Adding more layers would give us even greater amount of variations, but we wanted to stay reasonable and make sure that other aesthetic choices don't affect the performance.

Another compromise is very common for game development in general and mobile games in particular: we had to sacrifice certain portions of sound quality to reduce file size. Not too much, however. Audiokinetic Wwise offers several conversion formats for

different platforms, but ADPCM and Vorbis remain the most common and practical, especially for multiplatform release. ADPCM is a typical audio format for mobile platforms (Audiokinetic.com, 2018a) that is very light on CPU and memory but loses in sound quality to more CPU-intensive Vorbis. Wwise Vorbis implementation, however, allows audio designers to specify conversion settings per group of assets using the Quality Factor setting. This option provided us with enough flexibility to control quality of individual assets, so we choose Vorbis as our conversion format despite its higher usage of CPU.

4. Practical aspects of the Game audio design

The previous chapter provided an overview of general principles behind the Game audio design. This part of my thesis describes how these principles were applied in practice. Five sections of this chapter are focused on core gameplay audio design, UI and menu sounds, voice-over design and implementation, music implementation and mixing of the Game. Contents of this chapter shouldn't be seen as a guideline for mid-core game audio design. Creative decisions documented below are based on design principles of one project and may not work for other games of similar genre.

4.1. Core gameplay sounds

The Game is focused on battles, so gameplay audio mostly consists heavily focused on hits, stabs, punches, shots and supporting movement sounds. Battles are formed with three basic groups of entities: player avatars, spells and units. Each group features a set of actions or attributes that vary among different entities, but follow almost the same structure within a group:

Player avatars have sound events for following actions:

- Spawning a unit;
- Attacking an enemy unit;
- Attacking several enemy units at once;
- Getting damaged;
- Getting defeated.

The Game allows up to two player avatars to be present at the same time. In that case they have identical sounds, that panned to the opposite sides of the stereo field, according to avatars positions on the screen.

Spells have following sonic attributes:

- Spell trigger;
- Spell effect.

The "Trigger" part is played on player's avatar that casts the spell. "Effect" part is played at the center of area of effect or as a 2d-sound if the spell is applied to entire battlefield.

Units are divided into 3 subgroups: melee, ranged and totems. Those subgroups have slightly different structure. For example, melee units trigger sound events with following actions:

- Spawning on the field
- Landing on the field
- Swinging a weapon while attacking
- Collision – hitting an enemy unit
- Using a superpower
- Getting damage
- Getting defeated

Ranged units use a similar system but have additional sound for shooting. Swinging a weapon in that case is replaced with raising a weapon, and the collision sound event is triggered when the projectile reaches an enemy unit. Totems don't have a sound for defeating the last visible enemy, and in many cases don't have any attack-related sound

effects. Most units walk silently, but some of them have exceptional movement sounds when necessary.

Since almost every action with a sound effect might fall under a certain category, we developed a naming convention to automate the process of triggering audio events. Every audio event related to units was named along the lines of Play/Stop/Pause/Resume/Break*Unit name*Theme*Action. This allowed us to simplify the development process by reducing amount of times we had to trigger implementation from the programmer.

Our main creative goal for character sound design was to make sure that every unit sounds unique and distinctive without breaking overall sonic aesthetics of the Game. At the same time, we had to keep in mind sonic identity of the Franchise. And in addition to that we needed to make enough variations for every sound effect to avoid the "machine gun effect" – frequent repetition of the same sound that is considered very undesirable for an action-packed game (Chang, 2016).

We analyzed other Franchise products from sound design perspective and decided on a set of attributes to describe it. We came up with a set of descriptions that included words like "hasty", "rough", "simple", "clean" and "impactful". So, in general we tried to make our sound effects nice and pleasant, but also to add a feeling of roughness to give an impression that they were created in a hurry, in bold strokes.

First, we came up with a set of generic sound effects like punches, "whooshes" or body falls that perfectly fit desired style and can be used by different units. Those sounds were designed as a unifying component which would be layered with other unit-specific sound effects. Different units use those sounds with different pitch and volume settings

that depend on their visuals. For example, units that look slow and heavy would use lower and possibly louder version of the sound.

Second layer was designed for every unit from scratch, to make it sound distinct enough in the mix. This is the most prominent part of the sound effect, the part that defines what action sounds like. Some actions didn't use generic sound effects, so both layers for them were done from scratch.

Each of the layers is based on a random container with 3-5 sound variations and a rule, preventing system to play same asset two times in a row. In practice it means that every time sound event is triggered system will play random variation of sound 1 together with random variation of sound 2, making a layered sound 3 out of them. Having 3-5 sounds under each container allows us to create 9-25 variations of sound 3. Originally, we went even further, having up to 5 layers per sound effect, but we had to get rid of additional layers by baking sounds together during optimization phase. However, we added some more variation by slightly randomizing loudness, pitch and, in certain cases, high-pass and low-pass filter settings for individual layers.

Big part of units' individuality also comes from the voice-over lines recorded by the same actors who work on other products under the same franchise. Having those assets allowed us to focus on overall sound aesthetics so we didn't have to design our sound effects being too different from each other aesthetically to ensure their distinction.

As mentioned before, the game has four different settings, but creating different ambience or set of ambiences for each of them would be too expensive for memory budget we had. We chose a different approach having level ambience based on the same loop for every setting with additional theme-specific elements randomly playing on top of

it. This technique allowed us keep ambiences small but diverse and avoid audible repetition.

The main ambience loop was still too big to be loaded into the RAM, so we decided to stream it from the drive. This technique allows to exclude most of the file from the sound bank leaving there just a small portion of it from the beginning. Downside of this technique is that it might introduce small delay before starting the playback, but it in this case it was not considered important.

4.1.1. Core gameplay dynamic sound systems

Despite simplistic aesthetics of the game and technical constraints we still had to implement dynamic audio systems for some entities. Fortunately, we were able to achieve everything we need with standard Wwise Real-time parameter control (RTPC) tools.

The first one is additional attenuation system that decreases volume of entities based on their position. Despite all of the character sounds are 3d and follow general distance attenuation rule, we wanted to get a bit of extra control over individual unit volume. It would potentially help us clean the mix from sounds that play too far away from the visible area but keep ones that have informative value. Instead of modifying distance attenuation curves for individual sound effects we applied volume RTPC to the actor-mixer of every unit.

In most cases we used sine attenuation curve that would not affect units that are present on the screen (values 50-100 on X axis on Figure 1) but heavily attenuate ones that are located outside of visible space (values 0-500 on X axis on Figure 1). This setup allowed us to safely virtualize all sounds from units that spawn too far away from visible

area. At the same time, we were able to decide how far away certain actions would be audible on unit-specific basis.

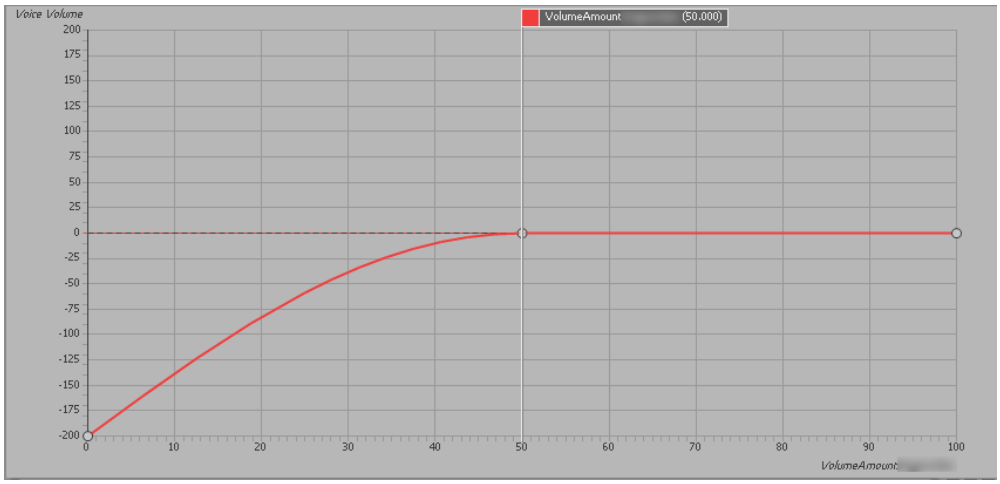

Figure 1. Custom volume attenuation curve for the unit

Another system modifies pitch of every sound unit has based on its speed. Certain spells and effects in the Game allow units to be sped up or slowed down via different buffs and de-buffs. These effects increase or decrease animation speed, and we had to make sure that audio reacts to these changes. Basic resampling that changes pitch of the sound with its duration was a perfect fit for our needs, creating a nice cartoonish effect for both sped-up and slowed-down characters. So, we decided to linearly automate basic Wwise pitch parameter on actor-mixers of our units based on the speed. Speed that can take any value between 50% and 200%, which corresponds to ±1 octave in pitch.

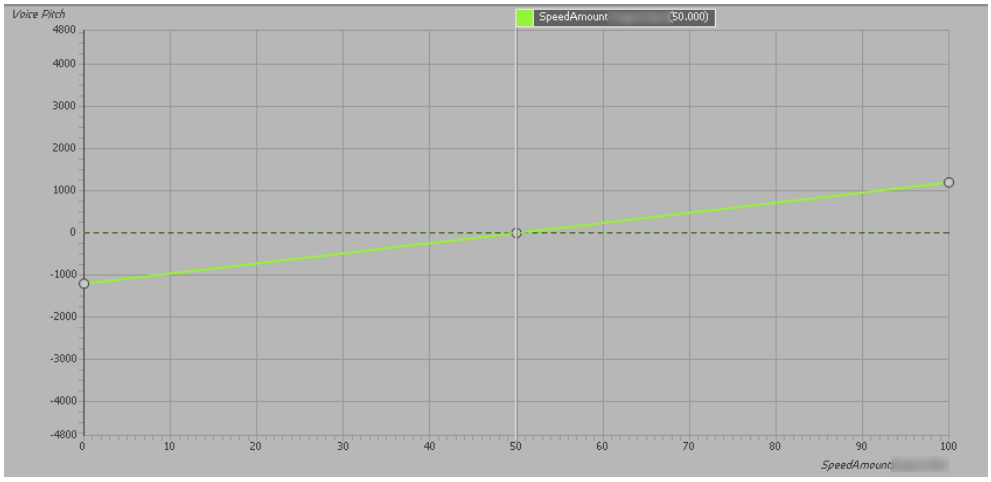

Figure 2. Pitch automation curve for the unit

During the testing phase we found out that sound of low-pitched (slowed-down) units has too much low-frequency content and added another set of RTPC to control high-pass filter to prevent it from happening. Additionally, we applied 0.7 seconds interpolation time to the game sync to make pitch changes sound smoother.

To get a precise control over every single unit, we created a set of game syncs (for distance and speed) and RTPCs (for volume, pitch and high-pass filter) for each of them. Sync objects are automatically recognized by the game when their name follows the naming convention.

4.2. UI and menu sounds

The Game features an extensive menu system where players can customize their avatar, modify decks, upgrade cards, interact with teams, make various kinds of purchases and perform other actions. As it was stated in Chapter 2, role of metagame in mid-core games is hard to overestimate. Therefore, we paid special attention to UI-related audio, trying to make it very responsive, satisfying to press and non-irritating.

Most of mobile games play UI sounds when user releases the finger from the screen. Playing sounds when user places finger is way more responsive – especially with longer presses – but potentially harmful for the user experience, since users sometimes press the button, then change their mind, move the finger away and releasing finger outside of the button area to prevent action from happening. In this scenario, playing sound events on tap would give user a false message of triggering the button – unless the button is actually triggered on press which increases the amount of accidental presses.

Our solution to this problem was to make UI sounds out of two parts. The subtler one is triggered on button press and the second, more prominent one – on release. Since the second part contains more information, we decided to make different sound effects for release of different button types, but the same sound effect for all of the presses: the only function of this sound is to tell users that something is about to be triggered. This kind of system is applied to most buttons in the game, except of ones that don't have "press" animation. Another advantage of the two-part UI sound system is that it reduces irritation from repetition of the same sound effects using natural variation of press duration. Subtle element of generativity makes two sounds, playing with slightly different amount of time in-between, cause less fatigue than one repetitive sound effect (Ihalainen, 2015, pp. 109-110).

We decided to make all button-related sounds 3d. It means they will be panned according to the button position on the screen. This is uncommon way of implementing UI sound effects: those are usually played as 2d sounds in the center or hard panned to different sides of stereo field. We had two main reasons use 3d sound approach. First, we wanted to increase responsiveness of in-game UI and make sure that every sound will be

played from the correct spot in the stereo field, even if this would be only perceived by players who use headphones. Second, we wanted to optimize the development process.

The Game has various menus with a lot of buttons. Position of those buttons was changing a lot during the development. Sometimes new elements were added to certain places and old ones were getting moved to a different place on the screen. If we followed conventional method with 2d UI sounds, we would have to follow every single change, modifying panning of existing sounds or adding triggers for the new ones. 3d approach allowed us to create prefabs for different sounds that would have proper triggers inside them, so UI artists were able to copy buttons to any place within the scene and stereo positioning always stayed correct.

However, 3d approach has its own downsides. First, it requires additional audio listener that will only react to UI. This necessity comes because UI elements don't usually exist in the same place within the scene as the level itself. Instead they are projected with a different camera, so main audio listener is unable to properly track their position. Second, 3d sounds cost more CPU power than 2d (Audiokinetic.com, 2018c), because audio listener needs to track their position in the scene.

Menu sound effects include audio feedback from receiving currencies, opening card packs, upgrading cards and units. All these moments relate to a feeling of empowerment: player either improves character and card stats or gets resources for future improvements. So, our creative goal was to make sound effects that will fully reflect the visuals and feel as empowering and rewarding as possible. This lead us to development of several dynamic systems.

4.2.1 Dynamic system for currency SFX

In the Game, the player can get different amount of currencies by achieving various goals or making in-app purchases. The amount of currency players might get varies from very few to tens of thousands and this difference was accented via amount and duration of visual effects on the screen. We wanted to reflect those amounts sonically to make sure that getting more currency sounds more rewarding. At the same time, we didn't want to remove all the excitement from getting smaller amounts. Our first idea was to define different tiers, like 1-9, 10-99, 100-499, and make individual sound for each of them. But this approach would require us to make lots of variations, neither of which would precisely match duration of the visual effect or its position in stereo field. Another way would be to fire a sound event from every particle of visual effect to achieve perfect audio-visual sync, but in this case bigger amounts of currencies caused voice starvation. So, we decided to build a dynamic system to handle currency sound effects.

The visuals were showing a flow of currency appearing where reward is claimed on the screen and flying towards the top with some sparkles in the end. We split this process to three stages, each triggering a sound event. The first event (Start) is fired when the first particle of currency visuals appears on the screen. The second event (Collect) happens when the first particle reaches the top of the screen where the currency counter is located. The third event (End) is triggered with a final visual effect, when the last particle reaches the top of the screen.

For every currency we created two sets of over 15 very short, granular sound effects: one for emission and one for collection when particles hit the top of the screen. Each grain in those sets is a one-shot sound of currency handling for emission and impact for collection. Those grains would be triggered one after another (overlapping) with a

different rate, depending on the amount of currency we wanted to represent. In addition to that, we added RTPCs that would slightly raise volume and pitch of the sound for getting bigger amounts of currency. Thus, we could flexibly vary the intensity of two main sound effects depending on amount of currency we wanted to represent.

Start event would start the emission loop with intensity received from a game sync. Collect event would start collection loop on top of that, following the same game sync. Then, End event would play a final sound effect and cut both loops. The system was working fine until some bug broke a visual effect that triggered End event, causing the system to play both loops infinitely until the sound bank is unloaded. This situation made us create a fail-safe that would cut those loops even if the End event hadn't been triggered. To do that we added stop actions to Start and Collect events and calculated necessary delays for minimal intensity of currency flow. Then we used intensity game sync to automate delay time based on different intensities and durations of a visual effect.

4.2.2 Dynamic system for card pack opening

Card pack opening process is done in a very physical, skeuomorph way. Animation of a card pack upper part tearing away follows player's physical swipe across the screen and magical glow appears and becomes more intense with the finger movement. This is a moment of excitement for the players who are about to get new cards for their collections. To support this moment, we built another dynamic sound system of several loops that will blend together while player opens a card pack.

First it starts with a normal sound of plastic cover being torn. This loop becomes louder and higher in pitch while the player moves the finger. Later magical glow loop sound starts blending with the first loop to support the visual effect. If the player moves

the finger to the opposite direction, glow loop filters out, following the movement with no plastic cover sound effects. When the pack is fully open, we trigger a different sequence of sound effects and quickly fade both loops out.

4.2.3 Dynamic system for card reveal process

This is the least complicated system we created, but is still worth mentioning. The Game features 4 types of cards of different rarity. When players open a card pack they see back of the cards and needs to flip them manually. More rare cards have more complex visual effects when players flip them, and we created a simple system to match the visuals.

We created four switches in Wwise, one per card rarity. When player reveals a card, we know which rarity it is and use a switch to play different sound from the switch container. To keep those sounds consistent, we created an additive system of layers. Revealing the most common card would trigger a single, generic sound effect. Second level of rarity will add another sound effect on top of that. Third level will play another sound effect on top of what we hear on second level and so on. In addition to that on levels 2-4 we play a voice-over line by narrator, commenting on rarity of the card player revealed.

4.3. Voice processing and implementation

As mentioned above, all voice assets for the Game were provided by the Third party. Files we received were already processed in a right way, so in most cases we didn't have to care about it outside of basic high-pass filtering at ~100 Hz to make sure there is no

unnecessary low-frequency content. Outside of that we were focused on voice-over structure design and implementation.

Some of the actions mentioned for units in section 3.3 are accompanied by a voice asset. Roughly, we divided all our voice assets into two categories: grunts and quips. Grunts are short vocalizations of getting hurt or making effort. Quips are catch phrases characters say when performing certain actions. The table below shows average structure of the character's voice design.

Action	**Voice assets**
Spawning on the field	1-3 quips
Swinging a weapon (melee units)	3-5 grunts
Shooting (ranged units)	3-5 grunts
Using a superpower	1-3 quips
Getting damage	3-5 grunts + 1 quip (optional)
Getting defeated	1-2 quips + 1-3 grunts (optional)
Victory	1-2 quips
Boss entry	1 quip

While most of the actions are self-explanatory, it is worth it to explicate on Victory and Boss entry actions that weren't mentioned before.

Victory is a short voice line one of player's units might say after cleaning the battlefield of enemy units. First game randomly chooses a unit to say that line and then

that unit will have 15% chance to fire the sound event. It means that players won't hear the same lines too often, – at least way less often than other lines, – and it feels interesting and meaningful every time it is played.

The Boss entry is a special line that accompanies custom animation when player meets boss-version of a unit. This animation happens only in pre-defined moments in PvE mode, so we put this line to a separate sound bank together other boss-exclusive audio content.

Overall, most of the characters had over 20 voice assets in different random containers for 7 possible actions. With many units on the screen the Game could easily start sounding too chaotic and cacophonous. We addressed this problem in two ways.

First, most of the repeating actions are only voiced with grunts – short sounds that don't overlap with each other that much. Grunts are mostly used for getting damaged, swinging a weapon and shooting. While the first case is very important to communicate that the player's unit got hurt, the second and the third have more of aesthetic value, regarding the fact that those actions also have sound effects. Thus, we decided that these effort grunts will only play in 50-75% of cases depending on character. This allowed us to decrease the amount of voice assets playing at the same time and thus have more space for quips.

Quips have a longer duration than grunts, so they might lose some aesthetic and informative value when they overlap with each other. Some overlapping, though, was considered normal due to originally chaotic aesthetics of the Franchise. So, we decided to cut quips in two cases: when units gets hurt and when unit gets defeated. It caused a funny effect of a phrase being interrupted with a grunt and helped us clean the sound space.

Other than core gameplay, we used some voice-overs in the menus. Those belong to shopkeeper characters who can welcome the players and comment on their purchases and narrator who comments on rare cards that players can get from the card packs.

4.4 Music implementation

Music for the Game was also supplied by the Third party. The larger part of it was composed specifically for the Game, but some tracks were borrowed from previous products under the Franchise. This was done both to match budget requirements and to add some familiar tracks for the fans of the Franchise. The game uses mostly linear crossfading score with some adaptive features.

Aesthetically speaking, the music follows the same style as other products under the Franchise. The style varies from rather simplistic but catchy and well-produced hybrid orchestral, inspired by film music of 90's and 2000's in core gameplay parts to pop and jazz tunes in some metagame menus. The music generally relates to "imaginary" parts of the design paradigm described in section 3.2. Orchestration and motifs reflect non-real situations and settings, exaggerating on-screen action and creating funny contrast with rough and mundane "real-world" sound effects. Overall music creates a fantasy mood, supporting aesthetic duality even when none of the otherworldly elements are present on screen. In addition to that, it creates a comical feeling by enhancing contrast between imaginary and real-world domains.

Metagame scenes have around 10 looping music tracks assigned to different menus and views. Every functional menu, like avatar customization screen, deck builder screen, or the store has its own music that matches the mood and the atmosphere defined by the

visuals. Duration of loops varies depending on how much time player usually spends on selected screens, from about 20 seconds to almost 4 minutes. The most important screens, where player spends most of the time, have playlists of two tracks that crossfade into each other. Music tracks, assigned to these scenes, can start from several preselected moments to make the music sound less repetitive and predictable.

The only exception is an Options menu that can be displayed on top of every other screen. It doesn't have any dedicated music theme but applies low-pass and high-pass filters to already playing music track when the menu is activated. This way we make players understand that focus has shifted from the game scene to an external menu with game settings, announcements and navigation shortcuts to quickly access different sections of the game.

The main screen has 4 different versions of the main theme that can blend on the fly depending on selected setting. Variations and composed with the same duration, tempo, time signature, scale and harmonic structure, but feature different timbres and arrangements. This allows us to quickly and seamlessly crossfade one into another when player switches levels that belong to different settings on the main view.

Core gameplay has setting-specific music: each of four settings is represented with its own music themes. There are two cues per setting: one for the level itself and one for the boss battles. Level tracks are usually around two minutes long, following the duration of average play session. Boss battle themes vary from one to three minutes because certain settings feature harder and longer boss battles. The beginning of every boss battle is also marked with a stinger – short music fragment that plays during the boss entrance animation. Due to the budget limitations, stingers are not setting-specific, so we could just choose one out of 16 we had to make the closest match for the animation.

PvP gameplay doesn't belong to any specific setting and has its own playlist of music tracks. The track is randomly picked from random container at start of the match. If the match lasts over two minutes, the system increases music playback speed by 5% to build up tension and inform the players that match is going to end soon. PvP music ends with one out of three stingers for "win", "lose" and "draw" conditions, depending on the battle result.

All music tracks except of stingers are handled as streaming assets in the same way as ingame ambiences to reduce usage of RAM.

4.5. Loudness and dynamic mixing

Final mixing was one of the most challenging parts of making the Game sound coherent with other products under the Franchise. As many times stated above, mobile games are a challenging medium for sound designers. For mixing, the main challenge comes from the mobile device speaker. Inability to reproduce low frequencies, problems with reproducing high dynamic range content, variety of devices with different speakers on the market create a full set of constraints for our work.

Additionally, as was mentioned in chapter 1, loudness standards for mobile applications are different from the ones that are used in console games, film and broadcasting. While bigger platforms follow AES/EBU recommendations of -23 LUFS +- 0.5 LU (R 128, 2014, p.4), mobile developers follow Sony ASWG recommendations for portable platforms of -18 LUFS +- 2 LU (ASWG-R001, 2013, p.6). Difference between those requirements shouldn't be seen as a constraint by itself. Conversely, it only benefits the end user who can stick to the comfortable volume level without changing loudness while switching between different applications.

But most media products under the Franchise were mixed to broadcasting standards. Mixing to mobile standards while maintaining the core aesthetics and feel of the Franchise seemed like a moderately challenging task. In the very beginning of development we designed an extensive audio bus structure, grouping different sounds into a complex setup of sub-buses:

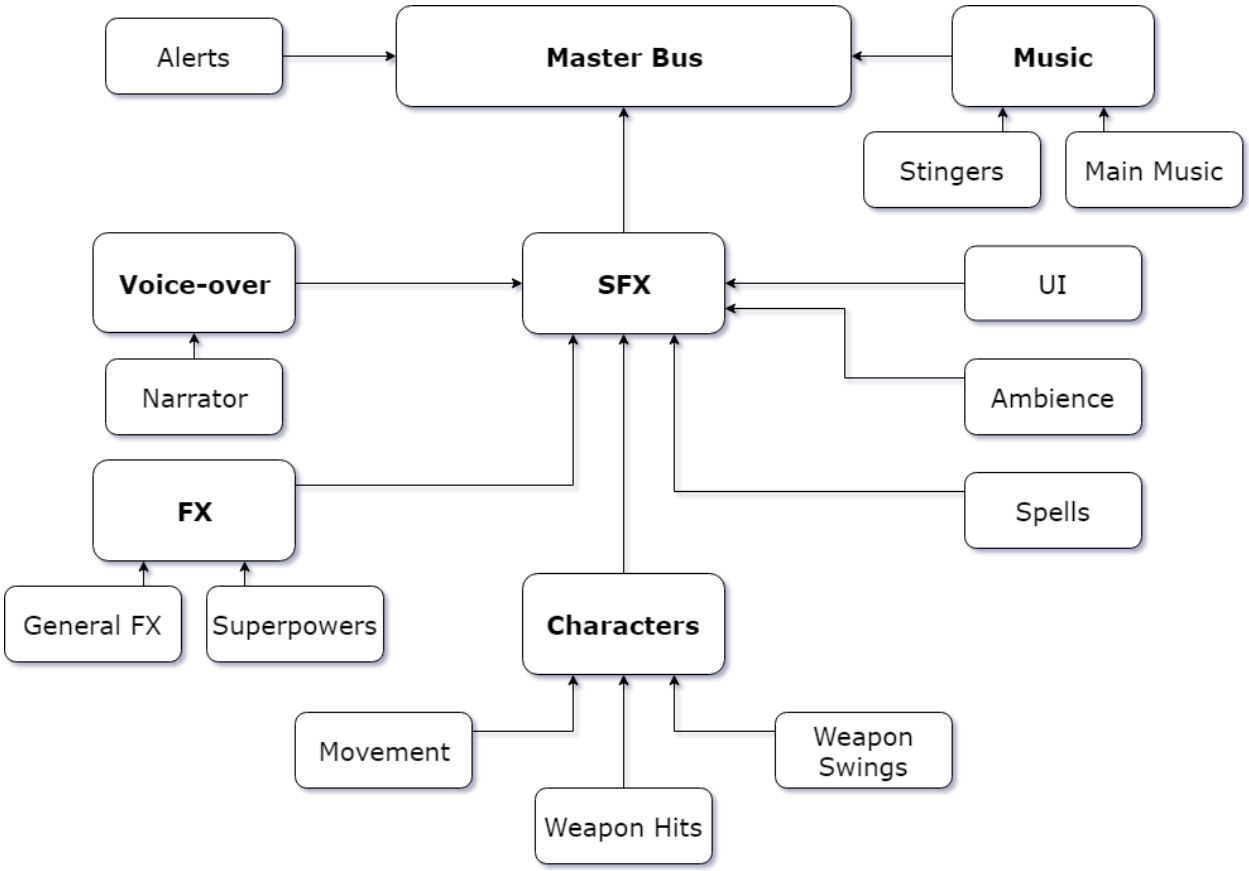

Figure 3. Mixing bus structure

Combined with a set of target loudness values for different categories of sounds, this structure allowed us to optimize mixing process, focusing on working with buses instead of individual sound effects.

After the first mix was done, loudness meter showed us the following data: Integrated loudness -18.4 LUFS, TP -1.5 dB, LRA -15.4 LU. Integrated loudness and true

peak level were just fine. LRA (loudness range average) was acceptable for listening through headphones but too high for a portable device speaker. However, if we decided to fully optimize our mixes for mobile device speakers, we would upset a sizeable portion of players who play with headphones on. And regarding how much we cared about correct audio representation in stereo field, we didn't want to disappoint the only users who play our game with a proper stereo setup. We addressed this problem by creating separate mixes for two different listening modes. Mobile speaker mix would stay at the same loudness level to match ASWG recommendations, but would have smaller average loudness range, closer to 9 LU.

We asked the programmer to implement a system that would detect whether headphones or some other kind of external speaker is connected to the device. Then we set a trigger that would switch states in Wwise according to received data. We used our existing "headphone" mix as a starting point because we were quite happy with the result. For the mobile mix we wanted to reduce loudness range while keeping the same integrated loudness level and not exceeding true peak level of -1 dB.

But first we had to find a straightforward way to monitor the result on a mobile device while mixing. Wwise can connect to the build that runs on a mobile device and give user full control over the mix, but we were unable use this function because of some network-related some technical constraints. Additionally, it didn't allow us to switch devices on the go to quickly check our mixing decisions on different speakers. We didn't want one specific device's acoustic characteristics to influence our mix, so we looked for a solution that would allow us to quickly switch between multiple devices while streaming audio from Unity editor on a PC

The solution was found in a free software called Stream What You Hear (Streamwhatyouhear.com, 2018) that allows to stream audio via HTTP protocol in local network. The best part of this method is that audio can be quickly accessed from any device with a media player capable of playing stream from the web (i.e. internet-radio). At the same time there are some downsides, like audio compression and almost 5 seconds of delay introduced by the network. Compression artifacts were considered neglectable regarding the quality of mobile device speaker playback. Delay made the work process less comfortable, but was not critical for the mixing purpose.

To adapt our mix for the mobile device speaker we applied a high-pass filter that cuts everything below 200 Hz on the master bus. Tiny speakers can't reproduce such low frequencies, but they still can affect audio playback when they are present (Kleiner, 2013, p. 507) in the mix, so we did it to avoid possible distortion. Music sounded a bit too quiet when played from the mobile device speaker, so we boosted corresponding music bus by 3 dB and compressed SFX submaster bus to compensate disbalance. After the compression we've noticed audible resonance around 3 kHz on our target device and attenuated this band by 3db using parametric EQ.

Resulting mix had integrated loudness -16.3 LUFS, TP -1 dB, LRA -13 LU. Integrated loudness was still within the acceptable range, but LRA decrease was not enough. We decided to take a next step in further iterations, following Audiokinetic's guidelines for decreasing loudness range, using compression on master bus. They suggest to apply low-level compression with a threshold under -40 dbFS, compression ratio of 1:1.2 – 1:1.5 and release time of over 1 second. (Audiokinetic.com, 2018b).

Conclusion

While working on this thesis I had to accept lots of exceptions and generalizations. The biggest of those is my understanding of mobile mid-core itself that I had to narrow down based on my observations of games, popular in 2017 and beginning of 2018. But the actual concept goes far beyond that. For example, Warman and Fowler (2012) point out, that Grand Theft Auto III (2001) is seen as a hardcore game when played on a console, but turns to mid-core when played on a tablet. This example, though, is too far from the existing market trend of service-based mobile games, so I consider it an irrelevant exception.

No surprise, but mobile mid-core offers the same set of audio-design challenges as mobile games in general. Hardware limitations and portability itself have such a dramatic impact on audio design process, that other aspects fade in comparison. But my findings and experience with the Game indicate these aspects still exist. In chapter 2 I was able to find some common grounds between successful mobile mid-core games. From these findings I derived a set of ideas about characteristics of mobile mid-core game audio design. Some of those ideas were equally inspired by the project I've been working on, others themselves helped to shape the Game and define or improve its audio design principles. These two processes were somehow interconnected, and it is hard to say if certain idea was inspired by existing problem within the Game or creative vision of the Game was somehow altered by my research.

The first and the most obvious one is increased complexity, which is more of a common case, but not an outcome of a game being designed for the mid-core segment. Some part of it comes from the players' expectation, other originates in the Games as a

Service model. Service-based games aim for long playtime, so require repetition control to be less irritative to the player on long-term perspective.

The second is special attention towards UI sound design. Again, this is not a mid-core specific requirement, but an outcome of extensive metagame most of modern mid-core games offer to the players. The same approach would be equally relevant for any hardcore or even casual game where players need to spend a lot of time interacting with different menus.

The third idea focuses on informative component in mobile mid-core game audio. While being limited by a requirement that games should be fully playable with sounds off, this aspect somewhat expands role and function of audio in mobile games. For example, we can say, that in casual games audio has purely aesthetic function and mid-core adds informative layer to the role of audio design. Hardcore segment probably extends that list with narrative and other functions, but this statement goes far beyond the scope of my thesis, leaving space for future research.

Working on the Game was a great educational experience where rather strict rules and guidelines were combined with a lot of creative freedom and space for experiments. I feel proud of many audio features we've made, but here I would mention two of them that I find unconventional for a mobile game. The first one is our approach to UI audio. Combining dual-part interface sounds with 3d audio sources we made UI elements sound natural and responsive. The second is a dynamic mixing system that adapts to the player's listening mode. Despite the system itself would benefit from better execution in terms of loudness range for mobile speaker mix, I still consider it an achievement. Since the Game is launched now, we might be able to improve this system in future updates.

Both these features will take a good place in my personal arsenal of professional methods and tricks.

On the other side, certain aspects of the Game could turn out better if the team had more time for pre-production. For example, we didn't think about physical channel limitation and CPU before we started working on sound effects and used too many layers for some of them. As a result, we had to spend time optimizing our audio assets by baking different layers together to reduce physical voice count. Still, it was a good learning experience that I will carry over to future projects.

Since the Game was released we haven't heard any negative feedback from the community about its audio side. Thinking how critical and demanding fans of big media franchises can be, we consider it a positive result. At the same time, we can't estimate how positive it is since we don’t have any data on how many players mute the sound entirely.

Overall, I hope this study provides a slightly different angle on how we view audio in mobile games and video games in general. Other than that, I believe that my findings would be useful to audio professionals working on similar games, or looking for inspiration for their ongoing or upcoming projects.

References

ASWG-R001. (2013). 1st ed. [PDF] Sony Audio Standards Working Group. Available at: http://gameaudiopodcast.com/ASWG-R001.pdf

Audiokinetic.com. (2018). Creating Audio Conversion ShareSets. [online] Available at: https://www.audiokinetic.com/library/edge/?source=Help&id=creating_audio_conversion_settings_sharesets [Accessed 4 Feb. 2018].

Audiokinetic.com. (2018). More on Loudness Range (LRA). [online] Available at: https://www.audiokinetic.com/en/library/2017.1.3_6377/?source=Help&id=more_on_loudness_range_lra [Accessed 2 Feb. 2018].

Audiokinetic.com. (2018). Working with 2D Objects. [online] Available at: https://www.audiokinetic.com/library/2017.1.1_6340/?source=Help&id=working_with_2d_sound_music_and_motion_fx_objects [Accessed 24 Feb. 2018].

Chang, A. (2016). Why People Turn Off Audio - Designing Audio that Accomodates Rather Than Dominates. [Blog] A Day in the Life of a Sound Designer. Available at: https://sparklystarburst.wordpress.com/tag/respectful-sound-design/.

Clark, O. (2014). Games as a service. Burlington: Focal Press.

Clash of Clans. (2012). Supercell. Video game.

Clash Royale. (2016). Supercell. Video game.

Collins, K. (2008). Game sound: An Introduction to the History, Theory, and Practice of Video Game Music and Sound Design. Cambridge, MA: MIT Press.

Das-Gupta, A., Katkoff, M. and Payton, A. (2018). Prediction for 2018: Mid-Core Games. [online] Deconstructor of Fun. Available at: https://www.deconstructoroffun.com/blog/2018/1/31/prediction-for-2018-mid-core-games.

Daw, H. (2017). Henry Daw on The Small Sounds That Make A Big Difference at TNW Conference 2017. [video] Available at: https://www.youtube.com/watch?v=XOMlQzyWk28

Developer.apple.com. (2017). Higher Limit for Over-the-Air Downloads – News – Apple Developer. [online] Available at: https://developer.apple.com/news/?id=09192017b [Accessed 25 Feb. 2018].

Freeman, W. (2018). Mobile tones: The audio tool and service providers pushing the sound of smartphone gaming. [online] pocketgamer.biz. Available at: http://www.pocketgamer.biz/comment-and-opinion/67504/state-of-play-mobile-audio/.

Graft, K. (2013). What the hell does 'mid-core' mean anyway?. [online] Gamasutra.com. Available at: https://www.gamasutra.com/view/news/183697/What_the_hell_does_mid-core_mean_anyway.php.

Grand Theft Auto III. (2001). Rockstar Games. Video game.

Hearthstone: Heroes of Warcraft. (2014). Blizzard Entertainment. Video game.

Heubel, B. (2015). Embracing the Mute Switch. [Blog] Immersion Corporation Blog. Available at: https://www.immersion.com/embracing-the-mute-switch/.

Horowitz, S. and Looney, S. (2014). The essential guide to game audio. Focal Press.

Ihalainen, K. (2015). Generative sound design: Complexity, realness, and quality including study cases of an interactive 3D environment sound research and a generative sound installation. Master's Thesis. Aalto University.

Imgur.com (2016). iPhone 7 Plus audio measurements. [online] Available at: https://imgur.com/gallery/DRbu5 [Accessed 13 Mar. 2018].

InMobi. (2014). Mobile Gaming Cross-Market Analysis. First edition, 2014 Q1. [PDF] InMobi, p.18. Available at: https://www.inmobi.com/ui/pdfs/Mobile_Gaming_Cross-Market_Analysis_(First_Edition).pdf.

Jacobsen, B. (2016). Informant Diegesis in Videogames. Master's Thesis. Aarhus University.

Jean, M. (2017). How to get a hold on your voices - Optimizing for CPU (PART 1). [Blog] Audiokinetic Blog. Available at: https://blog.audiokinetic.com/how-to-get-a-hold-on-your-voices-optimizing-for-cpu-part-1/.

Jørgensen, K. (2008). Left in the dark: playing computer games with the sound turned off. In: K. Collins, ed., From Pac-Man to Pop Music. Farnham: Ashgate, pp.163–176.

Kassabian, A. (2015). Sound and Immersion in Timekiller Games. Journal of Sonic Studies, [online] (10). Available at: http://sonicstudies.org/jss10.

Katkoff, M. (2014). Mid-Core Success Part 1: Core Loops – GameAnalytics. [online] GameAnalytics. Available at: https://gameanalytics.com/blog/mid-core-success-part-1-core-loops.html.

Katkoff, M. (2014). Mid-Core Success Part 3: Social – GameAnalytics. [online] GameAnalytics. Available at: https://gameanalytics.com/blog/mid-core-success-part-3-social.html.

Kleiner, M. (2013). Electroacoustics. Boca Raton, FL: CRC Press.

Lien, T. (2013). Core gamers, mobile games and the origins of the mid-core audience. [online] Polygon. Available at: https://www.polygon.com/2013/8/9/4604088/the-rise-of-mid-core-gaming.

Lofgren, K. (2017). Finding the Beat: Who's Listening to Mobile Gaming Soundtracks?. [Blog] Big Fish Blog. Available at: https://www.bigfishgames.com/blog/finding-the-beat-whos-listening-to-mobile-gaming-soundtracks/.

Long, B. (2012). The Insider's Guide to Music and Sound for Mobile Games. [ebook] Amazon Digital Services, Inc. Available at: https://www.gameaudio101.com/Freshh/GameAudio.pdf.

Lost in Harmony. (2016). Digixart Entertainment. Video game.

Luban, P. (2011). The Design of Free-To-Play Games: Part 1. [online] Gamasutra.com. Available at: https://www.gamasutra.com/view/feature/134920/the_design_of_freetoplay_games_.php.

Marvel: Contest of Champions. (2014). Kabam. Video game.

Mason, M. (2013). Demographic Breakdown of Mobile Gamers | Magmic. [online] Magmic. Available at: http://developers.magmic.com/demographic-breakdown-casual-mid-core-hard-core-mobile-gamers/.

Ng, P. and Nesbitt, K. (2013). Informative sound design in video games. Proceedings of The 9th Australasian Conference on Interactive Entertainment Matters of Life and Death – IE '13.

Noonan, P. (2013). SEVEN REASONS AUDIO WILL SAVE MOBILE USER INTERFACE DESIGN. [PDF] Akendi. Available at: https://www.akendi.ca/downloads/whitepapers/7-Reasons-Audio-Will-Save-Mobile-UX-&-User-Interface-Design.pdf.

Papa Sangre. (2010). Somethin' Else. Video game.

Pons, M. (2014). iPhone and iPad Speakers Frequency Response. [online] The Sound Design Process. Available at: https://thesounddesignprocess.com/2014/01/08/iphone-and-ipad-speakers-frequency-response.

R 128. (2014). 1st ed. [PDF] Geneva: EBU. Available at: https://tech.ebu.ch/docs/r/r128.pdf.

Stevens, R. and Raybould, D. (2011). The game audio tutorial. Burlington, MA: Focal Press.

Roseboom, I. (2016). How first session length impacts game performance - deltadna.com. [online] deltadna.com. Available at: https://deltadna.com/blog/first-session-length-impacts-game-performance/.

Star Wars: Galaxy of Heroes. (2015). Capital Games. Video game.

Streamwhatyouhear.com. (2018). [online] Available at: http://www.streamwhatyouhear.com.

Thomas, N. (2013). Casual, Social, Mobile, and the Audio That Makes Them Successful. [video] Available at: https://www.gdcvault.com/play/1017705/Casual-Social-Mobile-and-the.

Scolastici, C. and Nolte, D. (2013). Mobile Game Design Essentials. Birmingham: Packt Publishing.

Warman, P. and Fowler, S. (2012). Publisher 2.0 — The Emergence Of Mid-Core. [online] AListDaily. Available at: http://www.alistdaily.com/media/publisher-2-0-the-emergence-of-mid-core/.