Masters' Degree in Informatics Engineering

Dissertation

Final report

Improving game accessibility by exploring the audio-only game genre

David Gil Domingues Barbeiro

davidgil@student.dei.uc.pt

Adviser:

Licínio Roque

January 29, 2016

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Abstract

In recent years, video game developers have started to be more aware of game accessibility, and consequently, audio-only games are getting more popular as a genre that enables both blind and sighted audiences to appreciate immersive gameplay experiences in similar ways. In this dissertation, we propose two audio-only game prototypes that aim to explore different forms of participation from the player. Both games employ innovative Human-Computer Interaction and sonification techniques that stem from research on those areas to improve the usability between the player and the artifact. One game is used to assess the player's hearing acuity, which is useful to validate all the data collected during formal playtesting. The other game is used to fulfill two research objectives: the first objective is to collect data that is used to understand how the number of sound sources around a player, and their position relative to each other, influence the ability of the player to recognize them, which is important to level design. The second objective is to test if the proposed techniques ease the navigation mechanic in an audio-only First-Person Shooter, which could add value to future audio-only games. We used preliminary playtesting to get early feedback from two participants, which allowed to detect issues that were promptly fixed. Then, formal playtesting was done using different groups: sighted, blind, males and females. We used traditional and technical playtesting methodologies: different test scenarios to collect data to be analyzed, direct observations, Q&A and verbal reports. The results show that while blind people have much more hearing accuracy, they don't navigate better than sighted participants. It also indicated that males have a better sense of orientation and memorize sound references more easily.

Keywords

Audio-only games, Game accessibility, Human-Computer Interaction, Game development

Acknowledgements

I would like to thank my advisor, Prof. Licínio Roque, for giving me the opportunity to work in a project I am passionate about. I would not succeed without his guidance, words of wisdom and support. He is responsible for teaching me the importance of critical thinking when doing science research and of keeping an open-minded perspective about every hypothesis, and for that, I must also thank him.

My words of gratefulness also go to my lab colleagues, for all the encouragement words and for providing me an interesting work environment that made me feel welcome every day. In particular, I must thank Rui Craveirinha for all the enlightening discussions about the videogame medium and for his fun company.

I must also express my gratitude to everyone who playtested the games or contributed with input or feedback to the development of this dissertation.

My gratitude also goes to my friends, especially the close ones – they know who they are. They've always shared their laughs with me, along with their enthusiasm, which kept motivated throughout the whole project.

I also thank my amazing girlfriend Inês Balula for all the creative input and fun moments she shared with me. This report would not be the same without her support, words of motivation, and of course, last minute revisions.

Finally I have to thank with all my heart my brothers and parents. Without their love and support, I would not have been able to come this far.

Table of Contents

List of Figures

List of Tables

Glossary

PC – Player's Character

HCI – Human-Computer Interaction

GPS - Global Positioning System

FPS - First Person Shooter

TTS - Text-To-Speech

IDE – Integrated Development Environment

HRTF – Head Related Transfer Function

PLEX - Playful Experiences

MDA - Mechanics, Dynamics and Aesthetics

R-A-E-D - Relaxation Anticipation Engagement Decay

AI – Artificial Intelligence

BPM –Beats per Minute

HP –Health Points

API – Application Programming Interface

HLAPI - High level scripting API

HRTF - Head-Related Transfer Function

SDK – Software Development Kit

VR – Virtual Reality

GUI – Graphical User Interface

Q&A – Questions and Answers

1 Introduction

The advent of mobile devices such as smartphones and tablets took the gaming industry by storm. New forms of interaction appeared, as technologies like multi-touch interfaces, accelerometers, GPS and gyroscopes became more and more inexpensive. Channels of distribution such as the App Store, Google Play or Windows Store made it possible to anyone to create and share their Apps and Games with the world. These circumstances, along with the ever growing number of tools at the developers' disposal, created a whole new market full of opportunities for the mobile gaming industry, thus making it one of the more profitable entertainment industries and effectively overtaking the console game revenues by 2015 [1].

Despite all the potential enabled by mobile technology, audio-only games continue to be neglected as a game genre of its own, as we can see by the lack of offer in marketplaces. Indeed, since audio-only games don't require the player to be in front of a screen, they allow him freedom of movement. Therefore, the mobile platform is the natural solution to the requirements of most audio-only games, making it the platform which has the most potential to further push the boundaries of this game genre. We intend to do just that by researching the problems inherent to audio-only games and tackle them in a creative way, allowing us to produce experiences that are engaging to both the sighted and non-sighted community of gamers.

1.1 Motivation

It is undeniable that since the creation of Pong in 1972, video games have been consistently evolving to an intricate and powerful form of entertainment that can change the way we perceive the world. Nowadays, games are being used as more than just a pastime and different applications are being discovered at a fast pace, pushed by research on Gamification. In fact, we can already see examples of applications in education [2], training [3] and even science research

[4], to name a few. However, video games have also been in the media spotlight for bad reasons, such as the controversies about how the portrayal of sex and violence in videogames might affect players' behavior in real life [5].

We feel that games can have a positive effect in everyone's life and no one should be excluded from that. Game developers should take accessibility in consideration, designing games in a way that maximizes the audiences that can play them. Our major motivation for this thesis was to create games that can be experienced the same way by blind and sighted people, while being fun and engaging. To do that, we researched on Sonification and other techniques from Human-Computer Interaction (HCI), creating a basis of knowledge that ended up being useful in our own game design.

1.2 Objectives

This dissertation's main goal was to produce audio-only game prototypes that explored different forms of participation by the player, while giving emphasis to the acoustic aspects that are so often neglected in videogames. To that end, we idealized one First-Person Shooter that uses innovative HCI and sonification techniques to ease navigation, which is a challenging mechanic in the audio-only genre, while keeping the experience immersive.

Furthermore, we wanted to investigate how the number and angle between sound sources around the player affect the time and accuracy to detect each one. To that end, we did formal playtesting with different user profiles in different circumstances. Finally, we developed a game that is used to gauge the player's hearing acuity which was useful to put in context subsequent playtest results.

1.3 Methodology

One particularity of game development is that there is no way of knowing if the end result will be a success because there are different definitions of success – for instance, if the game is profitable we can call it a commercial success. However, in our opinion, a successful game must, in some way, improve the life of the player. Since everyone has different upbringings that affect the way events are interpreted, both opinions and experiences, in general, tend to be subjective. This has more implications in the game industry than in the software industry because, as Pippin Barr argues, "*People use software, but they play video games"* [6]. What this means is that software does not lend itself to interpretation like video games do, which means that video games have a much higher uncertainty of success than other forms of software because we can't know *a priori* if the mechanics, dynamics and aesthetics will be pleasing to the players. Consequently, software and video game development methodologies are divergent in many aspects.

Even though this dissertation does not follow the Scrum methodology, some of its concepts seemed appropriate to guide us, as we dealt with audio-only games in an exploratory manner, which inevitably lead to high risks that had to be addressed in a expedite way. Thus, we planned sprints with specific milestones in order to gauge our progress throughout this dissertation.

2 State of the Art

It is important to explore the historical background of audio-only games if we want to understand how this game genre became what it is today. This research will allow us to learn the motivations, technologies and works that have been pushing the boundaries of audio games, thus creating a basis of knowledge from which we can build upon. In this chapter, we will also look into different Game Design Models and Frameworks that might help us create a comprehensive analysis of several games, ultimately allowing us to find patterns that will, hopefully, give us some insight that will be useful in the creation process of our own games. Finally, we will explain how HCI can be used to improve the usability of games.

2.1 Audio and technology in the electronic game industry

Audio games are a subset of electronic games that are designed with the purpose of creating an experience that gives emphasis to the sound. More specifically, audio-only games use sonic artefacts to create gameplay mechanics, dynamics and aesthetics, differing from video games due to the total absence of visual feedback. Since all the information is conveyed to the player through his audition, this means that both the sighted and the visually impaired may experience the game in a similar way. For this reason, audio-only games have been mostly a genre made specifically as a way to make games accessible to the blinded. This game genre represents, so far, a small niche in the whole gaming spectrum, which has been a barrier to developers, as smaller audiences tend to increase the risk of commercial failure. In fact, most audio-only games available were developed by blind people and hobbyists [8].

The first audio game success happened in 1973 with the release of *Touch It*, a handheld device that featured four buttons, each with a corresponding tone. The device would play a sequence of tones that the player had to reproduce,

challenging his eidetic memory. After the successful reproduction of a sequence, a tone would be added to increase the difficulty.

Then, the computer era finally arrived and with it came text-based operative systems like MS-DOS that made the text-based games genre flourish. However, it wouldn't be until the proliferation of text-to-speech (TTS) software like MacInTalk in 1984 that those games would become accessible to blind gamers. Meanwhile, video games were becoming the dominant form of electronic entertainment and the improvements on graphical technologies and techniques were contributing to increase their popularity. Eventually, most video game developers stopped paying attention to game accessibility altogether, preferring to invest their efforts in more flashy features that would make their games look sexy. This conjecture did not change for many years but luckily, the past decade has seen a booming in independent video games and with them, came the creativity that is only possible when there are no external driving forces influencing game design. Since then, we have seen a handful of audio-only games that ended up being a commercial success and game designers are, once again, starting to be more aware of game accessibility, which is currently the most powerful driving force behind research on sonification [9].

Technology wise, recent years have been great for game developers. Long past is the time when game engines had to be built from scratch, as nowadays, everyone has access to them. Game engines like Unity 5 and Unreal Engine 4 have empowered game developers with tools that allow them to spend more time thinking about game design issues and less with implementation. The reason for this is that most of them are powerful IDE's that include physics, animations, sound and graphical engines, supported by large documentation and very helpful communities.

Finally, binaural sounds that make use of HRTF function can now be easily integrated with games through the use of open-source middleware which ultimately will improve the gameplay experience of both audio and video games.

2.2 Game Design Models

Game design models are extremely valuable to evaluate ideas in a methodic, rational and generally structured way, empowering the game designer with tools that help him iterate game proposals. Much like in other areas, no one has ever created a Grand Unified Model that is able to represent games in a fully comprehensive way. However, many different game design models have been created, allowing different perspectives and degrees of formality. For instance, The Playful Experiences (PLEX) Framework has shown good results when used by expert evaluators as a checklist [10]:

> *"We found that the PLEX framework provided experts a systematic and structured way to focus attention, in this case, a particular way to look at playfulness. (…) The experts reported that the PLEX framework provided anchor points for them to reflect and discuss different aspects of playfulness as they conducted their heuristic evaluations"*

The authors argue that the simplicity of the Framework is its own key strength and weakness, as it allows for some freedom of interpretation. However, its lack of additional structure can also limit the experts' analysis and make them miss some key observations. On the other side of the spectrum, we can find studies that describe games in a very formal and precise way. For instance, a study was done by Stefan M. Grünvogel, where a formalism for describing models of games is introduced by the use of abstract control systems [11].

Instead of using the aforementioned models, we chose the Participation-Centric Model of Gameplay Experience and the MDA – Mechanics, Dynamics and Aesthetics – framework because we believe that they give us two lenses that complement each other and cover all the relevant aspects of game design, allowing us to describe and discuss them in a broad but structured fashion.

2.2.1 Participation-Centric Model of Gameplay Experience

The motivation for this model arose from the necessity to analyze video games in a comprehensive way, regardless of videogame genre and context of use. The authors propose a model for *"supporting the design and analysis of videogames in order to achieve a rationalization between how the designer intended for the game object to promote a specific playing experience and the emerging experience as interpreted by players"* [12].

This model is built upon the premise that the videogame medium has an inherent participatory nature, which is corroborated by many authors (Aarseth 1997; Bogost 2007; Raessens and Goldstein 2005; Roque 2005; Salen and Zimmerman 2004). The player's participation in the game is regulated by the elements he is able to interact with, and, from this contexts, meaning emerges. Therefore, the game designer must think and allow the right forms of participation that will trigger the game experience he idealized.

Furthermore, the authors suggest to think of a game as a network of mediators of diverse nature. The reason for this interpretation comes from the necessity to synthesize and understand how the different forms of participation, through all the elements that compose the contexts, promote the gameplay activity.

This study results in a model that considers six forms of participation: *Playfulness, Challenge, Embodiment, Sensemaking, Sensoriality* and *Sociability*. This set of dimensions allows the designer to characterize and compare the intended gameplay activities with the actual way the player participates in them. Additionally, each form of participation is broken into three operative foci:

Intention: What is the participation ideal that the videogame is suggesting?

Artifact: How does the artifact support the idealized forms of participation?

Participation: What characteristics of the actual player's participation are consistent with or revealing of the participation idealized?

This model will be useful to analyze games through a player's perspective, using an empirical approach that results from playtest sessions and its summary can be seen in Table 1.

	Intention	Artifact	Participation
Playfulness	Exploring, discovering, recreating, customizing	The nature of a player's agency, the variety of interactive elements of the game (objects, characters, actions, etc.)	Degree, variety and tendency of exploration
Challenge	Overcoming a challenge, creating a strategy, defeating an opponent, mastering a skill	Nature of challenges proposed, type of penalties and rewards, intensity and organization of challenges	Control, pace, progress, efficiency in performing tasks
Embodiment	Physical involvement, physical performance	Representation of the physical game world, player's representation on the game world, interpretation of players movement	Control and rhythm of movement, aesthetics of the movement
Sensemaking	Interpretation of a role, fantasy, self-expression	Theme and underlying narratives, models and representations of phenomena, roles and motives, significant actions	Alignment between actions and roles, understanding and or critique of the represented phenomenon
Sensoriality	Contemplation, wonder	Style, nature of the stimuli, visual and sonic compositions, synesthetic explorations	Degree of exposure and responsiveness to stimuli, interaction or engagement with sources
Sociability	Competition, cooperation, friendship, identification, recognition	Diversity and nature of social interactions and relationships, models of social structures (team, hierarchy, etc.)	The intensity and types of interactions between players, effectiveness bonds

Table 1: Characterizing players' participation along the three foci of analysis

2.2.2 MDA - Mechanics, Dynamics, and Aesthetics – framework

The multidisciplinary nature of Video Game Development presents some cumbersome difficulties. Usually teams are divided in smaller groups that focus on specific areas, but, sooner or later, all the parts must be integrated, which means that teams must coordinate themselves and, consequently, learn about areas they are not familiar with. Therefore, even the teams that are in charge of implementing more technical features such as Physics Engine, the Rendering Engine or Artificial Intelligence, must adhere to the overarching design goals of the game. Inversely, the teams that are responsible for the Aesthetic side, which includes Level Design, Sound Design, Game Testing, etc, all areas that deal more directly with the end user experience, must be aware of the technical limitations and idealize the basic game mechanics with them in mind. As the authors refer, *"systematic coherence comes when conflicting constraints are satisfied, and each of the game's parts can relate to each other as a whole. Decomposing, understanding and creating this coherence requires travel between all levels of abstraction - fluent motion from systems and code, to content and play experience, and back"* [13].

Consequently, the MDA – Mechanics, Dynamics and Aesthetics – framework is proposed as a tool that promotes iterative, qualitative and quantitative analyses through two different perspectives. The first looks at the player's experience to see how the basic mechanics must be changed in order to get to the best end result. The second looks at the implementation, which ultimately is translated as source code, and sees if the emergent system behaviors are coherent with the desired Aesthetics.

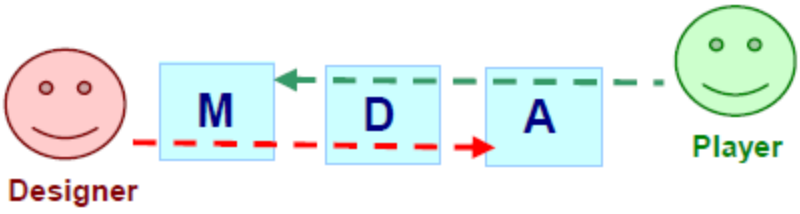

Figure 1-The two perspectives of the MDA Framework

The MDA breaks down this two perspectives into three distinct components:

Mechanics: Describes the particular components of the game, at the level of data representation and algorithms. They define the various actions, behaviors and control mechanisms afforded to the player within a game context. As an example, a First-Person Shooter has mechanics like ammunitions, health and spawn points.

Dynamics: Describes the run-time behavior of the mechanics acting on the player inputs and each other's outputs over time. Dynamics emerge from the different mechanics and ultimately translate into aesthetics. For instance, in a First-Person Shooter that has fixed, unprotected spawn points, dynamics like camping and sniping are bound to happen.

Aesthetics: Describes the desirable emotional responses evoked in the player, when he interacts with the game system. Mark LeBlanc defines eight different kinds of fun [14]:

1. **Sensation:** Game as sense-pleasure
2. **Fellowship:** Game as social framework
3. **Fantasy:** Game as make-believe
4. **Discovery:** Game as uncharted territory
5. **Narrative:** Game as unfolding story
6. **Expression:** Game as soap box
7. **Challenge:** Game as obstacle course
8. **Submission:** Game as mindless pastime

A First-Person Shooter such as Quake evokes Challenge as its main aesthetic goal. However, dynamics such as camping and sniping might degrade this goal and ultimately, the player's gameplay experience.

This framework provided itself useful by giving us semantics that allow us to better explain how and why the player's participation, that emerge from the artifact, are engaging and support the game's intended aesthetics.

2.3 Game Analysis

In the early stages of this dissertation we decided that it was important to research on previously developed audio-only games (prioritizing commercially successful ones), in order to get a better grasp on the current state of the genre. The result of this study gave us insight on the technologies and game design patterns used, enabling us to make more informed decisions throughout the development of our own game prototypes.

Each game analysis was produced using the two different aforementioned game design and research frameworks: MDA-Mechanics, Dynamics, Aesthetics, and the Participation-Centric Model of Gameplay Experience.

Using this two lenses allowed us to create a systematic and comprehensive analysis, providing us a way to fairly discuss game experiences and ultimately draw some conclusions that were useful to support our own implementation decisions. We also used Sound Design Patterns [15] [16] to understand how sonification is used to convey information to the player.

We chose to analyze more thoroughly the next four games because we think that they showed relevant and diverse mechanics, dynamics and aesthetics while promoting interesting forms of participation.

2.3.1 Papa Sangre

This audio-only horror adventure game is a prime example of how a uniquely imaginative world and narrative can be conveyed to the player exclusively through sound. In this game the player is a dead being, trapped in the fearful Papa's Sangre palace. This palace is full of dangerous monsters which the player will have to avoid in his mission to save his love and leave the Underworld.

This game requires the players to wear headphones. The screen has a compass that relies on the smartphone's gyroscope and defines which direction he is facing in the virtual world. To move forward, the player has 2 buttons on the screen's bottom that represent the left and right foot, which he will have to

tap alternately to move forward. If he taps them out of sync, the character will trip and fall, which can mean death in certain situations. Binaural audio is used to give the player clues of the direction and distance of the musical notes he has to collect.

Table 2 summarizes the most important sound patterns we identified during gameplay.

Sound Pattern	Description/Usage
Narrator Helper Voice	A "fluttery watery thing" accompanies the player throughout the game, giving him instructions and context when necessary
Aesthetics	In the beginning of the game, the helper voice sets the aesthetics by telling the player that "in this Underworld it is pitch dark. You cannot see a thing (…), but you can listen and move (…)". There is coherence between the game context and the actual player participation because the game has no visuals and the player might play blindfolded.
Directionality Beacon Locator	The main gameplay mechanic is following specific sounds in order to progress the narrative. Binaural audio is used to enable the player to better sense the direction of the sounds.
R-A-E-D	R-A-E-D is an iterated sequence that is extensively used in Papa Sangre. It starts in Relaxation, as the PC's wanders through the realm of the Underworld, followed by the Anticipation that is caused by a threat, which leads to Engagement and lastly, it ends in Decay, as the threat is over and the player enjoys a moment of safety.
Death	When the player steps into an enemy, he can hear a sequence of sounds that enable him to feel the embodied violence that is inherent to death inflicted by an attack.
Footsteps	The footstep sounds serve three purposes. The first is to convey information about the PC's movement. The second is to inform the player about the type of ground he's on. The third has to do with a core gameplay mechanic: if the player is heard by enemies, by tripping or walking to close from one, he dies.
Character's Soundprint Grunts Breath	Since the player is unable to visualize the PC's, it is important to communicate it's physicality through a correct use of sounds. This is achieved with Breath sound effects, that are used to define the pace (i.e. running or walking) or in extreme emotional conditions, and Grunts, that can be heard when the PC's trips and falls.
Acoustic Ecology	Since this is an audio only game, it is of the utmost importance to translate the PC's surroundings through a holistic soundscape that

Ambience	is coherent with the game design goals. The ambience pattern gives a significant contribution to achieve this.
Acoustic Ecology Sound Effects	During gameplay it's possible to hear a very diverse sonic composition made of sound effects that contribute greatly to the Acoustic Ecology

Table 2- Sound patterns in Papa Sangre

The aforementioned table serves to show how sonification was largely explored by making use of different sounds to inform the player of events. Additionally, in Table 3 we can see the forms of participation that emerged during our playtesting session.

	Intention	Artifact	Participation
Challenge	Master the ability to pinpoint sound sources; Walk or run in a controlled pace.	The core Mechanic of this game is to locate and walk to a sound source to progress the storyline. However, in certain situations, the PC's will have to run from enemies or lure them if they are blocking his path. This Dynamic proves valuable to communicate fear, which in turn is in accordance with the overall game design goal, since this is a horror adventure. Another important mechanic relates to the PC's movement. The player must control his pace differently, depending on the situation. If he is unable to do so, he might die.	The player wears headphones and rotates himself while holding a smartphone until the sound is "in front of him". The walking is done by tapping the left and right foot alternately. The speed of the tapping translates directly to the pace. Tapping faster makes the PC's run. When running it's hard to coordinate booth feet, which increases the chances of getting the PC's to trip and get killed.
Embodiment	The player embodies a soul in a realm called Underworld that is "pitch black".	The world is a continuous space described by sounds that can be moving or not.	The direction the player is facing in the physical world translates to the directions the PC's is facing in the virtual world. Additionally, the way the player taps the left and right foot simulates movement in a well thought manner.

Sensoriality	A sense of wonder and fear is induced in the player throughout the game by a rich sonic composition.	Most of the gameplay does not push the player into frantic action, allowing him to explore his surroundings at his own pace and letting him create his own interpretation of the world.	Since there are no visuals, the player's imagination is set loose to fill the gaps in the virtual world by using the information carried by sounds, allowing him an immersive experience.
Sensemaking	The player interprets the role of a soul in the first person	The PC's is trying to restore soul memories that will allow him to leave the Underworld.	The player is a soul that explores Papa Sangre's palace that is filled with memories. In his journey to save his love and leave the Underworld he faces many challenges.

Table 3-Forms of participation in Papa Sangre

This game excels on its goal as a horror game: inspire fear. Being a strongly narrative driven game, we can say confidently that it supports Sensemaking as its main form of participation. The stepping mechanic is useful not only as a method of input to make the PC go forward in different paces, but also to simulate walking/running in a believable way. The use of proprietary binaural technology also enhances the gaming experience and aids the player in the challenge of finding the direction of certain sound sources.

2.3.2 Deep Sea

Fear is an emotion that can makes us feel vulnerable in a very effective way and is usually induced when we sense a threat or when we feel that something we cherish is at risk. Some of the most primal instincts are triggered when we feel this emotion and this is exactly what Robin Arnott, the designer of Deep Sea, wanted to achieve. Deep Sea is all about survival in the most adverse of the environments. In this audio-only survival-horror game, the player wears a mask that devoids him from sight and takes over his own hearing, amplifying the sounds from his breathing and creating a sense of isolation and helplessness. The player is surrounded by alien creatures that are converging to him at a

crescent rate as he produces sound by breathing, rotating and shooting. Then, he has to use a joystick to control the direction he is facing while listening to commands given over radio which tell him exactly where the monsters are relative to him. When the player is confident that he has a creature directly in front of him, he can press a button in the joystick and try to kill it. As the number of creatures around the player increases, so does the need to suppress his own breathing in order to ear clearly the instructions given to him over the radio, further enhancing the sense of fear and claustrophobia. Table 4 summarizes the most important sound patterns we identified during gameplay.

Sound Pattern	Description/Usage
Voice Acting Radio Helper Voice Directionality	A human voice gives instructions to the player by radio, helping him to pinpoint the position of the enemies
Contextual Music	Used in the beginning of the experience to help create a setting
Trance Ambiance	Water sounds that give a sensation of being submerged
Entrainment Breath In-game Feedback	Breathing feedback
Beacon Locator Directionality Aesthetic	The sound the alien creatures produce when swimming around the PC's gives the player information about his location
Sound Effects Awareness	When the player tries to shoot a creature, he hears a sound that tells him whether he hit or missed the target
Death	The creatures attack and kill the player, ending the experience.

Table 4-Sound patterns in Deep Sea

Despite the limited number of sounds used in this game, the sound designer made sure that each one would effectively contribute to an immersive experience by making sure it was coherent with the game aesthetics. This approach shows that a good sonic composition does not necessarily come from a

large variety of sounds, but rather from their high quality. In Table 5 we can see the forms of participation that emerged during our playtesting session.

	Intention	Artifact	Participation
Challenge	Survive as long as possible by killing incoming creatures.	Every sound produced by the player attracts more creatures. Creatures produce sounds that allow the player to pinpoint their location.	The player must listen carefully to pinpoint the source of the enemies and shoot the incoming waves of creatures.
Sensemaking	The game is played in the first-person.	The player is inside an immersed compartment, completely alone, and shooting oncoming waves of enemies.	The player plays a role where he is cast in a subaquatic disaster, forcing him to survive while waiting to be rescued.
Sensoriality	Let the player feel the horror and fear that comes with unknown threats around him.	The soundscape is mysterious and frightening and the monsters produce sounds that provoke fear on the player. The player's breathing intensity controls the Breathing sounds the PCs produce. From this mechanic emerges an interesting dynamic, because the player will have to suppress his own breathing in order to better listen to the monsters.	Most times there is an anticipation that allows the player to fully appreciate the soundscape.

Table 5-Forms of participation in Deep Sea

The main problem with this game is its very limited forms of participation by the player. However, this was an experimental installation that showed how a smart use of device inputs, like the mask with microphones in the diaphragm that were used to "measure" the player's breathing, can be used to create dynamics that support the intended aesthetics.

2.3.3 BlindFold

This game throws the player into a situation where he is deprived from sight, forcing him to explore a world filled with blackness, but rich in its soundscape. The narrative starts with a traffic accident that makes the PC blind and disoriented. Then, it's up to him to explore his whereabouts while trying to learn how he got in that predicament.

To do that he uses the keyboard to move himself and interact with the objects. In Table 6 it is shown the different sound patterns used in-game.

Sound Pattern	Description of use
Narrative Cutscene	In the beginning of the game the player is presented with a cutscene that serves as a way to set the narrative, in this case, a vehicle accident.
Breathing	Used when the PC is running to give a sense of physical strain
Grunts	When the PC collides with a wall
Footsteps	Footsteps are used to inform the player about the pace (walking or running) and the floor material (grass or asphalt). The floor material is convenient to help the player create a mental map of the virtual world, thus facilitating his navigation.
Ambiance	When outside, it is possible to hear a background noise that communicates to the player the nature of the soundscape, in this case, an urban environment.
No can do	When the player tries to interact with some object that he is not supposed to, it.
Achievement	When an interaction occurs successfully, the player hears a positive sound effect.
Recordings Sound effects Narrative	There are Recordings throughout the game that the player has to find to try to unravel the narrative. Some sound effects are used as Signals to create spatial references to the player.

Table 6-Sound patterns in Blindfold

Once again, the sound designer knew how important it is to create a good sonic composition in order to give some meaning to the game's story. This was particularly important in this game, as it contains some scripted scenes that allow the player a lot of interpretation freedom. This was a deliberate decision by the game designers because it was used to test if it was possible to "*build*

understandings of user participation in the soundscape constituting the gameplay scenario" [17] .

Table 7 contains the most relevant aspects of the game that were revealed during the gameplay session.

	Intention	Artifact	Participation
Challenge	Explore the whole world and interact with its objects.	There is a specific order to visit each game object in order to fully experience the narrative. To do that, the player will have to create a mapping of the world in his mind by making use of the sonic artifacts that are placed there to help him creating references.	The player wears headphones and uses the cursor keys to move and rotate his character, plus one action key to interact with objects.
Sensoriality	Put the player's emotions to the test by providing him with a rich soundscape and ambience.	There are several interactive scenes that are meant to trigger different emotions in the player (e.g.: baby scene, dog scene).	The nature of the experience forces the player to focus on the auditory senses, enabling him to imagine the world in a way vision could not.
Sensemaking	The player interprets the role of someone who just had a traffic accident	This dimension is explored through a narrative that is not explicit, but rather gives the player clues and is open to interpretation.	The player tries to decode possibilities by discovering the interactive scenes that are scattered in the game world.

Table 7-Forms of participation in Blindfold

This gameplay experience was very useful to reveal major problems that arise from the challenge of navigation without any kind of visuals. Much like in most narrative-driven videogame, the player starts without a way to pinpoint his location in the Virtual Game Space/Level (the only exception are the videogames that provide a fully revealed map right from the beginning, which is not common). However, unlike what happens in videogames, the only way to build a mental map in an audio-only game is through sonic references. From this simple truth we conclude that the bigger the level, the more sonic references are

necessary, which leads to the difficult challenge of memorizing too many sound sources positions. This takes us to next the next problem: knowing which direction in 360º the PC is facing inside de Level, relative to the cardinal directions - North, East, South and West. Without any additional source of information, like a button that when pressed tells the player the direction the PC is facing, it is not possible to create references based on direction. However, if two or more sound sources are close enough and intercept each other, when the player hears them both, he can infer which direction he is facing relative to them (i.e.: locally, disregarding cardinal directions).

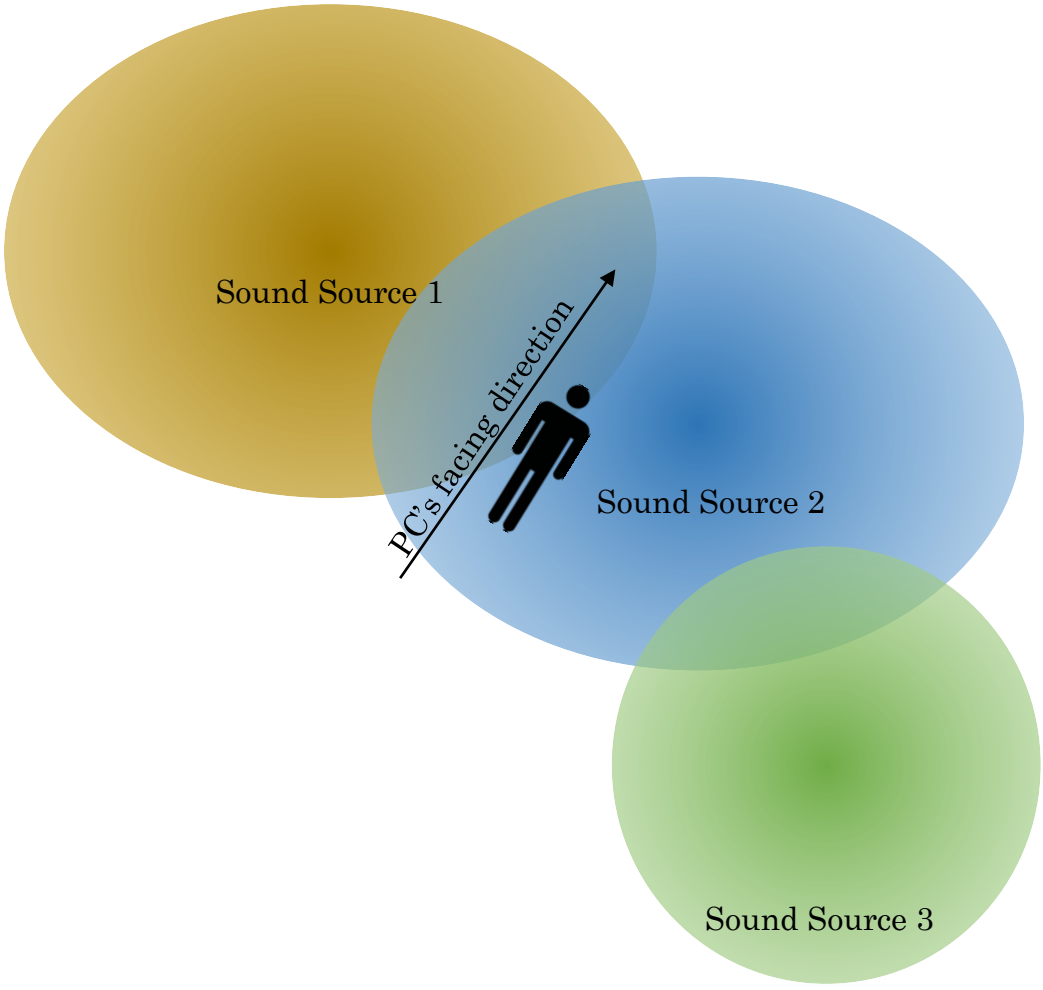

Figure 2-Player inferring his local direction by listening two Sound Sources simultaneously

This triangulation, which can be seen in Figure 2, allows the player to guide himself from one Sound Source to another as long as they keep intercepting each other. This means that sound sources must be carefully positioned during Level Design, since the player cannot be left without any sort of sonic reference during the exploratory exercise.

In BlindFold, this constraints were overlooked, resulting in a lack of sonic references in some parts of the level that made us feel lost and a bit frustrated. Figure 3 shows most sonic references in the BlindFold level, represented by icons inside circles. The icons represent a type of sound source, the diameter of the circles represent their range and the white circles belong to interactive objects, while the yellow ones serve only as references.

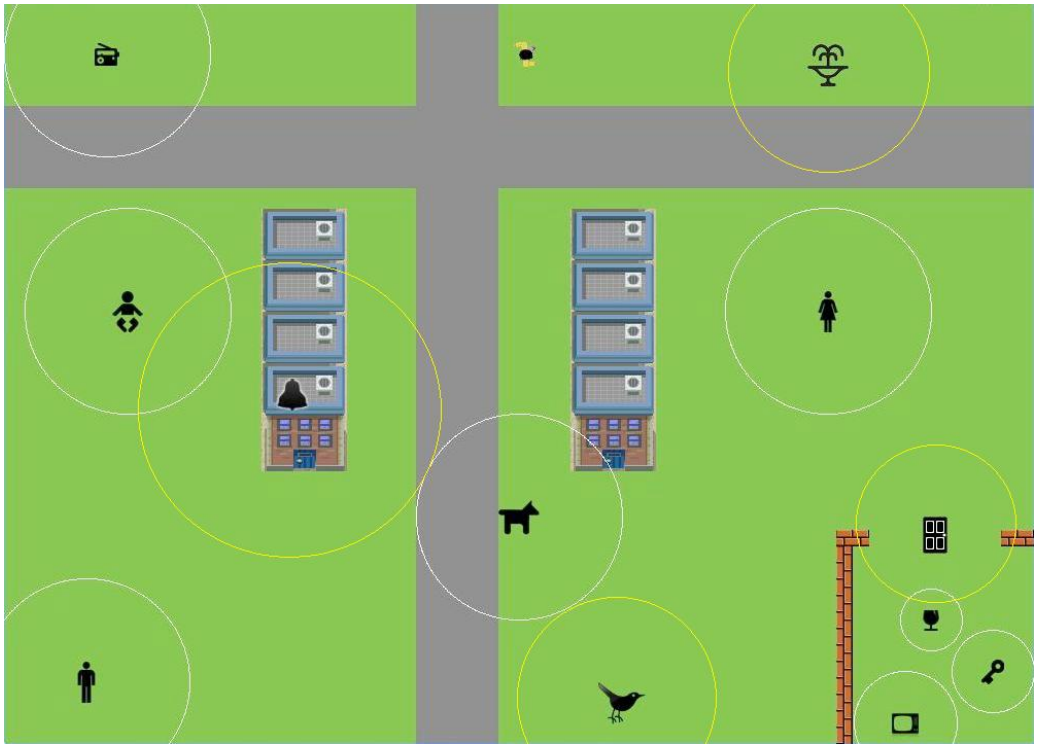

Figure 3-Map of the Level showing most sonic references composing the soundscape

As we can see, a big area of the Level does not contain sonic references, making it very hard for the players to explore it in a systematic way. The last critical barrier to the player's navigation revealed during one of the gameplay sessions

was the lack of feedback for the PC's movement. With a keyboard (or mouse) as the only device input, the player is unable to know the PC's walking/running speed, rotation velocity, or as we already mentioned, the facing direction.

The BlindFold proved itself very useful not only to expose the biggest navigation problems that are inherent with exploratory audio-only games, but also to revealed solutions that we will talk about in later chapters.

2.3.4 AudioQuake

Quake is considered by many to be one of the most influential games ever made. It popularized the first-person shooter by supporting full real-time 3D rendering and enabling players to play against each other in multiplayer arenas. AudioQuake is an Accessible Game that allows blind and vision-impaired people to play Quake through a series of modifications made to the original Engine. It applies sonification extensively to represent many different events and convey information to the player. For this reason, in the analysis of Table 8, instead of describing the sound patterns, we describe different events and information that the sound engine transmits to the player using sonification.

Information	Description
Toggle Sounds	Used to tell the player something was turned on/off
Generic Navigation Sounds	Navigation Helper is the engine module responsible for transmitting information about the PC's surroundings. Different sounds play when the PC is near walls, slopes and doors. The player can also hear a specific sound if the PC is scraping the wall, as well as wind sound, when there is plenty of space in the direction the PC is facing.
Vertical movement	Played when the PC's position is being changed vertically.
Hazard (Drop/Ledge) Warning	Different sounds are played when the PC is near a drop, depending on the height.
Jump Descriptions	Knowing there is a drop nearby, the player can press a key to obtain more information about it. If that information is available, he gets to know if he needs to make a running or walking jump to get through the gap and what kind of material lies bellow (water, lava, etc.).
Waypoint	A sound is played continually to notice the player about his location.
Compass Sounds	The player can press a key to know what direction the PC is facing.
Detector 5000	This module is responsible for detecting and informing the player about important items (e.g.: ammo, health, weapons, etc) that are in the PC's vicinities.
Weapons	The player hears different sounds when he picks/switches weapons. He is also noticed if he runs out of ammo.

EtherScan RADAR	This module informs the player of the presence of monsters, enemies and allies, as well as their vertical position related to the PC (higher level, same level or lower level).
End of level	Played when a level is completed successfully

Table 8-Sonification of AudioQuake

AudioQuake features various game modes, but since we already had analyzed several single-player games, we evaluated the Team Deathmatch Mode, in which there are two teams of players that play against each other in a server. The participation is described in Figure 9.

	Intention	Artifact	Participation
Challenge	The player has to navigate through a map and kill his opponents.	The game provides the player information to help him navigate through levels and coordinate with his team. Inside each level it is possible to find ammo, health, weapons and other resources.	The learning curve of the game is very hard, as there are many different sounds which have similar tones. Navigation is particularly difficult in labyrinthic maps. The player must search for the right resources at the appropriate times while killing opponents
Sociability	Two teams compete with each other	Players are aware of the presence of each other through the EtherScan RADAR.	Players use EtherScan RADAR to aim and shoot the enemies/monsters or to follow teammates. Vocal communication with teammates is strongly advised to improve the chances of winning.

Table 9-Forms of participation in AudioQuake

This game has the potential to provide a complex and exciting gaming experience to both blind and sighted players. It has high replayability value and puts to the test players' reflexes, hearing, communication skills and ability to create strategies and coordinate with other players and AI agents. However, despite all the potential, its game design ultimately creates an experience that is very artificial and far from immersive, as sonification is used too extensively, creating a barrage of similar sounds with very different meanings that are difficult to memorize, especially to newcomers in the audio-only genre. Not only

that, but translating an FPS video-game to an audio-only game ends up creating some navigation challenges that don't add anything of relevance to the intended experience. One good example is the presence of elevators, pitfalls/gaps, slopes, stairs, etc. With visual information, it is very easy to navigate through this kind of artifacts which actually allow more diversity in levels, but with only sound, they become barriers that prevent the players to focus on the intended challenges of an FPS. While AudioQuake ends up having similar navigation problems to BlindFold, we realized that with proper game and level design, it has a lot of potential to create an exciting gaming experience to blind and sighted audiences.

2.3.5 Conclusions

After playing BlindFold and AudioQuake, we realized that a new approach was necessary to the design of audio-only games that have navigation as a core mechanic. As of now, there are no games that we know of which allow navigation in an intuitive and immersive way. The biggest problems that we detected are the following:

1. **Lack of input feedback** – In videogames, pressing keys and/or moving the mouse to control the character position and rotation usually have an effect that can be seen on the screen. However, that is not possible in audio-only games and thus, other means of conveying information must be designed.
2. **Lack of adequate mechanisms to inform the player about the PC's surroundings and position in the Level** – From our gameplay sessions we realized that a new approach to how information is conveyed exclusively through audition is necessary. In BlindFold, there weren't enough sonic references for the player to navigate in a premeditated way. In AudioQuake, there were too many sonic references that would overlap each other, hindering the player's ability to know in which part of the map he was. The number of sounds used to convey information was also too big, creating a steep learning curve which is an obstacle for novice players. Finally, the semantic of sounds have an important role on making information easily accessible to the player, which was also an obstacle for us, as most of the time we were more focused on processing all the information, rather than thinking about strategies to improve our chances to win.

We propose solutions to these problems in the Design Chapter.

2.4 Human-Computer Interaction

One of the founders and of human-computer interaction (HCI) is John Carrol. He is best known for his theory of Minimalism in computer instruction and has made large contributions to the advancement of HCI since its inception. He claims the following: *"The original and abiding technical focus of HCI was and is the concept of* usability*. This concept was originally articulated somewhat naively in the slogan "easy to learn, easy to use". (…) However, inside HCI the concept of usability has been re-articulated and reconstructed almost continually, and has become increasingly rich and intriguingly problematic. Usability now often subsumes qualities like fun, well being, collective efficacy, aesthetic tension, enhanced creativity, flow, support for human development, and others."* [18]. In other words, HCI is a very broad field that aims to facilitate the use of computer technology while improving the experience of its users.

Most software can (and should) be a subject of study by HCI. However, this field has many principles and methodologies that were created with software in mind and thus, are not applicable to video games. Consequently, it is important to make the distinction between software and videos games to understand how software HCI differs from video games HCI. Video games enable different forms of participation: exploring caves in Minecraft (Playfulness), beating a record in Tetris (Challenge), playing Tennis on Wii (Embodiment), etc. Furthermore, when a player enjoys a game, he usually wants the experience to last as long as possible. Other forms of software are not prone to this kind of participations, as they are used, most of the times, as tools to achieve some goal in the most efficient way possible.

Pippin Barr, who created the Video Game Activity Framework [6], makes a clear distinction between both forms of HCI by comparing a video game, *Grand Theft-Auto: San Andreas* [19], to a software, *Microsoft PowerPoint* [20]:

> *"We can identify four key differences in interaction which help to characterize video game play as distinct. First, in using PowerPoint, a user's primary objective is the creation of a presentation and the interaction is a means to this end. In playing Grand Theft Auto: San Andreas, however, a player's primary objective is play and the interaction is an end in itself. Second, given this difference, users of PowerPoint expect the interface to be as unobtrusive as possible, while players of Grand Theft Auto: San Andreas are specifically focused on the interface as they play: the interface is the game. Third, while we generally think of PowerPoint as solely facilitating our work, Grand Theft Auto: San Andreas frequently assigns tasks such as killing gang members, evading the police, or navigating the world. Finally, in PowerPoint the ideal user experience is seamless and without error, but a successful playing of Grand Theft Auto: San Andreas will inevitably and acceptably include mistakes, challenges, and the frequent death of the player's avatar."*

The author wraps it by saying: *"The fundamental point of distinction can be summed up as follows: people use software, but they play video games".*

Developing an audio-only game implies an additional focus on HCI. The lack of visuals creates barriers to the input-output feedback loop that often forces the developers of audio-only games to create innovative designs and applications of input devices to circumvent those barriers.

2.4.1 Input Devices

Interaction implies some kind of action that affects two or more objects. When interacting with a computer in particular, it is necessary some kind of equipment used to send data and control signals to be processed. This kind of peripherals are called input devices.

Since we decided to develop on the mobile platform, we have some input devices that are available to us right out of the box and others that are also compatible, which are described in Table 10.

Input Device	Description
Touchscreen	A user can give inputs through multi-touch gestures by using a pen and one or more fingers. Different gestures can be mapped to different actions. For instance, tapping can be mapped as selecting, swiping can be mapped as next/previous, pinching can be used as zoom in/out, and so forth. Touchscreens are incredibly intuitive, providing a way for the user to interact directly with what they are seeing in the screen. One of their biggest uses is in applications or games that require the user to draw.
Mouse (optional)	The mouse is a pointing device that detects motion relative to a surface in two dimensions with great precision. The standard features of a mouse are two buttons. The left button is mostly used to make selections and the right button allows an accessible way to list alternative options. Very often there is also a scroll wheel that is used to navigate up and down through a page and also acts as a third button. This device is particularly useful for video games that require the player to look around in a very fast and controlled manner, like first-person shooters (FPS). A mouse can be plugged to a smartphone by Bluetooth or by cable, if it supports USB OTG.
Keyboard (optional)	A keyboard is a typewriter-style device that has many different keys, each mapping its own symbol. It's is possible to produce special actions or commands by pressing and holding some keys in a pre-determined order (e.g.: ALT+CTRL+DEL). This device is very suitable for video games that have a discrete space (e.g.: Tetris), 2D platformers (e.g.: Super Mario) or fighting games (e.g.: Mortal Kombat). A keyboard can be plugged to a smartphone by Bluetooth or by cable, if it supports USB OTG.
Accelerometer	An accelerometer is a sensor that measures proper linear acceleration (acceleration relative to free-fall). Smartphone's

	accelerometers usually have 3 orthogonal axis, each measuring the effect of earth's gravity force on the smartphone. They are used to set the correct orientation (portrait or landscape) of applications, depending on the way the smartphone is held. They are also used in video games, using the tilting of the smartphone as input.
Compass	The digital compass uses a magnetometer that provides the orientation of the smartphone relative to the Earth's magnetic field. Since the smartphone knows where the north is, it can auto rotate digital maps depending on its physical orientation, which can be used in video games.
Gyroscope	A gyroscope is a sensor that measures angular acceleration on 3 axis, providing a way to measure the smartphone's rotation - roll, pitch and yaw (Figure 4). It can be used with the accelerometer and compass to provide more reliable measures. Like accelerometers, gyroscope measures can be used as inputs in video games where the physical rotation is used in-game.
GPS	Global Positioning System (GPS) is a network of orbiting satellites that send precise information about its position relative to the earth. GPS receivers use that information to calculate with a certain degree of error their position, speed and time. Some games rely on the use of a GPS (e.g.: Geocaching).

Table 10-Description of input devices

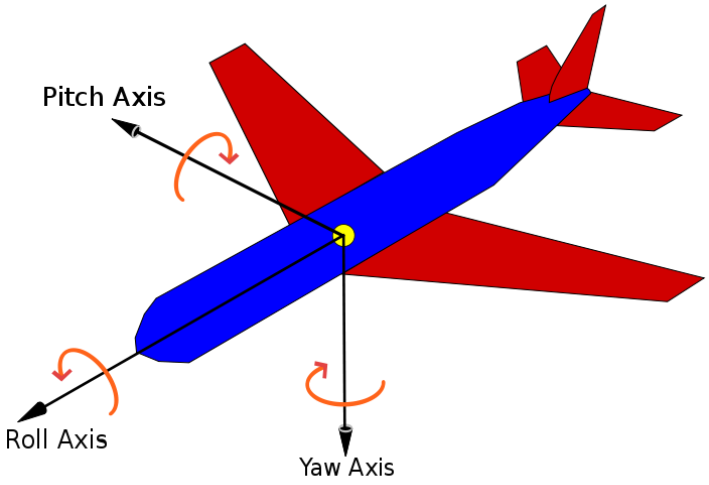

Figure 4-Roll, Pitch and Yaw 3D representation

This input devices can be used in a synergetic way, along with sonification, to improve usability and accessibility in games and software.

2.4.2 Sonification

"Sonification is the use of nonspeech audio to convey information. More specifically, sonification is the transformation of data relations into perceived relations in an acoustic signal for the purposes of facilitating communication or interpretation" [21].

In video games most of the information is fed to the player through visuals and sound is mostly used to prevent overload of information, enabling the player to change his focus in a non-intrusive way. Moreover, sound is frequently used to improve the immersion of the player through music and a complex soundscape. Sound is also of the utmost importance in rhythm games like Guitar Hero. In this particular example, it provides the player a way to detect the beat of the music which in turn helps him to know the right time to press the strumming bar button.

One of the biggest challenges presented to us was to convey information to the player exclusively through sound. We already described how some audio-only games use different Audio Patterns to guide the player and inform him about important events. In this process we got a good sense of the different ways sound can be explored and more importantly, what is intuitive and promotes an interesting gameplay experience.

3 Research

In this chapter, we start by defining the research objectives that stemmed from the State of the Art. This includes a brief explanation of the three game proposals and the rationale behind the research objective that was added in the second semester. Following, we present the initial planning for the second semester along with the details of each milestone. Lastly, we explain and disclose the reviewed planning after the objectives were reprioritized in the beginning of the second semester.

3.1 Research Objectives

One of the major difficulties we faced in the first semester was the problem definition. It was extremely important to set relevant objectives, guided by the right motives, as all the work to be executed in the second semester would be a direct result of them. After long deliberation, we decided that this dissertation would be about audio-only games and set out to learn more about the genre. This was achieved mainly through gameplay sessions with different games, followed by formal analysis of their game design. These gameplay sessions made us realize that most problems of the audio-only genre lie with the way information is conveyed to the player. That conclusion led us to research on sonification and HCI, two fields that exist to improve the usability of software and consequently, improve the user's experience.

After we identified the most troublesome problems in audio-only games, we started to hypothesize methods to solve them. At this point, we were able to start creating game proposals that were suitable to test our solutions, while exploring different forms of participation by the player.

Finally, during the second semester, after having a working prototype of the First-Person Shooter, we started to experiment different topologies, which made us notice that level design would need a more careful approach than we were initially expecting. We decided that a systematic study would be useful to

learn more about the human hearing and would allow us to design levels in a more informed way.

3.1.1 Initial game proposals

During the definition and refinement of the objectives and requirements of this dissertation, in the first semester, we committed to the decision that the best way to explore the potential of audio-only games was not to create one game with high production value, but rather create several prototypes in which different forms of participation and genres can be explored. The rational for this decision also took in consideration the possibility of researching a broader range of sonification techniques, which we hoped that would result in a relevant contribution to HCI.

In order to show that it is possible to create different forms of participation from the same mechanics and to shorten the development time of each game, some game mechanics are shared across the different games.

Each of the next game proposals describes briefly the mechanics, dynamics and aesthetics that we hope emerge from the gameplay experience. It was not possible to fully describe the game design model centered on participation in an early stage, because the actual forms of participation were only revealed during playtesting, after having working prototypes. As we already said, the next game proposals were made in the first semester and are brief because they only served as guidelines for the work to be developed. Additionally, due to changes in our research objectives that occurred during the second semester, the first game was changed and the second dropped. Those changes and a more detailed view into the planned design is given in the Design chapter.

3.1.1.1 Game #1 – Racing

In this game the player is controlling a spaceship that is navigating through a "wormhole". Since the wormhole is slowly closing, the player must reach its end as fast as possible, or he risks to be trapped inside it forever. The

player sets the direction of the spaceship by moving his head while using a head-tracking device, and controls the throttle and breaks with two buttons. If he gets too close to the boundary of the wormhole (which can be thought as a tortuous tunnel), the player's spaceship top speed is limited to a lower value. Consequently, the player has to stare straight to the audio source as best as he can through the whole track, as that audio source moves around in front of him at a speed controlled by the player. Finally, tracks are procedurally generated using ad-hoc methods, which in turn contribute to a bigger replay value.

3.1.1.2 Game #2 – Rhythm

This game builds upon the Racing game and introduces some new features that haven't been explored in the audio-only game genre. This is a rhythm based game that, similarly to the aforementioned Racing game, procedurally generates tracks, based on the music the player selects. That music must be stored in his smartphone and influences different features on the track generated. When playing a track, it has sound cues that inform the player that he should do some gesture with his head or tap a button at specific times. This implies the implementation of an algorithm of pattern recognition. Furthermore, a beat detection algorithm should be used to get the tempo – or beats per minute (BPM) – to allow the synchronization of the sound cues with the music. Additionally, we suggest the possibility of having more sound cues in the most energized parts of the music, which might contribute to improve the immersiveness that is consequence of the embodiment of the music.

3.1.1.3 Game #3 – First-Person Shooter

Our goal with this game is to explore the FPS genre, which we think that has much potential but hasn't been approached the right way by audio-only games so far, for the reasons already mentioned. Most of the information was collected during the analysis of AudioQuake. We found that it is very hard to make a proper sonification of all the elements of a game as complex as Quake,

since too many sound cues result in overload of information and consequently, a gameplay experience that is frustrating. For this reason, our implementation of a FPS is stripped from all the unnecessary elements and tries to fix the major problem we found in AudioQuake: navigation. It's important to mention that one of the strengths of this genre is multiplayer, which has the potential to greatly improve its sociability value. This value is particularly important, as there is a shortage of accessible games (especially audio-only) that allow players to interact between themselves.

The gameplay of our prototype is similar to the visual counterpart of a competitive FPS like Quake or Unreal Tournament. It is played as a 1vs1 Deathmatch with time limited rounds. At the end of each round, which lasts some minutes, the player who has more kills, wins. Each player has a weapon (longbow) and limited ammo (arrows). As the player navigates through the level, he might encounter an opponent. The adversaries' presence is announced by a continuous sound that informs the player of his presence. At this point, the player must turn to his enemy the best he can, similarly to the first game, and fire. If the shot doesn't miss, the damage is calculated and its health drops. When any players' health drops to below 0, he is killed and respawns at a different location with full health and ammo. However, when a player survives a confrontation, it is expected that he tries to restore the lost health and ammo. To do so, he has to navigate to places where those resources can be collected. We call those places Chambers, of which there are 5 types: Ammo, Health, Shield, Firing Speed and Neutrals. The first four will provide a resource to the player and the later act like a regular Chamber. Players can only accumulate resources to a certain point, so, for instance, a player with full health cannot collect a health resource when visiting a Health chamber. On the other hand, if his health is not full, he collects it and that resource only becomes available at that chamber 30 seconds later. The firing speed and shield are non-essential survival resources that can be obtained to get an edge against opponents and add some flavor to the game by motivating the players to navigate through the map to collect them.

3.1.2 Hearing acuity study

This research objective was added during the second semester, when it became apparent that the nature of sounds, their number and even the angles between them affected the gameplay experience by changing the time it took for the players to process information and act accordingly. For instance, when having two similar sounds sources too close to each other in front of him, the player sometimes would not be able to rotate and move to the intended sound source. Additionally, we realized that players have different sensitivities regarding sounds, and that should be taken in account when designing levels. This showed us an opportunity to research something that, to our knowledge, was never investigated in the academic world.

Fitt's law is an empirical model that predicts that the time required to rapidly move a cursor to a target area is a function of the ratio between the distance to that target its width. This means that an user takes less time to point and click in large targets, but that time increases if the target is far away from the initial position of the cursor. This scientific law has already been proven under a variety of conditions, using different limbs, physical environments and user profiles, but it was never tested with sound.

We think that it is of paramount importance to audio-only game genre to know the limits of our audition when hearing several sound sources at the same time. Thus, we decided to redirect some of our efforts to research that and dropped the development of the second prototype in order to have enough time to further improve the FPS, enabling it to also serve as a testbed. Additionally, we made changes to the racing prototype in order to use it to assess the testers hearing acuity and ability to follow sound sources, which was useful to validate the data gathered in subsequent tests.

3.2 Methodology

During our search for the design methodology that best suited our interests we found that good lessons can be extracted from almost everywhere. The most orthodox design methodology is Waterfall: it starts with the design of the entire game using extensive documentation, followed by the implementation phase, then the testing phase after which the game is ready for shipment.

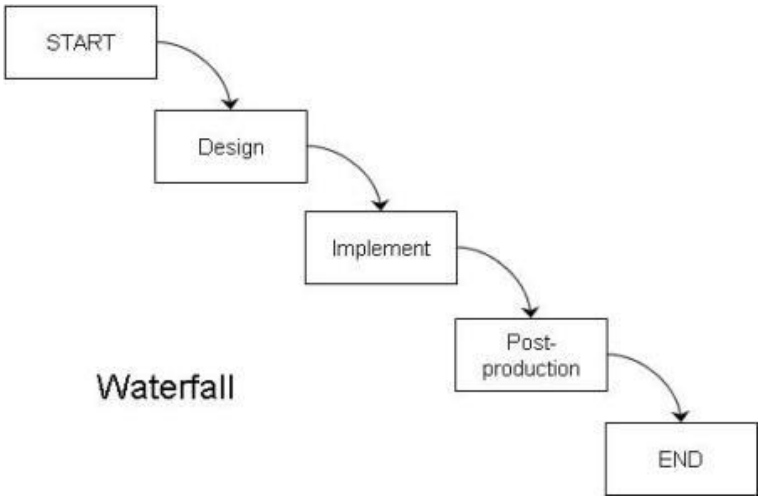

Figure 5-Waterfall model

In the waterfall model (Figure 5), the project is developed in sequential, unidirectional phases. Thus, this model is very adequate in projects where there's little risk and uncertainty, like sequels or expansions. However, since each phase can only start when the previous one has been finished, the project ends up having little room for evaluation and iteration, which is the major disadvantage of this model.

The second game design methodology we researched was Scrum. Its roots are based on the Manifesto for Agile Software Development [7]:

*"**Individuals and interactions** over processes and tools*

***Working software** over comprehensive documentation*

***Customer collaboration** over contract negotiation*

***Responding to change** over following a plan"*

This methodology promotes the division of the project in tasks that are allocated to small, multi-disciplinary teams that work closely to each other. At the

beginning of every week, some of those tasks are chosen by the Scrum Master, and are developed in short work cycles called Sprints (see Figure 6). However, the duration of those cycles depend on the nature of the project and are decided by the Scrum Master, whose responsibility is to keep the team focused on meeting the Sprints and serves as a mediator of the different teams. Every day the teams have a short meeting with the Scrum Master so that problems can be detected and corrective measures taken.

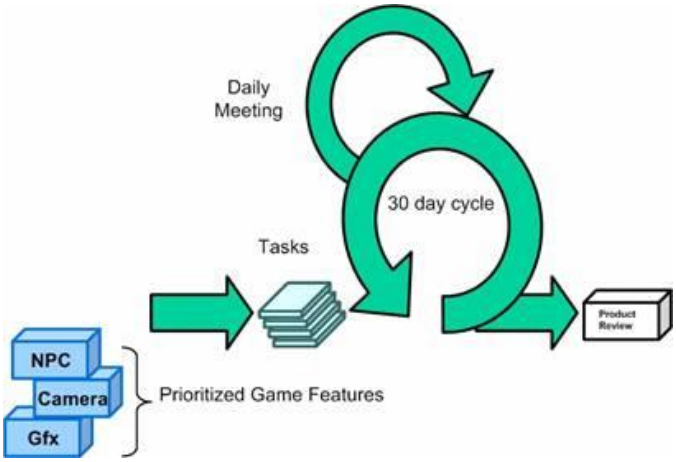

Figure 6-Scrum model

This approach stimulates iterative development, from which stems features that can be evaluated and tested for functionality, which is great to assess the progress of the whole project. These iterations also help to decrease some risks – being design risks and implementations risks the ones that we are more concerned with. Design risk is the risk of designing a game that is not engaging, while implementation risk is the possibility of facing technical challenges that prevent the development to continue without making some changes to the design.

For the aforementioned reasons, we followed a methodology that borrows concepts from Scrum. Our requirements were used as backlog, which allowed us to gauge our progress along the whole development by completing milestones.

3.4 Planning

In this chapter is described – in a chronological fashion - the work done during the first semester and projected milestones for the second semester. This dissertation's nature is highly exploratory, which leads to high degrees of uncertainty and in turn, prevented us to create a thorough planning. That being said, the projected milestones were useful to lead the development of the prototypes and allowed the assessment of their progress.

In the first semester we started without a well-defined problem. We knew that audio-only games have the potential to improve game accessibility, but had no idea how much this game genre in particular was being neglected by the gaming industry. Research on audio-only games that included gameplay sessions gave us insight on the major barriers to interesting gaming experiences. We found that some forms of participation and genres were really frustrating to play because of the poor HCI and sonification techniques used to convey information to the player. That made us recognize an opportunity in research and from there, the specifics of the problem definition started to be clear, which enabled us to actually start the documentation of the State of the Art, which ended up taking the whole semester. During this time we researched on many different areas that influence audio-only games. We researched Acoustic Ecology and Sound Design Patterns to understand the usefulness of sound to convey information to the player and create aesthetics that allow immersive experiences. This, along with the research on Game Design Models and Frameworks, allowed us to create formal evaluations of several games that are archetypes of different genres and support distinct forms of participation by the player. This analysis also allowed us to systematize the interaction problems we faced in those games and think about possible solutions to them using different input devices. Finally, we made some proof of concept tests to mitigate design and implementation risks, by making sure we had the technology to create interesting mechanics.

The main challenge of the second semester was the implementation of the three game proposals. After the conclusion of each prototype, we planned to

playtest and evaluate them using different metrics in order to assess if they are engaging to both blind and sighted audiences and use the players' feedback to improve and refine implemented features. Finally, we planned the integration of all these learning in this dissertation final report.

3.4.1 Planned milestones

Since we decided to some concepts borrowed from the Scrum methodology to gauge the progress of this dissertation, we planned one-month sprints with specific milestones to be achieved in each one. However, due to the exploratory nature of this dissertation, we did several adjustments to the milestones throughout the year, always prioritizing the aforementioned research objectives.

Table 11 includes all the milestones of this dissertation and their description, as planned in the final of the first semester.

Milestones	Sprint	Description
M1	February	Definition of the problem
M2	March-May	State of the Art
M3	April	Methodology
M4	May	Proof of concept prototypes
M5	June	Intermediate report
M6	July	Implementation of prototype #1
M7	September	Implementation of prototype #2
M8	October	Implementation of prototype #3
M9	November	Playtest and evaluation
M10	December	Integration of feedback
M11	January	Final report

Table 11-Sprint planning

3.4.1.1 Milestone 1 – Definition of the problem

Before starting the draft of the State of the Art, it was necessary to have a clear problem definition and propose a solution that would guide this whole

dissertation. This milestone was completed successfully, even though the research objectives were changed during the second semester.

3.4.1.2 Milestone 2 – State of the Art

The documentation of the State of the Art was one of the most challenging milestones to complete. It started with gameplay sessions of various audio-only games and as we researched about all relevant topics, the results would be added to the draft.

3.4.1.3 Milestone 3 – Methodology

In this milestone we researched about the methodologies used by the gaming industry. This milestone was completed in mid-April.

3.4.1.4 Milestone 4 – Proof of concept prototypes

During this milestone, we created a tech demo to make sure that we had all the technology necessary to develop the planned prototypes.

3.4.1.5 Milestone 5 – Intermediate report

The last milestone of the first semester was to write all the remaining chapters of the intermediate report.

3.4.1.6 Milestone 6 – Implementation of the first prototype

The first projected milestone after the delivery of the intermediate report was the development of the first game prototype. It was important to familiarize ourselves with the Unity development pipeline and learn more about the vicissitudes of audio-only games. The simple nature of this game, which is following sound sources by using a head-tracking device, would also provide itself an excellent opportunity to improve that feature, which would be used in the other two games

3.4.1.7 Milestone 7 – Implementation of the second prototype

Once the first prototype was completed, we could then start to implement an algorithm to recognize head gestures. This would allow us to start introducing sound cues in a song and experiment different mechanics, until we fulfilled the goal of producing an experience that explored the embodiment dimension in a fun, immersive way.

3.4.1.8 Milestone 8 – Implementation of the third prototype

Once the previous game was mature enough, we would be able to start our most ambitious milestone: the development of an FPS in which our proposed solutions for navigation could be implemented.

3.4.1.9 Milestone 9 – Playtest and evaluation

The expected output from this milestone was to test the three game prototypes with different user profiles, while evaluating their performances and making interviews to gather feedback that could be later integrated.

3.4.1.10 Milestone 10 – Integration of feedback

At this point, we should review all the feedback gathered during the previous milestone, and prioritize them in order to fix the most important aspects of the three prototypes.

3.4.1.11 Milestone 11 – Final report

The last milestone is the integration of all the learnings in the final report. All the writing and reviewing should be completed by the end of this milestone.

3.4.2 Reprioritization and executed planning

Even though we had planned to develop three prototypes, we started the second semester by prioritizing the work and realized that before polishing the first prototype, we should initiate the development of the FPS game, which was the one that offered most potential for research and allowed us to implement and test our proposed HCI and sonification techniques, with the purpose to solve navigation problems in audio-only games. Furthermore, the high complexity of the planned prototype resulted in implementation risks (technical issues, prolonged developing time, etc) that we wanted to suppress as soon as possible. Finally, we added a research objective to this dissertation and for this reason, the milestones for the second semester were redefined as follows:

- **Sprint 1 – September-October:** Changes to the racing game and development of the FPS.
- **Sprint 2 –November:** Preliminary playtests to gather feedback and following integration of changes to the FPS.
- **Sprint 3 – December:** Implementation of spectator mode using client-server architecture and formal playtesting.
- **Sprint 4 – January:** Writing of the final report.

All the sprint goals were completed successfully, and their output will be explained in the following chapters.

4 Design

As exploration/navigation is a recurring mechanic in most videogames, we think that the audio-only genre should not discard such mechanic solely for accessibility difficulties. The playtesting sessions gave us the opportunity to detect problems with navigation in more than one game which motivated us to create a set of HCI and sonification techniques with the aspiration to solve these problems, enabling the players to redirect their focus from navigation to more important goals. In this section we will explain those techniques and the rationale behind the important game and sound design decisions. Finally, we detail the requirements that emerged from the design.

4.1 Proposed HCI techniques for navigation

To deal with the problem of lack of input-output loop feedback, we propose the use of a head-tracking device. Fortunately, most models of smartphones have a gyroscope, an accelerometer and a compass, sensors that when combined, can give accurate readings of the orientation (roll, pitch and yaw) of the smartphone. Because of this, the player can have direct control over the rotation of the PC, as it is mapped directly to the orientation of his head. Furthermore, we propose external *stimuli* to give the player references for the cardinal directions. For instance, a regular fan can be used as a reference to North. To allow 360º navigation, the player should be sitting on a revolving chair, which also enables the use of different floor materials or objects that serve as spatial references (e.g.: carpets, pillows, etc). By combining the information provided by these external sources, the player can know which direction he is facing because he is able to feel the air flow on his face and the different materials on his feet. With this solution, the player always knows which cardinal direction he is facing, and has intuitive feedback for orientation, since the speed he rotates his head affects directly the speed the PC rotates. With the intricacies of orientation cracked, we

will now explain how we used sonification to convey information about the PC's surroundings in the FPS prototype.

4.2 Proposed Sonification techniques for navigation

Just like Quake, Unreal Tournament and other competitive FPS games, our Levels have resources spread throughout the map. In our case, these resources are:

1. **Health –** When collected, PC restores his HP completely.
2. **Ammo –** When collected, PC refills all his Ammo
3. **Shield –** When collected, PC gains an additional protection of 50 HP.
4. **Firing Speed Boost –** When collected, PC's weapon reload time drops to 1 second, instead of 2 seconds.

We can make a parallelism with the way sound works in sewers to better explain the way sound works in our Levels. Each Level (seen in Figure 7) can be viewed as a graph that has Nodes connected by Edges. Each Node is a Chamber and each Edge is a Corridor connecting two Nodes/Chambers. Chambers are circular with a specific radius and in their center, there is a sound source that represents the resource that Chamber has.

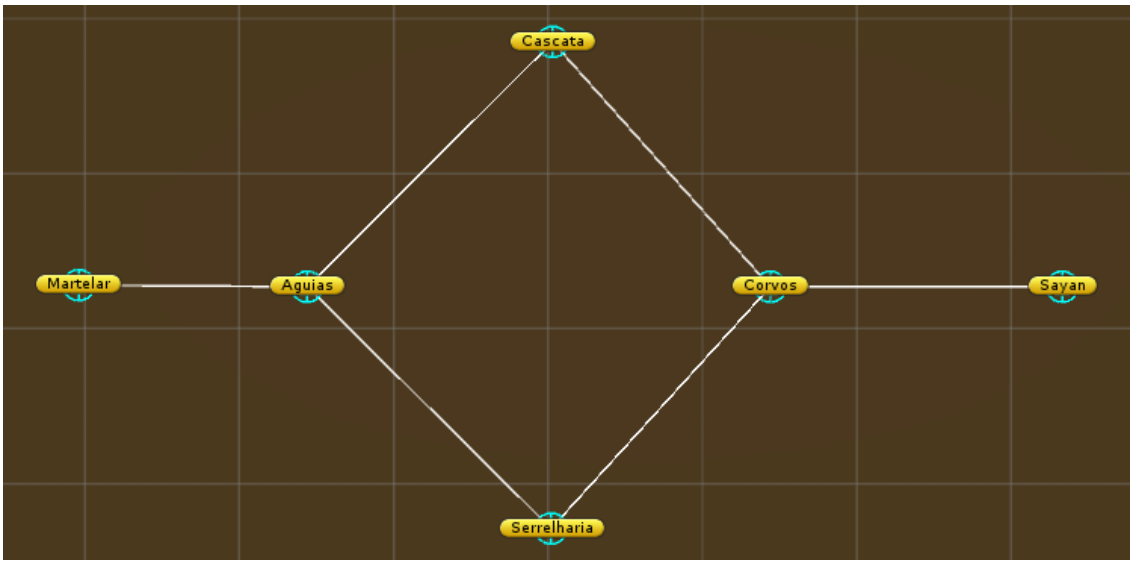

Figure 7-Topology of a Level

Obviously, not every Chamber can have a resource, because that would make the PC's navigation almost meaningless, as he would always have some benefit, no matter wherever he was navigating to. Since this is not desirable from a Game Design perspective, some Chambers have no resource, even though they still have a sound source in their center.

When the PC is inside a Chamber, the player can hear the sound sources coming from adjacent Chambers and the sound source coming from the center of the chamber he is currently in. If he decides to move in the direction of one of the adjacent chambers, once he leaves the radius of the Chamber he is in, he enters the Corridor and starts to listen only to the sound sources coming from both ends of it (i.e.: the Chamber he came from and the one he is heading to). As the PC continues to travel along the Corridor, the sound source volume he is facing continues to linearly increase to a maximum. Once the player gets to the end of the Corridor, he enters the Chamber and that sound drops rapidly to a minimum, in order to allow the player to better listen the sound sources coming from adjacent Chambers. This is not a realistic behavior of sound, but after several iterations we decided that this was the best option. The three options we tried, in chronological order of iterations, are the following:

1. **Fully realistic approach –** The closer the PC is to a sound source, the louder it is. This means that when a player is inside a chamber, the sound is at its maximum volume. This was actually our first approach because it represents sounds in a natural, intuitive fashion, but we concluded that, in this conditions, it was too hard to listen to the adjacent Chambers, so we discarded it. In other words, this approach is the best to inform the player of his current location, at the expense of the adjacent locations.
2. **Half realistic, half functional approach –** This approach is equivalent in everything to the aforementioned, with the exception that whenever a player is inside a Chamber, its volume is decreased to lower than the

adjacent Chambers. This facilitates the listening of adjacent Chambers, but creates two unrealistic situations. If the player is in a Corridor and enters a Chamber, its sound volume decreases rapidly, and if he leaves a Chamber and enters a Corridor, the sound source volume of the chamber on his back increases rapidly.

3. **Fully functional approach** – In this approach, the sounds furthest from the PC are louder and as we get closer to them, they decrease linearly, being at their minimum volume when the player is close to them. Consequently, when a player is inside a chamber he can listen clearly the sounds coming from adjacent Chamber. This approach is completely unrealistic and very counter intuitive for untrained players, but excels at informing the player about the adjacent Chambers, without having abrupt changes in sound sources volume.

As our goal was to produce a game with a high degree of challenge and competitiveness, there was a necessity to keep the player constantly well informed about certain aspects inherent to a FPS, enabling him to create strategies and navigate in the Level accordingly to them. For instance, when a player is low on health, he should navigate to a Health Chamber or he risks getting killed in a confrontation. For this reason, we favored functional approaches over realistic ones in most game design decisions. However, we discarded the fully functional approach because even though it allowed an informed navigation with smooth sound transitions, we found it so counter intuitive that immersiveness was broken, and for that reason we ended up discarding it.

With the aforementioned approach, the player can be in on of two situations when navigating the level:

1. The PC is inside the radius of a chamber and can only hear the sound of that chamber and the adjacent chamber(s) (Figure 8a).

2. The PC is inside a corridor and can only hear the sound of the chambers coming from both ends (Figure 8b).

Unlike many FPS videogames, where sound is implemented with a natural and realistic approach, we do not apply sound occlusion, but rather omit entirely all the chamber sounds the player isn't supposed to hear, even if those chambers are physically close to the PC. This design decision was made in order to keep the player without sound overload that contains information that does not contribute much to the way he plays. As an example, even though the PC is much closer to Node (1) than Node (3) in Figure 8, he cannot actually listen to Node (1) chamber sound.

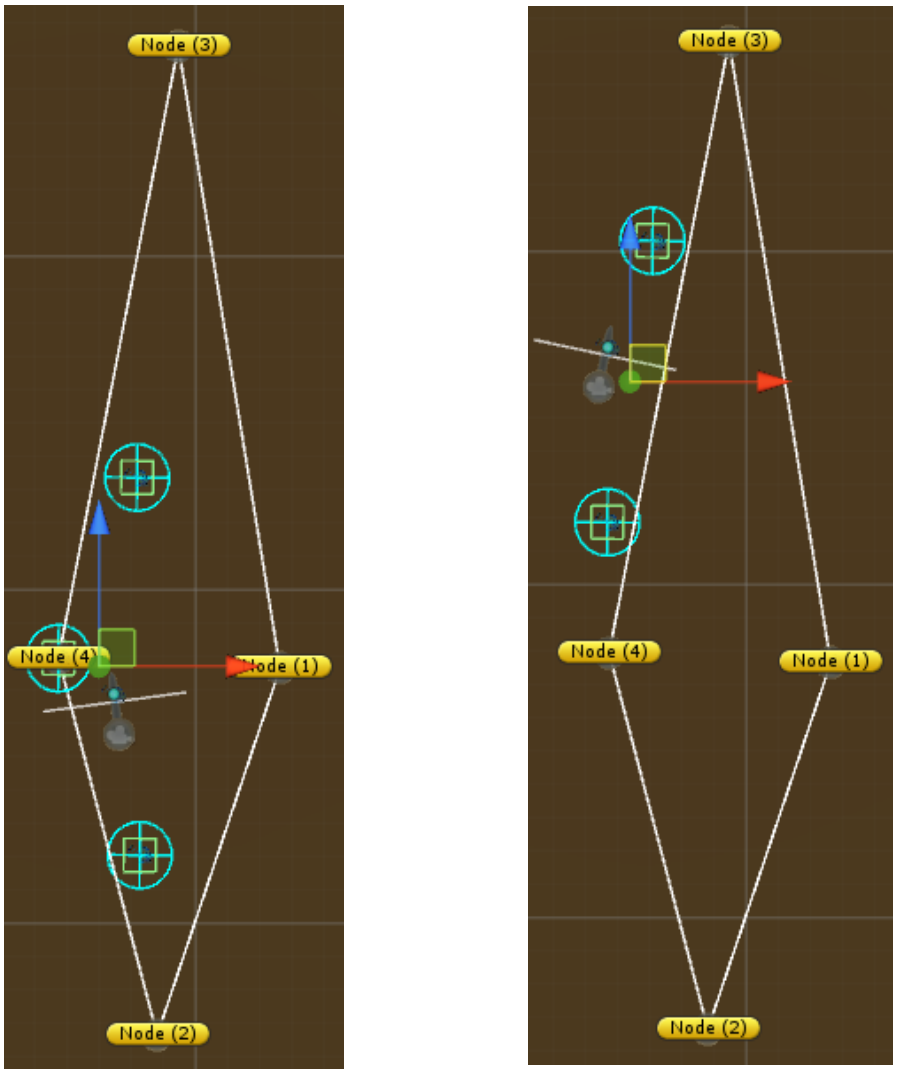

Figure 8-Sound sources behavior inside chamber (8a) and outside chamber (8b)

Additionally, it is necessary that, when the PC is inside a chamber, the player can hear equally well the sound coming from adjacent chambers, disregarding of their distance. For this reason, we decided that sound sources should be disattached from the objects they belong to (i.e.: center of each chamber). Instead, every update they should be repositioned in order to make sure they are at the same distance from the player. Then, depending on the position of the PC relative to both ends of the corridor he is in, the volume of each of the two chamber sound sources is updated. If, in the other hand, the PC is inside a chamber, that sound source is repositioned to its center and the chamber sound is reduced to lower than adjacent sound sources to allow him to focus on them.

To make everything clearer, we can once again use Figure 8 as an example. When the PC is inside a chamber (i.e.: Node (4) in the 8a) the player can listen the sound sources of Node (2), Node (3) and Node (4), but not Node (1). However, we can see that the distance from the PC to Node (3) is much bigger than the distance from the PC to Node (2). This happens because, as previously discussed, we decided that it is more important to inform the player about the adjacent nodes rather than the distance to them, the latter being usually done by making the volume of closer objects higher than distant ones. With our design, the sound sources of Node (3) and Node (2) are at similar volumes and the sound source volume of Node (4) is lower than the other two. Additionally, after the PC leaves the chamber and enters the corridor of Node (4)-Node (3), navigating towards Node (3), he ceases to hear the sound source of Node (2) and starts to hear Node (3) increasingly louder. As for Node (4), right after the PC left the chamber and entered the corridor, its volume increased very fast to the maximum value and then, decreased as the PC continued to walk along the corridor towards Node (3).

4.3 Sound Design

It is important that players are able to immediately tell the meaning of a sound when earing it. Sonification can be used to transmit information to the players, but this implies careful game and sound design to prevent information or sound overload.

The possibilities to represent chambers with sonic references are virtually limitless and, in the end, the easiest thing to do would be to teach the players about the meanings of each sound explicitly. However, we wanted to avoid the creation of a tutorial using text-to-speech because we value experience as a teaching tool, rather than (tedious) instructions. While some events can be intuitively conveyed to the players exclusively through sound, that does not hold true to the most abstract ones. Our analysis to Audioquake made us realize that when sonification is made with a lot of "artificial" sounds that aren't prone to any kind of interpretation, they are hard to memorize because they don't hold any kind of meaning or semantics. With this in mind, we realized that natural sounds always carry some information, and decided to use that to our advantage. Additionally, by using natural sounds, players would also be able to have some freedom of interpretation, which was useful to create a sense of aesthetic inside the level.

After brainstorming with more than 10 people about possible options to represent different types of chamber using sound, the final mapping between resources and sounds, as well as their rationale is presented in Table 12.

Resource	Chamber sound	Rationale
Health	Waterfall	Water is commonly associated with rejuvenation, peace and purity. For that reason, we used the sound of a waterfall to represent a place where players can cleanse themselves from their wounds, gaining Health.
Shield	Forge/Hammer hitting an anvil	We wanted to convey a sense of protection to the player. Since, the most widely known shields are made in metal (e.g.: shields seen on movies) and are used as means of protection, it seemed logical to use forging sounds to represent a place where shields are made.
Firing Speed	Super Sayan aura from Dragon Ball series	It is very hard to represent a concept as abstract as "the increase of speed/power" exclusively through sound. However, almost all interviewees that recognized the sound from Dragon Ball (a popular series that most people is familiar with) could correspond correctly the resource to the sound.
Ammo	Handsaw/Sawmill	After knowing that the PC uses a longbow and arrows as a weapon, which are commonly made in wood, most interviewees could guess by intuition that arrows need a handsaw to be made.
Neutral	Peacocks / Crows / Eagles / Parakeet	We wanted to represent all the neutral chamber with sounds containing equivalent semantics. Consequently, we decided to use bird sounds, since they also contributed the intended aesthetics of the game.

Table 12- Rationale behind the mapping of chamber sounds to resources

After deciding which sounds would better represent each resource, we needed a way to inform the player if a resource was available or in cooldown, preferably through sound and without the need to be inside the chamber. In videogames, a player can see if a certain place has a resource by simply looking for the icon or model that represents it. If the resource is not there, then the player doesn't waste time with it. This is particularly important when we think about resources that spawn in the end of an alley, because the time wasted to go there doubles. Even though in our prototype every resource's cooldown is 30 seconds, the possibility of having more powerful resources with longer cooldowns

still exists. Consequently, we thought of several options to inform the player about the availability of a resource. The first we tried was to get a similar sound, semantically related to each chamber sound when its resource is available. For instance, the sound of a waterfall represents abundance and thus, the presence of the health resource, while slow dripping water represents the same resource, which has been depleted. However, it eluded us how to represent this kind of duality with other resources in an intuitive manner. Additionally, this meant that the players would have to memorize twice the number of chamber sounds, which would make the learning curve steeper. Eventually, we realized that every sound has inherent properties that we could explore in our advantage to solve our conundrum: **volume**, **reverb**, **speed** and **pitch**. The **volume** property was already being used to inform the player of the PC distance to chambers, so it didn't seem wise to use it to convey any more information because the sound could become ambiguous and confusing. **Reverb** seemed an interesting option if used the right way. For instance, resources on cooldown could have reverb applied to their sound source, which would decrease with time until they were available again, in which case no reverb would be applied. This option ended up being discarded because it was impossible to notice the applied reverb to sounds with too much noise, like a waterfall or super sayan. Likewise, **speed** could be used in a similar fashion of reverb, deaccelerating a chamber sound when its resource disappeared, and then gradually accelerating it until the resource was available again. Although we could have used speed, **pitch** showed itself the best option because players could easily perceive the difference between the high and low pitched versions of each chamber sound, and we could even modify the pitch using RealSpace3D sound engine.

In order to ease the player navigation even more, we thought that the footsteps layer of sound could be used to inform the player of his whereabouts inside the level. For instance, we could divide the map in two regions: while inside the east region, the PC would hear steps on sand and in the west side, the player would hear steps on rocks.

It was also necessary to inform the player about the presence of enemies, which occurs when they are in the same corridor, including both chambers on each end. Our approach mimics what happens in a videogame FPS, where characters suddenly appear or disappear from the corridor the PC is in, just by going around the corner. We decided to use zombies because they represent evil and could convey a sense of fear to the players.

Finally, we decided that we would also need to inform the player about several actions and events, which are described in Table 13.

Action/Event	Sound used	Description/Rationale
Shooting	Arrow release	Since we decided that the FPS aesthetics would relate to wilderness, it made sense that the weapon would be longbow and arrows.
Reloading	Pulling longbow string	A longbow produces a characteristic sound of straining wood when its string is being pulled. This sound duration is 2 seconds.
Fast reloading	Accelerated reloading	Same sound as the above, but faster. This sound duration is 1 second.
Taking Damage	Random grunt	A list of grunting sounds is used to prevent repetitive sounds. In this case, the sound source is also positioned around the player in the direction of the enemy that shot the PC, helping the player to better notice his direction.
Dealing Damage	Zombie grunt	When the player hits the zombie with an arrow, a grunting sound is produced from the zombie location, informing the player of his success. Different sounds can also be used to inform the player about critical hits (e.g.: headshot).
Killing	Zombie sigh	When this happens, the player will hear a sigh in the position where the zombie was killed.
Dying	Human sigh	This informs the player that he was just killed and will respawn in a Node.
Zombie presence	Zombie sniffing	When the player sees enemies, he starts to hear their sound which informs about their position.

Table 13-Rationale behind the sounds used to represent actions and events

4.4 Static AI Bot

Before even thinking of developing a multiplayer version of the audio-only shooter, many iterations would be needed. However, this would be difficulty because we didn't have available testers. Consequently, the creation of a Bot that could simulate some of the most basic behaviors of a human player was the natural solution. Those behaviors are the following, in decreasing order of priority:

1. If possible, fire against the closest visible enemy target.
2. If the Bot has less than 50 Health, go to the Health Chamber.
3. If the Bot has less than 5 Ammo, go to the Ammo Chamber.
4. Navigate randomly

To allow the search for specific Chambers in behavior 2 and 3, we implemented a simple pathfinding algorithm based on Depth-First Search. This task was simplified by the previous decision to have our Levels represented as a graph, with each node having a list of connections to adjacent nodes. Furthermore, since it is hard to simulate the way a human aims to a target, we implemented a heuristic that calculates damage as a function that is inversely proportional to the distance of the target. This also allows us to easily control the difficulty of the AI, by changing the upper bound of the damage dealt.

4.5 Damage dealing heuristics

Unlike videogame FPSs, where a player can actually see his enemies on the screen, in audio-only games the only source of information about the location of enemies is conveyed through sonification. Consequently, the players' accuracy was expected to be much lower in the FPS and thus, an alternative method of calculating damage was needed. We decided that when a player fires, the angle between the PC' facing direction and the enemy is calculated. A lower angel will then mean that the player is looking straight to his target and, consequently, more damage is dealt. An angle bigger than 30º will completely miss its target.

Finally, since bots do not aim like humans, we decided that the closer they are to their target, the more damage they deal.

4.6 Spectator Mode

Once it became apparent that we would need to create a battery of tests in order to assess the users' ability to hear sounds, the necessity of visualizing their actions as they played in real-time became evident. However, since the game runs directly on the smartphone, which is inside the Cardboard, we couldn't simply look at a screen like in videogames. Instead, we decided to design a server-client architecture (see Figure 9) that would allow us the visualization of the game remotely and change some variables remotely.

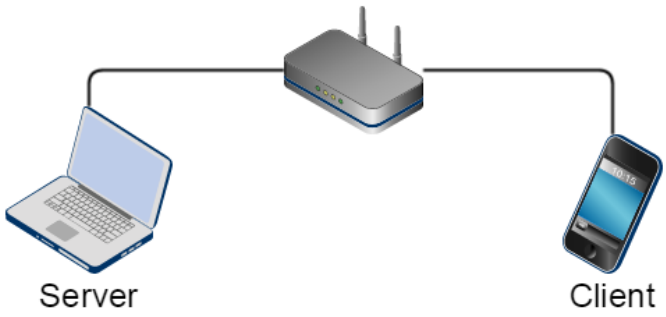

Figure 9-Server-Client Architecture

In particular, it should be possible to change remotely these variables:

- The current loaded level.
- The presence of a Bot.
- Toggle aiming aid.

When the client notices a change in any of this variables, he updates the local game accordingly.

4.7 Requirement Analysis

In this subsection we detail the requirements that were extracted from the design as a way to gauge our progress during sprints. Each requirement's rationale is detailed, along with its importance using the MoSCoW prioritization system [22].

Requirement	R1 – Navigate easily
Description	The player always knows which direction the PC is facing through external stimuli given by different floor materials and air blown by a regular wind fan. He also is informed and controls the speed the PC rotates.
Rationale	As we previously argued, this is a crucial requirement in an audio-only game that has a navigation mechanic.
Priority	Must

Requirement	R2 – Map editor
Description	The level designer must be able to create nodes, edit their positions, set their type (e.g.: neutral, ammo, health, etc.), add and visualize connections with adjacent nodes, define if characters may spawn in them and set the refresh time for resource nodes.
Rationale	Levels should be created and edited with minimal effort, making it easier to iterate them.
Priority	Must

Requirement	R3 – Automated sound loading
Description	When the level is initializing, each node automatically loads the sound that represents its type.
Rationale	Avoid errors that could happen by manually defining the sound of each node.
Priority	Must

Requirement	R4 – Level sonification
Description	Each node repositions and sets the volume of its own sound source, depending on the position of the PC in the level.
Rationale	The player must always be well informed about his position in the level. This is achieved by knowing which sound sources are around him and by having a sense of distance given by changes in volume.
Priority	Must

Requirement	R5 – Intuitive sound design
Description	Players are able to learn and memorize easily the meaning of each sound.
Rationale	Being an audio-only game, there are many sounds representing different events and actions. Ideally, the player can learn the information conveyed by every sound intuitively.
Priority	Should

Requirement	R6 – Collision system
Description	When the PC moves inside the level, there is a verification that forces it to stay inside corridors and chambers without having to use the physics engine.
Rationale	In videogames, colliders are usually attached to models like walls, floor and ceiling and are used by the physics engine to enable the PC navigation through the map. However, we didn't have the need for those models, just their colliders. Our own collision system simplifies the creation of levels by avoiding the trouble of manually dragging colliders to make up chambers and corridors. Its simple nature also takes off some the overload of calculating collisions through the physics engine, which is particularly important because the game runs directly on a smartphone, which has limited processing power.
Priority	Must

Requirement	R7 – Realistic sounds
Description	The sound engine must support HRTF to synthesize realistic binaural audio.
Rationale	Players must be able to accurately locate sounds sources around them. In particular, they should be able to tell if sounds are above, bellow, in front, in back, or at their side.
Priority	Must

Requirement	R8 – Computer controller
Description	It must be possible to control the PC with mouse and keyboard
Rationale	It would be very problematic if we would have to build for mobile every time we wanted to test out a new feature during development, especially considering the long building times. Consequently, to simplify the workflow, it must be possible to control the PC in computer builds.
Priority	Must

Requirement	R9 – Spectator Mode
Description	It must be possible to visualize on a computer what is happening in-game.
Rationale	During formal playtesting, it is convenient to evaluate the player's performance using direct observation. Since the game runs on the smartphone, there is the need of implementing a spectator mode.
Priority	Should

Requirement	R10 – Static AI Bot
Description	A Bot that simulates the behavior of a human must be present to serve as an opponent and thus, allow the iteration of the game.
Rationale	An alternative to human testers is to develop a bot that simulates their behavior.
Priority	Must

Requirement	R11 – Balanced damage dealing heuristics
Description	When the player or bot the shoot, the damage dealt must be calculated in a balanced manner.
Rationale	The behavior of the bot is hardcoded and the behavior of a human player emerges naturally when he interacts with the artifact. Consequently, the aiming system of the bot must be implemented in a way that, when combat occurs, neither party has advantage over the other.
Priority	Should

Requirement	R12 – Gauge playtest participants hearing acuity.
Description	The racing game must be modified to gauge the performance of the player when following a moving sound source.
Rationale	People have different hearing acuity and thus, that affects how they perform in audio-only games. Consequently, there's the need to relativize those results.
Priority	Must

4.8 Technologies used

It would be impossible to complete our endeavor without using some technologies. During this dissertation, we had the necessity to use a game engine, an external audio engine that supported binaural audio, an API to get the output from the smartphone's gyroscope and a software that allowed us to do some post-processing to the audio used in-game. In this subsection, we give an overview of those technologies.

4.8.1 Unity 5 (Game Engine)

Unity 5 is a cross-platform game engine that is used to develop video games for PC, consoles, mobile devices and websites. It has made game development universally accessible and thus, created a new wave of independently produced titles. Additionally, since it has a large, helpful and responsive community and we already had experience with it, we decided that it was the best option and used it to develop our prototypes.

In Unity (seen of Figure 10), every game must have at least one scene that contains game objects that may interact with each other. This game objects, are essentially containers that support components. When we select a game object in the scene, we can see in the inspector which components it contains. While unity provides access to many useful native components (e.g.: cameras, text labels, etc.), games always require some degree of personalization, which can be achieved with scripting. Scripts are behavior components that can be added to game objects, and are the cogs of the larger wheel that is a game. An overview of every script/class can be found in the Prototyping chapter.

It is also important to mention that Unity 5 has a multiplayer system (UNET), which would allow us to implement a server-client architecture to view remotely in the PC what was happening in-game in the smartphone. UNET has a High level scripting API (HLAPI) with some very useful features that allow us to:

- Control the networked state of the game using a Newtork Manager.

- Send and receive network messages.
- Send networked commands from clients to servers.
- Send networked events from servers to clients.

Unity Engine and Editor integrate UNET by providing the following:

- A NetworkIdentity component for networked objects.
- A NetworkBehaviour for networked scripts.
- Configurable automatic synchronization of object transforms.
- Automatic synchronization of script variables.
- Support for placing networked objects in Unity scenes.
- Network components

The existence of UNET was essential because it allowed us to save weeks in development time.

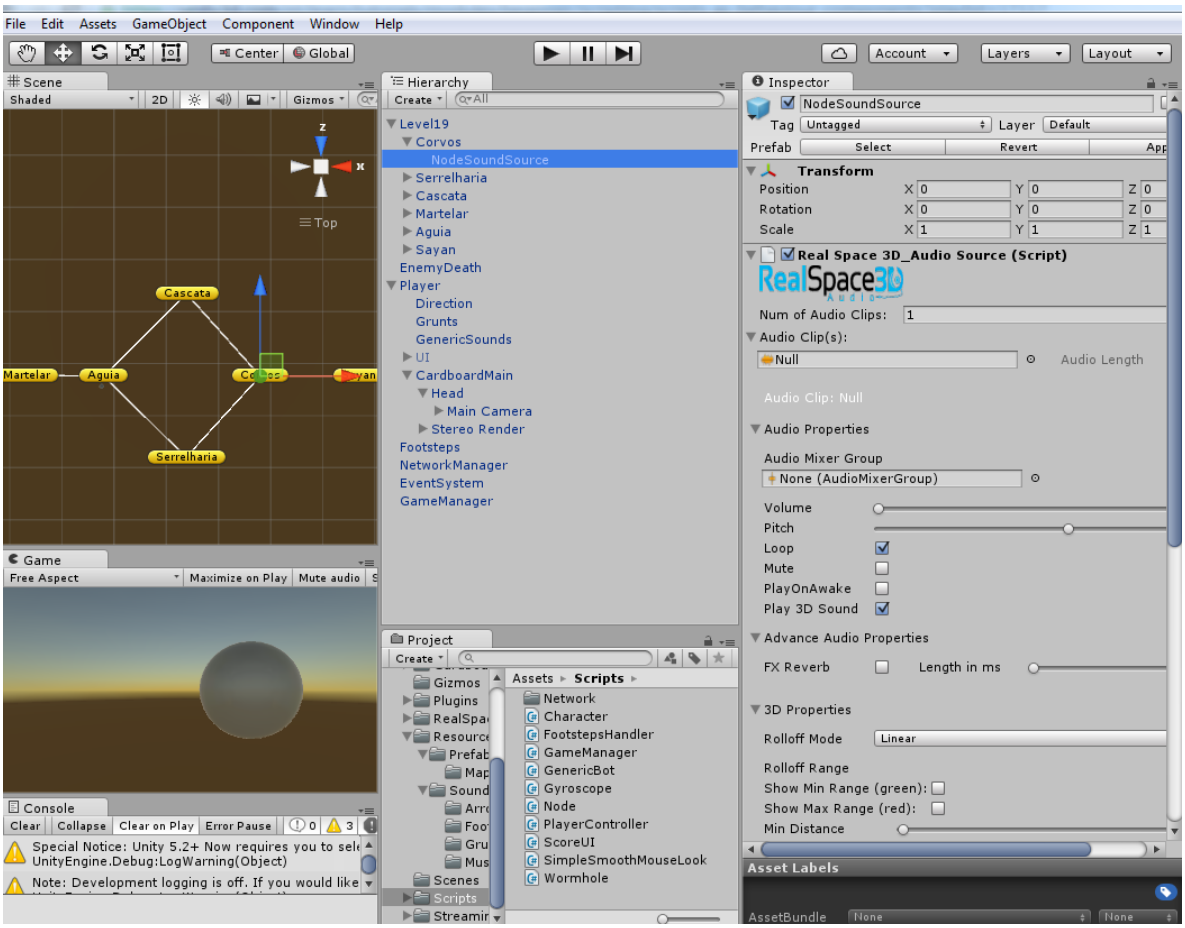

Figure 10-FPS project opened in Unity 5

4.8.2 RealSpace3D (Sound Engine)

The sound engine is extremely important in audio-only games because realism is necessary to promote immersive experiences. Additionally, certain sound characteristics can be used to carry information to the players, which is essential when using sonification. In particular, sounds should carry the information of their position relative to the listeners' head, which can be achieved using a head-related transfer function (HRTF).

A HRTF is a transfer function that describes the modifications that sound suffers due to the shape of the listener's outer ear, the physiognomy of its head and body, the acoustic characteristics of the space in which the sound is played, etc. It can be used to synthesize binaural sounds that accurately inform the listener where that sound is coming from in three dimensions – in range (distance) and in direction (above, below, front, back, or either side) – which requires the use of headphones.

Since we wanted to use binaural sounds to facilitate navigation and allow the players to accurately aim against enemies, we researched several sound engines that support HRTF. The results are detailed in Table 14.

Name	Realistic	Open-source	Unity support
OpenAL Soft	No	Yes	No
3Dception	No	No	Yes
RealSpace3D	Yes	No	Yes
Rapture3D	Yes	No	No

Table 14-Researched sound engines

The most important criterion was the realism each sound engine could provide. After testing them, we concluded that, in our opinion, the most realistic engine was RealSpace3D. Since it was proprietary, we contacted VisiSonics Corporation and talked with their CEO, which was kind enough to provide us with a development license, allowing us to use any number of sound sources instead of just four, for a renewable period of three months.

4.8.3 Cardboard SDK (Controller)

Since we wanted to create a prototype that wouldn't require expensive hardware like Oculus Rift or Samsung Gear VR, we initially decided to use a smartphone strapped to a headband or the headphones to serve as a head-tracking device.

In the initial phase of development of the first prototype, we developed a script that could be attached to a Game Object to control its orientation by using the Unity API that gives access to the gyroscope and accelerometer sensors. It allowed us head-tracking, thus mapping the physical orientation of the smartphone to the orientation of the PC head. However, we noticed that occasionally, the readings of the Yaw Axis would become unstable. For instance, the smartphone would be rotating at a constant pace while the readings showed rapid/slow rotation that did not make sense. We concluded that it had to do with lack of calibration and researched ways to solve that issue. We were unable to find a reliable solution at first, but eventually, we found Google Cardboard SDK (Software Development Kit), which is used to simplify the development of Android and Unity VR apps by providing head-tracking, binocular rendering and magnetic input. Its biggest advantage is the automatic calibration of the sensors and corrections that help stabilize their output and eliminate the drift that naturally occurs over time.

Eventually, we also decided that our solution to strap the smartphone to the headphone was not very solid, because rapid head movements, especially when looking up or down, would make the smartphone slide on the head. Consequently, we purchased a Google Cardboard (see Figure 11), which is a VR viewer in which a smartphone can be coupled.

Figure 11-Google Cardboard VR headset

Despite all the corrections made by the Cardboard SDK to the gyroscope output, we were still experiencing drift that was noticeable after rotating the head a bit. In other words, after resetting the Cardboard, making the physical North direction correspond exactly to the North in the virtual world, the player would rotate his head a bit, and afterward, the north of the virtual world would not be the same as the north in the virtual world. This meant that we had to fix that problem, or the player wouldn't be able to use external stimuli to know which direction he was facing in the virtual world. The solution, we later found, included the use of the smartphone's compass

The smartphone compass is a magnetometer, used to measure the Earth's magnetic field. It has the same problem of the gyroscope, which is inaccurate readings that render it useless to map the rotation on the Yaw Axis from the physical world to the virtual world. However, even though that mapping is not possible, when the smartphone is rotated to a certain direction, its compass output is always the same, with a negligible deviation. Knowing that, our solution is to make an initial calibration, by asking the player to face the four cardinal directions and saving the compass output for each one. Then, after 15 seconds, which is usually enough time to create noticeable drift, if the player rotates to one of the cardinal directions, a reset is made and the PC is rotated to the corresponding cardinal direction in the virtual world, correcting the drift.

4.8.4 Audacity (Audio editor)

Most audio clips used in-game needed some kind of post-processing before being imported to the Unity project. To do that, we used Audacity, a free open source digital audio editor and recording computer software application. The most used features are the following:

1. **Amplify:** This effect is used to change the volume of the selected audio track by changing the peak amplitude in decibels. This feature was used extensively to make all audio clips have similar loudness.
2. **Change Pitch:** This effect is used to modify the pitch of the audio track, without changing its tempo. This means that we can modify the perceived frequency of sound, without changing the speed/pace of the track. This was useful to create the audio clips that would inform the player about a resource in cooldown, by creating a low pitched version of the original audio clip.
3. **Change Tempo:** This effect is used to modify the tempo of the audio track, without changing its pitch. This was useful to fix the duration of the normal reload sound to 2 seconds and the fast reload sound to 1 second.
4. **Fade In:** This effect is used to make a transition from absolute silence to the original amplitude of the audio track.
5. **Fade Out:** This effect is used to make a transition from the original amplitude of the audio track to absolute silence. Both Fade In and Fade Out were particularly useful during the creation of audio loops.
6. **Reverse:** This effect is used to make the selected audio track play backwards. It was useful to create "seamless" audio loop.

5 Prototyping

Once we felt that the game design was mature enough and had a good idea of how most features would be implemented, the development phase begun. In this chapter, we explain what was done in each sprint and detail the architecture of the FPS, complementing it with an overview of all the classes we coded.

5.1 Activities developed

In this subsection we describe the activities developed chronologically, along with the rationale that guided our decisions along the second semester.

5.1.1 Milestone 7 – Implementation of prototype #1

During this milestone, we developed the Gyroscope class that would allow us to control the orientation of the PC using the smartphone's hardware. Our initial solution seemed to work just fine, but it wouldn't be until the development of the FPS that we noticed some drifting problems associated with full rotations on the Yaw Axis. During this milestone, we also coded the Wormhole class, which contained the game logic that allowed the player to control the velocity of the moving sound source, as well as a scoring system based on periodic assessments of the angle between the sound source and the player's facing direction. Before the end of the first semester (and summer break) we made the first preliminary playtesting session with two users. This made us realize that following a sound in constant movement resulted in considerable lag and poor accuracy when the speed of the sound source was increased.

5.1.2 Sprint 1 – Changes to racing game and development of the FPS

We started the second semester by prioritizing the work and realized that before polishing the first prototype, we should initiate the development of the FPS game, which was the one that offered most potential for research and gave us the chance to implement and test the HCI and sonification techniques that

we proposed to solve navigation problems in audio-only games. Furthermore, the high complexity of the planned prototype resulted in implementation risks (technical issues, prolonged developing time, etc.) that we wanted to suppress as soon as possible.

We needed controllers to move the PC inside the level, which requires the collision system. Consequently, these dependencies forced us to develop three main features simultaneously, which occurred during the first month: level builder, system collision and PC/Mobile controllers.

The level builder uses the Unity Scene View Window to allow the visualization of nodes and their connections. Furthermore, the inspector exposes some options that can be edited: chamber type, spawnable or not, refresh time and adjacent nodes. This was achieved by coding the Node class and proved itself very useful to quickly visualize and edit map topologies. It was also during this phase that we started to collect sounds that would convey intuitively the type of each chamber to the player, which was an iterative process that involved brainstorming with at least 10 persons. We then used Audacity to post-process most of the sounds we chose, which allowed us to create sound loops, normalize loudness and experiment which sound properties were more suitable to represent a chamber in cooldown.

The system collision, used by the player controller, was coded to allow movement in 3D because we wanted to explore sound both horizontally and vertically through binaural audio, supported by Realspace3D. In other words, players would be able to move and strafe in the horizontal, like in regular videogame FPSs, but also change their altitude. Our intention was to create an interesting dynamic, where players could be above or below each other, forcing them to turn their head in 360º to aim, which would add flavor to the game. It also allowed the creation of maps with more than one level, slopes and vertical corridors. This was never explored in playtesting because we knew that it would increase the difficulty of the gameplay and wanted to keep it simple. However, our implementation gives us the possibility to experiment that in future versions

without much hassle. Instead, the version used during playtesting didn't allow the players to change the altitude of the PC, which meant that all the chambers were in the same horizontal plane.

The player controller was initially coded to only support PC builds, using the input of keyboard to move the player character and the mouse to control its orientation. Later on, when the level builder and collision system reached an acceptable state of maturity, we used the Gyroscope class that we coded during the development of the racing game to also support Mobile builds. At this point, we were finally able to do a preliminary playtesting and assess if the proposed HCI and sonification techniques would actually work. The results were promising, but exposed some problems that we hadn't foreseen. These complications were all related with the difficulty of the players to hear certain sounds and turn to their direction, something that was essential for navigation. This difficulties were exposed in the following situations:

- Player was in a chamber which had adjacent chambers with similar sounds.
- Player was in a chamber with more than three adjacent chambers.
- Player was in a chamber with three adjacent chambers which had a small angle between them.

Initially, all neutral chambers used different parakeets sounds because we wanted to use the lighthearted aesthetics of a forest to suppress some of the discomfort of sensory deprivation. Nonetheless, whenever the player was in a situation where he could listen to two different parakeets sounds simultaneously, he couldn't tell the difference between them. This made us realize that we needed sounds semantically equivalent, but different enough for the player to hear them distinctly, which is why we ended up using different bird sounds to represent neutral chambers.

The other two problems weren't quite as easy to solve and, at the same time, influenced heavily the level design. We saw the similarities to Fitts's Law and decided that we should approach those problems systematically. However,

at that stage, the time allocated for the development of the FPS had already been used and we didn't even had implemented the FPS mechanics and static AI bot. For that reason, we decided to invest all our efforts in the FPS and dropped the Rhythm game. We also committed to the task of creating a formal evaluation that explored the aforementioned problems in a systematic way, and that proved itself very useful in future level design activities. Furthermore, since we had already noticed that the hearing acuity was affecting results, we decided to change the racing game so that it could be used to assess the ability of players to look for sounds, allowing us to relativize the results in the formal evaluation.

During the rest of October, we coded the Generic Bot class, putting the methods and attributes that were shared with Player Controller class in the Character class, from which they could be extended. We also implemented the Game Manager and Footsteps Handler classes, integrating all the planned features and gameplay logic on schedule.

5.1.3 Sprint 2 – Playtesting and integration of feedback

As we prepared to start the preliminary playtesting in November, we noticed that the Gyroscope would not behave consistently when rotating in the yaw axis, a problem that was not acceptable. Since we didn't find any definitive solution, we imported the Google Cardboard SDK to our project and solved that problem with apparent success. However, eventually we noticed that the PC orientation would sometimes drift after a while. A description of this problem and its solution is discussed in the Technology subsection.

After having a working prototype, we decided to get two users to playtest it, which revealed several lacking features and some others that needed fixing or refinement. The results can be consulted with more detail in the Evaluation chapter.

Finally, we became aware of the necessity to visualize what was happening in-game during these playtesting sessions, instead of relying on logging and the description of the players, whose interpretations were subjective.

5.1.4 Sprint 3 – Implementation of spectator mode and formal evaluations

We started the implementation of the spectator mode in the last week of November and it took us two weeks to complete this task, which was more time than we had planned. However, by visualizing the actions of the player in real-time, we were able to analyze formal evaluations in a more objective way, because we were able to give tasks to the players and see how they performed them. The results are detailed in the Evaluation chapter.

5.1.5 Sprint 4 – Writing of the final report

This last sprint was initiated in the third week of December and took a month to be completed. During this time, we did some more formal evaluations that gave us useful data for the final conclusions.

5.2 Architecture

The inherent complexity of software often requires thoughtful planning. To the process of documenting a system by describing all of its relevant elements and their relations, along with the structural high level design decisions, we call Software Architecture. The necessity of a good document tends to increase along with the complexity of a project, but the overhead of good documentation can prevent the necessity of costly modifications later in development. Furthermore, it also allows a good communication between all the stakeholders, which encourages the exchange of ideas and reasoning.

As we already said videogames differ from software in many aspects, but ultimately, their biggest difference is their purpose. Therefore, videogames may need much more iterations before reaching a state of maturity that results in the intended game experience. This was the case with our prototype, which corroborated the idea that following an agile methodology based on Scrum was the right decision.

In Figure 12, we can see the architecture of our project, with all the game objects of a level and their respective children, as well as the relations between them, which will then be detailed.

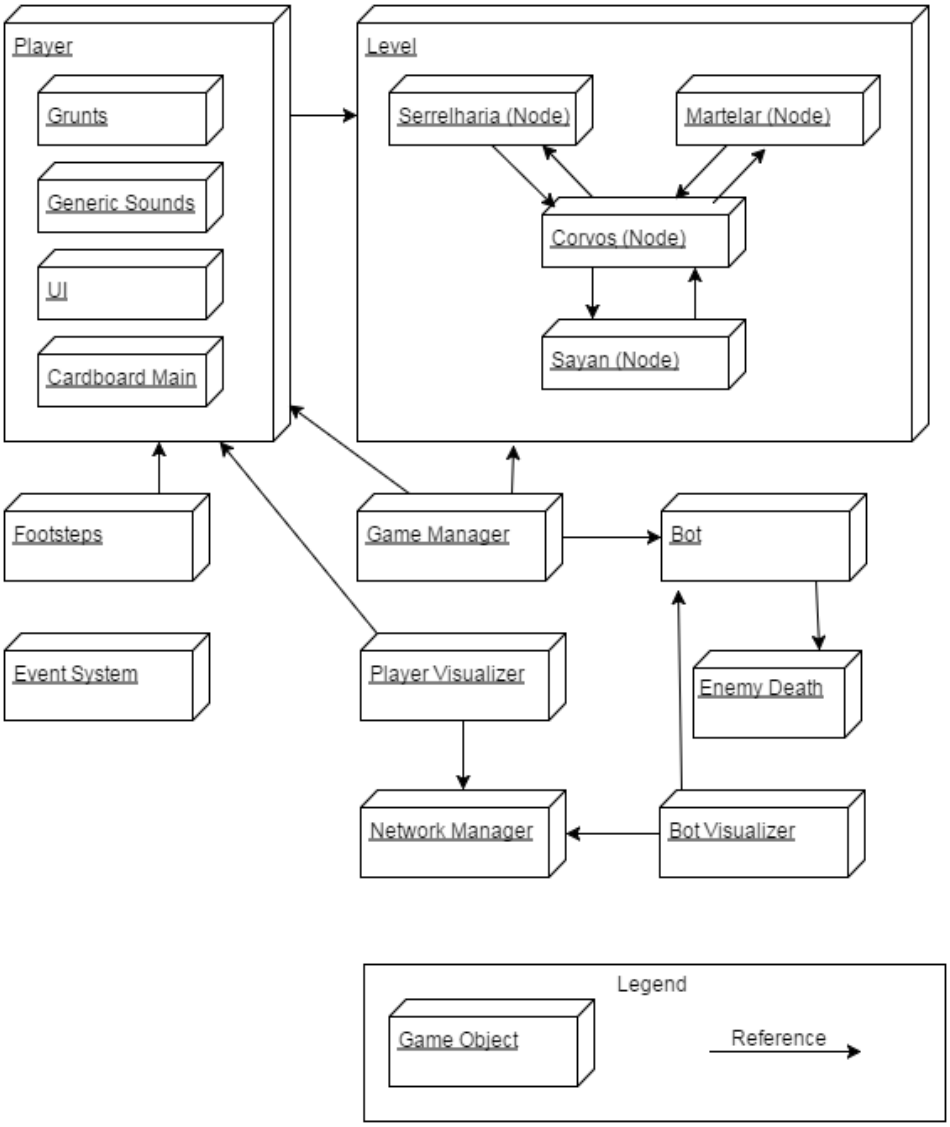

Figure 12-Dynamic View Architecture Diagram

- **Level:** Every Level is stored as a prefab that contains all the nodes and their connections, and can be instantiated at runtime. This means that there is no need to change the entire scene when changing levels, which would imply the destruction and creation of many game objects. By changing the Level Container instead of the whole scene, we managed to decrease loading times from 15 seconds average to less than 1 second.

- **Node:** Each Node represents a Chamber of a certain type and even though it's not represented in the diagram above, each Node contains a game object (as a child) which is the "NodeSoundSource". During the Level loading, each Node loads an audio clip to the respective "NodeSoundSource", based on its type (e.g.: the Health Chamber loads the sound of a waterfall). Finally, every node contains a list of adjacent nodes and it is possible to visualize a line when two nodes are correctly connected in the scene editor. It is also possible to define in the inspector if the node is spawnable and its refresh time.
- **Player:** This game object represents the PC inside the level. It has several components: an audio listener, that serves as the "ingame microphones", the Player Controller, which is used to control the PC inside each level, and Simple Smooth Mouse Controller, used during development to look around with the mouse.
- **Generic Sounds:** This game object has an audio source that contains all the sounds that don't have any kind of directionality and are played "inside" the player's head.
- **Grunts:** All the Grunting sounds are stored in an audio source belonging to this game object. Unlike Generic Sounds, Grunting Sounds have directionality and are useful to inform the player about not only the damage he is taking, but also the direction of the enemy that just shot him. This is done by repositioning this object around the player.
- **UI:** This game object is used only for debugging purposes, allowing the rendering of text to be read while using the Cardboard.
- **Cardboard Main:** This game object is an instantiation of Cardboard Main prefab, which was imported from Google Cardboard SDK. Among many other features, it cleans the smartphone's gyroscope output data, corrects drift and applies a shader to distort the rendered image to be

visualized through the bifocal lenses of the cardboard. For this reasons, it was used to allow head-tracking.

- **Footsteps:** This game object has a sound source with all the footstep audio clips and the Footsteps Handler script that is responsible for managing the playing of sounds. To allow a more immersive approach, the script also positions this game object in a way that simulates sound coming from the feet of the player, requiring a reference to the Player to get its position.
- **Game Manager:** This object contains the Game Manager class.
- **Bot:** This object has an audio source that contains all the audio clips played by the Bot. It also has the Generic Bot script running, which controls the bot behavior and has a reference to reposition the Enemy Death object.
- **Enemy Death:** This game object contains an audio source with an audio clip that plays when the Bot is killed by the player. We created it so we could respawn the Bot in another location, and have the player hear a sound informing him of the location of the Bot's death.
- **Player Visualizer:** This game object is used during playtesting to visualize remotely the PC position. It has a reference to the Player object to change its position accordingly to the Network Manager.
- **Bot Visualizer:** Similarly to the Player Visualizer, this game object is used to visualize the Bot remotely.
- **Network Manager:** This game object contains two native multiplayer components: Network Manager and Network Manager HUD. They were used to implement the client-server architecture and expose a ready-to-use GUI.
- **Event System:** This game object has an Event System component that enables the management of events coming from the Cardboard Main object.

5.3 Class Reference

It is necessary a more detailed explanation about all the scripts we coded in order to better understand how the whole system works.

- **Character:** This class contains all the properties and methods that are common to the Player Controller and Generic Bot classes.
- **Player Controller:** This class handles the player's input on the keyboard that controls the movement of the PC. It also contains all the logic for the collision system, preventing the player to walk through the walls, defined by chambers and corridors, along with damage dealing heuristic.
- **Generic Bot:** This class contains the logic that defines the basic Bot behavior, which includes his movement and damage dealing heuristic.
- **Game Manager:** This class is responsible for loading levels, spawn characters and manage input that is not related to gameplay: volume cardboard reset, current level, aiming aid and presence of bot.
- **Gyroscope:** This class uses the gyroscope API to control the orientation of a target game object.
- **Footsteps Handler:** This class handles the reproduction of the footsteps in a realistic manner.
- **Score UI:** This class is used to render some information used for debugging, like the player's score.
- **Simple Smooth Mouse Controller:** This class allows the player to control the camera vertically and horizontally using a mouse. It was used during development for PC builds.
- **Wormhole:** This class handles the logic of the first game prototype.
- **Node:** This class is responsible for loading the proper sounds and control the position and volume of the respective sound sources.
- **Player Sync:** This class is attached to the Player Visualizer prefab, which is automatically spawned when a client connects to a server. Then, in the server, the Player game object position is updated,

mimicking the position of the PC in the client. This class is also used to control certain variables from the server.

- **Bot Sync:** This class works similarly to the Player Sync. It is attached to the Bot Visualizer prefab and is used in the server to visualize the bot's position in the client.

6 Evaluations

In this chapter we detail the playtest methodologies we used. Additionally, we present and analyze the results from preliminary and formal playtesting.

6.1 Playtest methodologies

The exploratory nature of this dissertation implied many design risks, because we couldn't know *a priori* if the proposed HCI and sonification techniques would work as expected, or if the experience that emerged from the players' participation would be consistent with the intended participation. For that reason, we knew that many iterations would be needed, which should be done in a methodic and informed way.

The Valve's approach to Game Design and Playtesting seemed very appropriate to this dissertation, because it promotes iteration through the creation of a feedback loop between game design and playtesting (Figure 13).

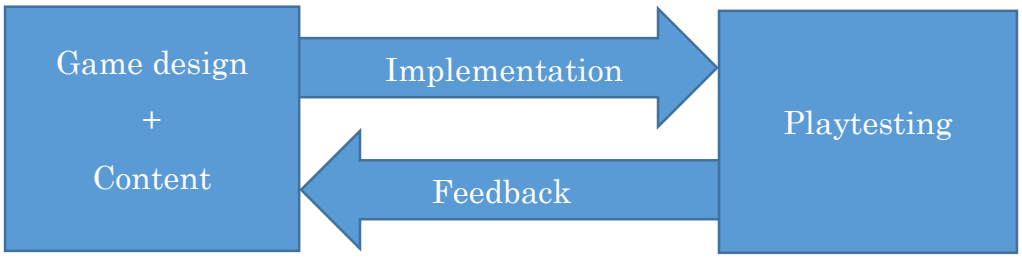

Figure 13-Feedback loop between design and playtest

The research of this methodology (and others) was important, not only for us to understand which techniques exist, but also to learn their advantages and disadvantages and in which situations they are best suited.

In "Valve's approach to playtesting: the application of empiricism" [23], the authors talk about some pertinent aspects of playtesting. In particular, they explain the difference between traditional and technical methodologies. The first one is more qualitative and usually resorts to:

- **Direct observation:** Watch the player play the game.

- **Verbal reports:** The player thinks aloud while playing the game.
- **Q&A:** Structured querying of playtesters (e.g.: interviews).

Traditional playtest methodologies are best suited to get a feel of what the actual player's participation looks like: which dynamics and behaviors emerge, what the players feel, what gameplay, navigation and content issues exist etc. Even though this kind of feedback can be very valuable, it must be used with caution because it usually requires some interpretation.

When there is need for more objective and measured approaches, technical playtest methods are more suitable:

- **Stat Collection & Data Analysis:** Logging of actions and events during gameplay.
- **Design Experiments:** Hypothesis testing and following validation with data.
- **Surveys:** Set of standardized questions with player categorization.
- **Physiological Measurements:** Analysis of body responses to stimuli.

This methodology allows the game design to evolve by using empirical data that can be collected in bigger scale (e.g.: logging). However, even though it can be used to tackle specific questions in a more methodic and objective way, it does not provide an insightful perspective to gameplay like traditional methods do.

We used traditional playtesting methods such as direct observation, verbal reports and Q&A, to evaluate the players' participation with the games produced, and later those findings were synthesized using the two studied game design models. On the other hand, technical playtest methods such as stat collection & data analysis and surveys were used to evaluate objectively the hearing acuity of players and how the number of sound sources and the angle between them influence their performance.

6.2 Preliminary playtesting and feedback

Having planned features facilitated the whole development process because it allowed us goals to have goals to achieve. However, in game development, the sole implementation of features does not guarantee a gaming experience that meets the intentions of the game designer and thus, there is need for playtesting to get feedback from the players. In the following subsections we present the results of the feedback of two testers using the two studied game design models. Additionally, we will discuss several game design details that were made relevant during interviews.

6.2.1 Modified racing game

During the preliminary playtesting of the FPS, we realized that the hearing acuity of a player has great impact on his performance when audio is the only source of feedback. For this reason, it was expected that people with lower hearing acuity would not do so well in the formal tests, which suggested that we needed a metric to relativize results. Because of this, we decided that the racing game could be used as a way to gauge each of the player's hearing acuity before starting the formal tests.

In order to do so, we changed the gameplay mechanics to make the scoring depend exclusively on the ability of the player to follow a sound, using time as a metric for success. The results are synthesized in Table 15.

	Intention	Artifact	Participation
Challenge	The player has to follow a moving sound source the best he can, moving his head.	The sound source is a music playing on loop. When the player manages to look directly into it (i.e.: less than 5 degrees), he listens to an Achievement Sound and the sound source moves to a new, "random" location and stops. Once the player manages to stare at it again, it moves again. This is repeated 30 times, and a time record is printed on screen. The seed is fixed, so that the moving pattern of the sound source is always the same.	During preliminary playtesting we realized that players who moved their head in a more paced way performed better. It also seemed apparent that players who closed their eyes had their performance enhanced. Finally, every player was much better at recognizing sound source movements in the Yaw Axis (horizontal) than in the Pitch Axis (vertical).
Embodiment	The player gets feedback from the movement of his own body, navigating more easily in the level.	The game uses the output of the gyroscope to rotate the PC in-game.	Player always knows which cardinal direction he is facing, and controls the PC rotation speed because it is mapped directly to the head-tracking device (Cardboard).

Table 15-Forms of participation of the modified racing game

During the formal evaluations, we were able to contextualize the testers' performance by recording the time needed to complete this game. However, some players were able to come up with a "strategy" that facilitated the pinpointing of sounds. That strategy could be divided in two simple phases:

1. Pinpoint the sound source in the Yaw Axis (horizontally).
2. Pinpoint the sound source in the Pitch Axis (vertically).

Since the sound source position changed mostly horizontally, this strategy minimized the time needed to locate it, which might have biased the results.

6.2.2 First-Person Shooter game

When all the main features (except spectator mode) were implemented, we finally started the preliminary playtesting of the FPS with two participants. This was our first external source of feedback and it ended up providing itself very useful to detect issues with navigation, gameplay, level design, lack of sonic feedback, etc. We also used this opportunity to assess the player's reaction to the aesthetics of the game and to see if the sound semantics could convey the information with success. The results were synthesized in Table 16.

	Intention	Artifact	Participation
Challenge	Memorize the level and navigate in it in a thoughtful way, shooting the enemies and creating strategies to improve survivability.	Each level is represented as a graph: nodes are chambers and edges are corridors. Some chambers can have resources that are represented by specific sounds. Those resources are: health, ammo, shield and temporary firing speed. They can be picked, and when that happens, they reappear 30 seconds later. There is also a bot, navigating in the level, trying to kill the PC. His behavior simulates, in a basic way, the behavior of a human player.	Both players understood easily the navigation system. They also found the aesthetics/theme adequate. However, resource sounds semantics were not intuitive. Both players were unable to memorize the whole map due to its size. Having two Health and Ammo nodes, represented by the same sound, made them confused because they thought there was only one of each. One of the players had difficulty in pinpointing the exact direction of audio sources. Both were unable to tell the nature of the bot's attacks (i.e.: melee or ranged). Sonic feedback was also lacking, as player couldn't tell when they were running against a wall or picked resources. They also felt uniformed about the remaining health and ammo.

Table 16-Forms of participation of the FPS

Verbal reports and Q&A exposed many issues that made us realize that there were still adjustments to be made and that we definitely would need to visualize what was happening in-game during formal evaluations, so that our

direct observations could become more informative. Each of the issues, their priority and our implemented solutions are detailed in Table 17.

Issue	Solution	Priority
No way to visualize what was happening in-game	Implement a server-client architecture to visualize the PC and Bot in real-time.	Very High
The map was too big to be memorized in such a short time, especially by players who don't have experience with audio-only games.	Create a smaller map to be used during formal evaluations.	High
Lack of information about remaining ammo	The player hears a voice when he reaches 10, 5 and 0 arrows.	High
Lack of information about remaining health	Player starts to hear a heavy breathing/panting sound when he is below 50 health.	High
Resources semantics were ambiguous.	We continued proposing alternatives until we found more suitable sounds to represent resources.	Medium
Lack of sonic feedback when collecting resources.	An achievement sound plays when the player collects a resource. A better alternative would be a different sound playing depending on the resource collected. For instance, the player could hear a relieved sigh after collecting health.	Medium
Lack of sonic feedback when running/walking against a corridor/chamber wall.	Player hears footsteps whenever the PC moves. A better alternative would be to play the footsteps in a way that informs the player about the speed he is moving. For instance, if the PC is walking right against a wall, his movement will be really slow, and footsteps playing speed should be too.	Medium
Players' were unable to tell if the bot attacks were melee or ranged.	Our proposed solution is the implementation of realistic projectiles, which inform the player of oncoming attacks and allow the implementation of a dodge mechanic.	Low

Table 17-Solutions to problems revealed from preliminary playtesting

All the issues with high or very high priority were solved, but some of the lower priority ones could not be fixed in time. However, we gave precedence to the issues that had easy-to-implement solutions.

6.3 Formal playtesting

After the issues revealed during preliminary playtesting were fixed and we could visualize remotely and in real-time what was happening in-game, we were finally able to start formal playtesting (Figure 14).

Figure 14-Playtesting with a participant

We were curious if certain participants' characteristics could in anyway influence their performance. For that reason, we created 3 groups: sighted male, sighted female and blind participants. All the sighted participants play videogames regularly, including shooters. The blind participant never had contact with any kind of electronic game.

Every evaluation followed the same structure and participants were given the same instructions before each phase to guarantee unbiased results. It is also important to refer that our results should not be used to confirm or refute hypotheses, since the participants' population is not large enough to be representative. Instead, we hoped that some patterns emerged from the results, which allowed us to propose hypothesis to be tested in future evaluations.

In the first phase, participants played the modified racing game, which requires them to follow a sound source that moves 30 times. The output was the time they could complete this challenge, in seconds. The results are shown in Table 18.

Sighted					**Blind**
Male			**Female**		**Female**
A	**B**	**C**	**D**	**E**	**F**
166	165	144	179	136	60

Table 18-Time (in seconds) each participant took to complete the modified racing game

Intuition made us expect that the blind participant would perform better. What we did not expect though, was such a large difference between the performances of the two groups. During Q&A, we were also told by participants that they found it easier to perceive horizontal movements than vertical movements of the sound source.

In the second phase, we evaluated how the angle between sound sources affected the participants' performance. Before starting, we told the participants which chamber sound they should navigate to (e.g.: waterfall, crows, sayan, etc). To do that, they had to start every test scenario facing north and walk forward until they reached the chamber at the end of the corridor. Then, they would hear two new adjacent chambers, and would need to walk towards the chamber sound we told them in the beginning of the test. Each test scenario had different angles: 90º in front, 45º in front and 30º turn left/right (see Figure 15). For each angle, participants had to perform the task three times, always with different sounds to make sure they couldn't memorize the previously executed actions.

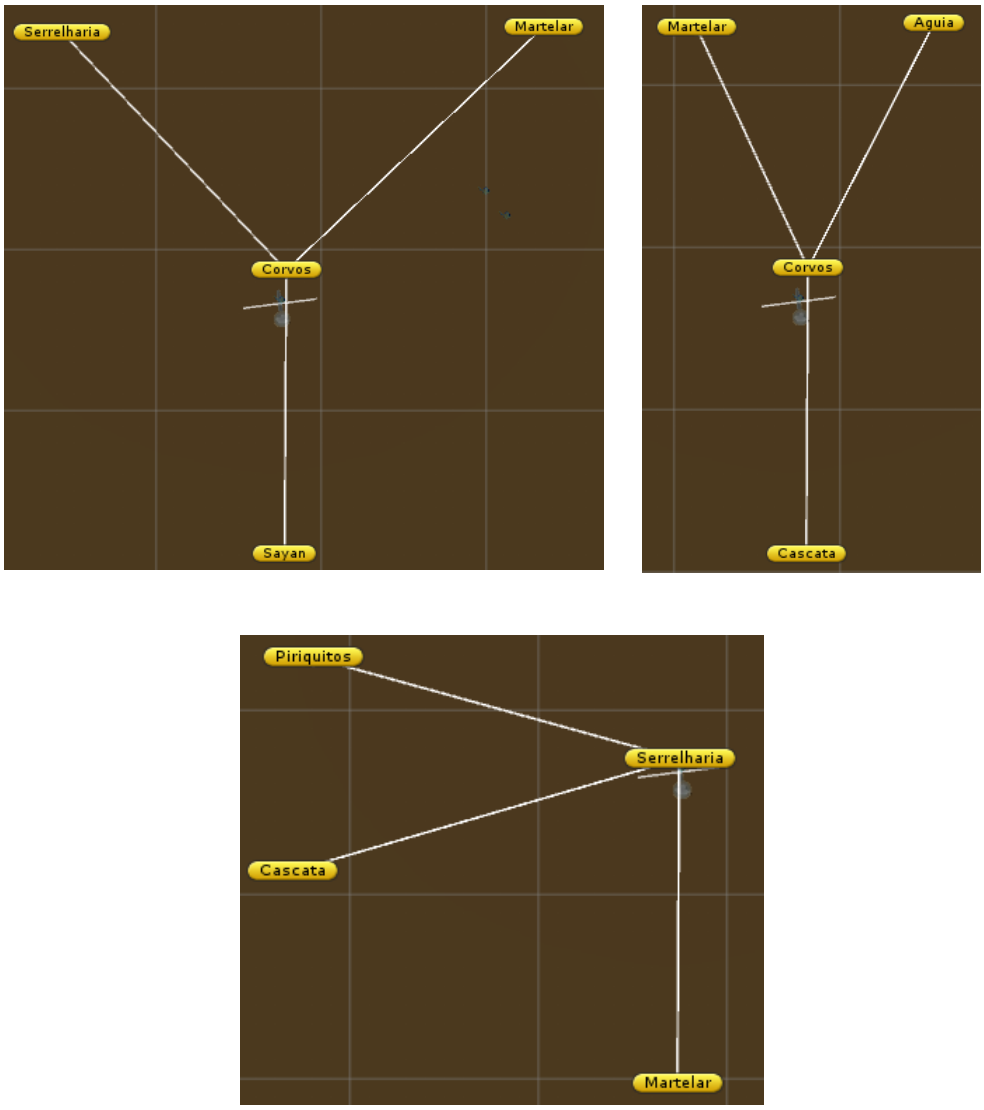

Figure 15-Test scenario for 90º, 45º in front and 30º turn left

While they did this, we observed and evaluated their performance according the criteria in Table 19, which was used throughout all the test scenarios.

Difficulty performing task	Score
Very easy (t<15 seconds)	5
Easy (15<=t<25 seconds)	4
Moderate (25<=t<35 seconds)	3
Hard (only on second try)	2
Unsuccessful (failed twice)	1

Table 19-Evaluation criteria

The results are detailed in Table 20 and show that the only circumstances where players had some degree of difficulty was in the third test scenario. In particular, sighted female participants performed worse than the other groups.

	Sighted					Blind
	Male			Female		Female
Level	A	B	C	D	E	F
90º	5	5	5	5	5	5
	5	5	5	4	5	5
	5	5	5	4	5	5
45º	5	5	5	5	5	5
	5	5	5	5	5	5
	5	5	5	5	5	5
30º turn	5	5	4	5	5	5
	5	5	5	2	4	4
	4	5	5	3	3	5

Table 20-Results of evaluation during phase 2

The third phase of tests used the same evaluation criteria, but this time, the task was to rotate and walk towards the chamber sound we asked. The test scenarios had 4, 5 and 6 adjacent sound sources and can be viewed in Figure 16.

The results, which can be seen in Table 21, are a bit unexpected. We found no correlation with the number of sound sources and the difficulty the players had to find a specific chamber sound. Instead, their performance depended more on the loudness/timbre of the sound they had to reach. For instance the easiest sound to recognize for every participant was the waterfall. We suspect that the noisy nature of that sound, which is accentuated by its constant presence, makes it more recognizable. On the other hand, sounds with intervals of silence were more difficult to recognize. In particular, one of the sighted female participants was not able to hear the super sayan sound at all.

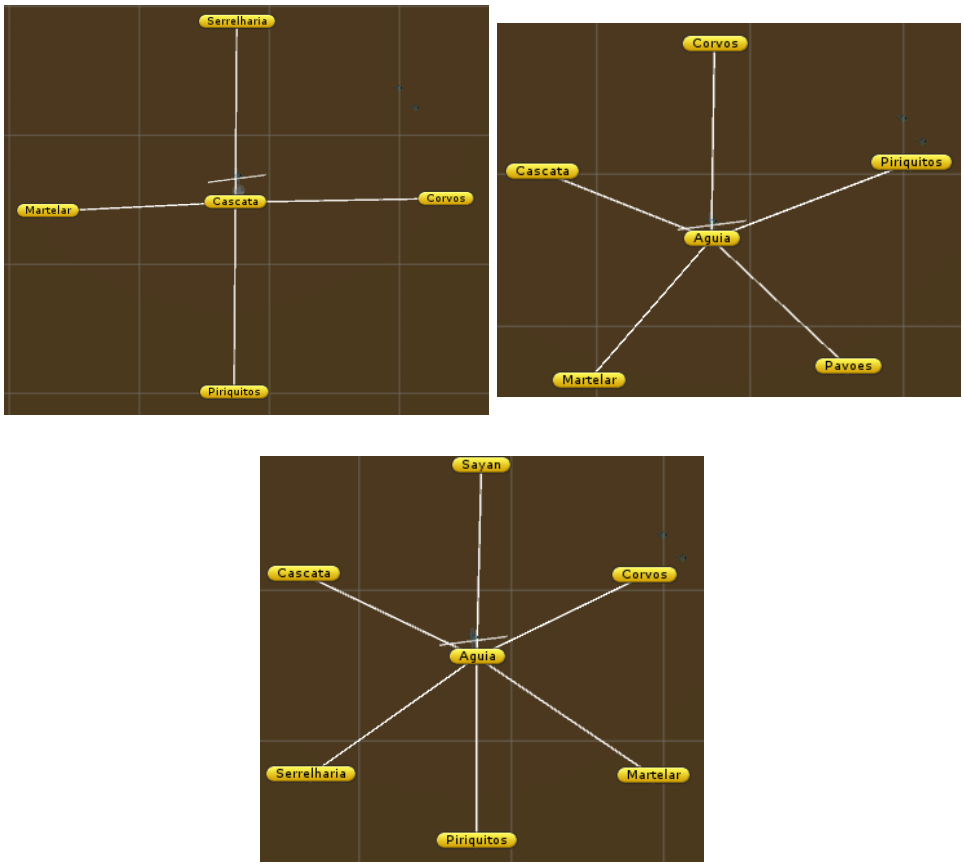

Figure 16-Test scenario with 4, 5 and six adjacent chambers

	Sighted					Blind
	Male			Female		Female
Level	A	B	C	D	E	F
4	5	5	5	4	3	5
	5	2	5	4	5	5
	5	5	5	5	5	3
5	5	4	5	4	5	4
	4	1	5	5	5	5
	5	2	5	5	5	5
6	4	4	5	4	5	5
	5	4	5	4	5	5
	5	5	5	5	5	5

Table 21-Results of evaluation during phase 3

The fourth phase of the formal playtesting was actually playing the FPS. By this time, players were already familiarized with all the chamber sounds and how sonification worked.

Each participant was given 15 minutes to explore the level without enemies. After those 15 minutes, we asked them to draw the topology of the map and make the correspondence between chamber sounds and resources. An example of a case of success, where the participant was able to memorize correctly all the chambers, their connections and position relative to each other can be seen in Figure 17.

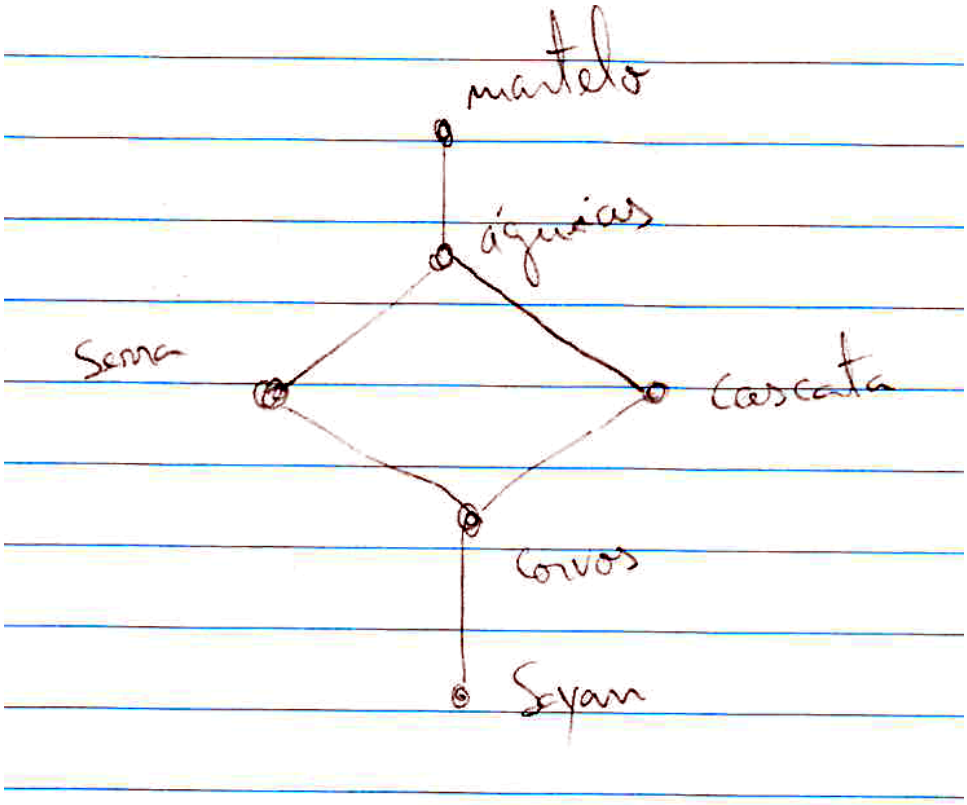

Figure 17-Participant memorized the topology with success

An example of partial success, where the participant was able to memorize correctly all the chambers and their connections, but not the angle between them, can be seen in Figure 18.

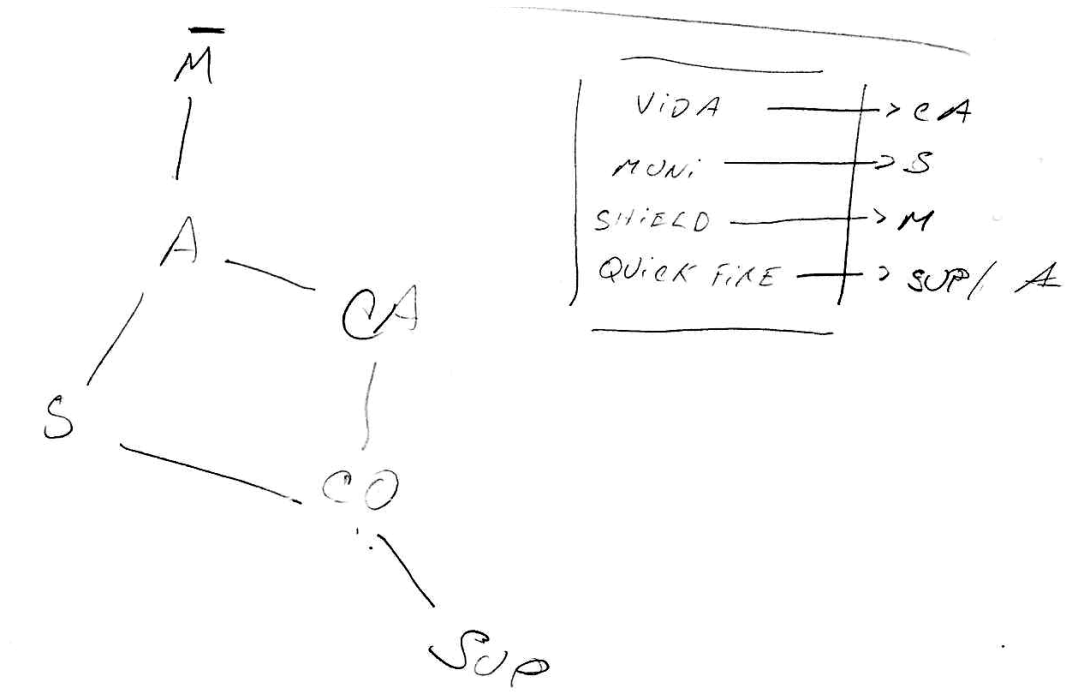

Figure 18-Participant memorized the topology with partial success

Finally, an example of failure, where the participant was unable to memorize correctly the connections of chambers neither their position relative to each other, can be seen in Figure 19.

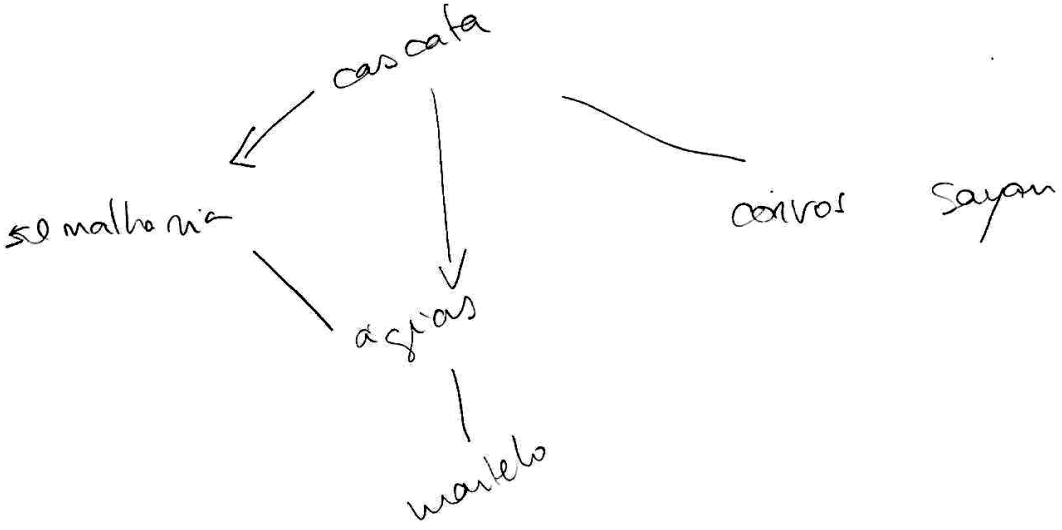

Figure 19- Participant was unable to memorize the topology with success

After this phase, all participants (except the blind one that got a description instead) were shown the actual topology of the map, shown on Figure 20.

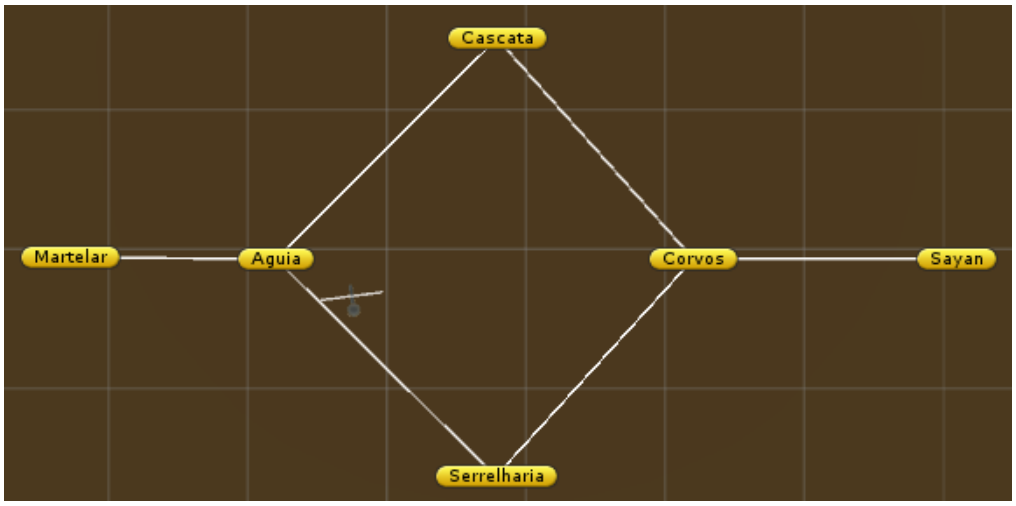

Figure 20-Topology of FPS map

We also explained the actual mapping between chamber sounds and resources, followed by an explanation of the FPS rules. Then, we let the participants play for about 10 minutes against the bot. The results of the drawings, mappings and final score against the bot are shown in Table 22.

	Sighted					Blind
	Male			Female		Female
	A	B	C	D	E	F
Drawings	Success	Success	Partial Success	Failure	Failure	Failure
Sound mapping	Success	Success	Success	Success	Failure	Failure
Score (Kills-Deaths)	7-4	12-9	8-3	6-2	6-8	4-6

Table 22-Results of phase 4 of playtesting

Since we didn't have a representative amount of people of each group playing the game, we couldn't confirm or refute hypothesis. However, our intuition told us the following:

1. Blind participants should memorize and draw/describe the topology more easily than sighted participants, regardless of genre, because that is a skill they use on a regular basis to navigate independently.
2. Participants who have experience with videogames regularly should perform better at playing the shooter than those who don't (blind participants).

The results showed that all male players were able to memorize and draw the topology with a high degree of success. All female players were unable to succeed in this task, which might indicate that this is a skill inherent to gender. It also shows that blind people aren't necessary better at memorizing sounds. This was unexpected by us, but was corroborated during Q&A by our blind participant, who had trouble describing the connections of nodes.

Additionally, all male participants and one female participant could correspond correctly chamber sounds to resources. It is worth mentioning that both participants who failed, did so by not recognizing the super sayan sound from dragon ball series and swapping it with ammo or shield. All players associated correctly health to the waterfall.

Furthermore, all males were able to win against the bot, with one participant being able to navigate with such ease that he was able to come up with a winning strategy: after each fight, collect health and collect fire speed boost, getting ammo only when needed. Finally, one sighted female player also performed very well, but the other was defeated. The blind participant was defeated, but we would need much more tests to know if the lack of experience with electronics games, actually influences the outcome.

This evaluations made us realize that there are many variables that might influence players' performance. Hearing sensitivity and acuity varies significantly, and consequently sounds are heard differently. Furthermore, the nature of the shooter genre might also influence the results, giving advantage to players who have more experience with shooting videogames. However, all the participants told us that they never had played a game that forced them to focus so much on audio, which in theory should make previous experience less relevant.

Finally, we were unable to see any correlation with hearing acuity and ability to play an audio-only shooter. Once again, more evaluations should be conducted to validate or refute the proposed hypothesis.

7 Future work

Throughout this dissertation, we continuously refined our problem definitions and proposed solutions, always guided by the motivations we set in the beginning. However, some of the modifications were made because of time and technology constraints, and even though we completed successfully all the research objectives, we feel there is still room for improvement of our implemented solutions.

In this chapter we present the improvements and corrections to research objectives during the development of the project. Furthermore, we propose additional features and share our vision for the future of the audio-only shooter.

7.1 Objective refinement

The exploratory nature of this dissertation forced us to adopt an agile methodology similar to Scrum, which allowed us to continually adapt and refine the proposed design and implementation solutions. In this subsection, we talk about several key moments and important decisions made along the way.

In the beginning of this project, we wanted to make an audio-only exploration game with high production value. This would require the investment of much of the development time learning about sound design, technologies and software, and since this dissertation is in the context of a Master's Degree in Informatics Engineering, more specifically in Software Engineering, we wanted to avoid spending most of our time with unrelated areas.

We started by researching and playtesting several audio-only games to learn more about the genre. That output was integrated in the State of the Art and was crucial to make us realize that the best way to explore this genre was not by developing one game with high production value, but instead, several game prototypes that had the core mechanics implemented, from which most of dynamics and aesthetics could emerge.

To that end, we idealized three game prototypes. Their game design had in consideration our requisite to explore different forms of participation. In particular, we wanted to explore the embodiment dimension because it showed much potential to create interesting mechanics. We saw two opportunities: embodiment as physical involvement and embodiment as physical performance.

When blind people navigate, they use feedback from many sources to help them perform that task: the material of the ground, its inclination and even certain characteristics of the nearby sound sources around (e.g.: room size influences greatly the way sound arrives at the eardrum). They also use their own footsteps as a source of information, counting them to gauge distances and always knowing how much they rotate. This made us realize that we could actually use our own body to create an input-output feedback that could help the player navigate in a level. At this point, we proposed HCI and sonification techniques that could solve the navigation problems that we found during playtesting of several games. The FPS showed itself the genre that better allowed us to test these techniques and at the same time, promoted the dimensions we wanted to explore, which is why we proposed its development.

The second opportunity, embodiment as physical performance, showed itself when we realized the lack of audio-only games that explore the body movements to create interesting gameplay experiences. There are many Rhythm videogames where players have to dance and play instruments, and it makes sense that this genre also is accessible to the blind community. With this in mind, we proposed an audio-only Rhythm game where interpretation of movements of the players was gamified.

Since the distilled mechanic of the rhythm game proposal was following a moving sound while wearing a head-tracking device, we decided that we could also employ this mechanic in the game design of a racing game with almost no extra effort. Additionally, the proposed game design of the racing game was so simple to implement that it would allow us to use it as a preliminary prototype,

or proof of concept. Consequently, the racing game ended up being the first to be developed.

The second key moment of this dissertation was the reprioritization of sprints in the beginning of the second semester. We decided that it was more important to test the proposed techniques in a game that promoted interesting forms of participation such as the FPS and thus, started to develop it and put the Rhythm game on hold. The audio-only shooter proved itself a bigger challenge than we expected, but at the same time, showed us an opportunity that we hadn't seen yet: research how the number of sound sources and their angle around a listener's influence the time he takes to make a decision. By designing test scenarios and employing a methodic approach, this research could prove itself valuable to level design of audio-only games and even the HCI field.

This marked the third key moment of the whole project, where, after some debate, we decided to drop the rhythm game and focus on designing and preparing test scenarios that would let us collect data to be analyzed, from which we could draw some conclusions.

Finally, after the preliminary playtesting phase, we were in a position where we could say confidently that verbal reports would not suffice and direct observations would be needed to draw more informed conclusions. Consequently, we had to invest some of the development time designing and implementing a client-server architecture to visualize remotely what was happening in-game.

7.2 Additional features and future usage

We feel that the FPS prototype served its purpose, since it created an interesting game experience that was enjoyable by both blind and sighted groups, and, at the same time, allowed us to test the proposed HCI and sonification techniques to solve navigation issues. However, we cannot forget that it is still a prototype and that implies that there is still much room for

improvement until it reaches a level of maturity that makes it worth being distributed to the community.

Due to time constraints, we had to do some modifications that were not part of the intended game design and some of the feedback we got during preliminary playtesting could not be implemented on time. For instance, it would be helpful to integrate more intuitive achievement sounds for when the player collects resources. We also think the game would benefit immensely from an expert sound designer, which certainly could contribute to a more diverse and interesting soundscape.

Furthermore, it would also be interesting to create bigger maps. This would be a pretext to implement different footstep layers that would be associated with different regions of the map to inform the player of his whereabouts. This was not implemented because early feedback made us realize that the duration of the playtests was not enough for players to memorize so many sound sources. Additionally, we propose that adding verticality to the level design would contribute to create a more challenging, but at the same time, diverse experience.

Another obvious improvement would be to include a sociability dimension in the game. The FPS genre is very suitable to multiplayer and different game modes: team deathmatch, capture the flag, dominion, etc. We envision a game that can be played by blind and sighted players, who must communicate with each other to create tactics that improve their chances of winning. This would break the barrier that currently exists to blind players, currently confined to game genres that are not prone to sociability.

Finally, it is apparent the need for further testing and evaluation using traditional and technical methodologies. It would be interesting to validate or refute our hypothesis, creating bigger and more diverse groups from where we could draw data to be analyzed, and in our opinion, future audio-only games and the HCI field would benefit from such a study.

8 Conclusions

In this chapter we reflect upon all the work developed along this dissertation and mention the contributions that emerged from it.

8.1 Reflections and Contributions

We started this dissertation by emphasizing the positive impact that games can have in people's lives. However, blind people have been discriminated by the game industry and that motivated us to explore the audio-only game genre, which can be played by blinded and sighted audiences, as a way to improve game accessibility.

We contextualized the reader by explaining how the audio-only genre has evolved throughout the years. Then, we argued that Game Design Models empower game designers with tools that help them iterate game proposals, allowing the process of evaluation to be done in a methodic, rationale and structured way. We also explained the rationale behind the choice we made when we selected the Participation-Centric Model of Gameplay Experience and the MDA – Mechanics, Dynamics and Aesthetics – framework to evaluate several games.

This explanation was followed by several game analysis that were made using the aforementioned game design models. Since there is a panoply of audio-only games and we were time constrained, we had to choose four that we found representative of the genre. The main criterion we used to decide which games would be evaluated was the presence of diverse mechanics, dynamics and aesthetics that promoted interesting forms of participation. Those game analysis allowed us to learn about common practices in the audio-only genre and were useful to make us realize one recurrent problem: navigation. Then, we argued that navigation is an essential mechanic that should not be discarded from audio-only games and summarized the problems we detected, which facilitated the process of designing solutions later on.

Additionally, we discoursed on the usefulness of fields like HCI and sonification. This fields allowed us to extract knowledge that ended up being integrated in the artifacts we developed, facilitating interactions and using sound as a carrier of information. We concluded the State of the art with a review of the available hardware that the mobile platform offers.

Once we had our problem definition, we were able to outline several research objectives. The first was to develop audio-only games that explored different forms of participation. The second was to design and implement several HCI and sonification techniques that could solve the navigation issues we detected early on. After some thought, we proposed the development of an audio-only FPS that allowed us to fulfill these two research objectives simultaneously and a racing game that would later be used in formal evaluations to measure the participants hearing acuity. Additionally, during the development of the FPS, we also added one important research objective that could contribute to the audio-only genre and HCI: evaluate how the number of sound sources and their angle around a listener influence his performance.

The intelligence of the proposed HCI techniques lies on the use of the player's own body to create an input-output feedback that gives him control and information during navigation. This idea mimics the actual navigation of blind people in the real world. These techniques, synergized with sonification that informs the player in a thoughtful way, were all integrated in the game design of the FPS.

After we completed the FPS game prototype, we did preliminary playtesting using traditional methodologies, consisting in verbal reports and Q&A. It exposed issues with gameplay that were corrected in the following weeks, and made us notice that direct observations should be used in later evaluations to draw more information from the tests.

Formal playtesting was then conducted, which included traditional and technical methodologies. We devised several test scenarios, with different number of sound sources and angles between them, and created three groups of

participants: sighted males, sighted females and blind female. All the sighted participants play videogames regularly, including shooters. The blind participant had no experience with electronic games. The number of participants was small, so we were unable to confirm or refute hypothesis. However, formal playtesting results allowed us to propose several hypothesis that should be evaluated with a more representative population.

Hearing acuity was measured using the time to complete the modified racing game as a criterion. Results show that the performance of the blind participant was much better than all the sighted participants.

In the next phase of tests, the results showed that males performed better at creating mental maps than females. It also showed that the blind participant had some difficulty memorizing the map, which indicates that experience navigating using exclusively sound in real life may not translate to the experience that emerges when navigating in our test scenarios. From this results we hypothesize that gender might influence the ability to create and memorize mental maps.

Finally, some players adapted themselves very well to the way navigation is performed, while others didn't. We also didn't see any correlation between hearing acuity and performance when playing the FPS against the bot.

We think that navigation can, in fact, be used as a mechanic in audio-only games, not only in shooters, but also in exploratory genres such as RPGs. We hope that this document and the developed audio-only first-person shooter may be of use to game designers and feel like our goals were achieved. Game accessibility is now richer, and we expect with optimism to see the audio-only genre flourish in the future.

References

[1] J. Gaudiosi, "Mobile game revenues will overtake console games in 2015. Here's why. - Fortune." [Online]. Available: http://fortune.com/2015/01/15/mobile-console-game-revenues-2015/. [Accessed: 20-May-2015].

[2] "Free Online Educational Games for Kids - Education.com." [Online]. Available: http://www.education.com/games/educational/. [Accessed: 02-Jul-2015].

[3] "X-Plane 10 Global | The World's Most Advanced Flight Simulator | X-Plane.com." [Online]. Available: http://www.x-plane.com/desktop/home/. [Accessed: 02-Jul-2015].

[4] "The Science Behind Foldit | Foldit." [Online]. Available: https://fold.it/portal/info/about. [Accessed: 02-Jul-2015].

[5] S. J. Kirsh, "The effects of violent video games on adolescents: The overlooked influence of development," *Aggress. Violent Behav.*, vol. 8, no. 4, pp. 377–389, 2003.

[6] P. Barr, J. Noble, and R. Biddle, "Video Game Values: Play as Human–Computer Interaction," *Interact. Comput.*, vol. 19, no. 2, pp. 180–195, 2007.

[7] "Manifesto for Agile Software Development." [Online]. Available: http://www.agilemanifesto.org/. [Accessed: 03-Jul-2015].

[8] "AudioGames, your resource for audiogames, games for the blind, games for the visually impaired!" [Online]. Available: http://www.audiogames.net/list-games/. [Accessed: 05-Jul-2015].

[9] K. Bierre, M. Hinn, T. Martin, and M. McIntosh, "Accessibility in Games: Motivations and Approaches," *… Pap. Int. Game …*, 2004.

[10] A. Lucero, J. Holopainen, E. Ollila, R. Suomela, and E. Karapanos, "The playful experiences (PLEX) framework as a guide for expert evaluation," *Proc. 6th Int. Conf. Des. Pleasurable Prod. Interfaces - DPPI '13*, p. 221, 2013.

[11] S. M. Grünvogel, "Game Studies 0501: Formal Models and Game Design by Stefan M. Grünvogel." [Online]. Available: http://www.gamestudies.org/0501/gruenvogel/. [Accessed: 11-Dec-2015].

[12] L. Pereira and L. Roque, "Understanding the Videogame Medium through Perspectives of Participation," *Authors Digit. Games Res. Assoc.*, pp. 1–15, 2013.

[13] R. Hunicke, M. LeBlanc, and R. Zubek, "MDA: A Formal Approach to Game Design and Game Research," *Work. Challenges Game AI*, pp. 1–4, 2004.

[14] M. LeBlanc, "The collected game design rants of Marc LeBlanc." [Online]. Available: http://8kindsoffun.com/. [Accessed: 31-May-2015].

[15] "SoundInGames.com - Sound Design in Games." [Online]. Available: http://www.soundingames.com/. [Accessed: 06-Jul-2015].

[16] V. Alves, "Design Patterns in Games : the case for Sound Design."

[17] D. Pires, B. Furtado, and T. Carregã, "The blindfold soundscape game: a case for participation-centered gameplay experience design and evaluation," *… 8th Audio Most. …*, 2013.

[18] J. M. Carroll, "Human Computer Interaction - brief intro," *Encycl. Human-Computer Interact. 2nd Ed.*, 2014.

[19] "Rockstar Games: Grand Theft Auto San Andreas." [Online]. Available: http://www.rockstargames.com/sanandreas/. [Accessed: 29-Jun-2015].

[20] Wikipedia, "Microsoft PowerPoint." [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_PowerPoint. [Accessed: 29-Jun-2015].

[21] T. Hermann, "Taxonomy and Definitions for Sonification and Auditory Display," *Icad.Org*, pp. 1–8, 2008.

[22] "MoSCoW Prioritisation | DSDM CONSORTiUM." [Online]. Available: http://www.dsdm.org/content/10-moscow-prioritisation. [Accessed: 24-Jan-2016].

[23] "Valve's Approach to Playtesting: The Application of Empiricism." [Online]. Available: http://www.valvesoftware.com/publications/2009/GDC2009_ValvesApproachToPlaytesting.pdf. [Accessed: 21-Jan-2016].