POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Merging augmented reality and virtual reality

A multiplayer experience between different platforms combined

Supervisors
Prof. Andrea Sanna
Dr. Federico Manuri
Dr. Francesco De Pace

Iñigo Lerga Valencia
266940

07 2019

Written in LaTeX on July 22, 2019

Programming today is a race between software engineers striving to build bigger and better idiot-proof programs, and the Universe trying to produce bigger and better idiots. So far, the Universe is winning.

Rich Cook

Contents

List of Figures

List of Tables

Abstract

Augmented and virtual reality are separate growing technologies that have started to impact areas such as video games or 4.0 industry. The aim of this thesis is to create a seamless experience between them, focused on a video game perspective to show the real capabilities of both platforms working together, in a multiplayer first-person shooter.

To achieve this goal, there will be a development in Unity, the video game development platform, using the standard libraries such as Mixed Reality Toolkit for augmented reality (AR) and SteamVR for virtual reality (VR). Additionally, the physical devices that will execute the experience will be the HoloLens from Microsoft (AR device) and the Oculus Rift from Oculus (VR device).

The combination of platforms will need a networking library, as every communication will be via Wi-Fi. The chosen library is UNET from Unity itself. It is being deprecated but it is still working until Unity releases the new networking alternative.

Based on the experience and lessons learned during this development, in this document will remain the different alternatives and choices made, as well as the limitations found in involved technologies.

Keywords: augmented reality, virtual reality, mixed reality, Microsoft HoloLens, Oculus Rift

Acknowledgements

I would like to thank:
Prof. Andrea Sanna and Dr. Federico Manuri
for allowing me to explore such an interesting
field, as well as lending me the needed devices
and providing advice.
Dr. Francisco de Pace for helping with crucial
parts in the development and testing of this project.
And finally, my family and friends for supporting me all along.

Chapter 1

Introduction

A video game is a non-physical type of game, in which the user interacts with a computing device via a controller, getting in return a collection of images and videos displayed to them. Currently, video game developing companies have become an increasingly important part of the entertainment industry.

This kind of games can be found for almost every computing platform that is sold today, being virtual reality devices a particularly rising one. In the case of augmented reality, there are no commercial devices yet, the only products available are focused in research and industry environments. For this reason, the amount of games available for augmented reality headset is almost null. Although AR games for mobile phones are starting to appear in the public.

Taking all of this into account, there is not a single commercial video game with support for both platforms at the same time. This master's thesis purpose is to develop a simple but functional video game that achieves these requirements under the use of mixed reality software available, with the theme of a first-person shooter.

Also called hybrid reality, mixed reality is the combination of virtual and augmented reality. This allows the development and use of new spaces in which humans, real and virtual objects interact.

Using these technologies, the goal of this development is to develop an immersive environment capable of working in a virtual platform, as well as in an augmented reality one, making the experience between both as similar as possible. They use a common map with an additional three-dimensional map of the space for the AR system, that makes possible the artificial occlusion of objects. The use of controllers is needed for both platforms and the players are themed as drones.

The motivation behind this project is to analyze the impact of user experience and game usability that a system with the mentioned requirements will have, to investigate if this field is worth studying in a future work and even to develop commercial products that take advantage of the positive aspects of both platforms.

This thesis includes testing with real users whose results will be discussed in the corresponding chapter. They are asked to fulfill a survey after trying each platform following the System Usability Scale guidance. After that, there will be a statistical analysis of the information gathered and the conclusions obtained from them, following a T-test distribution.

An extended section for each platform involved will be attached in the following chapters. From this chapter of the document until the end can be found a chapter with further information about the platforms involved, a state of the art in chapter 2, specifications of the devices involved (third chapter), the implementation itself in the next chapter as well as the problems found during the development. Moreover, the final chapters will contain an explanation of the user experience and tests made with real users, exposing the results and conclusions to finalize.

Chapter 2

Platforms

For the development of this thesis, the main fundamental pillars are the virtual and augmented reality platforms, as well as the networking library of Unity (UNET), which is an integral part of the bridge between one and another platform. Here below can be found further information about the three of them for better comprehension of the project.

2.1 Virtual Reality

Commonly known as VR, virtual reality is an immersive environment in which the user can interact with non-real objects as if they were. These kinds of experiences need of some specific hardware such as headsets, controllers, sensors to track the movement, gloves etc.

This technology is based in two factors: the user's real environment and the virtual one. Both communicate via the interface that makes the headset and sensors work together. For this reason, the user movement of a controller is translated into the movement of his virtual hand. Moreover, the virtual world can also manifest in the real one as sounds, vibrations etc. With this kind of feedback, the user's experience is complete and more realistic.

Figure 2.1: Example of a virtual reality kit being used by user

The experiences with these environments can be immersive or semi immersive. In the first case, a computing device creates a three-dimensional space that tries to be as realistic as possible, to create the feeling of reality to the user. The second option uses the same technologies but does not try to feel real to the user, it is more focused in industrial applications and tools. Moreover, it does not need hardware apart from a computer.

2.1.1 History

Virtual reality, as we know it today, was originated in the late 20th century when, in 1968, Ivan Sutherland created the first head-mounted display system with the help of students such as Bob Sproull. The device was primitive and so heavy that had to be hanged from the ceiling. It received the name of 'The sword of Damocles'.

From that moment until now, this technology as been present in multiple different fields, which some of them will be exposed in the following subsection.

2.1.2 Applications

This platform has been applied to multiple fields such as education, training, medicine, psychotherapy and entertainment amongst others.

Virtual reality for education is starting to become a very useful tool, as it makes possible to experiment things impossible or very expensive and visually learn from them, which is known to be the best way to learn and gain the attention from students. This is due to two factors: It is like watching a video that explains the lecture, but it needs to be interactive, so the students can't just ignore it. The experiences can be adapted to every topic imaginable, history, science, chemistry, math...

Figure 2.2: Example of application of virtual reality as a learning tool for kids

Military groups have found VR to be a very useful tool for training new soldiers, as it is less physically tiring and can be repeated with more frequency than a normal aiming or rescue practice. They even have developed their own system to simulate flights and landings, which are really cost effective, as there is no risk of destroying real equipment, and the environments used are almost true to life, to get the most realistic experience they can get for teaching purposes for pilots.

Medical uses are also the order of the day, as can be used in two main ways: for therapy and in surgery. In the last one, the limit is the sky. Doctors can use virtual reality headsets to immerse themselves into a virtual surgery room with a real model of a patient to see what they are dealing with, or even to learn in simulated scenarios how to proceed in certain occasions, leaving behind the possibility of damaging a patient in the process.

For the first application in the medical field, today there are some treatment plans for certain phobias and mental illnesses that include virtual reality as a tool to face scenarios that could cause stress, fear or a full episode of anxiety to a patient, in a controlled area in which they can be helped and improve in those situations. It is kind of a practice model that is less scary than real life [1].

The last field to be mentioned in this section is entertainment, which is the one that makes more money globally speaking. Video game companies that have expanded their circles to new projects with virtual reality headsets are growing as this thesis is being developed. There are big titles such as 'Beat Saber', 'Job Simulator' or 'Minecraft VR', that get the attention of hundreds of thousands of users.

Figure 2.3: 'Job Simulator' video game in action

2.1.3 Devices

There are three main types of devices for virtual reality consumption. Their main differences are:

- Mobile phones with a lens system attached: the amount of applications for this category is reduced due to the power these devices have, but simple titles can run without problem and are perfect for 360[a] media consumption.
- Individual headsets with processor and graphical power inside: they are like the last ones, but more expensive due to the limited space in which they must pack everything needed for a comfortable experience running the latest games.
- Headset connected to a PC: the most common approach, as the headset is lightweight and acts only as display, this separates the heat away from the

user even though limits the distance it can have between the computer and the headset itself. It depends completely in the graphical power of the computer it is connected to.

2.2 Augmented Reality

Augmented reality, also known as AR, is the term used to describe the technologies that allow a user to see virtual objects, sometimes in form of holograms, interacting in the real environment surrounding the user. Some of the experiences possible with this platform even interact with the real environment itself. All of this happens in real time for the user.

This technology is based in two factors as well as the previous explained. These are: user's real environment and the virtual one. However, these two interact in a different way, as in this platform there is no need to use a controller for the movement of the user in the virtual space. The real movement made by the user in the real world, like walking for example, is translated in movement in the virtual space. The holograms react to this readjusting their position according to the new user's location.

Because of the nature of this technology, it usually creates a feeling of immersion that is more natural than with a VR headset.

Figure 2.4: Augmented Reality example: Microsoft HoloLens

2.2.1 History

This is certainly a younger technology, as it was started between 1960 and 1990, but it was not until 2010 that it took off, when big companies like Sony, with 'The Playroom' in 2013, and Google, with the first AR headset in form of 'glasses', appeared.

The moment it got established to stay was without a doubt when Microsoft released the HoloLens in 2015, the first AR headset capable of interacting with holograms, capable of reaction to the environment and the changes in it.

2.2.2 Applications

The most important fields where augmented reality technologies have been applied are education, tourism, medicine and entertainment. But it is needed to take into consideration that, as said before, it is a new platform, therefore it has not been applied to many fields that will probably in the future.

For starters, in the educational ambit, the possibility to insert holograms in the middle of a classroom for every student to see, becomes a simple anatomy class into a totally new experience. It creates a different feeling than VR does, as it feels less like a game and more like an interactive scenario. With multi user capabilities, every former of the class can interact with a virtual element, being this visible for the rest in real time.

Tourism is a very different field amongst the rest, not being present in the virtual reality examples, that has improved the experiences of knowing new cities or historical places. This is due to the possibilities that a device can offer, superimposing new information or virtual elements with what the eyes are naturally seeing. Visiting important streets can be a lot more enjoyable with a virtual guide that knows what the user is seeing in each instant and reacts to it with information about the interesting place that caught the user's attention. When this technology is applied with a geolocational system, the information that can be provided with internet access is almost endless.

The third field mentioned is medicine. This one is also shared with virtual reality and very similar. Some of the uses are the same ones, even though this one is more focused in surgeon situations, as in the moment of operating a patient, the eyes of the doctor need to be clear to see what he needs to do. In this scene, the additional information that can be shown to the surgeon while looking to the patient can be a real improvement for the result of the intervention. Taking it a little bit further, with some image recognition and processing software and a augmented reality headset, a surgeon could even see in real life, superimposed to the patient open body, where to operate or what parts are critical [2].

At last, but not least, is the entertainment industry field, in which the growth is not as strong as in the previous ones or, at least, not in the same way. To this day, there is no commercial entertainment market for augmented reality in what headset or glasses is referred to. However, in the smartphone and tablet portion, some big games have started to pop up to the public and create huge success. But this part will be commented in the following subsection.

2.2.3 Devices

In augmented reality, there are three main different types of devices, which are:

- Mobile phones: these devices use the integrated cameras in the back to analyse the space behind the screen. With multiple sensors as phones are shipped today, they can map the distances between real objects and put virtual ones in those spaces. However, the virtual objects are only visible in the screen of the device. They don't provide an immersive experience as others do, but they are the most common among the public with games like 'Pokémon Go'.

Figure 2.5: Pokemon Go example of gamemplay with AR capabilities enabled

- Headsets: devices like Microsoft's HoloLens or Metavision's Meta 2 are the most notably in this area. They integrate a full array of sensors and a display that merges with a transparent glass. With this implementation, the user can see through as normal but with the possibility to include holograms or information in his field of view. To this day, these are not open to the public and are only sold to investigators and companies to develop for those platforms.
- Glasses: This one is the less common of all, as they started to be announced in the last few years, but they haven't been sold to a big amount of people yet. The differential factor of this devices is that they are focused on the daily life use. They try to be a substitute for real glasses, that allow to interact with the phone to get certain information like notifications, weather or directions without getting the phone out of the pocket. The most known device of this category is the Google Glasses.

Figure 2.6: Google Glass: Augmented Reality glasses from Google

2.3 Difference Between Virtual Reality and Augmented Reality?

After all the explanation above, the main idea is that virtual reality and augmented reality are two sides of the same coin. Augmented Reality offers the same kind of experience than VR does but with one foot in the real world.

In augmented reality headsets, the computer uses sensors and algorithms to determine the position and orientation of a camera. AR technology then renders the 3D graphics as they would appear from the viewpoint of the camera, superimposing the computer-generated images over a user's view of the real world.

In virtual reality, the computer uses similar sensors and math. However, rather than locating a real camera within a physical environment, the position of the user's eyes is located within the simulated environment. If the user's head turns, the graphics react accordingly. Rather than compositing virtual objects and a real scene, VR technology creates a convincing, interactive world for the user.

2.4 Multiplayer capabilities

In the video games sector, a multiplayer game is one that allows, in any mode, to interact with another user. This interaction can be achieved in a local machine (having two controllers attached to the same console), via short distance technologies such as Bluetooth or with a Wi-Fi connection.

This idea was born from the difficulty that was present decades ago to develop virtual opponents in games. These capabilities do not need to be implemented in every mode of a game, they can be restricted to one without affecting the rest [3].

There are two different approaches as mentioned. These are:

- Unique systems: all the computing is made in one machine and the players interact in three different ways: one after another, both at the same time with an overview of both players in the same screen or with a divided screen for each player.
- Network system: each player has his own computing device that runs the game, but the connection can be made choosing from two options: a local network that works only for the connected players if they are near to each other, and with an on-line connection via the Internet, which allow the players to be in different spots of the planet if they want to.

Chapter 3

State of the art

The research that has been made for this thesis can be divided in three main sections that are the following: virtual reality, augmented reality and hybrid designs.

3.1 Virtual Reality

3.1.1 VR commercial games

At the present time, the amount of commercial virtual reality games is considerable and rising, as new headsets and controllers appear. Some recent video games developed for this platform are, for example, 'Lone Echo' or 'MARVEL Powers United VR'. As virtual reality technology is very mature, the quality and quantity of known titles is considerable and keeps growing.

As for the most known video games developed for this platform, there are quite a few that deserve to be mentioned. For starters, 'Skyrim VR' is a version of the already well-known 'The Elder Scrolls V: Skyrim'. It is an open world role playing game developed by Bethesda Game Studios and released in 2011 [4]. The virtual reality version was released almost six years after that, in 2017.

Figure 3.1: Skyrim VR example of gameplay

The best game in terms of VR experience developed could be 'Superhot VR', as the main factor in favor that it has is an extraordinary controller system. It is a first-person shooter, as the purpose of this thesis, that was developed by Piotr Iwanicki and released in 2017 as the VR adaptation of 'Superhot' that had been released the year before [5].

Figure 3.2: Superhot VR example of gameplay

Lastly, 'Batman: Arkham VR' is one of the few games that was designed work in virtual reality from the first moment, as it was not an adaption of a previous game like the others mentioned. It is an adventure video game developed by Rocksteady Studios and published by Warner Bros in 2016. It had spectacular graphics for that time, and the interaction system was noticeably developed with VR in mind.

Figure 3.3: Batman: Arkham VR example of gameplay

3.1.2 VR scientific applications

This platform can be applied to various scientific fields, but the most important ones are medicine and military. Moreover, these two are often closely related with advances going from one side to the other.

For the medical applications, there is a big amount of studies that claim to get better results in surgeon residents learning to proceed with certain operations. The possibilities that this immersive technology generates allow new surgeons to learn much faster and with less risk for the patients [6, 7]

Figure 3.4: Example of possible scenario in medical training

Moreover, there are more possibilities inside this same field. Therapy in virtual environments is starting to be tested with great results. With these scenarios, a therapist can introduce a patient in a scenario that causes in him a disturbing feeling to be treated. This can be applied to phobias like fear in certain social interactions or even anxiety disorders [8, 9]. It can be a key part in the treatment of mental illnesses that were not improving in the last decades.

In the military side, soldiers can be prepared for the field with virtual scenarios in which they get to practice and be trained with tactics and behaviors they must obtain. This has much less cost and has no risk whatsoever [10].

Figure 3.5: Example of possible scenario in military training

Merging medicine with military, a variant of the previous applications is born. This treatment VR experiences can be applied to soldiers, before, during and after serving. With them, the very common Post Traumatic Stress Disorder can be addressed ad reduced with continuous sessions to improve life quality for them [11].

3.1.3 VR experience problems

Virtual reality, even though it is very mature, is not perfect, as there are still studies trying to solve some of the issues that are still apparent.

The most problematic one is motion sickness. This is a sensation of wooziness that, in this scenario, is caused by the unmatched sensation of movement that the brain feels when the eyes perceive movement (from the images in the headset) and the actual physical position of the body is not changing accordingly. For this reason, there have been various studies that focus on this problem and implemented different versions of movement. Another cause for this feeling of sickness can be a bad display of UI elements, as they are at eye sight, but it is minor and easily avoidable.

One of the possible solutions is 'point and teleport' [12]. There is even a modification that adds the possibility to choose the rotation state while the teleport is made. This kind of movement can be a good candidate to solve the problem, as it does not cause as much motion sickness as other methods.

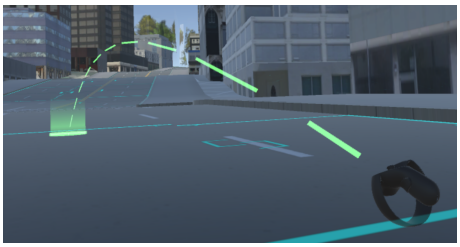

Figure 3.6: Valve's SteamVR point to teleport

There have been studies that compare multiple approaches that support these claims [13]. However, there are others that show the effectiveness of a movement that involves some physical input from the user. An example of this is the arm swing movement [14]. It has been proven that this kind of approach reduces the motion sickness effect, as the brain relates the arm swinging to the eye movement it perceives. With this technology, the experience of virtual reality movement becomes more natural to the human brain than with any other one.

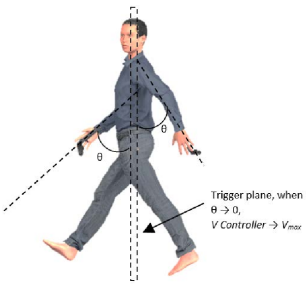

Figure 3.7: Armswing: Using arm swings for accessible and immersive navigation in AR/VR spaces

Another kind of approach is to change how the immersion mechanics work in a game. Instead of introducing the user in a virtual environment where the action takes place, there is the option of creating an in-between scene or environment where the player sees a screen in front of him with the real action. This allows the user's motion sensitivity to understand in a simpler way what is happening and effectively decreases motion sickness [15]. The counterpart of this implementation is that the immersive feeling is lost, but in some developments, it may make more sense to follow this path.

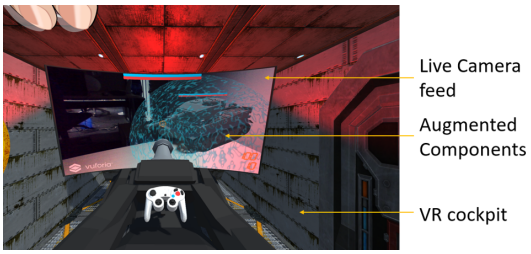

Figure 3.8: Clash tanks: An investigation of virtual and augmented reality gaming experience

3.2 Augmented Reality

However, in the augmented reality field things are drastically different. There are not companies developing commercial games for this dedicated type of platforms, as the main devices from big companies are restricted for investigation and developer use. This causes a total lack of user base to work with.

3.2.1 AR phone games

While it is true that AR games for phones and tablets are all around, they do not use the kind of hardware that is used for this project. For this reason, they cannot be compared in the same level of complexity. Some of the most relevant games in this specific realm are 'Pokemon Go', 'Minecraft Earth' and 'Harry Potter: Wizards Unite'. The last two are as recent as it gets, as they have been announced in 2019, the same year of this thesis.

However, the precursor of this type of mobile games is 'Ingress', which was released in 2012 and got a renewal in 2018 called 'Ingress Prime'. It is a tactical warlike and strategic online game, dependent on GPS positioning to play, as it uses the user's current location and the map around you to ambient the gameplay. The phone's main camera is the principal AR component of the game, as it is used to overlap holograms in the real world and show the result in the screen [16].

Figure 3.9: Ingress on the left and Pokemon Go on the right

Three of the mentioned games were developed by the same company, Niantic (originally a sub-company from Google but separated in 2015), except for the AR version of Minecraft, which is made by Mojang. All Niantic games have followed the same path as Ingress, with very similar technologies but with different ambiances. In terms of shooting games, 'Father.IO AR' was released for smartphones in 2017, using AR in a similar way than the previous mentioned games, and has more than a million downloads in the Google Play Store [17].

3.2.2 AR headset games

The few studies that have developed games for headsets of this platforms have encountered some common problems that are exhibited below. One example of game is 'RoboRaid' from Microsoft, which is pre-installed in the HoloLens. This was a way to express the kind of applications that could be built for this device, to encourage developers and investigation groups to try themselves to develop for this platform.

Nevertheless, the game that is considered almost the father in AR headset games is 'ARquake'. It was developed in 2002 by ID Software, when there was no dedicated commercial hardware for this platform. The game is an augmented reality adaptation of the original Quake. It is a first-person shooter released for PC in 1996. It was a revolutionary game, as it used three-dimensional models instead of bi-dimensional sprites, as well as a real 3D environment to play around [18].

A more modern game is 'HoloRoyale', which is characterized as a break through in terms of size of game area available for players in the same game. It counts with a proprietary system bases on merging previously 3D maps that allows players with HoloLens to play around an entire neighborhood in a multiplayer experience, themed as a sci-fi shooter [19].

Figure 3.10: On the left ARquake example of gameplay, on the right device than runs it

Figure 3.11: Holoroyale gameplay example

3.2.3 AR experience problems

In fact, there are still a lot of things to improve in AR experiences. Starting with the interaction, for the moment there are no controllers suitable apart from simple clickers. This is due to the nature of the environment, and origin innovative solutions as having a physical gun that acts as a controller in a first-person shooter augmented reality game [20, 21, 22].

Another approach for this input problem is developing a hand recognition system, which is much more complicated to do than using a simple clicker, even though it improves the game experience in a significant way [23].

The last input method found for this state of the art in this category has been the use of another device tracking system to show a holographic weapon (AR hologram) superposed to that device in the view of the headset [20]. This one in particular was considered an option but the total development of the project would engross probably too much.

Figure 3.12: Frame obtained from promotional video of Shooter AR

As the aim of this project is themed in a shooter game, there has been a small research in this area. It is true that there are augmented reality games that are first-person shooters too [24], but only for mobile devices or tablets, which do not have one of the main problems of a headset. In those games they show the UI and the weapon directly on the screen, which works just fine for that situation, but with the professional headsets, showing those elements would be intrusive and probably cause motion sickness as they would be directly in the eyes of the player.

Apart from all of this, there are another two main issues in regard to the studies that augmented reality has had in the last few years.

On the one hand, AR relies on holograms as its main system for interaction. These are positioned in the real world, so there is a need to develop an occlusion system. They are usually integrated in the headsets, using an array of sensors (which is the main difference between a mobile phone and a proper headset), even though the computing weight that they put in the processor of the headset is considerable and needs to be controlled. For the HoloLens in particular, the solution for this problem is in the official documentation [25].

On the other, there is a limitation in terms of the distance that can be included in an augmented reality experience. A large-scale scene is not possible for the current devices as they start to get lost in certain conditions as too much light on sensors, drastic velocity changes and large distances to origin for example. To solve this, it is possible to design a proprietary system that join previous maps with specific software and a powerful server side [19].

3.3 Hybrid designs

Taking all of the previously said into account, it is time to enter in the hybrid side of this project. Recent studies have been exposed in this section about each platform, but no word about a study or project that involves both technologies. This is for a reason, until the moment that the research for this project was made, there are almost no studies in this specific field.

One of the few found, is an implementation of a chess game that works as seamless as possible between both platforms [26], but due to the nature of that project, it is not similar enough to be compared with a first-person shooter.

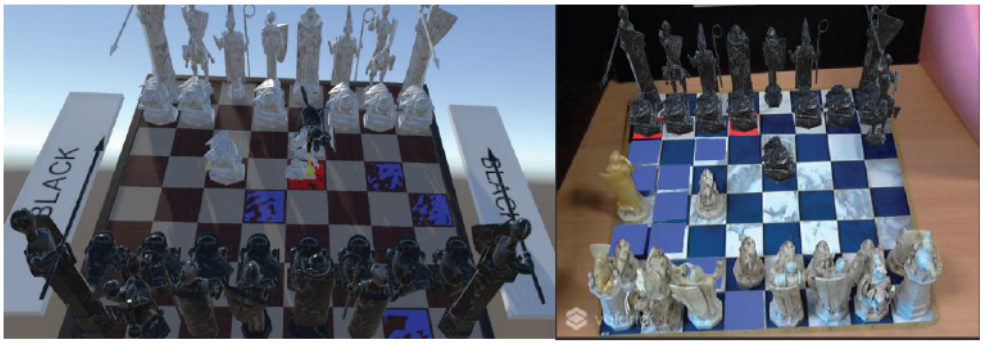

Figure 3.13: Virtual and Augmented Reality Interfaces in Shared Game Environments: A Novel Approach

Chapter 4

Devices

This thesis is, as said before, based in two platforms: virtual and augmented reality. For each one of them has been used a specific device for the implementation side of the project. This chapter includes a description of each, alongside the specifications and software included, as well as the SDK or APIs used in development.

The AR device of choice is the HoloLens, the first version of the headset, as in the middle of the development Microsoft announced the HoloLens 2. For the virtual reality part, Oculus Rift has been the chosen device, even though the Rift S model was available before starting this thesis.

4.1 HoloLens

Microsoft's HoloLens is the first unwired holographic computer, that counts with cutting-edge optics and sensors. This allows the device to provide realistic holograms attached to real spaces around the user.

Figure 4.1: HoloLens' optics configuration

4.1.1 Optics

This system is based on a specific optics configuration to be able to display the holograms in front of the user's eyes, without blocking the natural gaze through the transparent crystal. The result is a 35º window in which the holograms are shown. The components are:

- See-through holographic lenses (waveguides)
- 2 HD 16:9 light engines
- Automatic pupillary distance calibration
- Holographic Resolution: 2.3M total light points
- Holographic Density: >2.5k radians (light points per radian)

4.1.2 Sensors

Apart from a basic camera, this device has some other sensors that allow to have a better understanding of the environment where the user is located at each moment. As to the details in this section:

- 1 IMU
- 4 environment understanding cameras
- 1 depth camera
- 1 2MP photo / HD video camera
- Mixed reality capture
- 4 microphones
- 1 ambient light sensor

Figure 4.2: HoloLens' sensors configuration

4.1.3 Connectivity/Input/Output

In terms of the input and output of the device, it is formed by:

- Built-in speakers
- Audio 3.5mm jack
- Volume up/down
- Brightness up/down
- Power button
- Battery status LEDs
- Wi-Fi 802.11ac
- Micro USB 2.0
- Bluetooth 4.1 LE

4.1.4 Power and weight

- Battery Life of 2-3 hours of active use and up to 2 weeks of standby time
- Fully functional when charging
- Passively cooled (no fans)
- 579 g

4.1.5 Processors and memory

The HoloLens is a very capable device. Taking in consideration the reduced space it works with, it has enough raw power to manage the complex calculations necessary for displaying correctly the holograms. However, it is noticeable that a heavy scene can overload the system and generate some delays sometimes. The specifications in this section are:

- Intel 32-bit architecture with TPM 2.0 support
- Custom-built Microsoft Holographic Processing Unit (HPU 1.0)
- 64 GB Flash
- 2 GB RAM

Figure 4.3: HoloLens' motherboard configuration

4.1.6 OS and Apps pre-installed

- Windows 10
- Windows Store
- Holograms
- Microsoft Edge
- Photos
- Settings
- Windows Feedback
- Calibration
- Learn Gestures

4.1.7 Capabilities

With all the hardware mentioned above [27], the main functions that the device can handle are:

- Spatial sound
- Gaze tracking
- Gesture input
- Voice support
- 3D mapping of spaces in real time

4.1.8 Interaction

Microsoft developed a new way to interact with the device based on gestures. They are recognized by the combination of sensors and operative system and launch the pertinent action. They are the following:

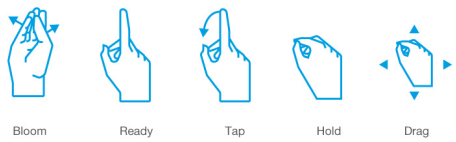

Figure 4.4: HoloLens' recognized gestures

- Bloom: this hand movement deploys the main UI of the HoloLens. It acts as a return home button, where every app and setting are.
- Ready: it represents the moment before a tap or click, like resting the finger in the left click button without pressing it. It is useful for obtaining a clearer tap.
- Tap: comparable to a normal click in a computer. In fact, this device comes with a clicker controller that, via Bluetooth, can be used as a button that executes a tap without the need to perform a gesture.
- Hold: comparable to a right click in a computer. It deploys sub-menus or more options.
- Drag: comparable to a scroll wheel in a normal mouse, or even to a left click held down and move of the cursor in a real computer. Useful to drag windows around or scroll inside them.

These gestures can be replaced also by voice commands, that act in a simple way: When the gaze is pointing at a button, its name pops up and the users says it out loud. Then it is recognized as a click in that button [28].

4.1.9 SDK and libraries

For the development of the project, the only SDK that has been used is the Mixed Reality toolkit that is documented in Microsoft official page [29].

4.2 Oculus Rift

It is one of the most popular virtual reality devices released to the public. When it was launched, it had cutting-edge technology, as it packed a considerable number of features comparing to the competition.

Figure 4.5: Oculus Rift device

4.2.1 Display and optics

The most important part of a VR headset is, without any doubts, the screen, as it is what the user will always see while using it. The experience depends completely on its quality. For measuring that quality, there are some factors involved, which are the following:

- Resolution: 2160×1200 (1080×1200 per eye)
- Pixel Density: 456 pixels per inch
- Screen type: Pentile AMOLED
- Refresh Rate: 90Hz
- Field of View: 110º
- Optics: Hybrid Fresnel lenses
- Interpupillary distance: Variable between 58 - 71 millimeters

4.2.2 Sensors

In comparison to the HoloLens, this device has less sensors, as there is no need for mapping the environment or any type of gesture recognition. Nonetheless, it is true that there are still two relevant sensors for the gaming experience:

- Gyroscope
- Magnetometer
- 6DoF external Constellation camera (external to device)
- 360-degree IR LED head tracking
- Passthrough

4.2.3 Connectivity/Input/Output

In terms of the input and output of the device, it is formed by:

- '3D audio' headphones
- Audio 3.5mm jack
- HDMI 1.3
- USB 3.0 (4-meter headset)
- USB 2.0
- Controllers Oculus Touch with 180-degree front-facing tracking support

4.2.4 Power and weight

- Battery Life of 0 hours, as it always needs to be connected to a PC to work
- Passively cooled (no fans)
- 470 g

4.2.5 Requirements

As this device works in conjunction to a PC, it has some minimum specifications necessary to provide a decent experience. In this subsection will be displayed the actual components used in the development of this thesis in comparison to the requirements for this kit [30].

	Minimum requirements	Actual components
Processor	Intel i3-6100	Intel Xeon E3-1245
RAM	8GB	32GB
Graphics card	AMD Radeon RX 470	Radeon Pro WX 5100
USB ports	1 USB 3.0 and 2 USB 2.0 ports	6 USB 3.0 and 6 USB 2.0 ports

Table 4.1: Requirements comparison between minimum and used

With this comparison it has been clarified that the computer used for the development of this project is more than capable to run the required environments for programming and playing the virtual world for the virtual reality side.

4.2.6 OS, SDK and libraries

The Oculus Rift have no operative system or pre-installed applications, as they work more similarly to a second screen for the computer than a individual device. However, it is needed to install the Oculus software directly from their web page for drivers and general software to set up the device [31].

Moreover, for the integration between the development environment Unity and the Oculus, SteamVR has been used as the compatibility plugin. It is an alternative software for virtual reality headsets, even though it is not made specifically for none of them [32]. It is a more general alternative to allow every VR device to work seamlessly in Steam's store of games.

4.2.7 Interaction

With this device, the main input system for the user is the use of the controllers included in the kit. They allow hand tracking in a 3D space, restricted by the view of the external sensors and the previous setup of the device. Moreover, these controllers include an additional sensitivity in the buttons: They are not just recognized when pressed, they know when the user is resting the fingers in each button and shows it accordingly in the game, as it is common to see your own hand in VR games for an improved immersion experience.

Apart from that, there is the normal use of the buttons, which are shown in the following image.

H

Figure 4.6: Oculus Rift's controllers

They can be divided in two sections:

- Front-facing: In this area can be found the A, B, Y, X buttons, which are mainly used in games. There is also a touchpad and a joystick for movement controls, and one key button on each controller: Home in one and Select/Menu in the other.
- Back buttons: There are two trigger buttons which are mainly used for games as shooter, speed or grab buttons.

Chapter 5

Implementation

5.1 Initial approach

At the beginning of the development process, there was an original idea of how the project would be like at the end, and a group of factors that needed to be addressed and solved before coding anything. The list of issues is the following:

- Reference system
- Tracking
- Weapon visualization
- Collision with assets
- Environment
- Point of shoot
- User experience
- Movement in VR
- Map visualization

5.1.1 Reference system

Starting from the top, it is found to be the most essential issue to be solved in this approach. Before the development started, there was a concern that it would be needed a proprietary system to merge virtual and augmented reality platform's reference system, as they could have worked in a different way.

Indeed, they both work differently, as for example, the HoloLens' centre of the Unity scene is the head of the player. This is due to the fact the usually augmented reality projects are made for a single room and to display holograms around the player. This system is not designed to move the player around a larger area like a virtual map or between physical rooms.

In the virtual reality system, the centre of the scene is not focused on the player but on the map. With this approach, the player moves around a virtual scene without difficulties. It is clear that this platform is made for gaming in small and larger scenes.

Taking this into account, there was a problem to be solved, but for simplicity reasons the first attempt was to use Unity's own reference system, which ended up working as needed.

5.1.2 Tracking

SteamVR plugin, which is used to connect the Oculus Rift to the Unity scene, allows the virtual player to move around with the help of the controllers. This was known before initiating the project.

However, the story is different for the AR device. Numerous studies have used before external tracking systems like GPS to get higher accuracy in the positional information. Due to the nature of the HoloLens, it does not have a great tracking system. It has problems with fast movement or rotations. Even too much light coming from the windows can make it get lost as the sensors get blinded.

For this reason, the initial idea was to develop the project trusting the sensors and basic tracking system of the HoloLens hoping for it to work decent enough. If further in the development of the thesis it was decided to not be enough, another solution would be needed.

5.1.3 Weapon visualization

As shown in chapter 3, there have been games developed in which the weapons are physical controllers or digitally shown in screens, but for this approach those solutions were discarded, as they would be very time consuming to implement or not possible considering that a headset device would display the weapon directly to the eyes, which could cause motion sickness or be disturbing.

For this problem, the theme of the game was the principal solution. Both players would be represented as drones with little turrets in the front, surrounding the cameras they use to see (which is where the cameras for the devices would be aligned). With this approach, there would be no need to display weapons, as the drones themselves would not see them.

5.1.4 Collision with assets

This point is just about how to interpret a collision between a game object and the augmented reality player, as he can pass through those objects because of the lack of any physical way to stop him. For this matter, the decision made was to not restrict this action in any way, as it would feel more natural to just go through a hologram than any other option.

The VR player does not have a problem with this, as with a full virtual world around him, he can not pass through virtual walls, roof or floor.

5.1.5 Environment

The game was going to take place into a specific area between two rooms connected by a hallway. For a change, the AR device has no problem with this issue, as the transparent lenses let the player see through and no further work is needed to show the scene to play. Only the other player and the possible obstacles would be visible as holograms.

The Oculus Rift's player would need to have a virtualized map of the real area where the game would be played. For that, the idea was to model that area with a 3D model software starting from the building's blueprints. However, this approach would make changing the game area not an easy task.

5.1.6 Point of shoot

Simplifying this process was a priority, as it is the main way to interact with the application. To achieve it, and following the idea for the weapon visualization, the shooting would be made pressing a button and shooting in the same direction as the gaze of the player. This approach can seem strange at first, but it came to be a very intuitive solution.

5.1.7 User experience

Originally, the idea was to make the experience as similar as possible for both platforms, using the same model for each player and making the controls such as shooting as similar as pressing a button in the same hand, even though each platform would use a different controller.

The main difference would be the movement, as the AR player needs to walk in the real world to play, and the VR one would use the joystick of the controller.

Both players would recognize the game area in both systems to be the same, as the subsection about Environment suggests.

5.1.8 Movement in VR

The first approach for this issue was to use the point to teleport type of movement described in previous chapters. At least as the initial solution, as it is easy to implement and does not cause motion sickness almost at all. However, further in development this changed to a movement with the controller's joystick to make the experience fair between players, to make them move in similar velocities, as with the first solution the VR player could move much faster and was harder to get shot.

5.1.9 Map visualization

As discussed in the subsection of environment, the VR player would be immersed in a three-dimensional model of the game area, which is obviously visible for him, and the AR player would be in the same space physically, but also in the same virtual space in the server, but that would not be visible for the player at all.

5.2 Software architecture design

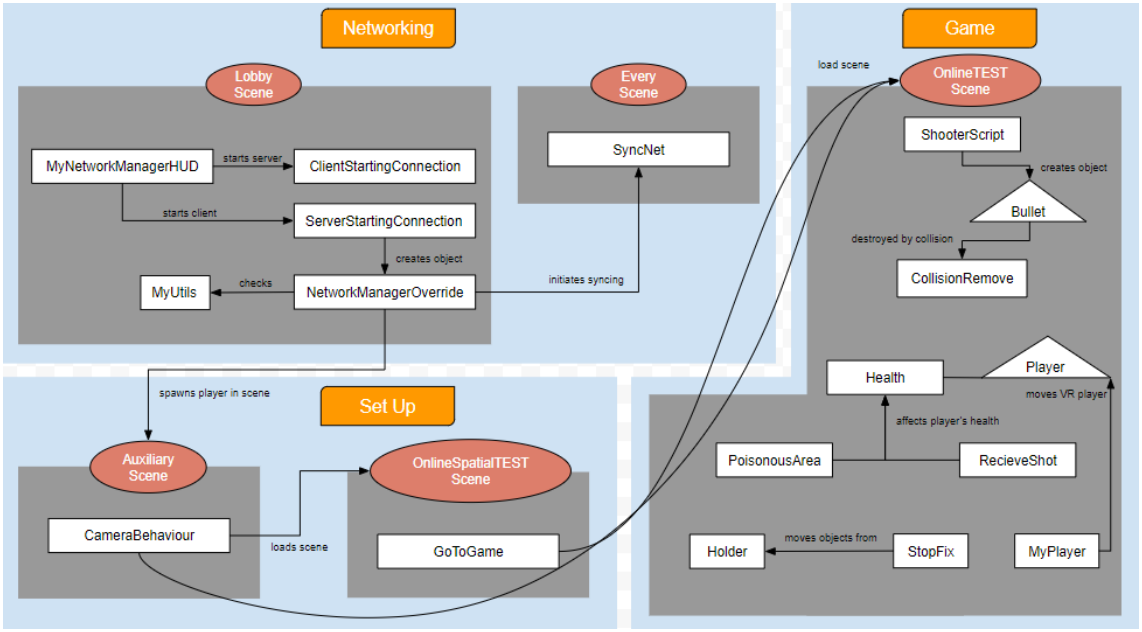

Figure 5.1: Structure of scripts used in the architecture of the project. Orange: Purpose. Reddish: Scene. White: Script. Triangle: Object

The structure shown in the previous image is essentially the sequence of scripts that act in the execution of the project. The scripts now written or modified for this project like common libraries are not shown nor described. One of those is for example the Spatial Mapping (explained before) and UNET internal classes.

It is visible that there are three main purposes that categorize the scripts. These are:

- **Networking**: Scripts and scenes purely dedicated to establishing the connection between server and client, as well as synchronizing the game's content and users.
- **Set Up**: These scripts are key for the good working of this project, as without then, no device could see correctly the scene nor play.
- **Game**: In this section can be found the code related to the game experience in itself, which can be interpreted as superficial in comparison to the other two categories.

5.2.1 Networking

This side of the project is based in two main factors. On the one hand there is the Lobby Scene, in which the entire connection is established. On the other can be found the SyncNet script that works in every scene to maintain the positions of the player synchronized.

Starting from the scripts used to establish the connection, there can be found the 'MyNetworkManagerHud' which starts either the server or client (whichever is needed in each occasion). 'ClientStartingConnection' and 'Server-StartingConnection' are the scripts that hold the code needed to start the respective platforms. Both create an object of the class 'NetworkManagerOverride' which is the final step to launch the following scene and spawn the player depending on the platform.

As described above, for the synchronization of the player's positions has been used SyncNet, a simple script that takes the local position (client side) and sends it to the server each frame, to have it always matching.

The rest of the synchronization needed for the project to work is handled by the inner functions of the library. They make possible to have the traps, movement of the objects in the scene or health of players synchronized between client and server. These things are achieved with 'Network Identity', 'Network Transform' and 'Network Transform Child' components and synced variables. For this reason, the networking side of the project has not been a major problem. A big part is already solved as the UNET library has been used for years.

5.2.2 Set Up

The scene loading sequence divides in two in this subsection. If the player is in virtual reality, the 'CameraBehaviour' scripts prepares the parameters as described in the following section and shown in appendix B. If the objective is to load the AR side, the same script changes parameters for it, as explained later and shown in appendix D.

After the Auxiliary Scene, for the VR experience the 'OnlineTEST' scene is loaded as it does not need any further preparation. However, the client (augmented reality side) loads the 'OnlineSpatialTEST' scene, in which occurs the load of the three-dimensional map previously saved, as well as the libraries in charge of the spatial anchors are initiated. After around twenty seconds of loading, the 'GoToGame' script lunches the 'OnlineTestSCENE' to match the VR device state.

5.2.3 Game

The final scene that is loaded for both players is 'OnlineTEST'. It contains the elements needed for the game to take place. In this scene are running the following scripts:

- ShooterScript
- RecieveShot
- CollisionRemove
- PoisonousArea
- Holder
- StopFix
- MyPlayer

ShooterScript

This script is responsible for the shooting capabilities in both platforms. It contains to different ways to trigger the shooting event, one for each device. The augmented reality player can shoot doing an air tap with the hand gesture or pressing the clicker, as both ways enter the same function in the code.

For the virtual reality user, it is even more simple. He just needs to press a button to trigger the event. However, both platforms include a delay after each shot, to restrict the number of clicks per second registered.

When the respective buttons are pressed, a sound of firing bullets is played to add realism to the gaming experience. Right after that, a bullet is spawn in each bullet spawn point of the player's model, which are added a velocity toward the gaze of the player. All of that is executed in the server, which is copied in the clients as well. This script is attached to each player.

RecieveShot

It is a simple script that registers every collision that each of the players receive. It checks if the collision comes from a bullet that is not from itself and calls the function in charge of taking damage from the Health script.

When a bullet collision occurs with a player, it is destroyed from the server and clients after registering the damage taken by the player. This script is attached to each player.

CollisionRemove

The functionality of this script is already described in its name. It destroys the bullets once they collision against any other thing than a player or another bullet. This script is attached to every bullet.

PoisonousArea

This script is attached to the restricted area models shown above. Its main function is to detect if a player is inside the area pre-defined as not allowed for users. If it detects one, it calls the function in charge of taking damage from the Health script of that player.

The process of checking for the user in the determined area is shown in the appendix E, as well as the system to activate the flashing damage instead of a continuous one.

Health

Its function is one of the most important of the scripts used for the game experience. It is in charge of holding the actual health of each player, as well as the maximum health possible. It also controls the canvas used to inform the players of the winner and loser, as well of to show when each player is being damaged.

The health variables involved are synchronized with the server from the clients, which allow them to be always controlled. Moreover, each model of players has a health bar in the upper part of the models that represents the health left to the adversary. This health bar is controlled in the Update method of this script as well.

Holder

It contains a collection of objects to move, as explained before, which are activated at the start of the execution and saved to be used later. It also has some objects that are needed to be disabled from the CameraBehaviour script for the VR experience to work properly. This script is attached to the OptionsHolder GameObject in the OnlineTEST scene.

StopFix

This script is responsible for the solution to the Tracking loss problem. In it is detected the exact moment in which the position of the augmented reality player is reset. In that moment, this code activates the warning for both players to stay still until it disappears in a synchronized event.

Moreover, it also calculates the opposite offset needed to be added to the scene objects' positions. When the movement is finished in both the server and client, both players are released to play with normality.

The code used for this process can be found in the appendix D.

MyPlayer

Last but not least, there is this script that controls the movement of the virtual reality player. It interacts with the controllers, to obtain the joystick position, which is used in a formula to generate a velocity depending on the direction and distance of the joystick to its center.

Finally, this velocity is attached to the VR model, giving the impression of movement. It is also in charge of attaching and removing world anchors for the model while moving, to increase the stability of the hologram for the augmented reality experience.

5.3 Theme and models used

5.3.1 Players

As exposed before, this game is based on drones flying in a real-world space. Both players are represented with the same model in the server, but with different configurations of components, depending on each one's necessities to work properly. In the gameplay, each player can see the other as this model, which makes the experience more similar between platforms.

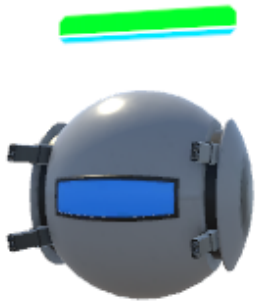

Figure 5.2: Model used to represent a player

On top of each player stands a status bar, which displays the amount of health of the player. This is shown to the other player, to see how far the adversary is from virtual death. Each player has four turrets attached to their virtual body as shown in the figure. The bullets are shot from those in straight lines.

This model is set up in each kind of player (VR and AR) to rotate with the actual rotation of the headset in which is being used. With this approach, the realism that the rotation of the drones is considerably bigger than with a joystick-based solution.

5.3.2 Map

To represent the real-world space in which the game takes place, it is used a specific model made from the map of the building. This one has also virtual representations for the real furniture that surrounds the game area, to reduce the difference between the virtual reality experience with the augmented reality one, as the second one can see the real world (and for that reason the furniture). There are only two rooms and a hallway that are going to be used, which are the only ones with more detail in them.

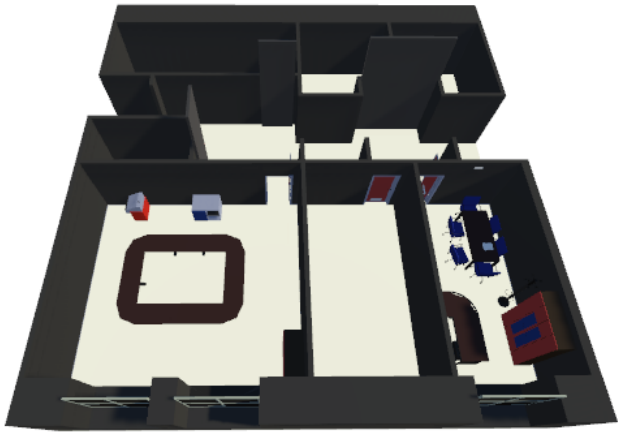

Figure 5.3: Model of the virtual game area based on the real-world one

However, there are other rooms with closed doors or walls to limit the playable area, even though could be used for a bigger game experience if wished. In addition to this, the hallway has some characteristic elements in it, like two restricted area signs and two different traps. The signs' function is to prevent users to get out of allowed spaces and, if they do get out, they would receive damage to their health as a way of 'punishment'. The traps are actually a way to make the game experience more dynamic, as the players need to be looking around to not get hurt moreover to shooting the adversary.

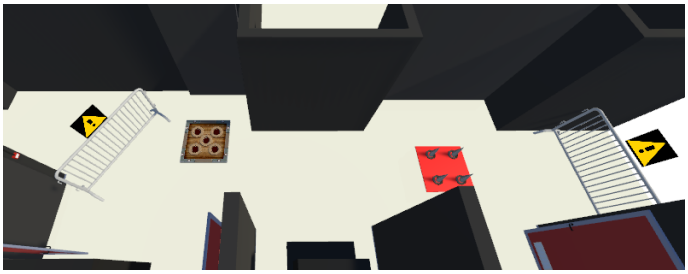

Figure 5.4: Additions to the map model

In summary, this is the map in which the game is played in the server, even though the augmented reality experience does not make use of it, as it is already in the server. This will be discussed in the next sections.

5.4 Common development between platforms

This Project is based in a multi-scene architecture, as the networking library UNET requires. There are four main scenes, which will be shown in this section, and some specific ones for each platform, that will be discussed in the following sections.

5.4.1 Sequence of scenes

The four main scenes to be explained are:

- For starters, the first Unity scene that is executed is the **Lobby**. Inside it there are just three important elements:
  - **NetworkManager**: It holds the UNET component in charge of the connection between server and client. It is set up to work in a specified IP direction and port. It also contains the list of spawnable objects, which are the ones that will be synchronized in terms of position in the map. Another important component of this object is a proprietary one that specifies the kind of player that is executing the program (VR or AR). This is especially important for the building moment of the project. Also, in this object it is specified the following two scenes to load. In case of a successful connection it goes to the Auxiliary scene and in the contrary case it goes to the Offline scene.
  - **AR and VR spawners**: These two empty objects represent the positions where each player is spawned in the map when the game starts. This is due to the need to have these positions pre-measured to achieve a reliable location-based game.
- **Offline scene**: This is a very simple empty one which shows in a panel that the client/server is disconnected.
- **Auxiliary**: It is an empty scene which has only one function, to separate the loading process depending on the platform in which the game is being executed. As seen in the appendix A, there is a part of the code that is executed when this scene is loaded that separates the execution line into two:
  - **OnlineTEST** is the next scene if the platform is virtual reality, which is common to both platforms and is explained bellow.
  - **OnlineSpatialTEST** in the augmented reality case, which is explained in the augmented reality specifics section. After this scene, the OnlineTEST scene is loaded as well for the game itself.

- **OnlineTEST**: This scene contains more objects than the previous ones, as it is where the actual game is played. However, the map for the game is not present, as it will be explained in the virtual reality specifics section. The present objects that it has are:
  - **Traps**: These traps are prepared to make damage to players when they are hit by them. They are synchronized across server and client to generate a good game experience, as both players can see the traps moving at the same time. All of this is thanks to UNET's component 'Network Transform'. They also have attached a spatial anchor, which is useful for the stability of holograms for the augmented reality experience, but this part will be explained later.
  - **Restricted Areas**: They are used to limit the game experience, causing damage to players when they get out of the game area. They work for both kind of players, as they use the position of the player in the server to function.
  - **Height Limit**: This object makes sure that no player can go higher than a specific point. This is just for restricting the movement in the game area.
  - **Options Holder**: It is an auxiliary object that collects the traps and restricted areas in the same component. This is useful for a feature explained in the augmented reality specifics section.

5.5 Virtual Reality specifics

5.5.1 Components of the player's model

The virtual reality player has a specific structure in its hierarchy to allow it to work properly.

The most important parts are the left and right hands and the 'VRCamera'. The first ones represent the virtual hands visible for the player but not for the adversary. They are used to create an improved feeling of immersion in the game. The second one, the camera, is another object from the SteamVR library but with some modifications such as having the model of the drone shown above as its child (which allows the model to move following the real movement of the head of the player), as well as two canvas to show information to the user and four bullet spawners where the shot bullets appear in the virtual world.

Apart from the hierarchy, the root of the object, the parent, counts witch scripts that make possible to be damaged by bullets or restricted areas, improvements in stability of the hologram and move around the scene.

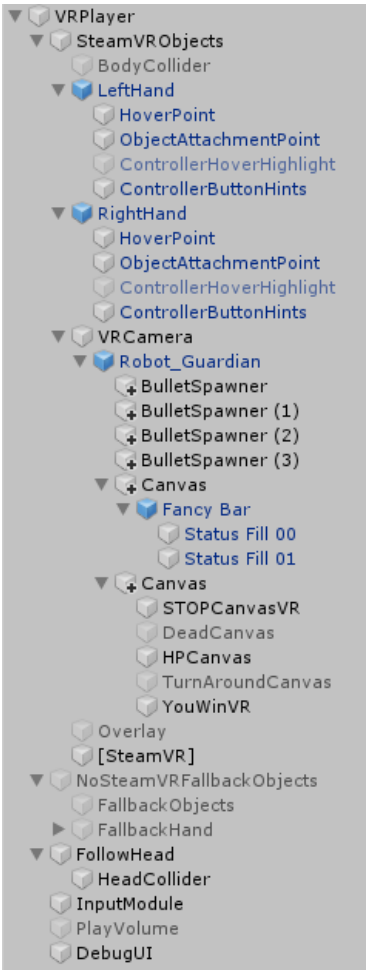

Figure 5.5: Hierarchy of the virtual reality player's GameObject

5.5.2 Scenes loading process

As the virtual reality experience does not require any other special loading, the process is the one described above adding some configurations that need to be made when the virtual player arrives to the game scene.

As seen in the appendix B, after checking that the required scene is loaded, every camera and audio listener needed are activated in a specific order, as it can't work otherwise. At the end of the code, the lines that call the SetActive function with the parameter false deactivate some objects that are used for the augmented reality player but locally, they don't have a function for the virtual experience.

In the middle of the code is the most important part, where the Instantiate methods are called. They spawn the map in the scene in a way that allows to have them in a list for easy access to them. This is useful due to the need to move them around the scene in certain situations (it is explained in a following section).

5.5.3 Behaviour of the connection

The virtual reality experience is at the same time the server of the game. Since the map and elements are present in its game scene, every action is triggered in it. In other words, when a bullet is shot in the client, it is spawned in the server and when it hits a server wall, floor or roof it is destroyed. When a trap is activated, the movement happens in the server as well, and the movement of the adversary is sent to the server as well.

With this approach, everything happens on the server and the client, in this case the augmented reality device, only sends events or copies the result of the server.

5.5.4 Controller implementation

For using the Oculus Rift's controllers, it is used the SteamVR plugin again, as it has compatibility with almost every virtual reality device controller out there. It is based on actions triggered by buttons binded in a proprietary Steam UI. Pressing a button generates an event that can be tracked by code and execute the needed code for each occasion.

The joystick, which is also used for this experience, returns with the event two coordinates, which represent the direction with values between -1.0 and 1.0 for the x and y coordinates. They are used in a movement formula that gives speed to the player and changes its position, as seen in the appendix C. The RemoveWorldAnchor and AttachWorldAnchor methods are relevant but they are explained in the next section.

5.6 Augmented Reality specifics

In terms of the hierarchy of the augmented reality player, it is the following one.

5.6.1 Components of the player's model

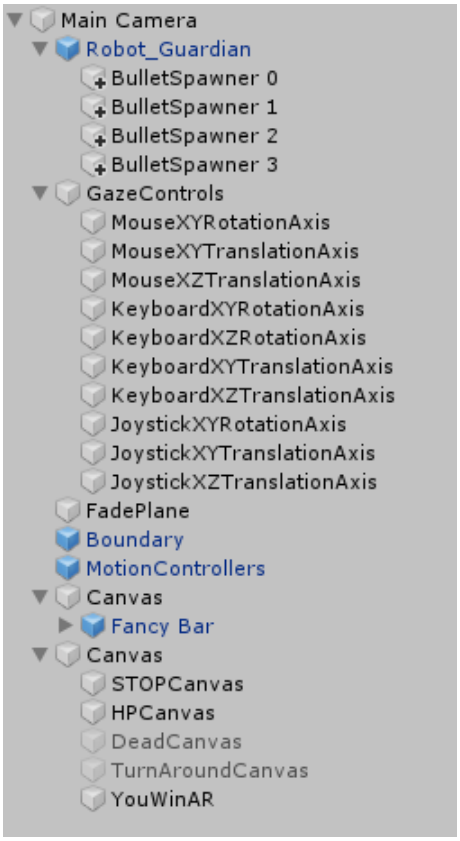

Figure 5.6: Hierarchy of the augmented reality player's GameObject

For this object, the most important part is the Main Camera itself, which is the root for the rest and contains the most amount of the scripts and components that allow it to work in this environment. It has been extracted from the Microsoft official library for the HoloLens with some proprietary modifications.

These modifications include the capabilities to have health and lose it, get shot by the other player, take damage by traps or restricted areas, as well as to set up the scene as explained bellow.

As children can be found the model of the drone, structured to move with the rotation of the headset, four bullet spawners and two canvas to show information as well as in the VR player.

5.6.2 Scenes loading process

In the case of the AR player, the scenes it needs to load are different, as it needs an additional one 'OnlineSpatialTEST', This scene is the responsible for loading the three-dimensional map previously recorded of the game area, as well as starting the sensors and functions in charge of update this information. All of it is part of the Spatial Mapping capabilities of this side of the project that are explained in the next section.

Figure 5.7: 3D map of the game area obtained with the HoloLens tool

The setup needed to work properly in the scene can be seen in the appendix D, in which it is shown that, after checking that the required scene is loaded, every camera and audio listener needed are activated in a specific order. Later, every kind of object with the tags of floor or wall is deactivated, as there is no need to use them, the needed objects that do not need to be shown to the player are made invisible with a dark material (HoloLens cannot show black in its screen) and the traps and restricted areas are activated.

The main difference with this approach is that there is no map to be used in the scene. This is since the structure of the project has the map already loaded in the server. As the augmented reality experience does not require to see the map (because the user already sees the real space) it is not included in the scene.

5.6.3 Spatial Mapping

Spatial mapping provides a detailed representation of real-world surfaces in the environment around the HoloLens. It is a key factor of this approach for three main reasons:

- **Occlusion**: As the map is not used in the scene, the hologram would be visible through walls and that would destroy the immersion feeling. This is where the occlusion comes into place. With a previously mapped area loaded, it can be used to generate a black mesh (transparent for the HoloLens) that creates the illusion of not seeing through objects.

Figure 5.8: Occlusion generated in the corner of a door

- **Hologram stability**: Spatial anchors are a useful tool that works with the spatial mapping capabilities of the device used. They attach a virtual object to a real-world space, improving its stability when it's near the player, as the sensors of the device recognize the area and reallocates the hologram to where it should be. The previously methods mentioned 'RemoveWorldAnchor' and 'AttachWorldAnchor' are the ones that create and destroy anchors. They are called from the virtual player model and from the traps and restricted areas.
- **Tracking**: The sensors used for the spatial mapping work also for the tracking of the HoloLens. As mentioned before, there is a collection of objects to move around the scene, such as the map, in a specific moment. That moment is when this tracking is lost, but it is explained in detail in a following section.

5.6.4 Controller implementation

Microsoft's HoloLens toolkit allows to track the air tap event and execute the consequent code. Moreover, use of the included clicker has also been made in the development of this project, which simulates an air tap wherever the gaze is pointing at in that moment.

For the movement, the included scripts in the toolkit allow to track the displacement of the user to use it in the virtual environment.

5.7 Comparison between platforms

Because of the already explained implementation, both platforms are able to connect to each other and play in the same environment with different platforms amongst the mixed reality spectrum.

With the intention to develop a fair game for both players, the experience between platforms is designed to be as close as possible, including even a limitation for the movement of the VR player, to have as peak speed possible a similar one to walking with the HoloLens.

Here below are some images that show the implementation working between platforms. For example, in the following picture is shown the visual difference between platforms. In each one the adversary is in the same spot, being the first one the virtual reality player seen from the AR device and the second one the other way around.

Figure 5.9: Different perspectives of the same case scenario

These other images represent the differences that can be seen in the environment. In the first one, it can be seen a restricted area sign from the perspectives of the AR player (top) and of the virtual reality one (bottom). In the second photo it can be seen the real room and the model used for the game, which is what the augmented reality device lets see to the player.

Figure 5.10: Different perspectives of the same case scenario

5.8 Problems found during development

5.8.1 Field of view

As the development kept going forward, it became clear that the field of view of the augmented reality device, the HoloLens from Microsoft, were not a good fit for this project, as it covers a 35º angle in front of the player.

This means that the gaming experience is restrictive in terms of visualization of holograms. For a game that needs to be played fast and with a good range of sight, as it is a shooter-based game, it is clear that more degrees are needed in the field of view of the device.

However, the development continued knowing this factor could have a negative impact in the results, without adding any modifications to the project, as it would still be comfortably playable.

5.8.2 Tracking loss

At the moment of testing the first prototypes of the project it became apparent that the augmented reality device had a problem with the tracking. It is not designed to work in large scenarios, which caused it to get lost. This was a combination of negative factors such as distance, fast movements, excess of light hitting the sensors coming from the windows (the development area where it was being tested had walls covered in windows) and surfaces being too close to the sensors (like a door that needs to be open).

The result of this loss of tracking was a restart in the Unity scene position, in other words, if the user started in point A (real and virtual) and moved to B (real and virtual), the position reset itself to be virtually in A but physically in B. This causes an offset of what the augmented reality user sees in the real world with what is in the virtual one.

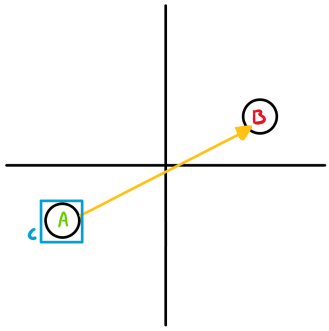

Figure 5.11: A: Initial position. B: Position after moving. C: Position after reset equal to A

To fix this problem, the first approach was to move the virtual player of AR to match its physical position. This was an easy solution; it was only needed to catch the moment in which the reset happens and correct it. However, the HoloLens internal implementation does not allow to be moved in the scene if it does not match physical movement of the headset. That is why that solution was discarded and another was needed.

The second approach was the one that worked and stayed until the end of the project. Instead of moving the player, the server map would move to match the physical position of the augmented reality player's position. For this approach it was needed to have a collection of the objects that needed to be moved (which is the one in OptionsHolder in the scene OnlineTEST. In addition to the map, the virtual reality player needed to be moved as well, but it is not visible for him as it happens in a single frame and keeps seeing the same virtual space before and after as he moves along with it.

When this solution was tested it worked as expected but had a problem. When the adversary (VR player) moved while resetting the positions as described, he would end in a different position from where he should be and could end up outside the virtual map or in areas where it is not possible to enter otherwise. Same thing happened with the AR player, as if he moved during this process the offset would be apparent again.

To solve this problem, the decision made was to explicitly announce to the player the need to stop for a second with a warning in both headsets. After this addition, the solution was completed and functional. If the tracking was lost, both players would receive a synchronized warning to stay put for a second, which is the amount of time that is needed to process the offset and move the corresponding objects of the scene to correct the experience.

It should be emphasized that when the project was tested in the real testing area, with better light conditions meaning less windows which were smaller, the stability of the tracking was noticeably better to the point of not failing almost at all.

Chapter 6

User experience

6.1 Start of the application

To start the game, each player would start getting the device ready. This process is different in each platform.

On the one hand, the virtual reality experience requires to set up the game space and sensors with the Oculus tool included in the drivers needed to use the device with the selected computer. Once the process in that program is finished, it is time to execute the project. It does not require any further configuration; it is a matter of pressing play and putting on the device.

On the other hand, the AR user does not need to setup any game area. However, the start of the project is considerably different. When the app is launched, it needs to be put down on the floor in a specific predefined position. Once the three-dimensional map is loaded and the scene is changed to the game holder one, it makes a sound that notifies the user that it is ready. From that moment on is play as usual.

Figure 6.1: Augmented reality device loading the game

6.2 Controllers

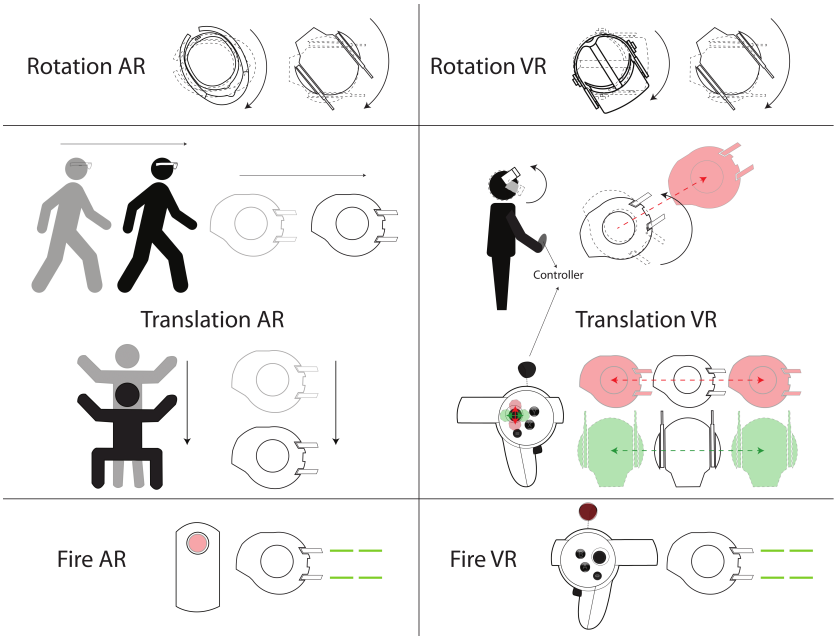

Figure 6.2: Comparative figure of AR and VR controllers

As shown in the previous image, the controllers' functions can be divided in three:

- **Rotation**: Both platforms have a free rotational input from the headsets. In other words. The movements of the head of each player are transcribed in rotational movement in the virtual world.
- **Translation**: This is the most different factor in terms of controllers. The differences are the following.
  - Augmented reality: Free movement in the real world is transcribed in movement in the virtual one. The only limitation is that they are told not to run just to have a fair experience with the virtual experience, as that one has a limit in speed.
  - Virtual reality: This one has two main axes. When the joystick is moved up or down the virtual movement is forward or backward to the gaze's direction. However, when the joystick is moved in the horizontal axe, the movement is parallel to the gaze of the player, as shown in the previous image in this section.
- Fire: Both players shoot with a controller's button (with different controller type but with similar experience) aiming to where the gazing is at the moment of firing.

6.3 Feedback to the user

During game time, the server can interact with users giving them some extra information in a more specific way. There are four cases in which this happens:

- **Getting hurt**: A red canvas is activated each time the player gets hurt. The intensity of color of this warning is inversely proportional to the amount of health left of the player. In other words, the more damaged the player is, the redder the warning is. From the user's point of view, the scenes become red for a second when he loses health points. It is not opaque, which makes the rest of the environment visible. This canvas acts just like a filter in the rest of the image.
- **Track lost**: As explained before, whenever the track is lost in the augmented reality device, this image is shown to both players, who are told to stop when the see it until it disappears.

Figure 6.3: Warning shown to users to stop due to the lost of tracking in AR

- **Entering a restricted area**: A warning is shown to the player to inform about the need to turn around and get out of the restricted area. This is shown only inside one of these areas, appearing once inside and disappearing as soon as the player gets out.

Figure 6.4: Warning shown to users to make them get out of restricted areas

- **Winning or losing**: Its function is as basic as it sounds, to inform the players about the winner and loser of the game as soon as it is over.

Figure 6.5: Feedback of the result of the game to the users

The last three use cases make use of specifically optimized canvas for each platform, to make the text shown to the users as clear as possible considering the difference in optics between headsets.

6.4 Examples of execution

There has been created a small amount of short videos that show a live execution of the project. They are the following:

- Virtual reality experience: `https://youtu.be/uruNWw6ibI8`
- Augmented reality experience 1: `https://youtu.be/TV7uTfjvHaQ`
- Augmented reality experience 2: `https://youtu.be/74fgV6a3ldE`
- Shooting example clip: `https://youtu.be/VuIBuyDCLbw`
- Hologram stability in AR: `https://youtu.be/8aIXHwYcKpI`

6.5 Tests with real users

6.5.1 Survey

Before testing the project, the users are asked to fill the initial part of the survey used. It contains:

- Name
- Age
- Sex
- How many times have you used an AR Application?
- How many times have you used an VR Application?
- How many times have you used a head mounted display?

After that part, each player starts testing one of the platforms. When the first run is done, they are asked to fill their corresponding sections. In other words, each player answers questions about the platform that has just tested. After the second run (in which they switch platforms), they do the same thing.

The questions follow the System Usability Scale (SUS). It provides a quick and reliable tool for measuring the usability. It consists of a 10-item questionnaire with five response options for respondents; from Strongly agree to Strongly disagree. SUS has become an industry standard as it is a very easy scale to administer to participants, it can be used on small sample sizes with reliable results and can effectively differentiate between usable and unusable systems [33].

The scale questionnaire from 1 to 5 for both platforms goes as follows:

- I think that I would like to use this system frequently
- I found the system unnecessarily complex
- I thought the system was easy to use
- I think that I would need the support of a technical person to be able to use this system
- I found the various functions in this system were well integrated
- I thought there was too much inconsistency in this system
- I would imagine that most people would learn to use this system very quickly

- I found the system very cumbersome to use
- I felt very confident using the system
- I needed to learn a lot of things before I could get going with this system

The open questions added to the survey are:

- What is the thing that you have liked the most?
- What is the thing that you have liked the least?

6.5.2 Results

Starting from the basic initial questions, the population (10 users) of these tests has the following characteristics:

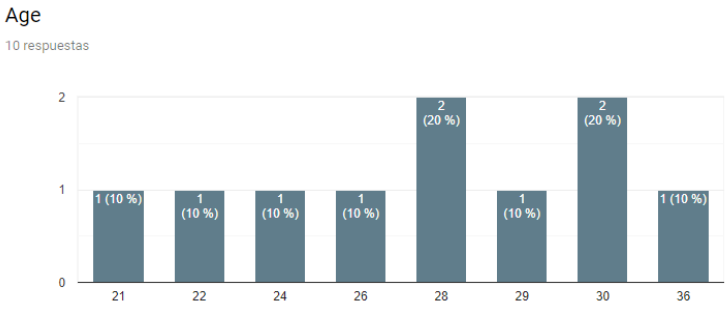

Figure 6.6: Age data obtained from the survey

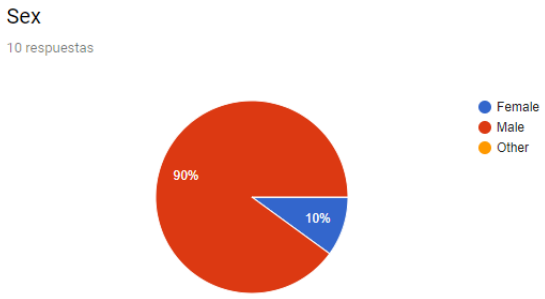

Figure 6.7: Gender data obtained from the survey

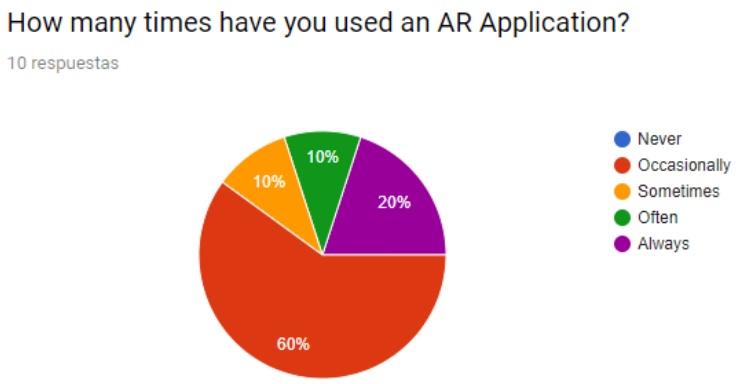

Figure 6.8: Frequency of AR experiences data obtained from the survey

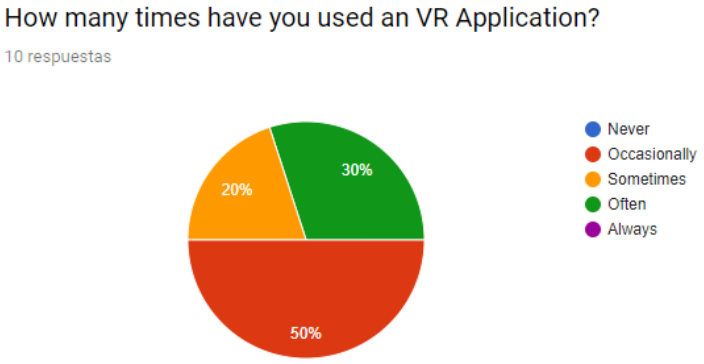

Figure 6.9: Frequency of VR experiences data obtained from the survey

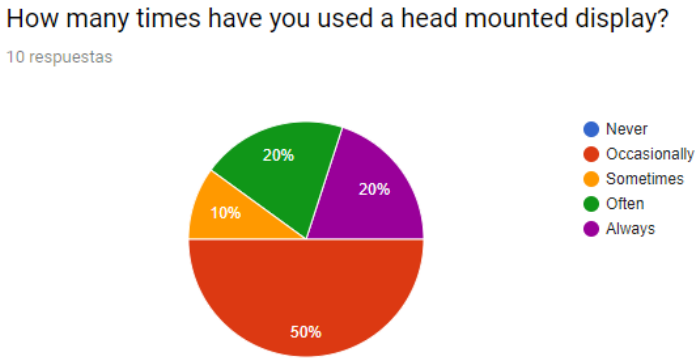

Figure 6.10: Frequency of mounted headsets experiences data obtained from the survey

With that information taken out of the way, it is time to enter the form results with its following analysis:

I think that I would like to use this system frequently

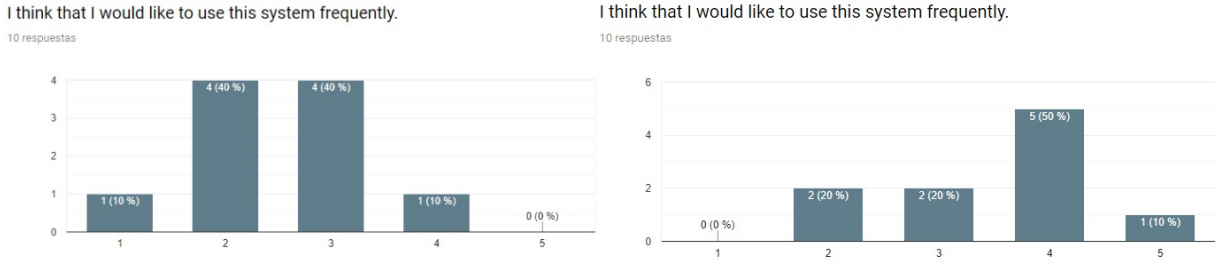

Figure 6.11: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

I found the system unnecessarily complex

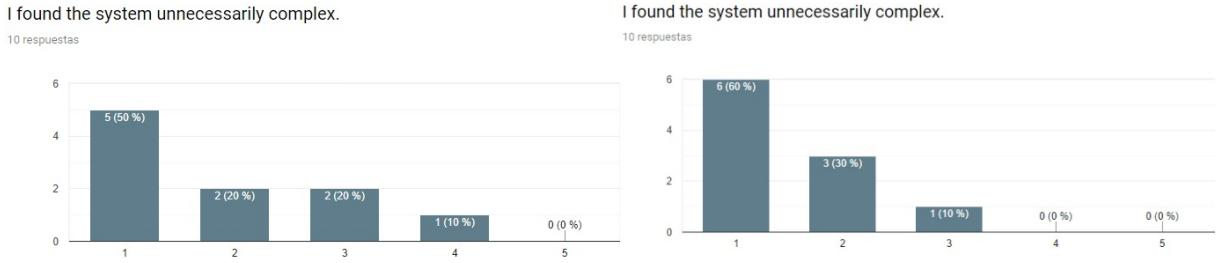

Figure 6.12: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

I thought the system was easy to use

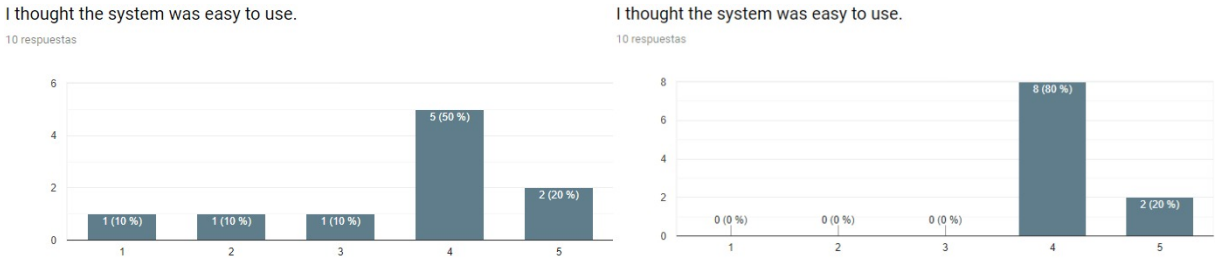

Figure 6.13: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

I think that I would need the support of a technical person to be able to use this system

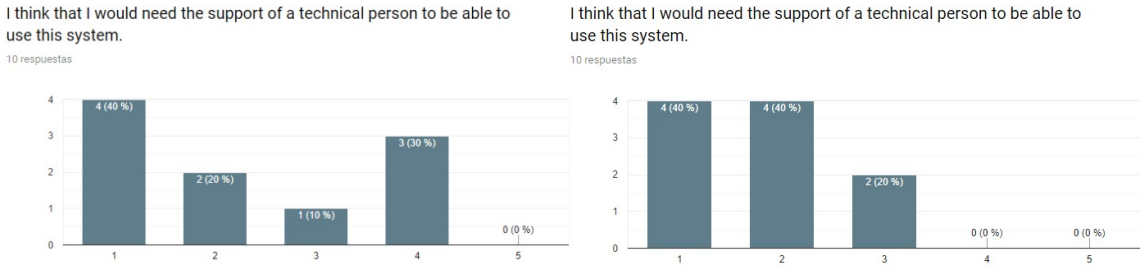

Figure 6.14: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

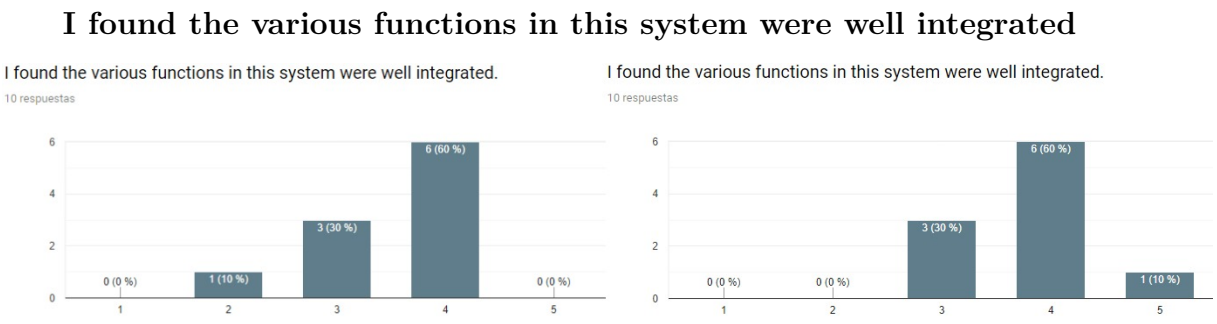

Figure 6.15: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

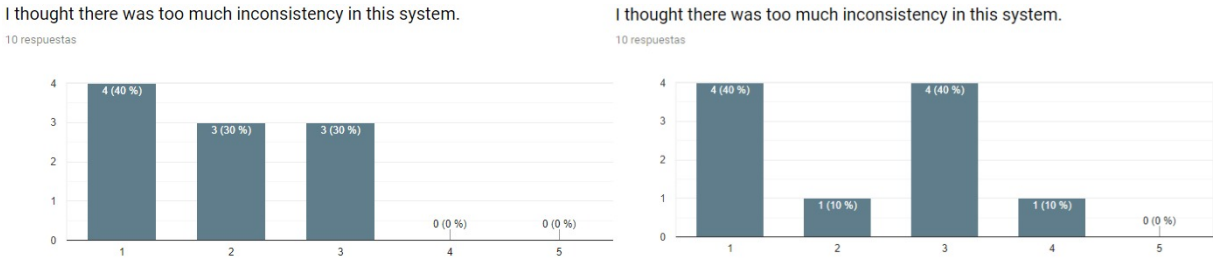

Figure 6.16: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

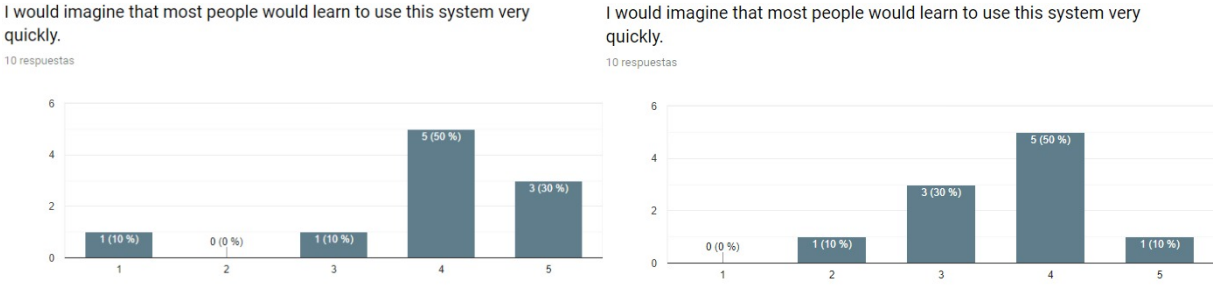

Figure 6.17: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

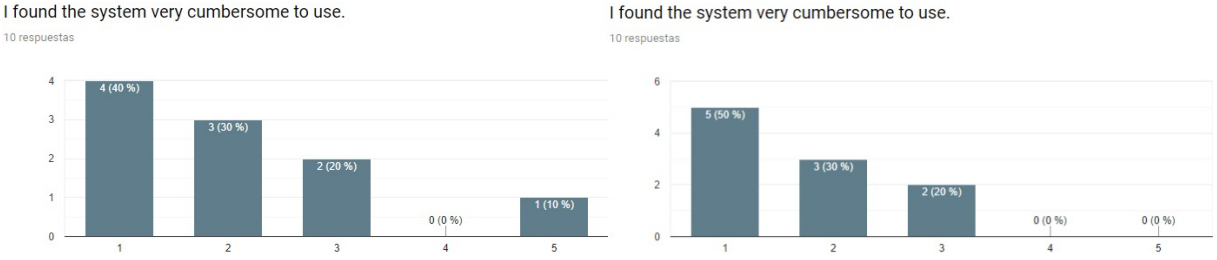

Figure 6.18: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

I felt very confident using the system

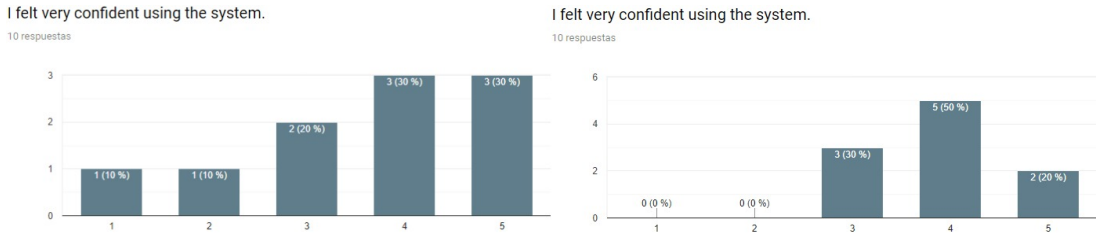

Figure 6.19: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

I needed to learn a lot of things before I could get going with this system

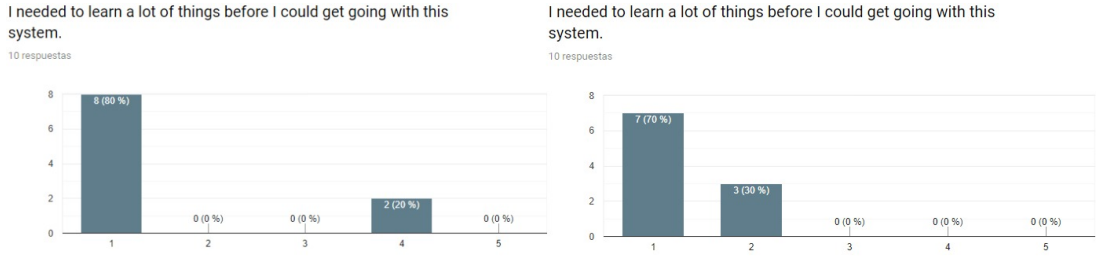

Figure 6.20: Augmented reality (left) and virtual reality (right). 0: Strongly disagree. 1: Strongly agree.

What is the thing that you have liked the most?

Figure 6.21: Observations made by the players. Augmented reality (left) and virtual reality (right).

What is the thing that you have liked the least?

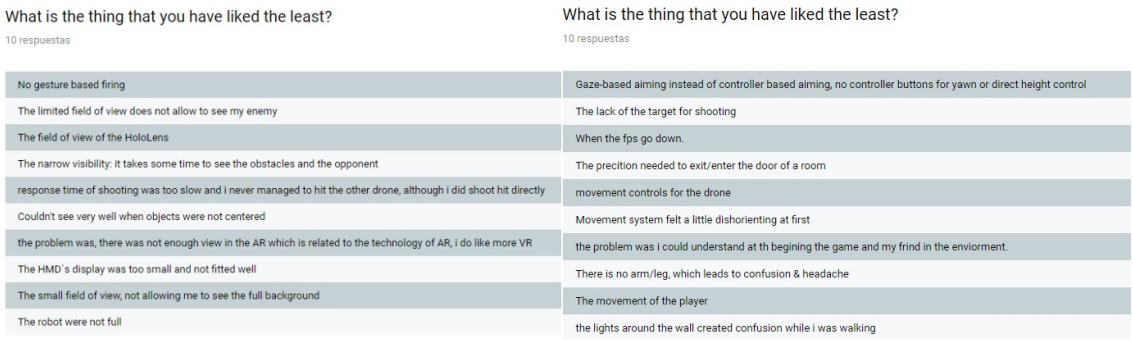

Figure 6.22: Observations made by the players. Augmented reality (left) and virtual reality (right).

6.5.3 Analysis

The data used for the following calculations is based on the grouped results bellow.

It is proposed a H_0 like $\mu_{AR} = \mu_{VR}$ and a H1 of both means being different. Considering that $\mu_{AR} = 27.3$ and $\sigma^2_{AR} = 67.3$. For the virtual reality experience, $\mu_{VR} = 30.5$ and $\sigma^2_{VR} = 23.1$.

Following an un-paired student t-test distribution, the probability obtained is 0.2937, higher than 5%. This means that the null hypothesis cannot be rejected. In other words, it cannot be claimed that an application is more usable than the other one, even though a noticeable preference for VR has been spotted.

Even though the statistical analysis does not provide an evidence of user preference, there is still some knowledge obtained from these tests. For example, a common complaint has been the HoloLens' field of view, as it can be difficult to see the adversary in some occasions. However, the capability of free movement in the game area has been the most positive comment.

For the VR case, the realism in the models used for the game have been the most positive appreciation. For contrary, the worst has been the motion sickness suffered by one of the testers due to the motion system implemented.

Repeating the testing with a much bigger population would be necessary to get clearer results.

User	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	3	4	3	4	3	4	4	4	4	4
2	1	3	2	4	2	2	3	4	2	4
3	2	3	3	1	3	4	2	3	1	1
4	1	4	3	3	3	4	3	2	3	4
5	1	4	3	4	2	2	3	3	4	4
6	2	4	3	4	3	4	3	4	3	4
7	2	2	1	2	2	3	3	2	3	4
8	2	2	4	1	3	3	4	3	4	4
9	0	1	0	1	3	2	0	0	0	1
10	1	4	4	3	1	3	4	4	2	4

Table 6.1: Results obtained by testing the augmented application

User	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1	1	3	3	4	2	2	3	3	3	4
2	3	4	3	4	3	2	3	3	3	4
3	3	3	3	2	3	4	2	4	3	3
4	3	4	3	3	3	4	3	4	3	4
5	3	4	3	3	2	2	1	2	2	4
6	2	3	3	3	3	3	2	3	2	3
7	2	2	3	2	2	1	2	2	3	4
8	4	4	4	4	4	4	4	4	4	4
9	3	4	3	4	3	4	3	4	2	3
10	1	4	4	3	3	2	3	4	4	4

Table 6.2: Results obtained by testing the virtual application

Chapter 7

Conclusions and future work

As seen above, this project has accomplished to develop a multiplayer first-person shooter cross platform between augmented and virtual reality devices. It is true that the statistical data obtained does not show preference for either sides of the thesis. However, it is clear that this kind of game is viable and enjoyable for the open public.

The limitations found during this thesis would be the next objective if this was a work in progress. For that reason, it is interesting to mention that there are new generation devices for each platform chosen in this development.

Microsoft released the HoloLens 2 in the middle of this project. This new device counts with double the field of view, which would probably solve the most commented thing as the negative side for the augmented reality experience of the application. It also counts with native hand recognition, which could have opened new possibilities for the shooting system and weapon visualization. Tracking capabilities have been updated too, which could have had a great impact in the performance of the application.

Figure 7.1: New device: HoloLens 2

For the VR device, Oculus' Rift S model was already available from the start of the project, but the used model had already been handed to this development. The main improvement that this device would add to this thesis is, apart from the higher resolution of the display, the tracking system, which is no longer external but internal. It uses two cameras and a proprietary system to track the controllers without the need to set up the game area before playing or each time that the location is changed.

Figure 7.2: New device: Oculus Rift S

Lastly, the chosen networking library is being deprecated, which will cause the need to update that part of the code. This would benefit the project, as the actual library used is quite old and do not offer the best performance possible with the hardware used.

For future works, apart from using the previous devices mentioned and with the aim of extending this project, it could be possible to design and develop a proprietary solution for the tracking in large areas in the augmented reality experience. With this implementation, it could be possible to play in spaces as big as cities, needing for that a server to process previously mapped areas digitally combined.

Another possible addition to the project would be the ability to automatically generate the models for the virtual reality experience from the maps generated with the HoloLens three-dimensional system. It could save time in the preparation process for each new location. With this new feature and the previous one, the scalability of the project would be almost limitless.

The previous two additions are selected amongst much more options such as, for example, recognizing objects with the HoloLens and reacting in consequence. This project could be as complete as desired because of the open nature that it has.

It is possible that, in the near future, commercial games (or even other industries) using this kind of approach start appearing to the public, as the hardware is already in the necessary state in which it can be enjoyable and affordable at the same time.

Acronyms

AR	Augmented Reality
VR	Virtual Reality
SUS	System Usability Scale
UI	User Interface
API	Application Programming Interface
LED	Light-Emitting Diode
TPM	Total Productive Maintenance
HPU	High Power Unit
GB	Gigabyte
IMU	Inertial Measurement Unit
MP	Megapixel
HD	High Definition
3D	Three-dimensional
OS	Operative System
SDK	Software Development Kit
AMOLED	Active Matrix Organic Light Emitting Diode
DoF	Degrees of Freedom
PC	Personal Computer

Appendices

Appendix A

Common development between platforms: Sequence of scenes

```
public void Start()
{
    StartCoroutine(Change());
}

IEnumerator Change()
{
    yield return new WaitForSeconds(1);
    if (sceneName == "Auxiliar" &&
    this.name.StartsWith("Main Camera") &&
    isLocalPlayer)
    {
        SceneManager.LoadScene("Online-SpatialTEST");
    }
    if (sceneName == "Auxiliar" &&
    this.name.StartsWith("VRPlayer(Clone)") &&
    isLocalPlayer)
    {
        SceneManager.LoadScene("OnlineTEST");
    }
}
```

Appendix B

Virtual Reality specifics: Scenes loading process

```
private void Update()
{
    if (requiredOnce)
    {
        scene = SceneManager.GetActiveScene();
        sceneName = scene.name;

        if (sceneName == "OnlineTEST" && isLocalPlayer)
        {
            foreach (var objectToEnable in objectsToEnable)
            {
                objectToEnable.SetActive(true);
                GameObject aux = objectToEnable.gameObject.GetComponent
    <Camera>();
                if (aux != null)
                    aux.enabled = true;
                aux = objectToEnable.gameObject.GetComponent<
    AudioListener>();
                if (aux != null)
                    aux.enabled = true;
            }

            if (!this.name.StartsWith("Main Camera"))
            {
                foreach (GameObject x in
                Resources.FindObjectsOfTypeAll(typeof(GameObject)))
                {
                    if (x.name == "OptionsHolder")
                    {
                        x.GetComponent<Holder>().ObjectsToMove
                        = new List<GameObject>(3)
                        {
                            Instantiate(AreaTest,
                                AreaTest.transform.position,
                                AreaTest.transform.rotation),
```

```
                            Instantiate(Roof,
                                Roof.transform.position,
                                Roof.transform.rotation),
                            Instantiate(PlaneTeleport,
                                PlaneTeleport.transform.position,
                                PlaneTeleport.transform.rotation)
                        };
                        x.SetActive(true);
                    }
                }
            }

            Holder.shared.DefaultCursor.SetActive(false);
            Holder.shared.InputManager.SetActive(false);

            this.transform.GetChild(0).gameObject.SetActive(true);

            requiredOnce = false;
        }
    }
}
```

Appendix C

Virtual Reality specifics: Controller implementation

```
void FixedUpdate()
{
    horizontal = moveAction.GetAxis(SteamVR_Input_Sources.LeftHand).x;
    vertical = moveAction.GetAxis(SteamVR_Input_Sources.LeftHand).y;

    if (horizontal != 0 || vertical != 0)
    {
        Vector3 aux1 = Vector3.zero;
        Vector3 aux2 = Vector3.zero;

        aux1 = camera.forward * vertical;
        aux2 = camera.right * horizontal;

        this.GetComponent<CharacterController>().Move((aux1 + aux2)/70)
    ;
        this.GetComponent<CharacterController>().center = new Vector3
    (-0.29f, 1.23f, 1);

        this.gameObject.GetComponent<TapToPlace>().RemoveWorldAnchor();
        this.gameObject.GetComponent<TapToPlace>().AttachWorldAnchor();
    }
}
```

Appendix D

Augmented Reality specifics: Scenes loading process

```
private void Update()
{
    if (requiredOnce)
    {
        scene = SceneManager.GetActiveScene();
        sceneName = scene.name;

        if (sceneName == "OnlineTEST" && isLocalPlayer)
        {
            foreach (var objectToEnable in objectsToEnable)
            {
                objectToEnable.SetActive(true);
                GameObject aux = objectToEnable.gameObject.GetComponent
    <Camera>();
                if (aux != null)
                    aux.enabled = true;
                aux = objectToEnable.gameObject.GetComponent<
    AudioListener>();
                if (aux != null)
                    aux.enabled = true;
            }

            if (this.name.StartsWith("Main Camera"))
            {
                foreach (GameObject x in
                GameObject.FindGameObjectsWithTag("Floor"))
                {
                    x.SetActive(false);
                }

                foreach (GameObject y in
                GameObject.FindGameObjectsWithTag("Wall"))
                {
                    y.SetActive(false);
                }
```

```
                Color32 c;
                foreach (var objectToHide in objectsToHide)
                {
                    Material aux1 = objectToHide.GetComponent<Renderer
    >().material;
                    c = aux1.GetColor("_Color");
                    aux1 = MatTransparent;
                }

                foreach (GameObject x in
                Resources.FindObjectsOfTypeAll(typeof(GameObject)))
                {
                    if (x.name == "OptionsHolder")
                    {
                        foreach (GameObject y
                        in x.GetComponent<Holder>().RestrictedAreas)
                        {
                            y.SetActive(true);
                        }
                        x.SetActive(true);
                    }
                }

                requiredOnce = false;
            }
        }
        if (this.name.StartsWith("Main Camera") &&
        once &&
        GameObject.Find("Main Camera") != null)
        {
            GameObject aux2 = GameObject.Find("FloorQuad(Clone)");
            if (aux2 != null)
            {
                aux2.SetActive(false);
            }
            once = false;
        }
    }
}
```

Appendix E

Software architecture design: PoisonousArea

```
void Update()
{
    if (isLocalPlayer)
    {
        if (poisonousAreas.Length == 0)
        {
            if (GameObject.Find("OptionsHolder") != null)
            {
                poisonousAreas = GameObject.Find("OptionsHolder").
    GetComponent<Holder>().RestrictedAreas;
                foreach (var x in poisonousAreas)
                {
                    Areas.Add(x.name, false);
                }
            }
        }
        if (poisonousAreas.Length > 0)
        {
            foreach (GameObject poisonousArea in poisonousAreas)
            {
                Transform p1 = poisonousArea.transform.GetChild(0);
                Transform p2 = poisonousArea.transform.GetChild(1);
                if (FindPoint(p1, p2, this.transform))
                {
                    if (!IsInvoking("TakeDamage"))
                    {
                        canvasTurnAround.SetActive(true);
                        InvokeRepeating("TakeDamage", 0.0f, 1.5f);
                    }

                    Areas[poisonousArea.name] = true;
                }
                else
                {
                    if (Areas[poisonousArea.name])
```

```
                    {
                        contador += 1;
                        canvasTurnAround.SetActive(false);
                        CancelInvoke("TakeDamage");
                        Areas[poisonousArea.name] = false;
                    }
                }
            }
        }
        CmdPrintCount(contador);
    }
}

static bool FindPoint(Transform p1, Transform p2, Transform p)
{
    Vector2 p1_processed = new Vector2(p1.position.x, p1.position.z);
    Vector2 p2_processed = new Vector2(p2.position.x, p2.position.z);
    Vector2 p_processed = new Vector2(p.position.x, p.position.z);

    float x = Vector2.Distance(p1_processed, p_processed);
    float y = Vector2.Distance(p2_processed, p_processed);
    double z = Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2));

    double insideResult = Vector2.Distance(p1_processed, p2_processed);

    return (z <= insideResult);
}
```

Appendix F

Software architecture design: StopFix

```
 private void Awake()
{
    lastPosition = this.gameObject.transform.position;
    origin = this.gameObject.transform.position;
}

void FixedUpdate()
{
    if (isLocalPlayer)
    {
        actualPosition = this.gameObject.transform.position;

        if (Mathf.Sqrt(Mathf.Pow(actualPosition.x - lastPosition.x, 2)
    + Mathf.Pow(actualPosition.z - lastPosition.z, 2)) >
    acceptableValue)
        {
            StartCoroutine(StopMoving());
            Vector3 delta = new Vector3(actualPosition.x - lastPosition
    .x, actualPosition.y - lastPosition.y, actualPosition.z -
    lastPosition.z);
            Debug.Log("LAST POSITION: " + lastPosition + " ACTUAL
    POSITION: " + actualPosition);
            CmdMoveTo(delta);
            if (!isServer)
            {
                Debug.Log("Antes de hacer algo");
                GameObject y = GameObject.Find("RestrictedArea1");
                y.transform.position += delta;
                y = GameObject.Find("RestrictedArea2");
                y.transform.position += delta;
                y = GameObject.Find("Trap_Needle");
                y.transform.position += delta;
                y = GameObject.Find("SpearTrap");
                y.transform.position += delta;
                y = GameObject.Find("VRPlayer(Clone)");
```

```
                y.transform.position += delta;

                Debug.Log("Despues");
            }
        }

        lastPosition = actualPosition;
    }
}

public IEnumerator StopMoving()
{
    CmdStopVR();
    Debug.Log("StopMoving HOLOLENS");
    GameObject aux = GameObject.Find("STOPCanvas");
    aux.GetComponent<Image>().sprite = hand;
    aux.GetComponent<Image>().color = new Color32(255, 255, 255, 255);
    yield return new WaitForSeconds(secondsToRemap);
    CmdrenoveVR();
    aux.GetComponent<Image>().color = new Color32(255, 255, 255, 0);
    aux.GetComponent<Image>().sprite = null;
}

[Command]
public void CmdStopVR()
{
    GameObject aux = GameObject.Find("STOPCanvasVR");
    aux.GetComponent<Image>().sprite = hand;
    aux.GetComponent<Image>().color = new Color32(255, 255, 255, 255);
    Debug.Log("Stopped");
}

[Command]
public void CmdrenoveVR()
{
    Debug.Log("CmdrenoveVR HOLOLENS");
    GameObject aux = GameObject.Find("STOPCanvasVR");
    aux.GetComponent<Image>().color = new Color32(255, 255, 255, 0);
    aux.GetComponent<Image>().sprite = null;
    Debug.Log("Renewed");
}

[Command]
public void CmdMoveTo(Vector3 delta)
{
    Debug.Log("--------MOVING SCENE--------");
    Debug.Log("DELTA: " + delta);
    string s;
    foreach (GameObject x in Resources.FindObjectsOfTypeAll(typeof(
    GameObject)))
    {
        if (x.name == "OptionsHolder")
        {
            List<GameObject> collection = x.GetComponent<Holder>().
    ObjectsToMove;
```

```
            foreach (GameObject y in collection)
            {
                s = y.name + " PAST: " + y.transform.position.ToString
    ();
                y.transform.position = y.transform.position + delta;
                Debug.Log(s + " NOW: " + y.transform.position);
            }
        }
    }

    s = "VRPlayer" + " PAST: " + GameObject.Find("VRPlayer(Clone)").
    transform.position.ToString();
    GameObject.Find("VRPlayer(Clone)").transform.position = GameObject.
    Find("VRPlayer(Clone)").transform.position + delta;
    Debug.Log(s + " NOW: " + GameObject.Find("VRPlayer(Clone)").
    transform.position);
    Debug.Log("-----------------------------------------------");
}
```

Bibliography

[1] Wikipedia contributors. Virtual reality — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Virtual_reality&oldid=903474622, 2019. [Online; accessed 10-July-2019].

[2] Wikipedia contributors. Augmented reality — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=903527025, 2019. [Online; accessed 10-July-2019].

[3] Wikipedia. Videojuego multijugador — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 10-julio-2019].

[4] Wikipedia. The elder scrolls v: Skyrim — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 10-julio-2019].

[5] Wikipedia contributors. Superhot — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Superhot&oldid=904308268, 2019. [Online; accessed 10-July-2019].

[6] Kurinchi Selvan Gurusamy, Rajesh Aggarwal, Latha Palanivelu, and Brian R Davidson. Virtual reality training for surgical trainees in laparoscopic surgery. *Cochrane database of systematic reviews*, (1), 2009.

[7] Anthony G Gallagher, E Matt Ritter, Howard Champion, Gerald Higgins, Marvin P Fried, Gerald Moses, C Daniel Smith, and Richard M Satava. Virtual reality simulation for the operating room: proficiency-based training as a paradigm shift in surgical skills training. *Annals of surgery*, 241(2):364, 2005.

[8] Brenda K Wiederhold and Mark D Wiederhold. *Virtual reality therapy for anxiety disorders: Advances in evaluation and treatment.* American Psychological Association, 2005.

[9] Doug A Bowman and Ryan P McMahan. Virtual reality: how much immersion is enough? *Computer*, 40(7):36–43, 2007.

[10] Ajey Lele. Virtual reality and its military utility. *Journal of Ambient Intelligence and Humanized Computing*, 4(1):17–26, 2013.

[11] Albert Rizzo, Jarrell Pair, Ken Graap, Brian Manson, Peter J McNerney, Brenda Wiederhold, Mark Wiederhold, and James Spira. A virtual reality exposure therapy application for iraq war military personnel with post traumatic

stress disorder: From training to toy to treatment. *NATO Security through Science Series E Human and Societal Dynamics*, 6:235, 2006.

[12] Evren Bozgeyikli, Andrew Raij, Srinivas Katkoori, and Rajiv Dubey. Point & teleport locomotion technique for virtual reality. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 205–216. ACM, 2016.

[13] Noah Coomer, Sadler Bullard, William Clinton, and Betsy Williams-Sanders. Evaluating the effects of four vr locomotion methods: joystick, arm-cycling, point-tugging, and teleporting. In *Proceedings of the 15th ACM Symposium on Applied Perception*, page 7. ACM, 2018.

[14] Yun Suen Pai and Kai Kunze. Armswing: Using arm swings for accessible and immersive navigation in ar/vr spaces. In *Proceedings of the 16th International Conference on Mobile and Ubiquitous Multimedia*, pages 189–198. ACM, 2017.

[15] Shubhankar Ranade, Mingshu Zhang, Mohammed Al-Sada, Jaryd Urbani, and Tatsuo Nakajima. Clash tanks: An investigation of virtual and augmented reality gaming experience. In *2017 Tenth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pages 1–6. IEEE, 2017.

[16] Wikipedia. Ingress — wikipedia, la enciclopedia libre, 2019. [Internet; descargado 10-julio-2019].

[17] Proxy42 Inc. Father.io ar fps. `https://play.google.com/store/apps/details?id=com.proxy42.father.io`. Accessed: 2019-07-10.

[18] Wayne Piekarski and Bruce Thomas. Arquake: the outdoor augmented reality gaming system. *Communications of the ACM*, 45(1):36–38, 2002.

[19] Damien Rompapas, Christian Sandor, Alexander Plopski, Daniel Saakes, Dong Hyeok Yun, Takafumi Taketomi, and Hirokazu Kato. Holoroyale: A large scale high fidelity augmented reality game. In *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*, pages 163–165. ACM, 2018.

[20] Zhiwei Zhu, Vlad Branzoi, Mikhail Sizintsev, Nicholas Vitovitch, Taragay Oskiper, Ryan Villamil, Ali Chaudhry, Supun Samarasekera, and Rakesh Kumar. Ar-weapon: live augmented reality based first-person shooting system. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 618–625. IEEE, 2015.

[21] D Weng, D Li, W Xu, Y Liu, and Y Wang. Ar shooter: An augmented reality shooting game system. In *2010 IEEE International Symposium on Mixed and Augmented Reality*, pages 311–311. IEEE, 2010.

[22] Evike Extended. Shooter ar promo. `https://www.youtube.com/watch?v=dREvZRRGd4c`. Accessed: 2019-06-20.

[23] Peter Weir, Christian Sandor, Matt Swoboda, Thanh Nguyen, Ulrich Eck, Gerhard Reitmayr, and Arindam Dey. Burnar: Feel the heat. In *2012 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 331–332. IEEE, 2012.

[24] fatherio. Father.io ar fps - real life battlegrounds! `https://www.youtube.com/watch?v=Y_TZTnxgD1I`. Accessed: 2019-06-20.

[25] Microsoft. Hololens spatial mapping official documentation. `https://docs.microsoft.com/en-us/windows/mixed-reality/spatial-mapping`. Accessed: 2010-09-30.

[26] Francesco De Pace, Federico Manuri, Andrea Sanna, and Davide Zappia. Virtual and augmented reality interfaces in shared game environments: A novel approach. In *International Conference on Intelligent Technologies for Interactive Entertainment*, pages 137–147. Springer, 2018.

[27] Microsoft. Hololens (1st gen) hardware details. `https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details`. Accessed: 2019-06-21.

[28] Fologram. Learning to use voice, gestures and gaze. `https://learn.fologram.com/hc/en-us/articles/360000919873-Learning-to-use-Voice-Gestures-and-Gaze`. Accessed: 2019-06-23.

[29] Microsoft. Mixed reality documentation. `https://docs.microsoft.com/en-us/windows/mixed-reality/`. Accessed: 2019-06-23.

[30] Jeremy Horwitz. Oculus rift s versus oculus rift: the spec comparison chart. `https://venturebeat.com/2019/03/20/oculus-rift-s-versus-oculus-rift-the-spec-comparison-chart/`. Accessed: 2019-06-23.

[31] Oculus. Primeros pasos en la realidad virtual. `https://www.oculus.com/setup/`. Accessed: 2019-06-23.

[32] Steam. Steamvr. `https://store.steampowered.com/app/250820/SteamVR/`. Accessed: 2019-06-23.

[33] System usability scale (sus). `https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html`. Accessed: 2019-07-16.