POLITECNICO DI TORINO

Master's Degree Course in Computer Engineering

Master's Degree Thesis

Field of View Evaluation in Augmented Reality Games

Chess Game for HoloLens and Meta2 Devices

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

Author
Sonia Elizondo Martínez

July 2019

Written in LaTeX on July 22, 2019
This work is subject to the CC BY-NC-ND Licence

"That brain of mine is something more than merely mortal; as time will show."

Ada Lovelace

Contents

Acknowledgements

I would like to thank Prof. Andrea Sanna and Dr. Federico Manuri for the great opportunity of developing my Master's Thesis under their guidance and Dr. Francesco De Pace for his daily support at the laboratory.

Thanks to my family and friends for giving me the strength to be able to surpass my own expectations.

Abstract

The aim of this Master's Thesis is to compare the Fields of View of two different Head-Mounted Displays: Microsoft's HoloLens and Metavision's Meta 2.

To accomplish this goal, an Augmented Reality table-top game has been developed. After noticing its popularity and enhancement with computers through decades, the popular board game *chess* was selected as game to be implemented. This game has been redesigned, not only in a visual context, but in the way players would interact with it, providing a new experience.

Once implemented for both mentioned devices, the chess game has been evaluated by numerous users, giving their opinions about the usability of the head-mounted displays in the form of a questionnaire.

The data obtained during these user tests has been analyzed to reach a conclusion to a main question: which device players prefer to utilize in order to properly involve with an Augmented Reality game.

Moreover, future purposes in relation to this project have been presented in the form of investigation proposals and improvements to the current work.

Keywords: Augmented Reality, Field of View, Meta 2, HoloLens, chess game.

Chapter 1

Introduction

The upcoming market of not so rudimental wearable devices is making possible to develop and experience Augmented Reality (AR) applications as never before. A deepest look into the backwards of AR games development will be taken at next chapter.

In this past decade (2010's), leaving behind the primitive Head-Worn Displays (HWD) and Handheld Displays, Microsoft and Metavision gave a new perspective with their Head Mounted Displays (HMD) - HoloLens and Meta 2, respectively.

The possible interactions with the virtual and real worlds multiplied due primarily to the fact that these devices let the user free-handed. Despite of having acquired this great advantage, AR devices keep failing to another very important characteristic for players, but very difficult to improve for developers: their Field of View (FOV).

FOV refers to an optic property. It is defined as the expansion of the appreciable world at any precise moment.
The visual field of the human eye is generally 200 degrees horizontally and 135 degrees vertically. This wide range permits humans to see a great portion of the landscape and at the same time, detect changes in motion in their surroundings. This extraordinary mechanism is the biggest challenge AR must face: achieve a FOV that can behave as widely and dynamically as human eyes. [1]

Due to the importance given to this aspect, it was decided to set it as main goal of this Master's Thesis: evaluate and compare both HoloLens and Meta2 FOV.

To do so, a table-top game scenario was brought up to be a good way to confront both devices. Once the classical chess game was found to be an adequate option, it had to be interpreted into a new AR design (refer to Chapter 4) and implemented for the two mentioned AR devices.

In fact, it was also necessary to display the same exact UI, in order to make the situation suitable for a comparison. To understand the technology behind these two devices, both hardware and software details will be described in Chapter 3.

The development process consisted in implementing two chess games, one for each device, counting on the games' developing platform Unity. All this procedure, including essential phases of the project such as first thoughts, problems found, changes in perspective and final scenarios, will be extensively described in the implementation chapter, founding the core of this work.

Once finished, the project was evaluated by various users. Letting them play both chess games during a limited amount of time, they could experience the feeling of changing the FOV from one device to another and then express their opinions in the SUS questionnaire.

As expected, the results concluded that players feel more comfortable with the Meta 2 display, due to its wider FOV.
The full analysis of users' feedback and final conclusion to this work will take place at one of last chapters dedicated to the experimentation explanation.

Furthermore, improvements to this project in the form of future aims will close this Thesis.

Chapter 2

Augmented Reality

Augmented Reality (AR) is not a strange term anymore, due to its various applications and continuous development. But, the first time it was formalized, it sure sounded rare. It was in 1994, when Milgram and Kishino established the *reality-virtuality continuum* to explain the conceivable variations and configurations of real and virtual objects. [2]

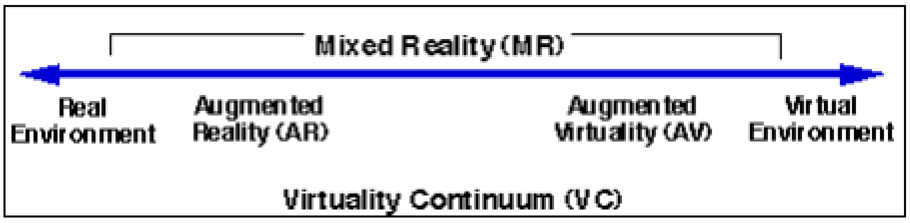


Figure 2.1: Simplified representation of a "virtuality continuum" [2]

Based on this definition, AR can be described as the interaction between the real and the virtual worlds. This way, real and computer-generated objects can coexist in real time and in the same environment. But not only in a visual way: it can be applied to all senses, including smell. [3]

2.1 Brief History

Although its starting point dates in the 1960s, it is not until the 1990s that AR was referred to as an actual research field: several important conferences and symposiums landed and important organizations focused on the development of new tools. [3] Nowadays, AR has hit its top development with brand-new devices that are opening the path to what will come in the near future.

These decades are full of revealing moments that can be rapidly reviewed [4]:

- In 1968, Ivan Sutherland developed the first HMD. This system used computer-generated graphics to show simple wireframe drawings.
- In 1974, Myron Krueger built an 'artificial reality' laboratory: the Videoplace. It combined projectors with video cameras to emit onscreen silhouettes, involving users in an interactive environment.
- In 1990, the term "Augmented Reality" was coined for the first time by Tom Caudell, a Boeing researcher.
- In 1992, Louis Rosenberg developed Virtual Fixtures - one of the initial working AR systems, built for the Air Force. The exoskeleton upper body let the military control virtually guided machinery to perform tasks from a remote operating space.
- In 1994, Julie Martin created the first AR Theater production: Dancing in Cyberspace. It was composed of acrobats dancing within and around virtual objects on their physical stage.
- In 1999, NASA X-38 spacecraft was flown using a Hybrid Synthetic Vision system that used AR to overlay map data to provide enhanced visual navigation during flight tests.
- In 2000, Hirokazu Kato created the ARToolKit, an open-source software library that utilized video tracking to overlay computer graphics on a video camera. The kit is still used widely to compliment many AR experiences.
- In 2009, ARToolKit brought AR to web browsers.
- In 2013, Volkswagen MARTA app (Mobile Augmented Reality Technical Assistance) provided virtual step-by-step repair assistance, allowing service technicians to foresee how a repair process will look on the vehicle in front of them.

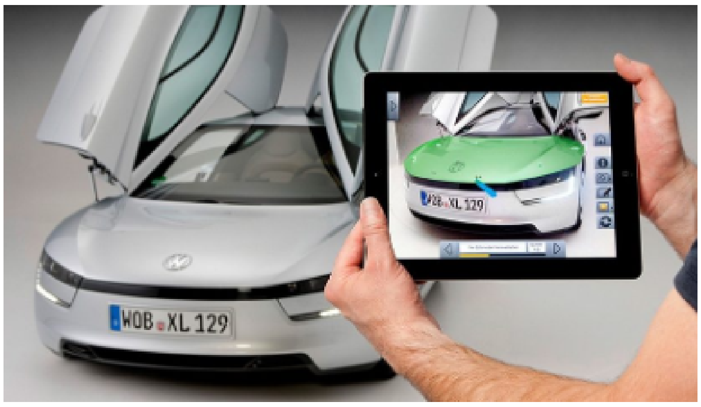


Figure 2.2: Volkswagen's MARTA application [5]

- In 2014, Google announced shipment of Google Glass devices for consumers, thus starting the trend of wearable AR.

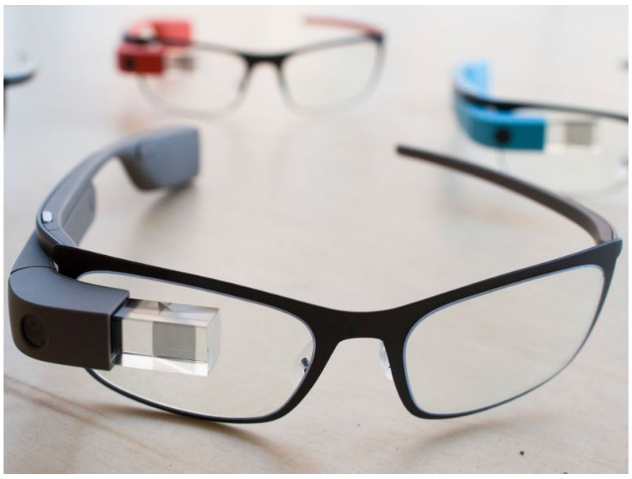


Figure 2.3: Google Glass [6]

- In 2016, Microsoft HoloLens Developer Kit and Meta 2 Developer Kit set to ship this year.

2.2 AR Technologies and Architectures

There exist several ways to implement an AR application. It mostly depends on the objectives set and the type of technology involved.
Based on this technology, AR can be divided into the following categories [7]:

- *Marker Based* or *Image Recognition* AR uses a camera from any device to recognize a simple pattern marker, such as a QR code, that are quickly differentiated and do not need much processing to be read. The idea is to produce a result whenever this marker is identified by the camera. Generally, some kind of content is displayed overlaying the marker.

- *Markerless* AR is the most broadly implemented application. It uses a device's GPS (digital compass, speedometer or accelerometer) to provide location information. The reason for it to be so spread, is the large number of available smartphones and their location detection features. Mapping directions or nearby businesses search are the most widely used applications.

- *Projection Based* AR displays artificial light onto real world surfaces and senses the human interaction on this light. This interaction detection is possible by discerning a known projection from an altered projection (because of human interference). Another path to this category is the projection of three-dimensional interactive holograms into mid-air using laser plasma technology.

- *Superimposition Based* AR partially or fully replaces an original view of an object with a newly augmented view of it. Certainly, the object recognition plays a crucial role: the original view cannot be replaced if the object is not differentiated by the application. A consumer-facing example is the display of virtual objects extracted from a catalogue into the users' environment.

Regarding the display of virtual objects, there are three defined paradigms. They differ from one another in the way the technology is used to display correctly these virtual objects into the users' view.

See-trough The first paradigm is based in see-trough devices, which allow users to see the real environment with their own eyes and virtual assets are overlapped by on optical effect. This solution demands special devices such as AR glasses which can be monocular or binocular (making possible to precisely perceive 3D display). The virtual items can be projected over lenses or on semitransparent monitors placed between the eyes and the lenses.

Hand-held The second paradigm is based on mobile devices such as smartphones and tablets that include all the required hardware to implement an AR system: a camera to record video of the real world, a display to show the augmented situation and the computational power to compute the position and orientation necessary for the camera and to combine the generated virtual assets with the video. This approach was named Mobile AR (MAR).

Monitor-based is conceptually comparable to the hand-held paradigm, but the camera, the display and the computational power are not encompassed in one unique device. This option is used whenever a large display is needed, or the camera must be independent. Obviously, this kind of display entangles several challenges.

Firstly, dealing with performance, primarily related to the camera's position and orientation computation. Another issue involves the precision of aligning the virtual to the real world. The last problem would be the lack of a keyboard or a mouse, which are the users' most traditional ways to interact with the virtual environment. [8]

2.3 AR Applications

The exploration of AR has led to a large variety of applications concerning any daily-based activity: maintenance, repair, sports, marketing, construction, education and other fields later discussed.

In **Medicine**, AR technologies were introduced as a tool to bring patients

and their medical data into the same space. The first real AR application in this field dates to 1986, when computer tomography data was integrated with an operating microscope.
Nowadays, the advances in medical imaging have made possible to, for instance, support diagnoses based on preoperative and intraoperative data. Yet still, this field must face three main issues: tracking precision, misperception and interaction with synthetic data.

Another remarkable field onto which AR has improved its experience is **Archaeology**. Showing computer-generated models of ruins, buildings and landscapes gives the user (for example, a museum visitor or a researcher) a new perspective of the archaeological site. An application of this kind can as well present additional location-based information or display audio and 3D-enhanced narrations. [9]

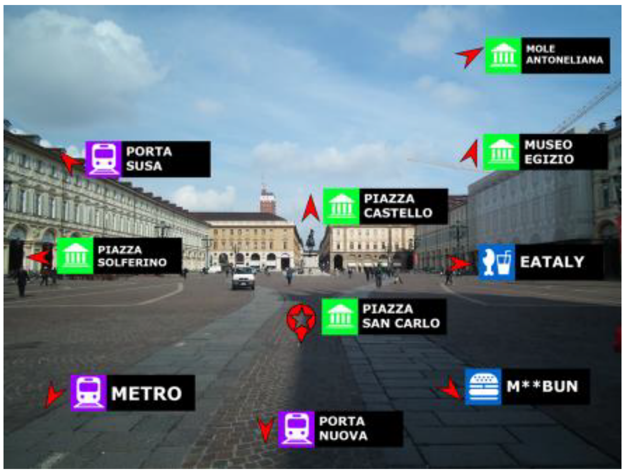
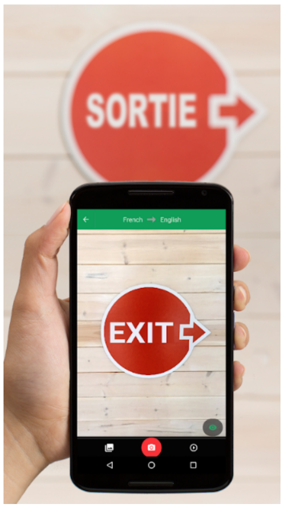


Figure 2.4: Tourism application showing user's location and nearby points of interest [8]

For touristic purposes, AR can play a key role. **Tourism** can be enriched by adding multimedia and personalized contents according to the tourists' demands. Since most tourists own a smartphone (and that these devices are already equipped with GPS and network connections), location-based AR services can be used on them. [8]

In addition to those serious investigation fields, AR is well-recognized because of other mainstream applications: mobile *apps* and videogames.

A world-wide known AR mobile application is *Google Translate*. In this case, AR is used to interpret signs and menus and display them in the desired language as printed subtitles in front of the user.



(a) Google Translate App [10]

(b) Playing Eye Toy 3 [11]

(c) Pokémon Go Display [12]

Figure 2.5: AR popular applications examples

Examples of gaming experiences could be the EyeToy saga and Pokémon Go.

Eye Toy was first released in 2003. It used its camera to augment computer graphics onto live footage.
In fact, these videogames used computer vision and gesture recognition to process the images taken by the camera. Players could interact with the games using motion, color detection and even sound. [11]

In the case of Pokémon Go, it was released in 2016. This game revolutionized iOS and Android apps market, for being a free-to-play location-based AR game that just needed the players' mobile devices to display its potential. The game depended on the mobile GPS to locate, capture, battle and train virtual Pokémon, which would appear as they were in the player's real-world location. [12]

Chapter 3

Augmented Reality Devices

Microsoft's HoloLens and Metavision's Meta 2 are two of the newest HMDs available. As Microsoft released its headset, the first of this type, few years before Metavision's did so with its second generation, the comparison was inevitable. Meta 2 was considered the main rival to HoloLens (and for some it still is, even after HoloLens 2 was released).

Not only their main features have been debated because of ergonomic issues, development documentation or pricing, but for the vast difference in their FOV: their specifications kept clear that HoloLens' 35º made Meta 2's 90º victorious in this competition. [13]

Despite of the provided statistics, there was still an unanswered question to which was the actual FOV perception of real users. And this question was paraphrased into which device would be preferred to be used by players of an AR chess game.

The hardware and software similarities and differences for these AR devices set up the basis for this work.

3.1 HoloLens

Microsoft's HoloLens was first released in 2016. It is the first fully untethered holographic computer. Using breaking new optics and sensors, it is capable of binding 3D holograms to the real world.

3.1.1 Specifications

It is necessary to remark some of this device's specifications.

Figure 3.1: Microsoft HoloLens [14]

Hardware

First of all, HoloLens hardware specifications could be summarized in this way [14]:

- Optics
  - See-trough holographic lenses
  - Two HD 16:9 light engines
  - Automatic pupillary distance calibration
  - Holographic resolution: 2.3M total light points
  - Holographic density: more than 2.5k radiants (light points per radian)
- Sensors
  - One IMU
  - Four environment understanding cameras
  - One depth camera
  - 2 MP photo / HD video camera
  - Mixed reality capture
  - Four microphones
  - One ambient light sensor
- Input/Output/Connectivity
  - 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
- Power: Battery Life
  - Two to three hours of active use
  - Up to two weeks of standby time
  - Entirely functional when charging
- Processors
  - Intel 32-bit architecture with TPM 2.0 support
  - Custom-built in Microsoft Holographic Processing Unit (HPU 1.0)
- Weight of 579g
- Memory
  - 64GB Flash
  - 2GB RAM

To be able to work with this kind of devices, it is common to fulfill some specific requirements for the computer. In the case of desktop computers, as the one used to develop this project, they need the following main features in order to work properly with HoloLens [15]:

- Processor
  - Minimum
    * Intel Desktop i5 6th generation CPU
    * Dual-Core with Hyper Threading *OR* AMD FX4350 4.2Ghz Quad-Core equivalent
  - Recommended

    - Intel Desktop i7 6th generation (6 core) *OR* AMD Ryzen 5 1600 (6 core, 12 threads)

- GPU
  - Minimum
    - NVIDIA GTX 960/1050, AMD Radeon RX 460 (2GB) equivalent or greater DX12 capable GPU
  - Recommended
    - NVIDIA GTX 980/1060, AMD Radeon RX 480 (2GB) equivalent or greater DX12 capable GPU
- GPU driver WDDM version: 2.2 driver
- Thermal Design Power: 15W or greater
- Graphics display ports
  - One for each graphics display port for headset
    - HDMI 1.4 or Display Port 1.2 for 60Hz headsets
    - HDMI 2.0 or Display Port 1.2 for 90Hz headsets
- Display resolution: SGVA (800x600) or greater Bit depth: 32 bits of color per pixel
- Memory
  - Minimum
    - 8GB of RAM or greater
  - Recommended
    - 16GB of RAM or greater
- Storage: more than 10GB additional free space
- USB Ports: one per available USB port for headset (USB 3.0 Type-A)
- Bluetooth 4.0 (accessory connectivity)

As an extra hardware source, HoloLens can be paired with the *Clicker*. It is just a peripheral device with a button and two LEDs inside it, which can exchange a click for a hand gesture, but not all of them. It usually replaces the Air Tap gesture (explained in detail in the subsequent section).

This device connects to HoloLens via Bluetooth LE. To know about the connection, it is enough to look at the white LED - blinking light means pairing - and the amber LED - solid light means failure.

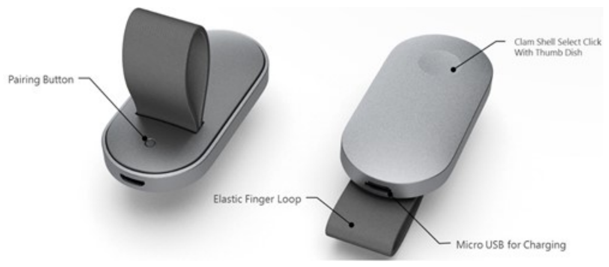


Figure 3.2: HoloLens Clicker [16]

The gaze movement is still necessary to target an object when using the Clicker. The advantage is that it is not compulsory to point it or orient it in an explicit way.

To, for example, scroll over a menu, the Clicker just needs a smooth wrist rotation to get to its maximum speed.

Its battery can be expected to last two weeks or more on a regular use on a full charge. To know its current battery situation, the white LED indicates charging when solid and the amber LED reveals low battery when blinking. [16]

Software

In addition, there are some software requirements to take into account. For this device, it is preferable to install the most recent version of Windows 10. This way, the computer's operating system will match the platform for which the applications will be built.

Another possible installation is the HoloLens Emulator. It is not mandatory, but it gives the possibility to run applications on a virtual machine image, when the developer has no physical HoloLens available.

The Windows 10 SDK is enforced to be installed. It provides the latest headers, libraries, metadata and tools for building Windows 10 applications on the HoloLens device.

This SDK has several minimum system requirements depending on the desired type of applications to be developed [17]:

- Universal Windows Platform (UWP) app development
  - Windows 10 version 1507 or higher
    - Home
    - Professional
    - Education

        * Enterprise

    - Windows Server 2019
    - Windows Server 2016
    - Windows Server 2012 R2 (Command line only)

- Win32 app development
    - Windows 10 version 1507 or higher
    - Windows Server 2016: Standard and Datacenter
    - Windows 8.1
    - Windows Server 2012 R2
    - Windows 7 SP1

Moreover, some hardware requirements are necessary:

- 1.6 GHz or faster processor
- 1GB of RAM
- 4 GB of available hard disk space

Besides the SDK, developers can keep track of all the features offered by HoloLens Development Kit via its Application Programming Interface (API).

Its documentation confers detailed information about the core building blocks that set numerous development guidelines. Those mentioned blocks provide structure to the main functionalities: gaze, gestures, motion controllers, voice input, spatial mapping, spatial sound, coordinate systems and spatial anchors [18].

Being the gestures the most frequent way to interact with the virtual world in an AR application, they are being explained thoroughly.

3.1.2 Main Gestures

HoloLens permit some interesting inputs in order to make the user interact in a comfortable and close-to-reality way. These inputs are commonly gestures, either hand gestures or gaze movements.

In the case of hand gestures, they do not need to be done in a specific location of the space. This makes the user start acting upon the virtual objects the same instant they put HoloLens on, as an immediate interaction. To adjust the target to which the hand gesture is directed, the head gaze is used. The resulting situation is a combination of looking at an object to target it and making a meaningful gesture to act upon that target.

HoloLens can recognize two main component gestures: *Air tap* and *Bloom*. These two interactions conform the lowest level of spatial input data that a developer can access. In fact, they create a basis upon which it is possible to construct multiple different user actions.

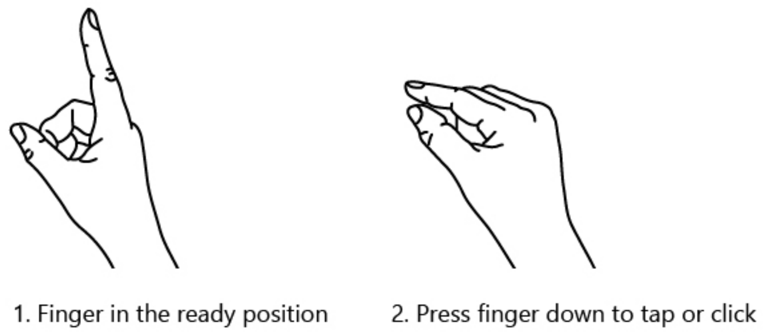

1. Finger in the ready position
2. Press finger down to tap or click

Figure 3.3: Air Tap gesture [19]

Air tap The Air tap gesture is inspired in a "click" on a mouse or a select routine. To do so, the user just have to target the entity with Gaze and perform the gesture: as simple as tapping with the hand held upright.
This "click" interaction can be used in any situation and it is easy to learn. It is considered to be a discrete gesture, but is not the only one, as any developer can create other gestures of this kind by putting together main components.

Bloom The Bloom gesture is also known as the *home gesture*. It is used to go to the Start Menu. It is referred as a special system action due to its reserved interaction.

3.1.3 Composite Gestures

After getting to know the core gestures for HoloLens, it is advisable to remark that they are not the only possible gestures to interact with.
Combining several taps, holds and releases with hand movement, new and more complex composite gestures can be performed.

Tap and Hold *Hold* simply implies maintaining the down-finger position of the Air tap. The combination of both makes possible to generate "click and drag" interactions (of course, it is necessary to make an arm movement too). This is equivalent to show a context menu or picking up an object.

Manipulation A whole branch of movements can be addressed for manipulation goals: to move, to re-size or to rotate a hologram. The perspective is to let the user move the objects around like what would be a real interaction with them. As the "click" interaction, there should be a gaze movement to target the object and then the gestures should be performed.

Navigation Similar to a virtual joystick, it can be used to access and operate UI widgets. It is commonly applied to velocity-based continuous scrolling or zooming gestures. It tries to ensure a similar experience to the one of a user clicking the middle mouse button and moving it up and down. [19]

3.2 Meta 2

Metavision's Meta 2 HMD was presented as a hard tether device. This feature has been justified to be an advantage: it gives the device the computer's resolution capability to display detailed AR models and sharp text contents. [13]

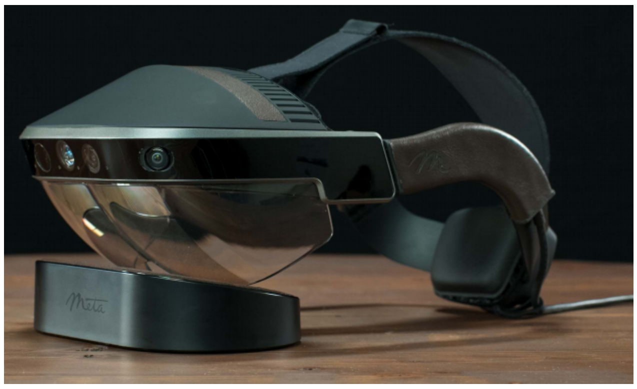


Figure 3.4: Meta 2 Development Kit [20]

3.2.1 Specifications

Meta 2 was thought to be a new holographic experience for content creators, artists and makers, as its principal goal was to re-define their workplace. [13] To achieve this, the device was designed with the following features.

Hardware

The technical characteristics of the device are [21]:

- 2550 x 1440 resolution
- 60Hz refresh rate
- Four speaker near-ear audio system
- Unobstructed view of the eyes
- Nine-foot HDMI cable for video, data & power
- 720p front-facing camera
- Sensor array for hand interactions and positional tracking
- Sensors
  - Two IMU

  - IR emitter
  - Cameras
    - Depth Camera
    - RGB Camera
    - Two Monochrome Cameras

The required computer recommendations to work and develop for Meta 2 are [21]:

- Graphics
  - NVIDIA GTX 960 *or* AMD R9 280
- CPU
  - Intel Core i7 for Desktop computer
- Memory
  - 8GB RAM
- Storage
  - 10GB
- Video
  - HDMI 1.4b
- Sound Card
  - Intel HD-compatible
- USB Ports
  - USB 3.0
- Operating System (OS)
  - Windows 8.1 64-bit (minimum)

Software

Meta 2 Development Kit includes SDK to build and share tools. It offers many built-in components and features for application development.

As well as the SDK, Meta 2 API is available. It is considered a primary source to understand its main characteristics implementation background.

Some of them are: the *calibration stage*, related to its tracking; the *Meta configuration*, showing the features that can be customized; the *palm state*, concerning the hands input; and the *virtual key codes*, keeping track of the Meta Mouse interactions.

Some of the devices' characteristics must be briefly explained in order to get in touch with Meta's mentioned technology.

3.2.2 Main Interaction Features

Camera and Sensor Features The most important part of the device are the cameras themselves, because they make possible to display the holograms correctly and keep track of the real environment where AR is happening.
To access and work with the varied cameras, some components and coding solutions are included in the kit [22]:

- **Meta Camera Rig** is a set of graphical components that provide Meta integration for Unity applications.
- **Tracking(SLAM)** means Simultaneous Localization and Mapping and is defined as the computational problem of constructing or updating an unexplored environment's map while, at the same time, keeping track of an agent's location within it. It represents Meta's tracking system.
- The **Surface Reconstruction** builds a mesh representing the current environment.
- **Meta Locking** is a script with the possibility of being integrated as a component to a *Game Object* to permit positional and rotational locking relative to the *Meta Camera Rig*.
- **Meta Gaze** is an interaction technique that lets users act upon objects in the center of their view.
- **Sensors Data** leaves free access to raw data from the devices' on-board sensors for advanced scripting.
- **Meta 2 Webcam** is a virtual device that exposes two feeds: a composite view enclosing both AR content (from Unity scene) and RGB camera feed, as a way of simulating what the user is seeing; and an only RGB camera feed.

Display Features Metavisions' device has different ways of communicating with the computer it is tethered to. The possibilities are:

- **Direct Mode**: primary rendering mode. The idea behind it is to bypass Window's display management system and to connect directly to Meta 2.
- **Extended View** is an alternative to the previous mode which uses Windows display management system as an intermediary to render to Meta 2.
- **Meta Compositor** controls rendering and enables advanced graphic functionality.

In the case of the device's audio, the main functionality can be found in **Meta Audio**, which connects to Unity Audio Manager to set the optimal project settings for playing audio on Meta 2.

In addition to the cameras already explained, a critical Meta 2 factor is the interaction. To start with, the main permitted interaction features are:

Meta Hands is the system that allows the user to add a collection of hands interactions to objects within their scene. This feature was thought as a way of interacting with the holographic objects in a natural way.
Therefore, the hand gestures are not composed of specific restrictive movements, but actually the device tries to understand the usual hand actions that a human can make with the computer-generated objects displayed.

The SDK provides a bunch of different hand gestures to be used within the scenes that provide distinct results to the objects. These called *interactions* are actually scripts to be attached to the desired objects on a scene, so the moves can be recognized. Adding more than one interaction to the same object, makes it possible for a multiple-action hologram. [23]

- Standard interactions
  - **Two Hand Grab Scale Interaction** to translate and scale an object with two hands.
  - **Two Hand Grab Rotate Interaction** to translate and rotate an object with two hands.
- Supplementary interactions
  - **Orbit Rotate Interaction**: touch to rotate in an orbital manner.
  - **Turn Table Interaction** to rotate a carousel about the Y axis.
  - **Turn Table Swipe Interaction** is a carousel interaction with discrete steps.
  - **Two Hand Scale Interaction**: grab with two hands to scale but not allowing translation.

   - **Two Hand Grab Switch Rotation Interaction**: grab with two hands, and depending on their orientation, rotation around either X o Y axis.

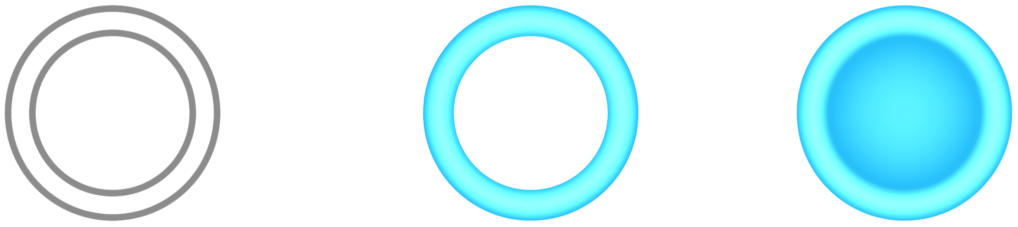

(a) Hand recognition (b) Possible hand action (c) Object is being grabbed

Figure 3.5: Meta Hands' detection and information display

In case these already coded hand-gestures scripts are not sufficient for the interactions required in a project, Meta SDK provides direct access to hand tracking data. The *Hands Provider* system gives tracking information to support the implementation of custom hands-based interfaces. The information that can be managed for both left and right hands are:

- A hand is detected
- Current position of the center of the hand
- Hand is in a closed position
- Hand is in an open position
- Current position of the top of the hand

Meta Mouse In addition to hand interaction, a new interpretation of the common computer mouse is available and it is called Meta Mouse. It is an input method that supports the use of the computer mouse in the 3D AR environment. When it is activated, the Windows mouse position is locked and a special mouse cursor is visible in the application window for the user to interact with the 3D display [24].

Figure 3.6: Meta Mouse prompt when it is activated

Meta Gaze Moreover, another possible feature is available to interact with objects placed in the center of view. It is composed of a ray cast from the center of

Meta Camera Rig in the look's direction. It is usually used to change the colors, highlighting and materials of the objects identified as the gaze interaction moves and focuses on them [25].

Chapter 4

Design

The first idea for this Thesis was to implement a table-top game that could confront a Virtual Reality player and an Augmented Reality player.

The scenario would be deployed as an AR user playing on top of a real chess board with real pieces and seeing the virtual pieces of the opponent on top of that same board; and a VR user playing in an immersive virtual environment representing the full chess board experience.

This decision was based in a previous work concerning this kind of mixed environment that used the Microsoft's HoloLens as AR device and Oculus Rift as VR device.

4.1 Useful Contextual Terms

4.1.1 Virtual Reality

To begin with, Virtual Reality (VR) refers to a simulated or immersive three-dimensional computer-generated environment where the player becomes a part of it and is able to explore their surroundings and interact with this virtual world.

The development of this type of technology is not only attached to the entertainment industry, such as films and videogames, but to medicine and other serious applications too. [26]

To experience VR, some devices are nowadays available. One of the most well-known headsets is Oculus Rift.
It was released in 2016 as Oculus' first-generation device. The whole VR experience is obtained through the headset itself, two touch controllers and two sensors. [27]
The headset provides the display engine via its custom optics and achieves high

visual fidelity thanks to its wide field of view. The controllers confer an intuitive hand presence in VR and the sensors track constellations of IR LEDs to transcribe the player's movements into VR.

4.1.2 Platforms

The project was developed in Unity, a cross-platform game engine with which it is possible to create three-dimensional, two-dimensional, virtual reality and augmented reality games [28].

In order to understand following references to Unity basic elements, they are being reviewed:

- A **Scene** contains the environments and menus for a game. Usually, games are built from several scenes, considering each scene a different level; but in this case, there will be a unique scene: the chessboard display. [29]
- A **Game Object** is the most important concept in a Unity Editor. It can be defined as any object, from characters and collectible items to lights, cameras and special effects. It cannot do anything of its own. [30]
- A **Script** is a coding file that respond to input from the player and arrange for events in the game to happen when they should. It gives faculties to the Game Object it is attached to, for example, physical behaviour. [31]
- An **Asset** is a representation of any item that can be used in a game or Project. It might come from a file created outside of Unity such as a 3D Model, an audio file, an image or any other supported file. [32]
- A **Prefab** is a stored Game Object with all its components, property values and child Game Objects that can be configured and reused in any scene as an Asset. [33]

The chosen coding language was C#, as it is a possible scripting option to develop for Unity. The IDE paired with the project was Microsoft Visual Studio, for its code editor supporting IntelliSense (code completion component) and code re-factoring. [34]

4.2 Chess Game

Another important part of this work is the development of table-top games - they can be simply recognized for being played on top of a table or any other flat surface. They can be separated into different subgenres, such as board games

(*Monopoly*,*Risk*), card games (*Poker*, *Blackjack*), dice games (*Dungeons and Dragons*) and tile-based games (*Dominoes*, *Mahjong*).

Taking advantage of the fact that using an AR device, the user can play with their bare hands, these kind of games can be re-experienced: classical games can be merged with the virtual world to redefine the way the player participates in the game.

But not only that: using virtual objects can make these games' usual static boards, pieces, dices or even cards come into life. Notice that this can be design for both AR and VR environments. At the end, the selected board game was *Chess* due to several reasons.

Firstly, because of the popularity the game enhances all over the world and the fact that most of the population knows how to play in a basic way or, at least, how the pieces should be moved on the board.

Secondly, because it was the board game chosen in various previous investigations for Augmented Reality applications.
And finally, due to the large history involving computing and chess games. [For more information about chess in computing history, refer to Appendix A].

A relevant part to understand this project's development is to comprehend the basic chess rules. They can be quickly reviewed via the pieces moves (they will be better visualized in following pictures of the game):

- **Pawns** can open the game with a two-step straight-forward move. At any other time, they can make one straight step. A diagonal one step move is permitted if an opponent's piece is placed one diagonal tile away from the pawn (to capture it).
- **Rooks** can be freely moved (referring to number of tiles) while they always move along their column or row directions.
- **Knights** have the peculiar L-movement. They can move wherever (forward, backwards, rightwards or leftwards) if the move is composed of a two-tile same direction move followed by a ninety-degree change of direction for a one-tile final move.
- **Bishops** can be freely moved on the board (referring to number of tiles) while they always stay in its diagonals.
- **Queens** can be freely moved on the board, either in number of tiles or in directions.
- **Kings** can be freely moved referring to the board directions, but one step at a time.

The way to determine the end of the game and the winner is to accomplish a checkmate: the king cannot move anywhere to be save, because it is completely surrounded by opponent's pieces that can capture it anytime.

4.3 Game Interpretation

Based in an already mentioned previous work, the chess principal components were designed to give a new interpretation to the classic chess. It can be seen in the following images that the 3D interpretation of the pieces gives the game a fresh look in comparison to the usual 2D interpretation.

(a) 2D interpretation

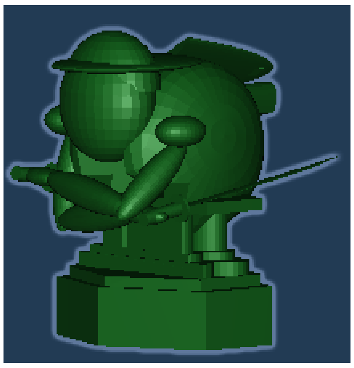

(b) 3D interpretation

Figure 4.1: The Pawn

(a) 2D interpretation

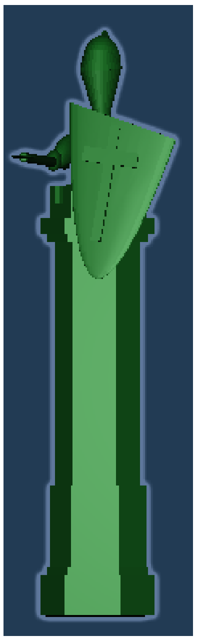

(b) 3D interpretation

Figure 4.2: The Rook

(a) 2D interpretation

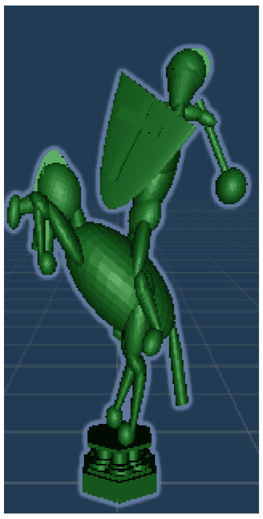

(b) 3D interpretation

Figure 4.3: The Knight

(a) 2D interpretation

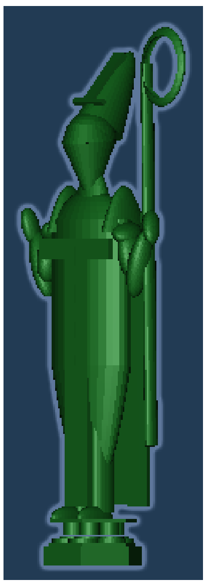

(b) 3D interpretation

Figure 4.4: The Bishop

(a) 2D interpretation

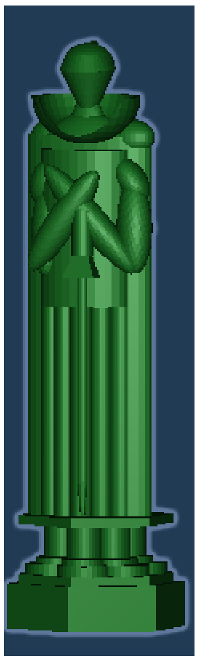

(b) 3D interpretation

Figure 4.5: The Queen

(a) 2D interpretation

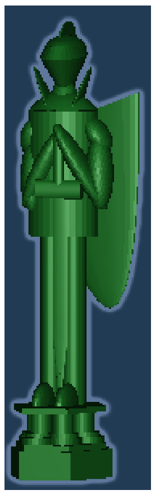
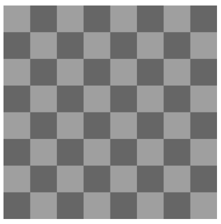

(b) 3D interpretation

Figure 4.6: The King

The chess board was designed to allow the AR player to see the pieces in their proper positions but without covering the real board. This makes the board a simple green mat that serves as tile separator.

Keeping in mind that this is the first interpretation of the board, but not the definitive, it is shown below.


(a) 2D interpretation

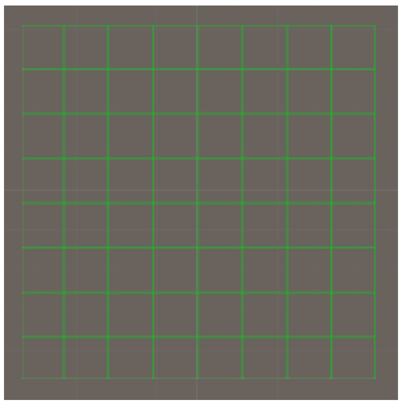

(b) 3D interpretation

Figure 4.7: The Chess Board

It is remarkable that to achieve a better experience for the players, the chess game incorporates some animations for the pieces. These animations are based on the pieces' physical design, so each piece would move distinctly to the ones of different type. Following this path, a piece can be animated in three different ways depending on the action it is performing:

- Whenever the piece is moved to another position, the piece would make a change of place animation, similar to sliding on top of the board.
- If the piece is going to capture an opponent's piece, the piece would approach that targeted piece's position, turn to it and make an attack move such as a sword movement. Then, it would arrive to the final position and turn around (if necessary) to face the proper direction.
- If the piece has been killed, it explodes in several bits when the attacking piece has performed an aggressive move and after few seconds, it disappears from the board.

4.4 Programming Characteristics

First of all, the pieces for the game had to be well distributed: each of them in its established initial position.
Hence, the board was considered an 8x8 matrix, as these are a chessboard actual dimensions, and it would make the game run smoothly when, for instance, a search for a piece in a certain position was required.

Knowing the matrix and the already shown 3D-interpretation for the scene, it was essential to know which positions in Unity's three-dimensional axes corresponded to the two-dimensional matrix interpretation.
To achieve this, a couple of functions were implemented in order to quickly change

coordinates from one situation to the other. Obviously, not only the initial setup needed this change of coordinates, but any other scene's update too.

The primitive player-game interaction using the computer's mouse was chosen for this initial approach as the devices possible inputs had not been analyzed yet. This meant taking the Unity coordinates of the mouse whenever it was *clicked* to select a specific piece.

In fact, in the very beginning, a yellow tile followed any mouse movement to help the user perceive the current position of the mouse cursor - thus, the conversion between coordinates was even more frequent - but it was discarded in following adaptations, so it will not be analyzed in detail.

Once the coordinates, where the click had happened, were known, they were converted to 2D coordinates and used to search for the required piece. At this time, some considerations had to be taken to ensure a proper game flow:

1. The game verifies if the coordinates correspond to an actual position of the chessboard.
    (a) If the coordinates are outside of the 8x8 possible positions, they are ignored and the game waits for another click (back to step 1).
    (b) If the coordinates correspond to a correct position, it continues.
2. The game verifies if that position contains a chess piece.
    (a) If there is no piece in that position, the current coordinates are ignored and the game waits for another click (back to step 1).
    (b) If there is a piece in that position, it continues.
3. The game verifies if that piece belongs to the current player.
    (a) If it is an opponent's piece, the current coordinates are ignored and the game waits for another click (back to step 1).
    (b) If it belongs to the current player, the piece is selected.

When the piece is selected, the game waits for another input from the mouse: this time to know where to move the piece. The procedure is like last one:

1. The game verifies if the coordinates correspond to an actual position of the chessboard.
    (a) If the coordinates are outside of the 8x8 possible positions, they are ignored and the game waits for another click (back to step 1).
    (b) If the coordinates correspond to a correct position, it continues.

2. The game verifies if that position contains a chess piece.
    (a) If there is no piece in that position, it jumps to step 5.
    (b) If there is a piece in that position, it continues.
3. The game verifies if the piece belongs to the current player.
    (a) If it is an opponent's piece, it continues.
    (b) If it belongs to the current player, the current coordinates are ignored and the game waits for another click (back to step 1)
4. Capture the opponent's piece placed on top of the selected position (and it continues).
5. Move the selected piece to that new position.

Of course, each time a piece is captured or moved, the game situation must be updated. In the capture situation, that piece has to no longer appear available (it is no more displayed on the scene). After a move, the previous piece position must be freed and the new position must be occupied by this piece.

Plus, the turns were controlled, in order to disable the input for the user that had already make a move until the other player moved a piece and so on.

4.5 Software Architecture

In order to achieve a well performed chess game, several scripts were coded. These scripts contained the main functionalities for the game such as the ones previously mentioned: update the game state, consider the available moves for a specific piece and alter the displayed hologram based on each current situation.

To understand the way all these methods work together to accomplish these goals, it is advisable to present the applications' software architecture. To do so, the main functionalities for each script will be explained as follows:

- **Game Manager**: its main goal is to store the game logic. This means it keeps track of, for instance, the allowed pieces' moves and the initial chessboard setup. It is the game's core script and it is usually used by the other parts of the game.
- **Board**: it monitors the holographic display of the pieces. The same way, it handles the highlighting of individual pieces.
- **Geometry**: it is responsible for the coordinate's conversions. It can convert

either 3D-coordinates into row-column notation for the matrix representation or matrix coordinates to 3D ones.

- **Piece**: it defines enumerations for any instantiated piece, as well as contains game logic to determine valid moves. For each piece it is defined a specific script that inherits from this main script.
- **Player**: it makes possible to discern between the pieces from each player. It also keeps track of the pieces available at the game and the ones that each player has captured from the opponent. The permitted directions for the pieces are as well revised.
- **Tile Selector**: it responds to the player interactions that correspond to the selection of a piece to move. It interacts to other parts of the game to, for instance, highlight the selected piece or check the possible moves for it.
- **Move Selector**: it is responsible for the proper move of a piece, analyzing if the move is possible or if it entails the capture of a piece.
- ***Piece*Animation**: each type of piece has its own script (substituting *piece* by its type in the script's name). It controls the animations that the pieces have to perform in each given situation.

Knowing the core activities for each part of the application, it is remarkable to put them together to form the appropriate game flow or script dependencies.

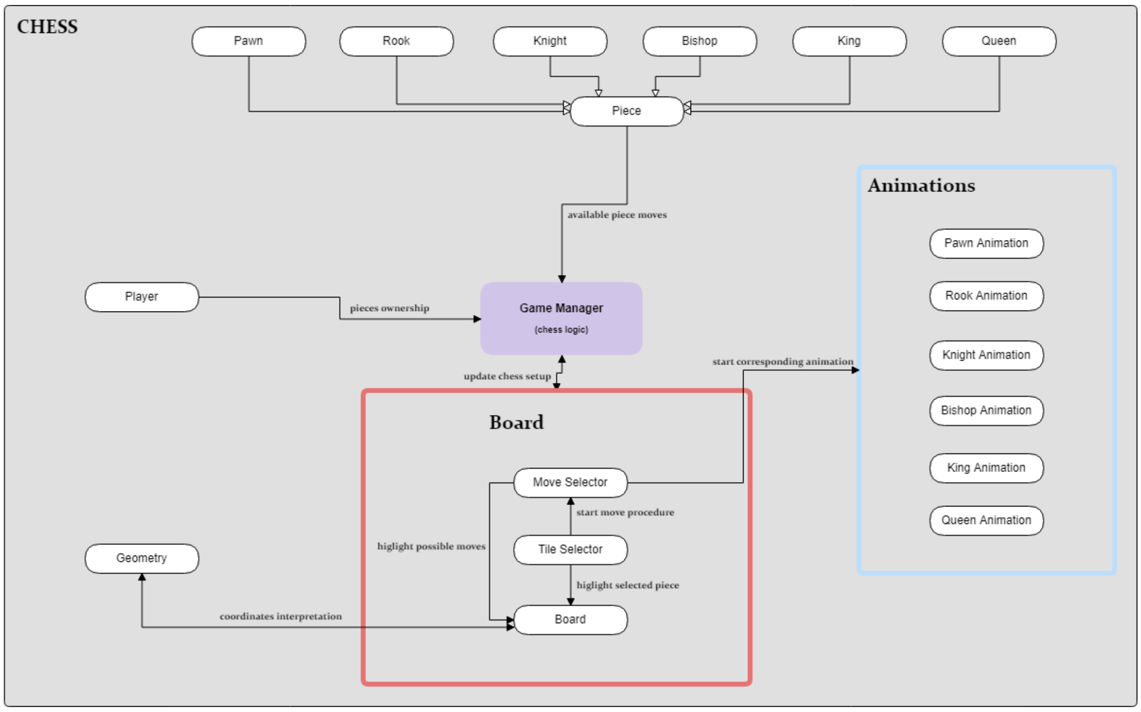


Figure 4.8: Interaction between Game Components

4.6 Color Selection

During the explained implementation process, other customized features were included.

It is worth noting that the initial pieces were white and black for being the colors generally used in chess. But, for visualization issues during high presence of light on the devices, they were changed to green and red, only for being opaque colors and distinguishable from one another.

To each piece (Game Object) a behavior script was attached. These scripts' core functionality is to determine which are all the possible moves for a specific chosen piece.
This information was not only required to play accordingly to chess rules, but to introduce color-based highlighting to the game, so that the player could receive a slight support to decide their moves.

The already explained playing procedures can now be shortly disclose in terms of colors:

1. Whenever a piece is chosen to be moved, it is completely displayed in *yellow* so that the player realizes that the desired piece has been accurately selected by the game.
2. Once the piece is highlighted, all its possible moves are shown by coloring in *blue* the tiles where the piece can be moved to.
3. If in any case, one or various possible moves imply catching any opponent's piece, the tiles are then colored in *red*.

The best way to understand these color conditions - as well as the already explained pieces moves - is to show them graphically.

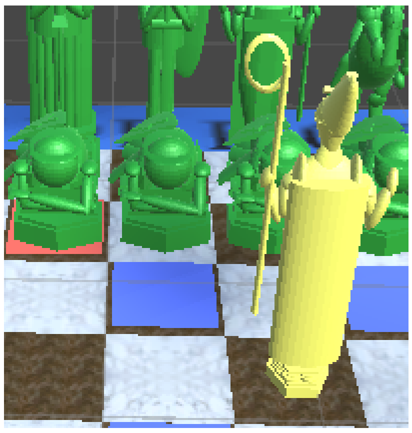
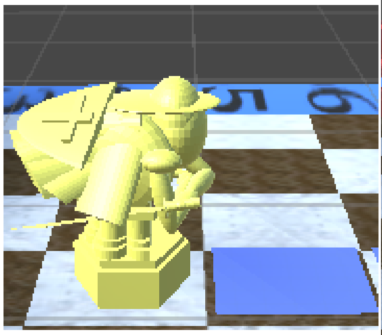
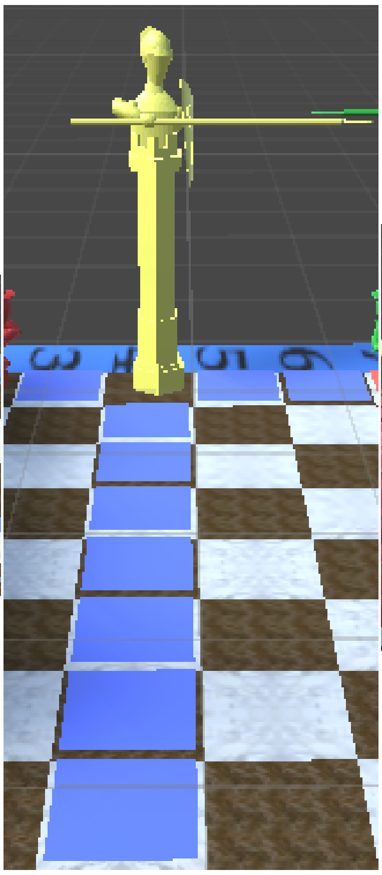
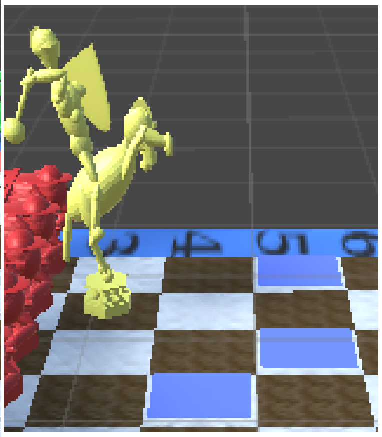
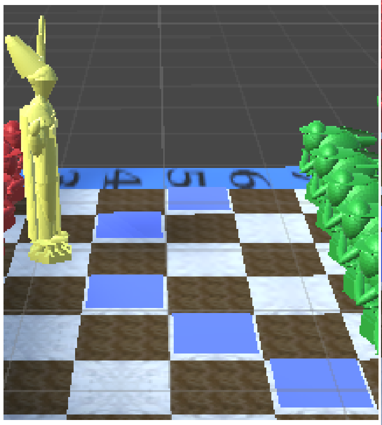


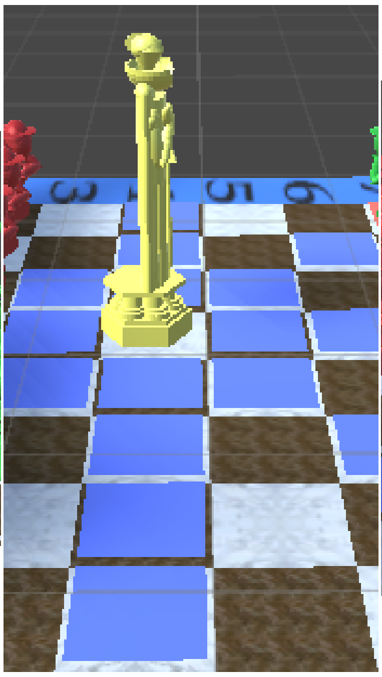
Figure 4.9: *Red* highlighting represents possible capture of a piece


(a) Pawn Moves


(b) Rook Moves


(c) Knight Moves


(d) Bishop Moves


(e) Queen Moves

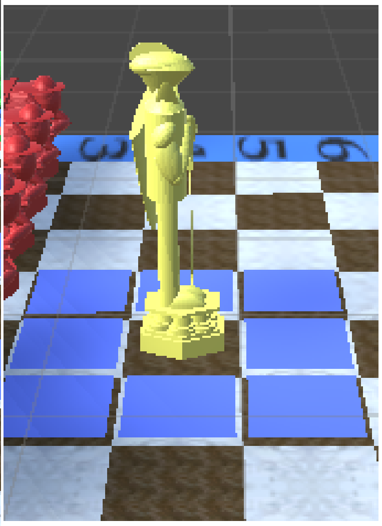

(f) King Moves

Figure 4.10: *Blue* highlighting represents possible moves

The aforementioned colors to identify different game states were chosen based on significant studies about the emotions that colors may evoke from players. Initially, Plutchik research [35] was revised in order to comprehend the basic emotions that each color could arise in each person:

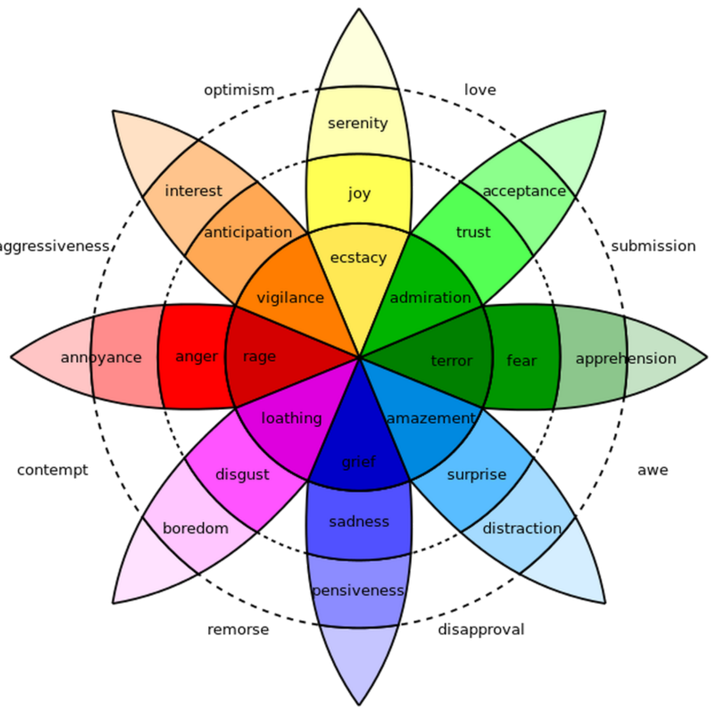


Figure 4.11: Plutchik's Wheel of Emotions [35]

Moreover, an interesting investigation about colors in videogames, inspired by Plutchik's work, was found. Based on the created links between emotions and colors, some experiments were taken to confirm the users' reactions while playing in different color environments.

The selected colors for this specific experimental work were:

Emotion	Color
Surprise	Light Blue
Fear	Dark green
Joy	Yellow
Anger	Red

Table 4.1: Emotion-color associations as represented by Plutchik [36]

The investigation found relevant effects on players' emotional responses with the colors *red* and *yellow*. Indeed, the conclusion was that *red* evokes a highly aroused, negative emotional response and that *yellow* evokes a positive emotional

response.
Furthermore, it was advised to game developers to utilize these specific colors to manipulate player's emotions. [36]

Going back to the chess game implementation, the decided tile-highlighting colors were blue and red.

As shown before in the previous picture, blue can evoke pensiveness, a desired emotion when the player should think calmly in its next move.

Following the experiments' conclusion, red arouses the player, so it was decided to be applied to remark the possibility of capturing a piece, as a way of calling out the player to commit this action and no other.

Finally, the pieces' materials changes to yellow when they are selected. This color was chosen because it gives a positive emotional response. The idea was to recall a joyful feeling in the players when they watch the wanted piece well selected.

Chapter 5

Implementation

By the time the basic chess game was implemented, a new AR device became available: Meta 2. The most remarkable feature of this new HMD was its FOV: 90º (it more than doubled the one of HoloLens).

As in a past chess game implementation, HoloLens had been already tested, a new idea came along: instead of improving that work (the initial thoughts), it was a better solution to implement this chess game for the new device, as it would be the first AR chess game for Meta 2.

The first step to achieve the new goal was to understand how the new device worked.

5.1 Meta 2 Evaluation

As for any initial work with a new device, the first task was to install all the necessary software. Luckily, apart from the specific device's software, it was not needed to work with any other platforms than the already considered: Unity and Visual Studio.

5.1.1 Meta 2 Tools

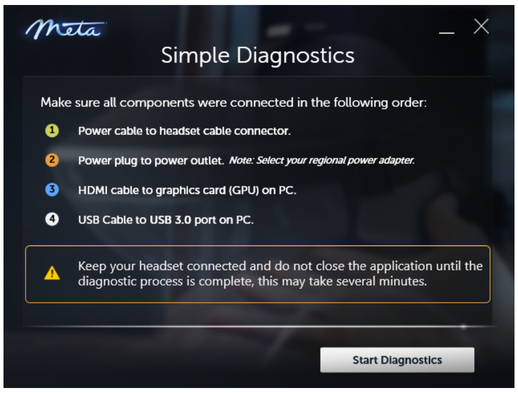
The device's software required a simple installation and then, it provided the main support features for the headset. Only the most used features are going to be disclosed.

For starters, the software brings some utilities.

Setup Headset is a computer application that is only required to be run the first time the device is plugged to a new computer.

This program guides the device's developer or user through the connection process and ensures everything is working as expected. Then, it starts an application to adjust the visual elements of the display.

Headset Diagnostic evaluates the current situation of the device and alerts the user if there is any problem with it, such as any sensor issue or any lacking wire connection (evidently the headset has to be plugged in while the diagnostic is running).

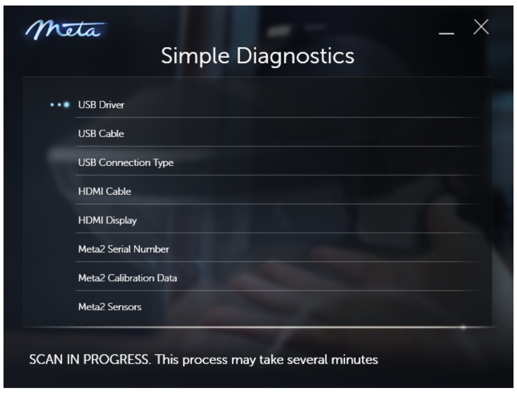

Simple Diagnostics

Make sure all components were connected in the following order:

1. Power cable to headset cable connector.
2. Power plug to power outlet. *Note: Select your regional power adapter.*
3. HDMI cable to graphics card (GPU) on PC.
4. USB Cable to USB 3.0 port on PC.

Keep your headset connected and do not close the application until the diagnostic process is complete, this may take several minutes.

(a) Initial warnings


Simple Diagnostics

USB Driver
USB Cable
USB Connection Type
HDMI Cable
HDMI Display
Meta2 Serial Number
Meta2 Calibration Data
Meta2 Sensors

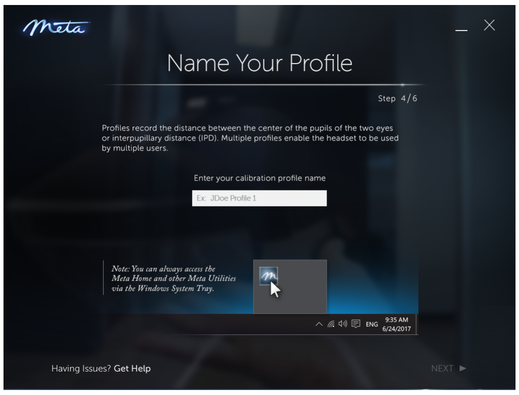
SCAN IN PROGRESS. This process may take several minutes

(b) Drivers being checked

Figure 5.1: Headset Diagnostic

Calibration Besides these utilities, another crucial feature is the calibration component. It consists in several eye tests displayed in the headset that will determine individual features a user needs for a well visualization (distance between the eyes, for instance).

The user must align some virtual objects in order to give the program enough feedback for it to understand the looking necessities of that specific user. Then, a calibration profile (with a custom name) is created, so that the user can use it whenever is working with the device. This information translates into camera configuration when, for example, a scene in Unity is played.



Name Your Profile

Step 4/6

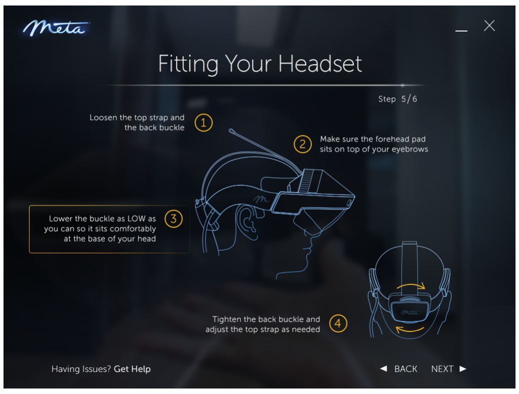
Profiles record the distance between the center of the pupils of the two eyes or interpupillary distance (IPD). Multiple profiles enable the headset to be used by multiple users.

Enter your calibration profile name

Ex: JDoe Profile 1

Note: You can always access the Meta Home and other Meta Utilities via the Windows System Tray.

(a) Name for the Calibration Profile



Fitting Your Headset

Step 5/6

1 Loosen the top strap and the back buckle

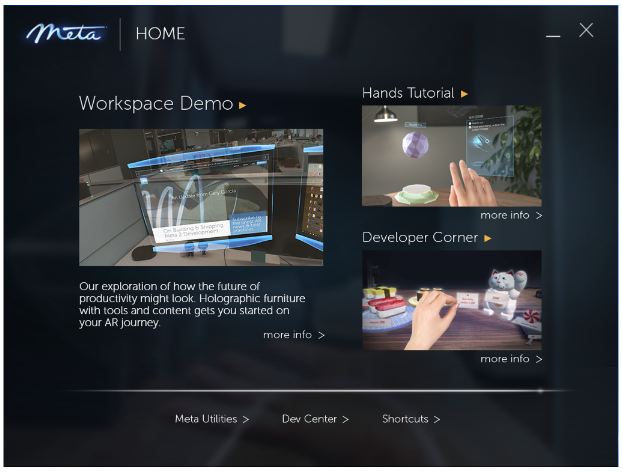
2 Make sure the forehead pad sits on top of your eyebrows

3 Lower the buckle as LOW as you can so it sits comfortably at the base of your head

4 Tighten the back buckle and adjust the top strap as needed

(b) Suggestions to wear the Headset correctly

Figure 5.2: Calibration



Our exploration of how the future of productivity might look. Holographic furniture with tools and content gets you started on your AR journey.

Figure 5.3: Meta Home Prompt

Meta Home is the last piece of software worth mentioning. It is a Meta prompt that contains three different activities or programs that can help a player or a developer in different ways:

- *Workspace Demo* presents an interpretation on how Metavision's developers imagine the future of any work station. It displays a bunch of shelves with different virtual objects, such as a brain or a bee. These objects are initially small so they can be stored in the shelves and they can be taken, moved in the space and re-sized - all of it using hands input.
- *Hands Tutorial* consists in several exercises to make the user understand how to do recognizable hand gestures. As explained in previous chapters, Meta 2 does not include limited gestures but, instead, tries to interpret the usual hand movements into already programmed actions.
- *Developer Corner* displays an AR game that tries to make developers aware

of the several input options and hand gestures that Meta 2 supports. And not only that, some designing features are visualized too, such as Meta's border highlighting.

5.1.2 Meta 2 Input Methods

An interesting fact for developing with Meta 2 was the various input methods that were included with its software. These possibilities had been already discussed in a previous chapter, but they are needed to be recalled again.

The main input features were: *Meta Hands*, *Meta Mouse* and *Meta Gaze*. This last one was the first to be considered.

Meta Gaze

One of the example scenes for the Meta Unity SDK showed that it was possible to change some virtual objects color whenever they attracted the user's attention.

The gaze input was used to accomplish this task: whenever a gaze movement ended up with the user looking to an object, the virtual object changed to a different color; when the gaze was pointing somewhere else, the object changed to a third color; and after some time without bringing looks to it, the object changed back to its initial color.

The gaze input could have been used for any other task, but the switching colors situation brought an idea for the chess game: follow the users' head movements to anticipate and detect which piece they will select to move. In this case, the yellow piece display will be done based on a focused look instead of on a click.

As interesting as it might seem, the gaze input was discarded due to a lack of information on how to change colors on a restricted part of a Game Object. The whole chessboard reacted to the gaze gestures as the pieces were part of it and there was no found solution to make a change of color in one piece at a time.

Also, the gaze input was forgot when the Meta Hands were contemplated as a better approximation to an actual chess game.

Meta Hands

The player would be able to move the virtual pieces with usual hand moves as grabbing and moving actions were already programmed features. And beyond that, the user would be able to play with real pieces while the Meta would get these normal hand moves and interpret them to update the position of invisible holograms.

After extended tests with different scene setups, the conclusion was discarding this input method too. The reality of the hand recognition was very different to the one experienced in simple one-object situations. The key word for the hand input was *inaccuracy*. Some proven faults were:

- The one-hand or two-hands interactions functioned correctly (most of the time) but when the virtual objects started getting smaller, the efficiency dropped. For the chess game, the movements had to be identifiable for the pieces, so not having a secure event-reaction solution was not an option.
- The detection of hands around the device was another issue for the game. A user can be taught how to make a grab movement and how to make a drop movement - it is the same movement that anybody would do with a real object: open one hand and close it around the object and the opposite to drop it. But, as focused as a user can be, normally the hands get relaxed, instead of staying in a completely open position. This casual hand placement was a problem for the Meta perception: in some situations, it was interpreted as a grab movement, so the pieces moved when they should not have.
- The objects around the display of the headset were few times misinterpreted as possible hands making gestures, so, even keeping the real hands in the users back, the pieces moved around with no user handling them.

After the decision to abandon the use of any type of hand input on the chess game, when earlier it seemed the best and most promising approach, the *Meta Mouse* was as well examined.

Meta Mouse

Similar to the gaze situation, the mouse interaction was analyzed in example scenes. Obviously, it was a serious issue to develop for a device with incomplete online documentation, but it seemed a good opportunity to let the user play in a 3D environment.

After familiarizing with the prefabs needed to make use of the mouse, they were included in simple scenes to understand the moves permitted and which where the necessary scripts to attach to a Game Object for it to react to the mouse interaction.

Virtual objects were found capable of being moved around with a basic mouse control: one *click* meant a *select* action; *click-and-hold* was used to grab the selected object and moving the mouse made the grabbed object move in a similar way.

The Meta Mouse was then included in the chess game and this carried several implementation alterations.
To begin with, the required prefabs and scripts had to be added. Then, the tile

responses were redefined as the input information was no longer taken from the actual computer's mouse, but from the Meta Mouse.
It is worth remarking that the device's mouse is a 3D mouse displayed in the headset, so working with it is not the same as working with the usual mouse.

An issue found for the mouse was the strange sensation of moving it physically on top of a plane surface (a table, usually) and at the same time, seeing it moving before the eyes in a 3D environment.
This happened because the action of moving a three-dimensional object trough two dimensions does not come with ease.

Although it would take some time for the user to get used to this new type of mouse perception, the Meta Mouse continued being a part of the chess game implementation.

5.1.3 Holographic Display

Apart from the difficulties found during the examination of the input methods, other concerns were related to the Meta 2 holographic display.

One of the characteristics of the prefab *MetaCameraRig* (the asset that represents and handles all the Meta 2 cameras in a Unity scene) was the *depth occlusion*.

This parameter could be checked, so that during the display the virtual objects were not occluded by those real objects interfering between the users' eyes and the virtual objects; or unchecked, so the virtual objects were partially or completely occluded by the real objects depending on where they were placed.

Playing a scene where occlusion could happen made the chessboard disappear almost completely, so a game could not be well performed, in the vast majority of the attempts.

Discarding the occlusion, for the chessboard to show up properly, made the user lose depth perception: the distances between the user and the virtual board and the ones between pieces and the mouse, were not well observed and caused visual confusion.
This found problem was finally worked out as it will be explained in subsequent explanations.

An even more serious issue involved the holographic stability of the device. This problem could be tackled into various points:

- Each time the virtual board was displayed, there was a high probability for it to appear in a different location. The positions did not extremely differ from one another, but they were indeed rapidly recognizable for not being well located.

For example, instead of being displayed straight in front of the user, the chessboard commonly showed up rotated around 30º left or right in the x-axis - it was an obvious bad disposition to play.

- In few occasions, the chessboard completely lost track of the defined position in the scene and appeared in rare placements, such as at the back of the player or very high from the users' eyes.
- The mentioned changes of display happened even after the first prompt of the virtual board. Whenever the calibration engine of the Meta 2 asked for the re-mapping of the current environment (too many times, even in controlled situations), the user had to do some specific gaze movements to keep everything on track.
  But, as this calibration took place, instead of displaying the chessboard back to a good position, it moved to one of the described ones.

These cases were a major problem for a project that was meant to be tested on real users, because in addition to the lost time, the scene display must be exactly the same for each individual test.

Trying to, at least, solve the calibration problem, one of the *MetaCameraRig* aspect was revised and the effect to the scene was to not show the calibration UI during the scene display. This way, if for any chance, the first virtual board prompt happened to be well positioned, the calibration would not misplace the board after that.

But this option brought a non-desired scenario: the Meta 2 cameras lost the perception of its position and made the hologram follow the users' head moves, as it was attached to the users' position. This was not desirable, as the best AR experience should place the virtual content and let the user move around to see in different perspectives. This alternative was finally discarded.

Knowing the holograms display issues, it was considered to change the chess board for a full-surface one. Being so, the player will see a virtual chess board and all the pieces involved in the game; no real pieces were involved in the game then, so a miss-calibration would not mean an impossibility to play.
The board was thought to be helpful to the user in another way: the coordinates for the row and columns were visible, so the player could tell quickly where each piece was placed.

5.2 AR player vs. Chess Engine

As at this time, there was no found solution to the calibration issue, the project was redefined in order to cope with all these obstacles. The first idea of a game for virtual and augmented reality was then substituted for a game where the

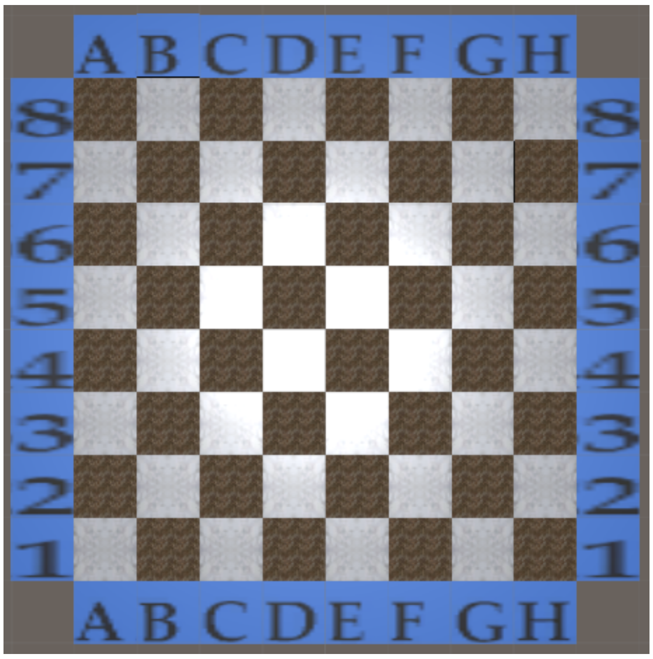


Figure 5.4: New Design for Chess Board

AR player would play against a chess engine (AI). Of course, this decision meant changes in the implementation.

The essential change to the implementation implied discarding any VR functionality and finding a way to keep a great experience for the AR user even while playing against a chess engine. Also, it had to be kept in mind that, implementing an accurate chess AI or correlating an existing one with the Unity project, would not have to mean giving up other more important features for the game.

Based on this, it was decided to develop a basic engine. It consisted in a script with two main functions: one to determine the piece that had to be moved and the other to choose a move from the possible ones. In both cases, the selection was made randomly. As said before, the goal was to have a playable game, not a perfect AI.

Being possible to play in an AR environment against a chess engine as any two people could play in a real setup, a new possibility came up: instead of just developing this game and test how real users will play, it was better to analyze how this game could be played in two different situations. In this sense, one situation was defined as playing an AR chess game with Meta 2 and the other situation meant playing an AR chess game with HoloLens.

5.3 Meta 2 and HoloLens Development

It was found interesting to know which one of the devices would be better to play with in a table-top game environment as the one created. There were important reasons behind this decision.

In first place, that as the experience AR player vs. VR player had to be discarded, the project could be led to something more than just the development of a videogame.

Secondly, because at the beginning of the development, Meta 2 and HoloLens were the most recently released devices of this kind and a comparison between them was key for future AR development.

And finally, due to the fact that it was necessary to experimentally prove if the difference between the given FOV for the devices, was noticeable for a player in a board game display (commonly in this kind of games the player has to be able to see the whole board and pieces situation).

Then, the project had to be rethought so that both Meta 2 and HoloLens scenarios matched and made possible an accurate comparison between them. It is remarkable to say that this meant a double work, as now there had to be developed two different Unity projects for two different devices but with same UI.

5.3.1 Implementation Evolution

Following this path, the Meta Mouse input method was then discarded, as there was not an exact equivalence for it among any of the available HoloLens inputs. It was then developed an equal keyboard input method for both.

It consisted on the process of the information received via the computer's keyboard. For this matter, the player had to input the selected piece's coordinates and the ending tile coordinates (the tile where the piece wanted to be moved). The coordinates were described as those of a normal chessboard, from A to H and from 1 to 8.

The game itself, each time some coordinates were introduced, translated them into the game's actual coordinates. This means that, as the chessboard was defined as an 8x8 matrix, the coordinate A1 will imply the coordinate [0,0] for the game. This conversion was done after the player had pressed the *return* key and verified that the coordinates introduced were formed by two characters. This led to the already explained situations to confirm if the coordinates belong to a possible tile and so on.

To encourage the experimentation with real users for both devices, the main priority was to be able to set a scene where the chessboard would be displayed in an exact position at any given execution time and in any of the devices.

To accomplish this, Vuforia Engine was integrated in both Unity projects, as it is an AR application development platform which counts on robust tracking and performance on various devices. [37]

A commonly used image target was then used, so that each time the game started and the camera of the wore device identified the image, the chessboard was displayed on top of it.

For HoloLens this Vuforia implementation is common and there are plenty of documentation about it. But, in the case of Meta 2, it was a brand-new approach that had to be applied with no previous support.

In fact, even after obtaining a good result for this specific scene display, it is thought that, for any other more complex situations, the Vuforia arrangement with Meta 2 would have to be deeply explored in order to work as desired.

The already mentioned position of the chessboard was part of the requirements to make an equal UI for both projects, because to test the FOV in both devices, all the distances concerning the game had to be the same, starting by the distance between the player and the board.

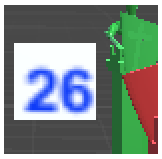
5.3.2 Testing Features

Any other components for the game had to maintain this exact UI specification. These components refer to some add-on features that were introduced in the scene for a better understanding of the player's comprehension of the game.

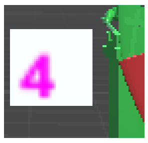
To start with, a timer was introduced. It was thought to be a countdown from 30 seconds that signified the time left for the user to make an accepted move. Also, it was expected to be a sign to keep the user aware of when it was their turn and when it was the engine's turn. If the FOV did not make possible for the user to see this timer, it would of course make a difference in the user's performance.

The timer's color lettering changed when there were 5 seconds left to the end of a turn's time. While the timer was shown in blue, in that case it flipped to pink, so it will call the player's attention.

In addition, the number of times a user made the timer go down to 0 seconds without making a move, was gathered in order to compare it for a same player in each of the devices.



(a) Count down from 30 seconds



(b) Pink lettering (< 5 seconds left)

Figure 5.5: Timer

For a similar purpose, a canvas was set to display a message whenever necessary. In first place, when the timer showed 0 seconds left for the user, a message remembered the user that even if they were out of time, the turn was still theirs, so they had to make a move to keep the game going.

When the game was finished because a checkmate had occurred, a message defining the winner of the game was also displayed.

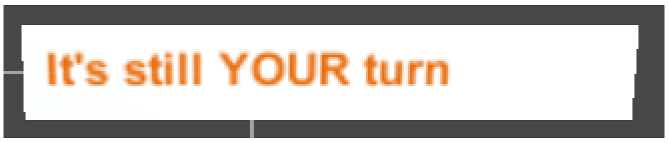


Figure 5.6: Message as reminder for the player

The large turns counter was saved into a file each time a game ended. A finished game was defined whenever a checkmate happened or when a time limit was surpassed (for the tests to get fluent, this time was set in five minutes, enough time to play and be able to realize the main facts about each device experience).

In fact, another message was displayed in the canvas when this situation happened, in order to inform the users, they could no longer play.

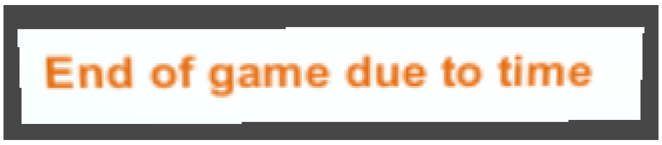


Figure 5.7: Message for the End of the Game

At this point, it is advisable to know the ultimate chess game display. For this, some pictures of the scene that would be displayed for the players are shown below.

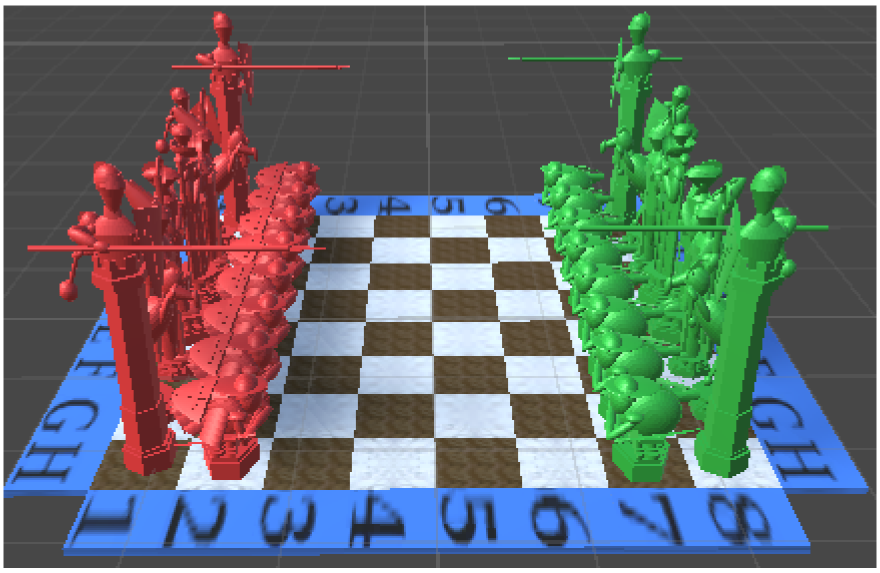


Figure 5.8: Initial Chess Game Setup

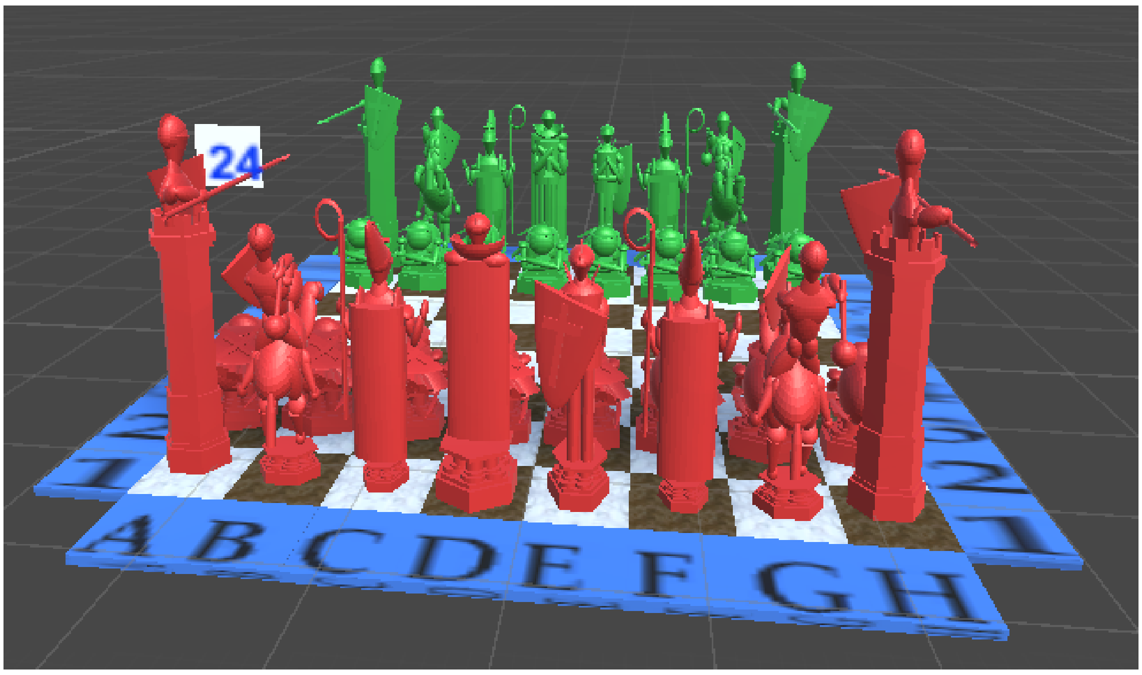


Figure 5.9: Chessboard and Timer Display

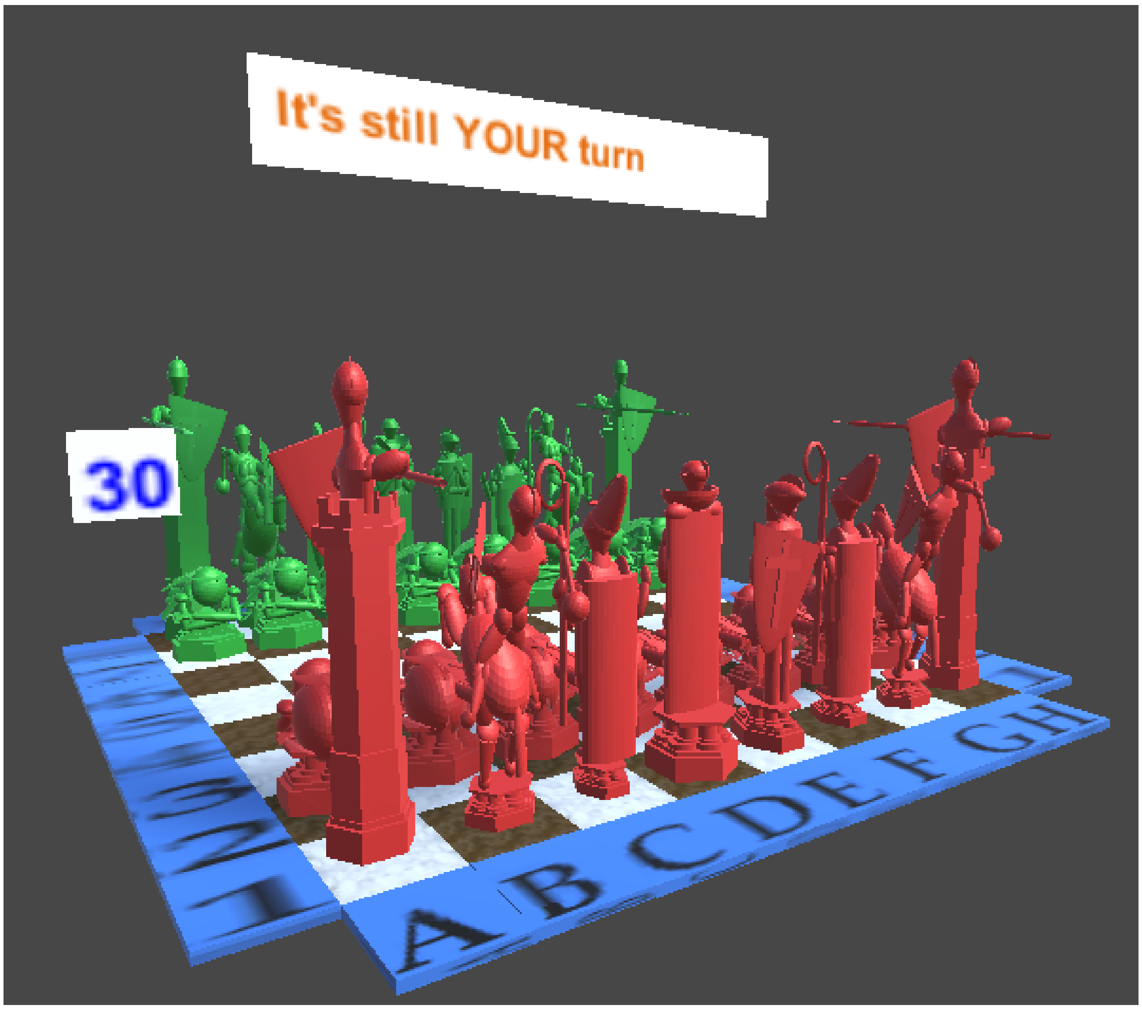


Figure 5.10: Full Board-Timer-Message Display

5.3.3 Client-Server for HoloLens

It is noticeable that for the HoloLens' project, a client-server interaction was needed. This situation only appears to this device, because it just needs to be plugged in to the computer while deploying the application for the first time, but Meta 2 is a tethered HMD.

For the communication between the server (computer) and the client (HoloLens) the *LiteNetLib* library was used, as the sending information were simple strings easy to interpret.

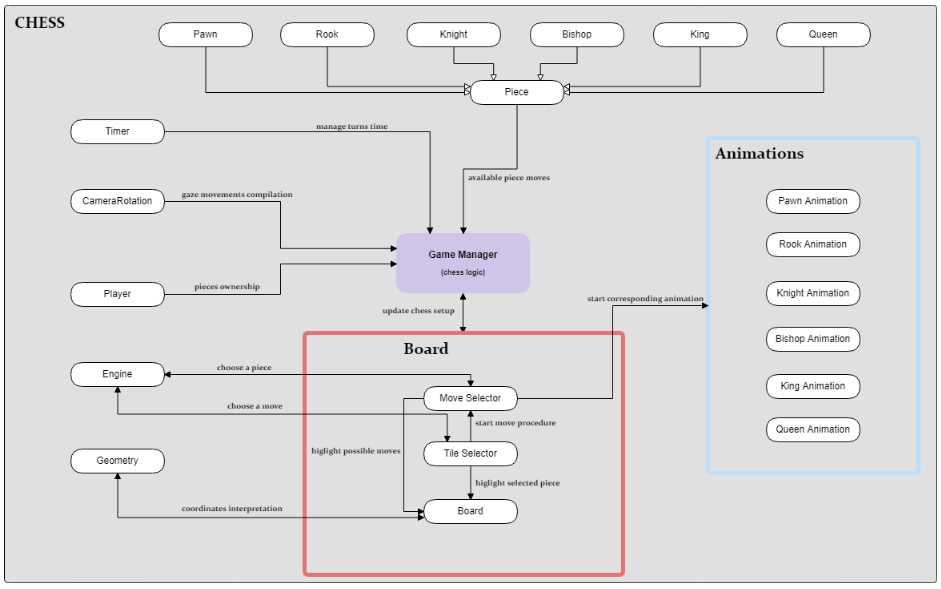
To achieve this communication, the HoloLens' project had a script concerning the main functions and some important characteristics such as the server IP and port to connect to, and a new Unity project was needed to incorporate the server script as well as important features of the game such as the keyboard input or the file saving.

5.3.4 Final Software Architecture

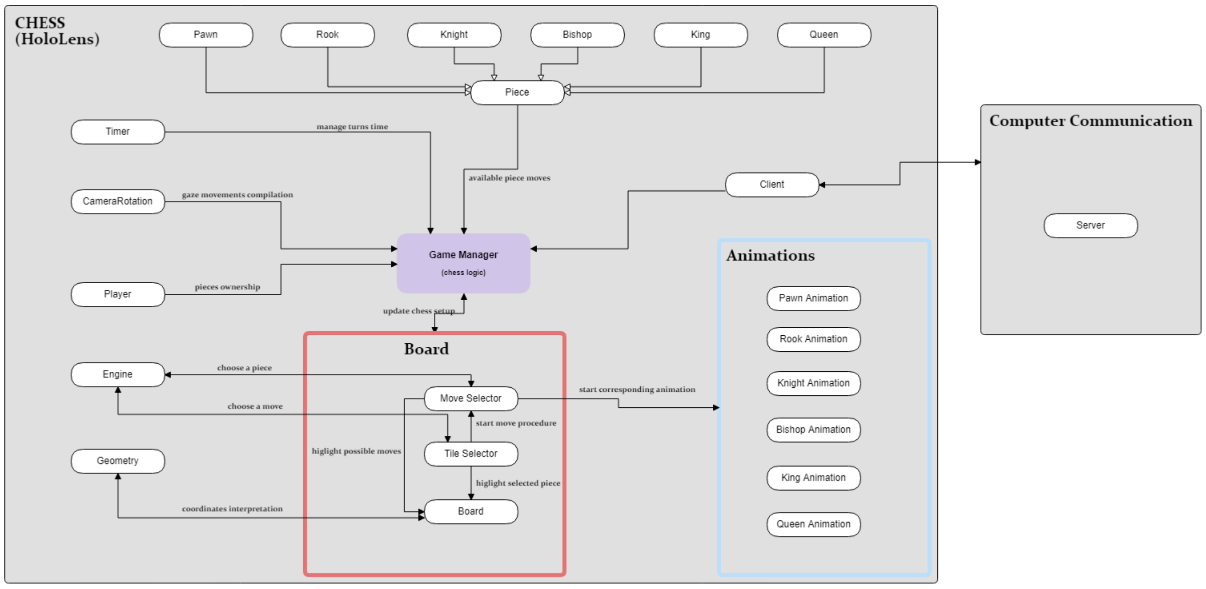
In the previous chapter, the core Software Architecture was shown for the chess game itself. But now, that all the implementation has been explained, it is important to revise this architecture and complete it with the final scripts and functionalities. The new scripts functionalities can be explained as shown below:

- **Engine**: it covers the chess AI functions for choosing pieces and moves whenever required.
- **Timer**: it controls the display of the counter and the messages for the player to be aware of the time left for a move and the turns flow. It also sums the number of turns lost during a game.
- **Camera Rotation**: it compiles the sum of the degrees made in each of the three axes in terms of the player's gaze movements during the game.
- Specific HoloLens' Scripts
  - **Client**: it contains the usual functionalities such as connecting to a server, sending and receiving information. The script corresponds to the HoloLens.
  - **Server**: it defines functions to make possible the connection of clients to it and to send and receive messages. This script corresponds to a computer which has to communicate to the HoloLens so the holograms keep updating correctly.

To understand the differences between the Meta 2 project and the HoloLens project, it is better to refer to a visual diagram of the updated version of the aforementioned Software Architecture:



(a) Meta2 Project



(b) HoloLens Project

Figure 5.11: Final Software Architecture

Chapter 6

User Experience and Testing

After the implementation of the projects, they had to be tested in real users so the goal of the Thesis could be accomplished: compare the FOV of Meta 2 and HoloLens devices.

6.1 Experiment Design

The experiments were thought to be as simple as a user playing while wearing one of the mentioned devices at a time.

First, the image target was restricted to a certain position on top of a desk for it to maintain the same distance from the player in all the done experiments. For the same reason, the player was told to stay still while sitting on a chair in front of the desk.

The chair was always found in the same place because the idea was that the distance from the player to the table was equal for all testers. Plus, they were only allowed to make head movements if some chess parts were not seen correctly, so that they would not alter the distances my moving around.

After the camera would notice the image target, the chess game would setup in that exact position (board and pieces in their initial arrangement on top of it). To avoid a bad initialization or a large explanation about how to use the devices, this step was done by the experimenter and then, the device would be handed over to the player.

The player would put on the device in an adequate way so that they feel as comfortable as possible (a bad positioning of the device could lead to fear in doing gaze movements or the implication of the player's hands to hold the HMD over the eyes).

A brief explanation on how to play this particular chess game was given to the player (the basic chess rules were taken for granted except when the player demanded them).

A summary would be that they had to say out-loud the coordinates of the tile underneath the selected piece and, then, the coordinates of the tile where the piece wanted to me moved. The experimenter would write this data down so that the game would update as expected.

6.2 Questionnaire

To evaluate each user's experience, a test was then taken.

6.2.1 General User Information

Before playing, the user would have completed a general information questionnaire to know trivial data and select which one of the devices had been tried first (the order in which the player would try the devices meant a lot to the data gathering about their FOV):

- ID: identifies anonymously each user so that the file where the degrees and lost turns information can be related to this questionnaire
- Name
- Age
- Sex
- How many times have you used an AR Application?
  - Never
  - Once or twice
  - Sometimes (once a month)
  - Frequently (twice or three times a week)
  - Every day
- How many times have you used an AR Head Mounted Display (HMD)?
  - Never
  - Once or twice

  - Sometimes (once a month)
  - Frequently (twice or three times a week)
  - Every day
- Which Interface did you try first?
  - HoloLens
  - Meta 2

6.2.2 SUS Questionnaire

After the general questions, the users had to fulfill the System Usability Scale (SUS) test in order to evaluate each of the devices. The completion of the test was made once for each device, right after playing with one and before playing with the other. The SUS questionnaire consists in the following statements:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with system.

These comments had to be rated between a 1 and a 5. 1 was set whenever the player strongly disagreed with the read statement and a 5 if they strongly agreed with it. And, if the player's opinion was not so polarized, any number between 1 and 5 was used to try and identified their opinion.

6.3 Results

6.3.1 General User Information

For the first part of the experimentation, the general user information, the results were:

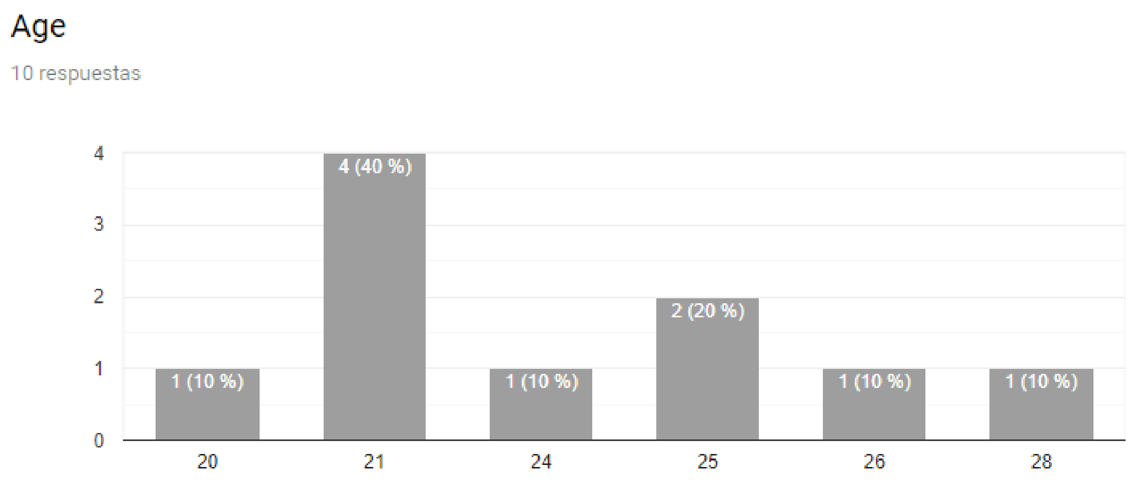


Figure 6.1: Users' Age

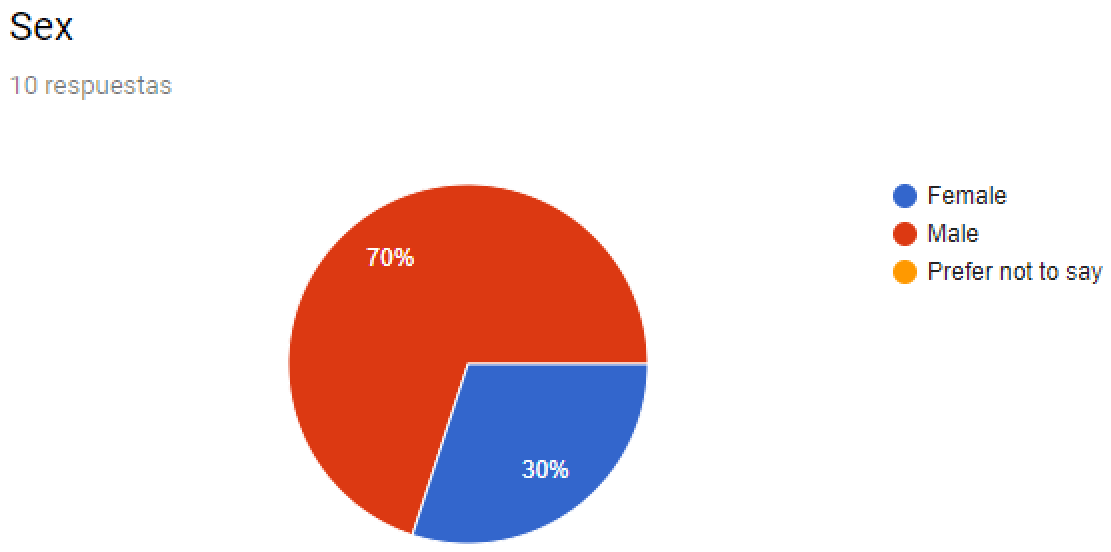


Figure 6.2: Users' Sex

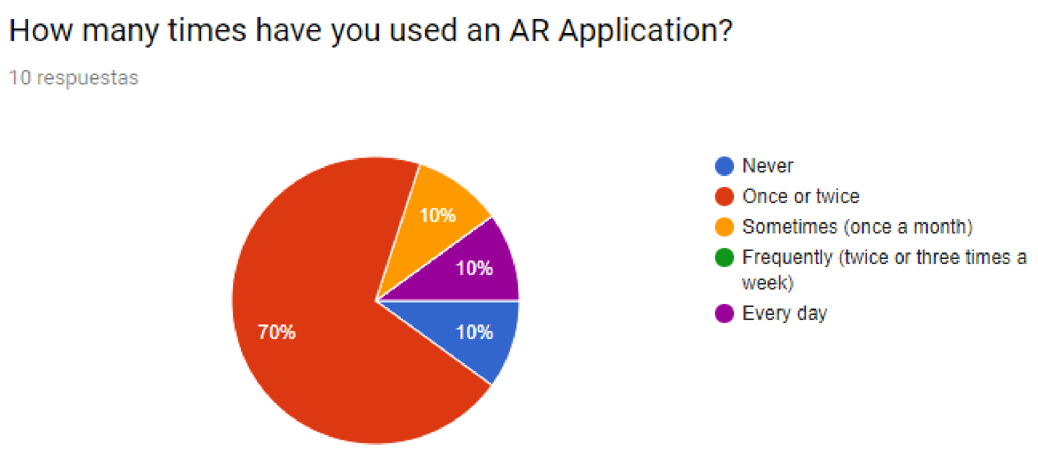


Figure 6.3: Frequency for AR Applications

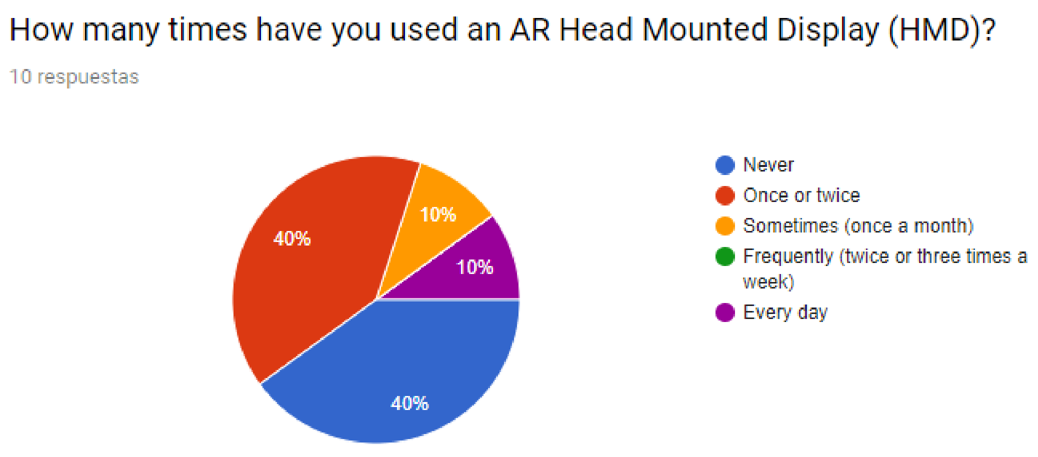


Figure 6.4: Frequency for HMD

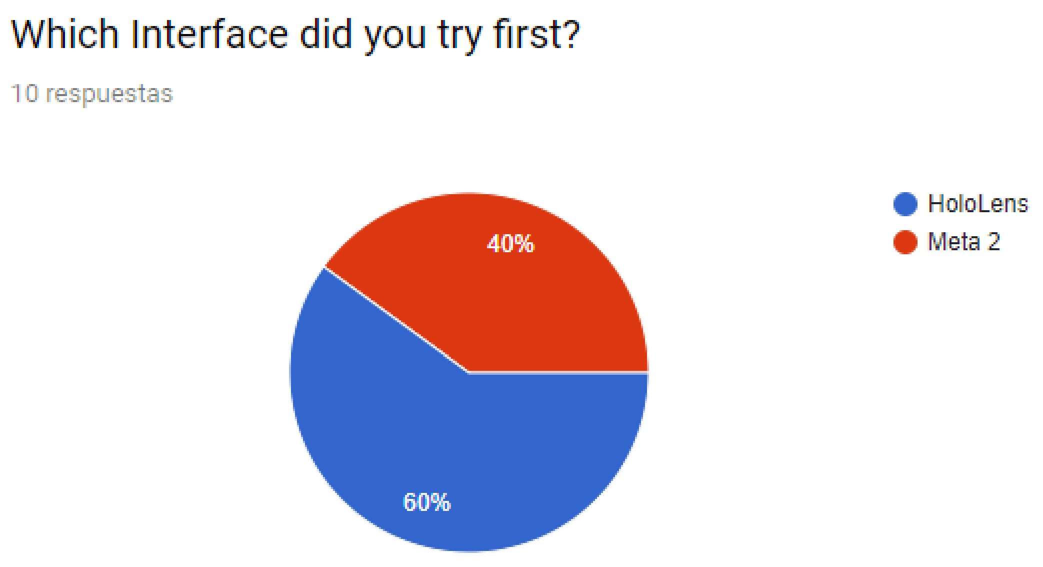
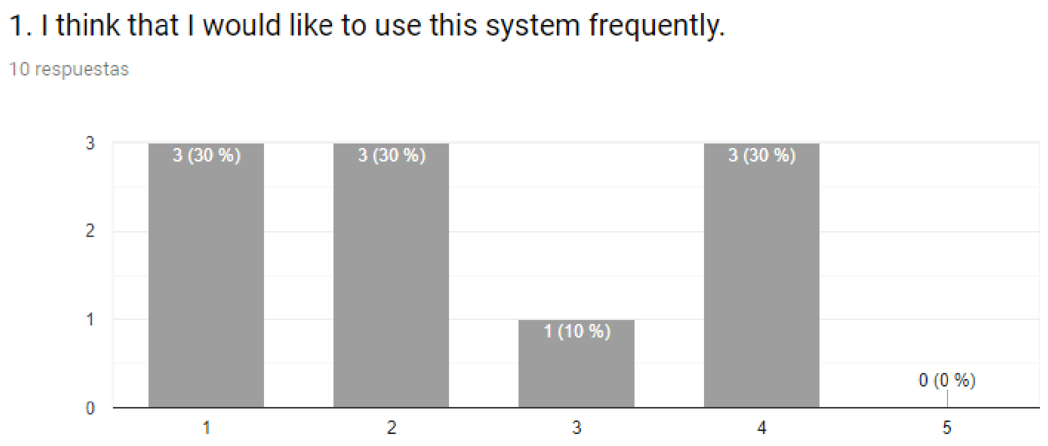


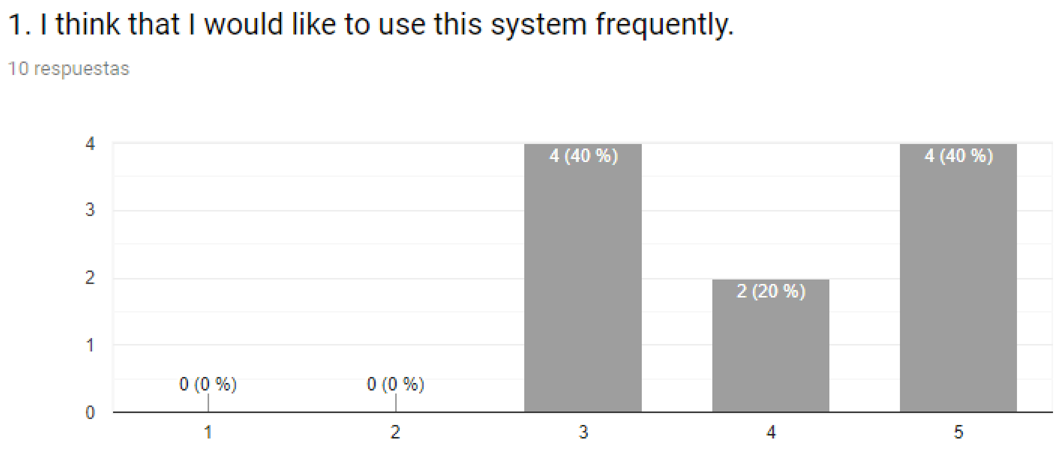
Figure 6.5: First Device Tried

6.3.2 SUS Questionnaire

The SUS questionnaire responses can be expressed in the form of bar graphs that summarize the different answers for each question. On the upper position, the qualifications for the HoloLens can be found and the Meta 2 evaluations are placed below them.

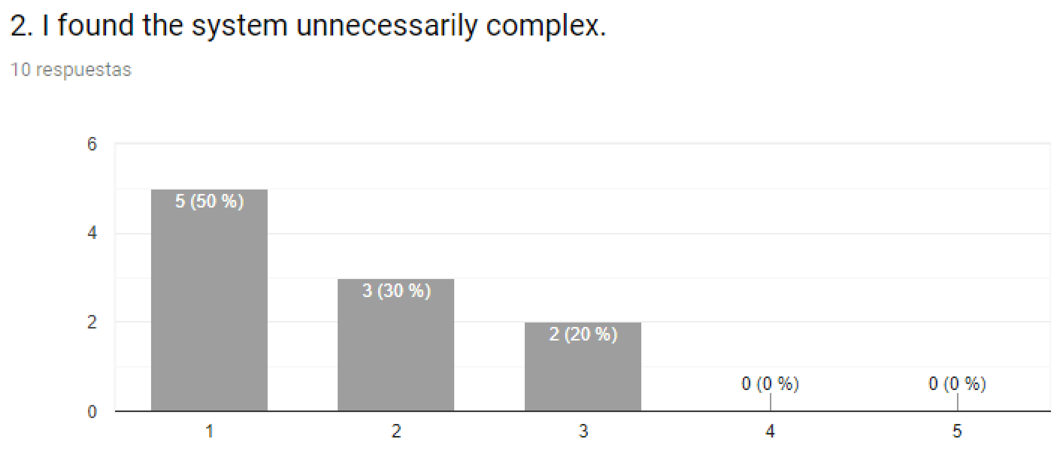


(a) HoloLens evaluation

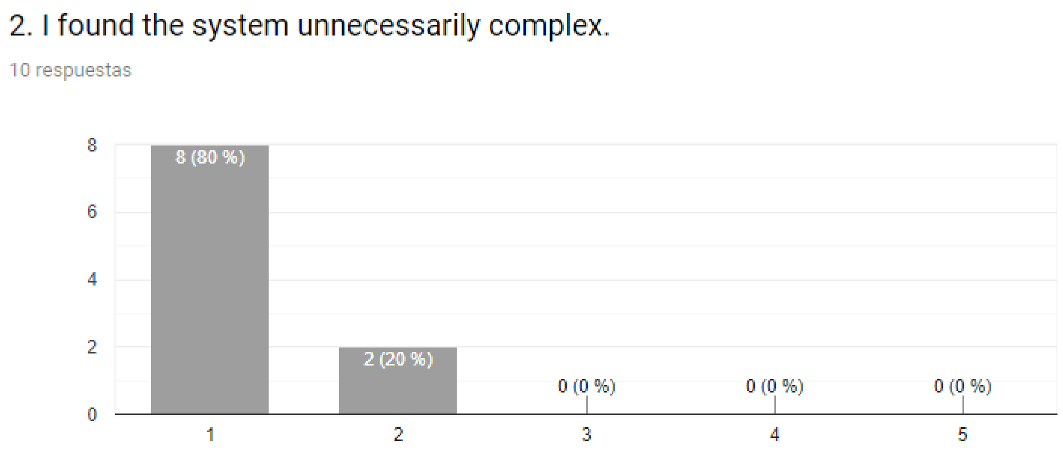


(b) Meta 2 evaluation

Figure 6.6: SUS Question 1

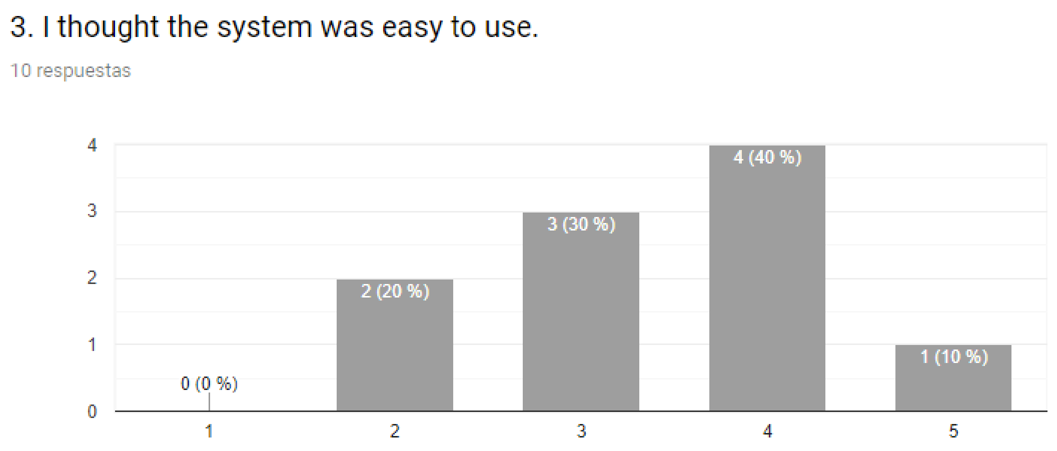


(a) HoloLens evaluation

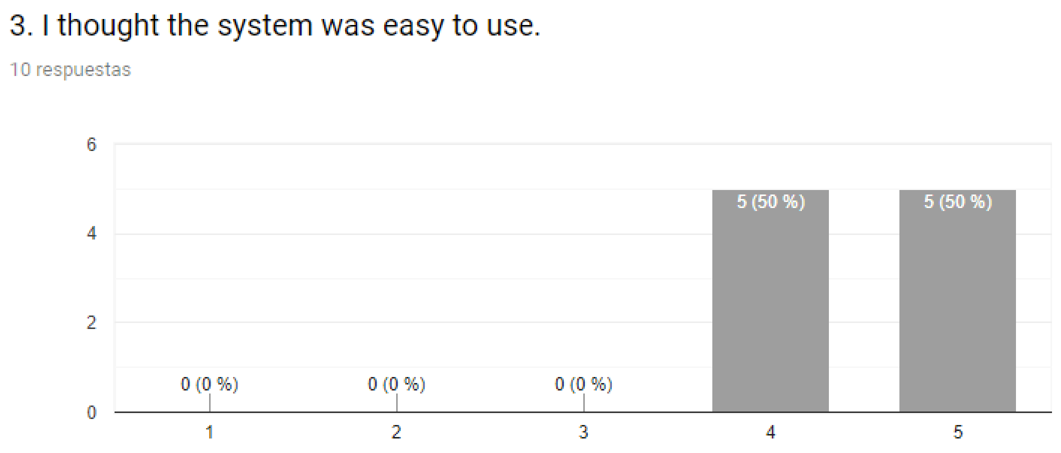


(b) Meta 2 evaluation

Figure 6.7: SUS Question 2

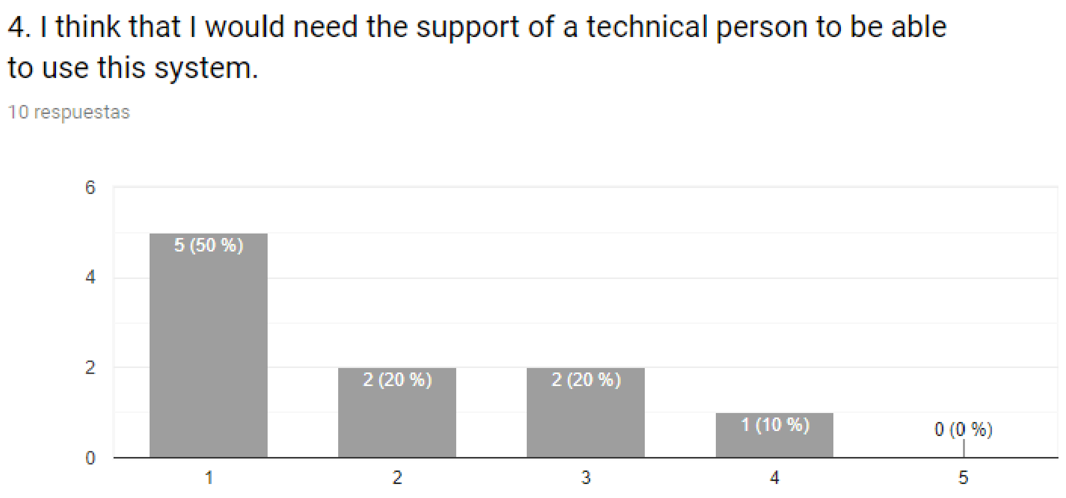


(a) HoloLens evaluation

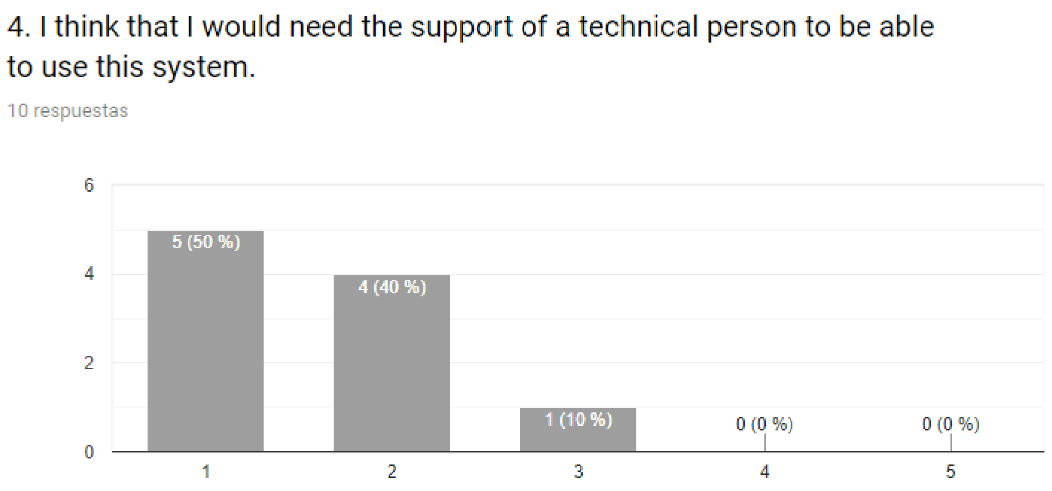


(b) Meta 2 evaluation

Figure 6.8: SUS Question 3

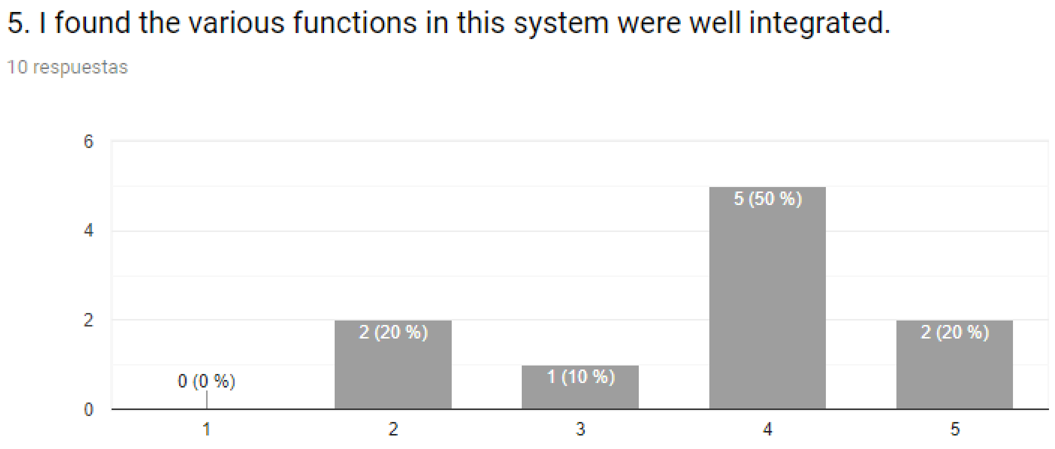


(a) HoloLens evaluation

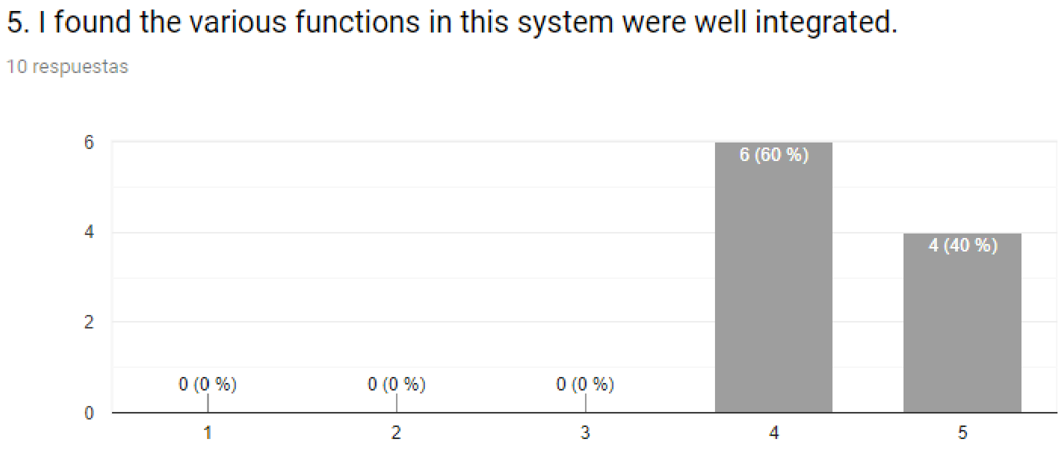


(b) Meta 2 evaluation

Figure 6.9: SUS Question 4

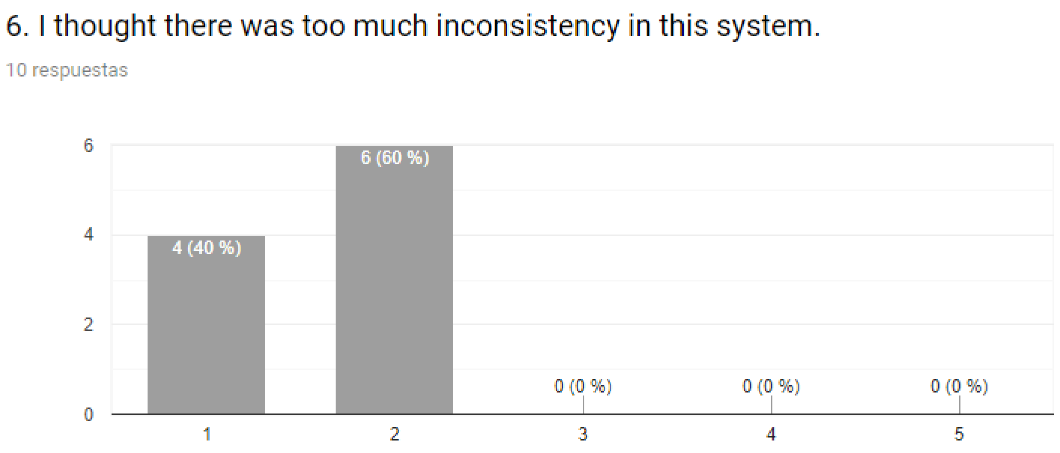


(a) HoloLens evaluation

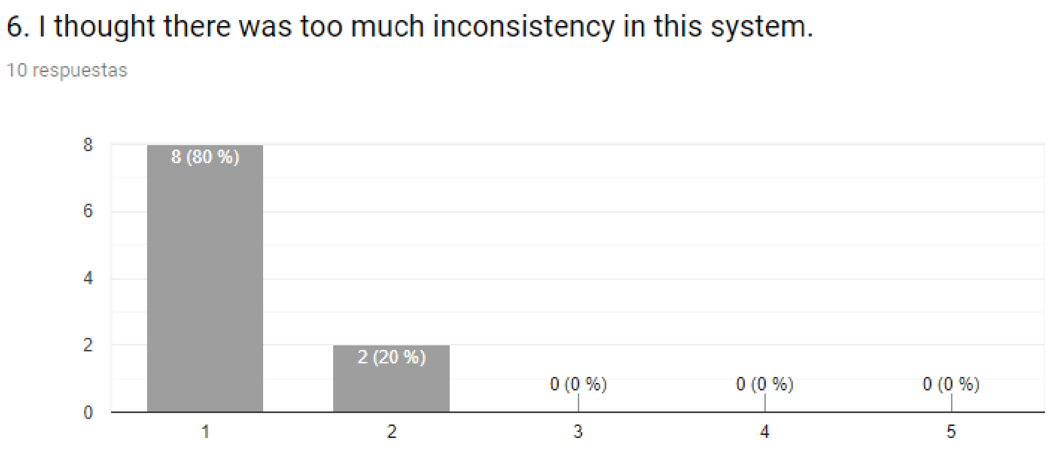


(b) Meta 2 evaluation

Figure 6.10: SUS Question 5

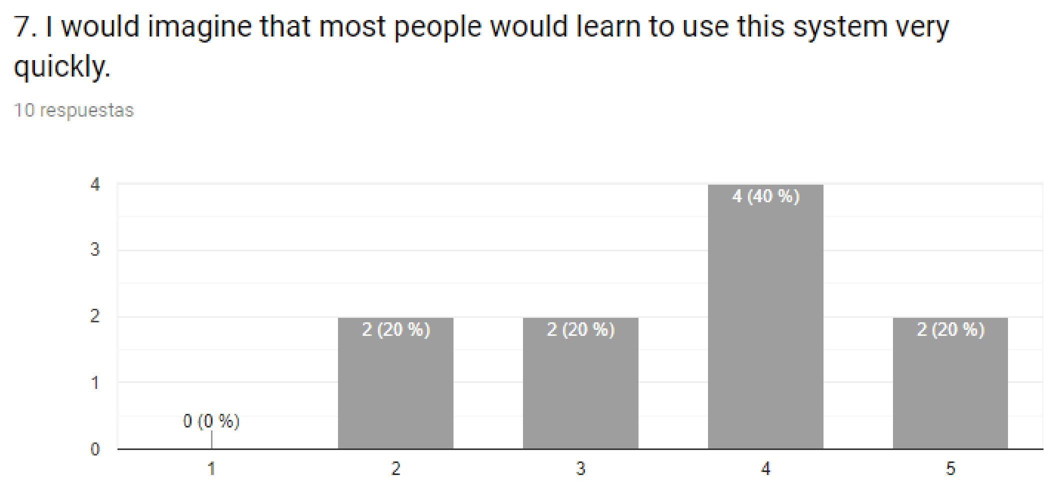


(a) HoloLens evaluation

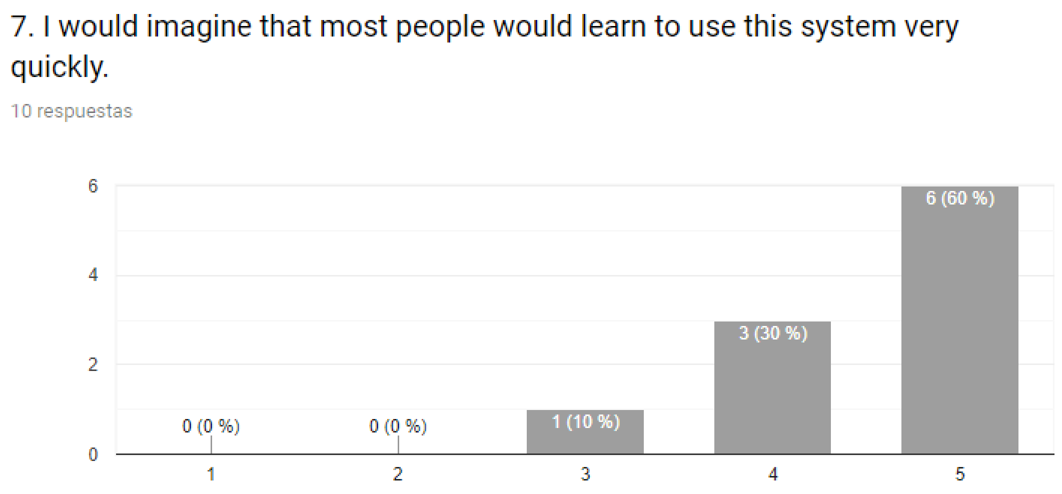


(b) Meta 2 evaluation

Figure 6.11: SUS Question 6

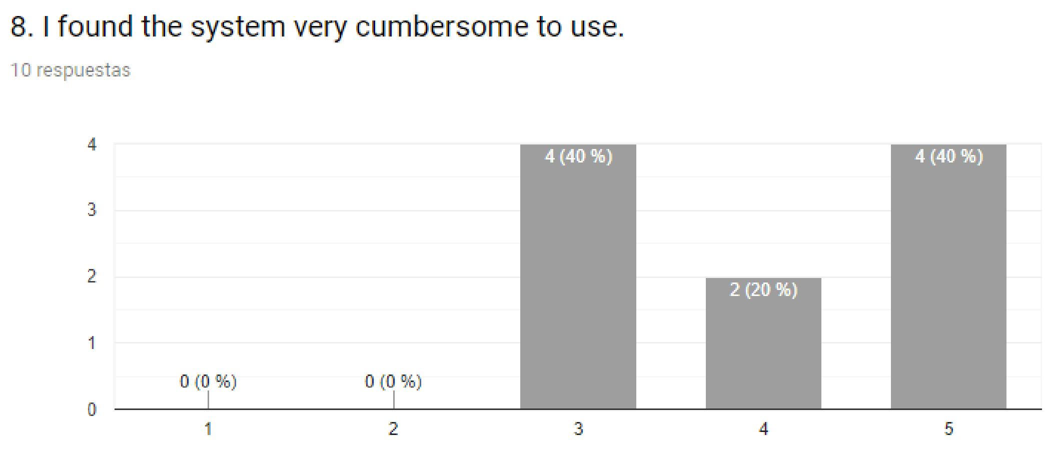


(a) HoloLens evaluation

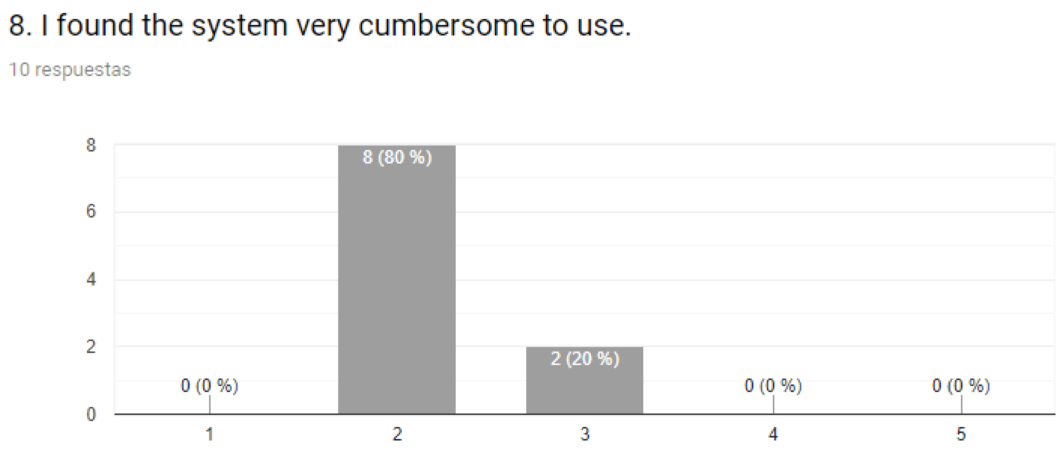


(b) Meta 2 evaluation

Figure 6.12: SUS Question 7

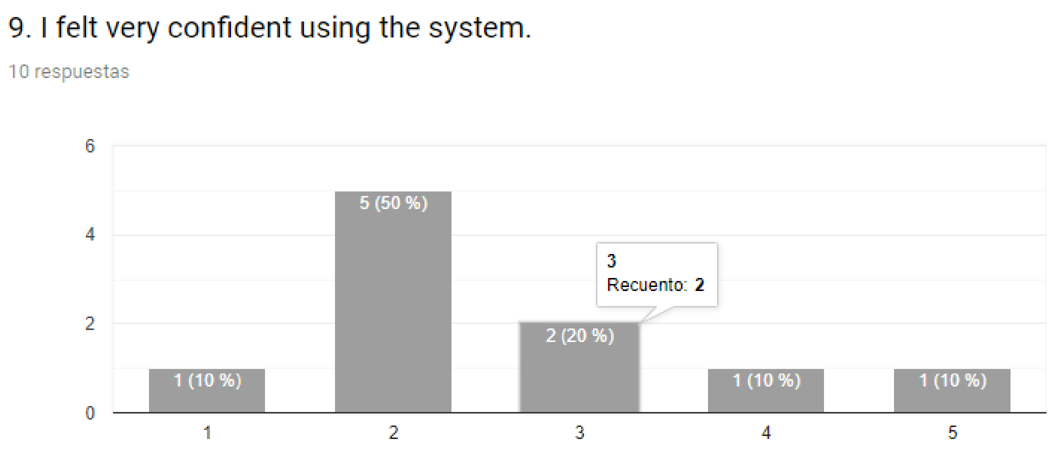


(a) HoloLens evaluation

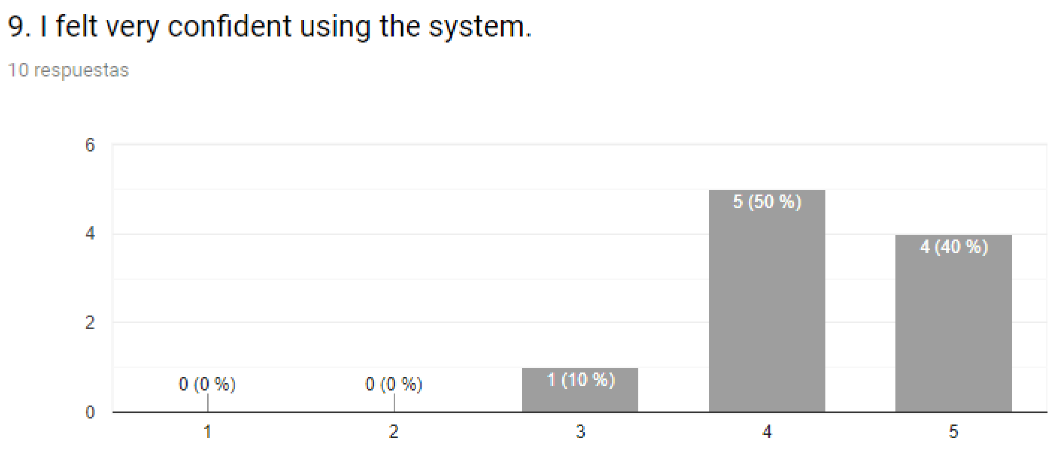


(b) Meta 2 evaluation

Figure 6.13: SUS Question 8

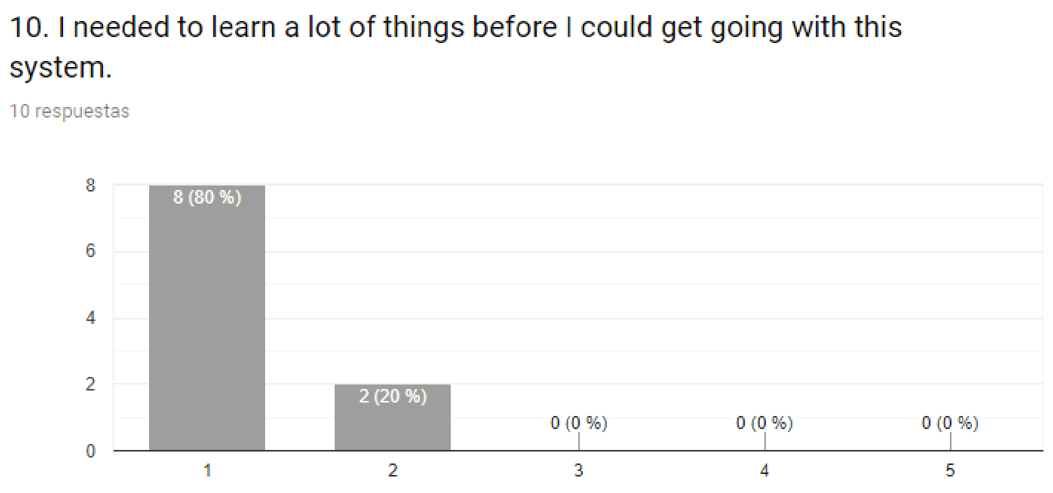


(a) HoloLens evaluation

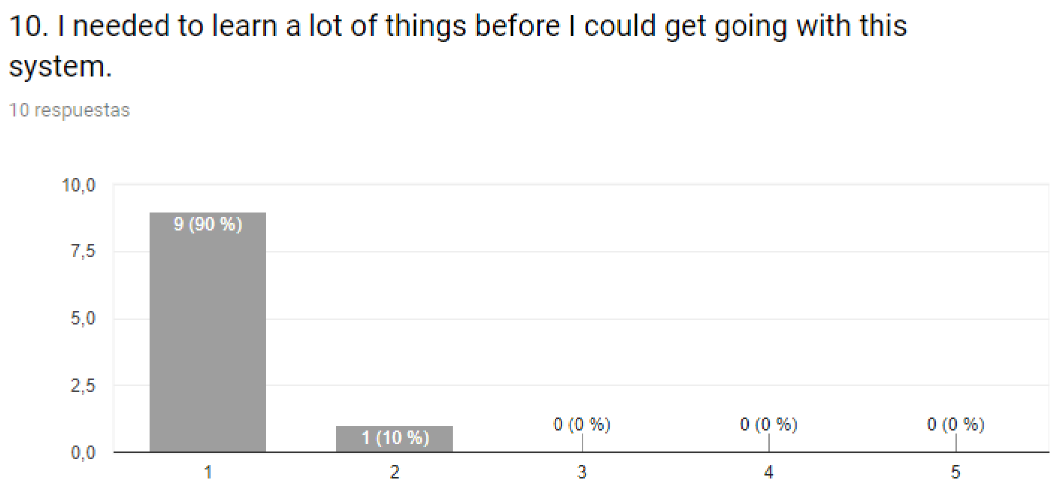


(b) Meta 2 evaluation

Figure 6.14: SUS Question 9



(a) HoloLens evaluation



(b) Meta 2 evaluation

Figure 6.15: SUS Question 10

6.4 User Tests Analysis

Up to ten users were tested to obtain quite significant feedback to compare the FOV in a proper way. Once all the tests were fulfilled, the users' answers were interpreted.

6.4.1 General User Information

The ten users selected for the project's experiment were between the ages of 20 and 28, being the average approximately 23 years old. The majority of them identified themselves as male (70%) while the female presence was only of 3 people.

Most of the users had tried an AR Application once or twice before using the AR chess game. It is remarkable that apart from this tendency, one of the individuals had never tried an application of this type, another one exposed that it sometimes used AR and a last one was using it on a daily basis.

For the utilization of a HMD, there was not a defined tendency, but a tied 40% had never tried one or had tried it only once or twice. A similar situation to the AR Application was recorded here: a user was familiarized with the HMD because of a daily use and another one used it about once a month.

It was established that each new user had to try first the device that their predecessor had tried on a second place. This rule was followed strictly except for one misplaced case.

6.4.2 SUS Questionnaire

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

An H_0 is proposed like $\mu_{AR} = \mu_{VR}$ and an H1 of both means being different. Considering $\mu_{AR} = 25.3$ and $\sigma^2_{AR} = 86.9$ for the HoloLens results and, for the Meta 2 experience, $\mu_{VR} = 34.5$ and $\sigma^2_{VR} = 13.39$.

Following an un-paired Student's t-distribution, the probability obtained is 0,00038 (lower than 5%). This means that the null hypothesis can be rejected. In other words, it can be claimed, for now and only based in these results, that the Meta 2 device is noticeable preferred to play with than the HoloLens one.

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

Table 6.1: Results obtained by testing the HoloLens device

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

Table 6.2: Results obtained by testing the Meta 2 device

6.4.3 Lost Turns

The results for the turns that the players did not manage to play can be summarize as follows:

User	Turns HoloLens	Turns Meta 2
1	3	2
2	3	1
3	2	0
4	2	0
5	2	3
6	2	0
7	2	2
8	2	0
9	2	2
10	2	1

Table 6.3: Turns lost during HoloLens and Meta 2 testing

Based on the data presented above, an H_0 is proposed defining $\mu_{AR} = \mu_{VR}$ and an H1 proposing that both means are different. Grating that $\mu_{AR} = 2.2$ and $\sigma^2_{AR} = 0.18$ for the HoloLens experience and $\mu_{VR} = 1.1$ and $\sigma^2_{VR} = 1.21$ for the Meta 2 results.

Following an un-paired Student's t-distribution, the probability obtained is 0,013 (lower than 5%). This means that the null hypothesis can be declined. This means that it can be confirmed, based on the turns data, that Meta 2 HMD is a better option than HoloLens.

6.5 Conclusion

The users' marks for HoloLens were clearly lower than the ones given to Meta 2. Although before the tests were taken, it was considered that a wider field of view will make the difference for a table-top game, it was not expected an obvious and defined result like the one obtained.

During the experimentation, it was observed that the players who tried HoloLens first, did manage to play by moving their heads quite much. When they wore Meta 2, they felt surprised to see the complete board without even turning their gaze. They seemed then relieved and played more fluently.

In fact, they usually talked about their first impressions and thoughts, and the most repeated one was the fact that with HoloLens they had omitted some of the features, for instance, the messages in the canvas, because they did not even know they existed.

When the opposite situation happened - Meta 2 was experienced before HoloLens - they played calmly at first, as in a usual chess game.

But when putting on the second mentioned device, they started to feel similar to trapped, to do nervous movements and to claim that there had to be something wrong with the display.
After explaining that the HMD was just like that, the users understood it was not a mistake, but they kept feeling uncomfortable.

In addition to the questionnaire, a non-compulsory comment section was available for the users to write down any thoughts about the experiences they had with both devices. They were informal statements, so they have not been formally analyzed, but they are worth mentioning.

Many of the comments referred to HoloLens for not making the whole chess setup visible and for not being adapted to people wearing eyeglasses. The only objection to Meta 2 was that it was a heavy device to wear on the head.

After all these informal data and, more importantly, observing the aforementioned statistical results, it can be concluded for this Master Thesis that the Meta 2 FOV is significantly preferred to the HoloLens FOV.

Chapter 7

Future Work

The project could be defined as complete and finished, but it is obvious that a more detailed experimentation to compare the HoloLens and Meta2 FOV could be designed. The users' experiences could be registered not only via the SUS questionnaire but by means of other respected tests such as the NASA-TLX. [38]

Other future aims that could arise from this project could be related to previous ideas that were discarded during the implementation.

The first idea of developing an ARvs. VR chess game could finally find its way counting on Meta 2 implementation with the Vuforia Engine for the AR and, for example, the already considered Oculus Rift for the VR.

This development would follow the path of trying to improve the chess game user experience by means of holographic superposition to a real board and pieces. Common hand movements or natural voice commands could be a step forward this searched experience.

On the other hand, an interesting assignment could be developing the AR chess game for the HoloLens and Meta 2, but this time with the goal of playing together. This way the real chessboard would be enriched with holographic displays for both players.

One last purpose for a future investigation could be updating the actual work by comparing the FOV of Meta 2 and HoloLens 2. When this project started this last device had not been released yet, but now, it seems like a very appealing comparison: HoloLens 2 FOV is rated in 70º. [39]

Figure 7.1: HoloLens 2 [40]

The FOV of Meta 2 is still larger than HoloLens 2, but it would be intriguing to know if players would still favor Meta 2 over HoloLens 2 in an evident way as they did with HoloLens in this project's tests. A not so big difference between FOVs could lead to a large variety of opinions to which device is better to play with.

To conclude, several works could be related to this one in a near future, from improving this project to starting a new one based on ideas inspired by it. Even the pieces and animations designs for the chess game or the use of another type of game to compare the devices, could be topics of debate and evaluation.

Acronyms

AR	Augmented Reality
HWD	Head Worn Displays
HMD	Head Mounted Displays
FOV	Field of View
UI	User Interface
SUS	System Usability Scale
HD	High Definition
MP	Mega Pixel
IMU	Inertial Measurement Unit
LED	Light-Emitting Diode
TPM	Total Productive Maintenance
LE	Low Energy
RAM	Random Access Memory
GPU	Graphics Processing Unit
SDK	Software Development Kit
UWP	Universal Windows Platform
CPU	Central Processing Unit
OS	Operating System
IR	Infrared
API	Application Programming Interface
VR	Virtual Reality

IDE	Integrated Development Environment
AI	Artificial Intelligence
TLX	Task Load Index

Appendices

Appendix A

Chess in Computing History

Chess game possibly arrived at the Western world from India through Persia in the sixth century. Its origins happened to be royal, as it was considered a test for one's intelligence and knowledge of the battlefield.

It is a two-player board game composed of several pieces that can be moved only by some restrictive rules. These six tokens are: King, Queen, Rook, Knight, Bishop and Pawn. Also, it is described as a zero-sum, perfect information game.

Zero-sum means that one player's good move implies the other player's detriment; and perfect information signifies that the entire game, at each and every single step, is entirely visible for both players. The game also involves the Markov property: prior moves are not necessary to figure out what move should be played next.

For computer scientists, chess is combinatorial: each move generates a combination of possibilities. In fact, in terms of search, chess can be seen as a bounded branch problem, where a large set of branching opportunities are constrained by bad moves at the end of the branch and rich moves at the top of it. This constraint can be pruned, due to the possibility of ignoring one branch after one has decided that a particular path is not better than other.

The first attempt of a chess playing machine was the so called "Turk". Wolfgang von Kempelen, a diplomat and inventor, self-proclaimed it in the year 1770.

It consisted of a board lying on top of a large box where the pieces moved, supposedly by an enigmatic hidden machinery, in response to a human rival. The fake was ultimately discovered, but it had already set up a building chess playing machine competition.

Although a chess playing automaton was expected to land sooner or later, the invention of the electronic computer in the twentieth century made a twist in the chess automatization.

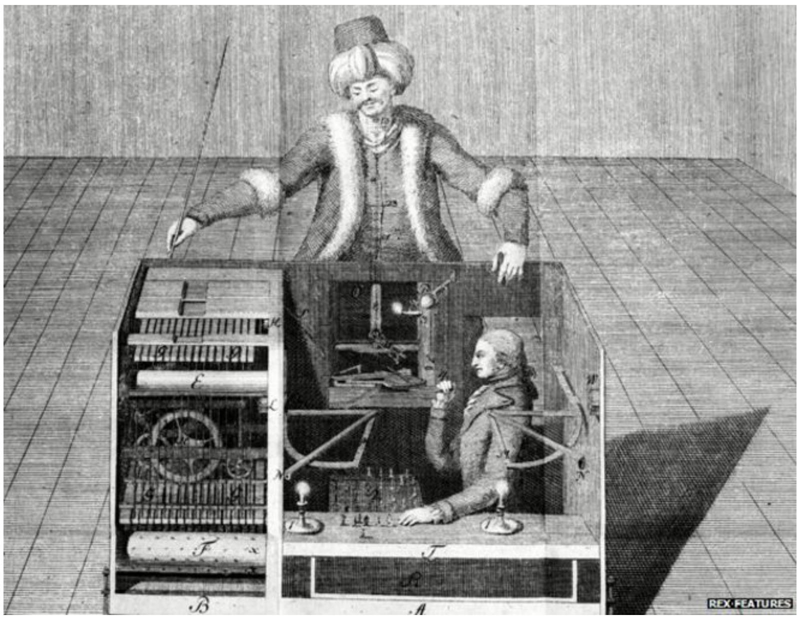


Figure A.1: The Turk [41]

The well-known computer pioneers Alan Turing, in England, and John Von Neumann and Claude Shannon, in the United States, put out the question of making a machine think like a person. This same question was tied with making a machine play chess.

In fact, Turing started an investigation on the topic with a paper-and-pencil written system that was continued by Shannon. The actual aim for it was not exactly the chess game itself, but to be able to build a brain.

In 1951, Turing's colleague, Dietrich Prinz published the first automated chess playing program. This program solved the "mate-in-two" problem: find the best move knowing it is two moves from checkmate, but it could not make it to a complete game due to memory and computational of the Ferranti Mark I computer (Manchester University).

Seven years later, in 1958, an IBM researcher named Alex Bernstein wrote the first complete program running on the IBM 704 mainframe.

Was then Von Neumann, who anticipated the MiniMax algorithm, so successful in playing chess. It tries to maximize one player's score while minimizing the opponent's one. By the end of the 1950s, this algorithm was improved with heuristics after examining several tactics and strategies from human players.

After that, Allen Newell and Herbert Simon combined the MiniMax with the alpha-beta technique: pruning (speeding up the search by ignoring known poor moves) into the NSS program.

By the decade of the 1960s, there existed a real automated chess who could play against a human opponent. Indeed, by 1962, a MIT students' program could beat amateur chess players. By 1967, an improved MIT system achieved a 1400 score, comparable to a very good high school player.

During the 1970s, not only the already written chess programs were improved by effective heuristics and newly designed searches, but hardware sped up and was specially created for chess gaming purposes.
In 1978, after the ten years expressed in the famous International Master David Levy's bet, a match was scheduled to determine this bet outcome. Levy won to the CHESS 4.7, the top program at that time.

The 1980s launched the era of personal computers. Being so, chess playing games were sold in the public market and played everywhere.

Then came the World Microcomputer Chess Championships (WMCCC): the first official computer-only chess competitions. It is remarkable to mention the $100,000 award offered to any computer program that could defeat a current World Chess Champion, the so called Fredkin Prize.

Carnegie Mellon researchers landed Deep Thought, the first program capable of playing Grand Master level chess. By the end of the 80s, this system had already won games against several human Grand Masters.

IBM purchased Deep Thought with the only goal of winning a game to Garry Kasparov, the current world champion. The chess programming group was led by Carnegie original team. But, in 1989, Kasparov won smoothly.

It was not until 1997, that the rematch materialized, this time with the renamed Deep Blue, which could evaluate two hundred million chess positions per second. The game was recorded in a television studio in order to broadcast the event. The match lasted up to six games and ended up with Deep Blue victory.

(a) Kasparov before starting the game

(b) Kasparov 7 steps into losing to Deep Blue

Figure A.2: Deep Blue vs. Garry Kasparov [42]

Nowadays, Deep Blue's chess abilities are available on our laptops, smartphones and even, as this work discusses, in our HMD. And chess happens to be the most popular board game. [43]

List of Figures

List of Tables

Bibliography

[1] Christian Crisostomo. Introduction to ar headsets technology: The field of view. https://arpost.co/2018/09/27/introduction-to-ar-headsets-technology-the-field-of-view/, 2018.

[2] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[3] Ronald Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, and Blair MacIntyre. Recent advances in augmented reality. *IEEE computer graphics and applications*, 21(6):34–47, 2001.

[4] Huffington Post. The lengthy history of augmented reality. http://images.huffingtonpost.com/2016-05-13-1463155843-8474094-AR_history_timeline.jpg, 2016.

[5] Volkswagen. Virtual technologies. https://www.volkswagenag.com/en/group/research/virtual-technologies.html, 2019.

[6] John Minchillo. Google glass to go on sale to the u.s. public for one day next week. https://business.financialpost.com/technology/google-glass-public-sale, 2014.

[7] Reality. The ultimate guide to understanding augmented reality (ar) technology. https://www.realitytechnologies.com/augmented-reality/, 2018.

[8] Andrea Sanna and Federico Manuri. A survey on applications of augmented reality. *Advances in Computer Science: an International Journal*, 5(1):18–27, 2016.

[9] Vlasios Kasapakis, Damianos Gavalas, and Panagiotis Galatis. Augmented reality in cultural heritage: Field of view awareness in an archaeological site mobile guide. *Journal of Ambient Intelligence and Smart Environments*, 8(5):501–514, 2016.

[10] Google Play. Google translate. https://play.google.com/store/apps/details?id=com.google.android.apps.translate&hl=en, 2010.

[11] Martin Coxall. Eyetoy: Play 3. https://www.eurogamer.net/articles/r_eyetoyplay3_ps2, 2006.

[12] Josep M. Berengueras. Pokémon go añadirá 80 nuevas criaturas esta semana. https://www.elperiodico.com/es/tecnologia/20170215/pokemon-go-nuevos-actualizacion-5839511, 2017.

[13] Michael Alba. First look: The meta 2 ar headset. https://www.engineering.com/ARVR/ArticleID/16474/First-Look-The-Meta-2-AR-Headset.aspx,

2018.

[14] Microsoft. Hololens (1st gen) hardware details. `https://docs.microsoft.com/en-us/windows/mixed-reality/hololens-hardware-details`, 2018.

[15] Install the tools. `https://docs.microsoft.com/en-us/windows/mixed-reality/install-the-tools#installation-checklist`, 2019.

[16] Microsoft. Head-gaze and commit. `https://docs.microsoft.com/en-us/windows/mixed-reality/gaze-and-commit`, 2019.

[17] Microsoft. Windows 10 sdk. `https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk`, 2019.

[18] Microsoft. Mixed reality documentation. `https://docs.microsoft.com/en-us/windows/mixed-reality/`, 2019.

[19] Microsoft. Gestures. `https://docs.microsoft.com/en-us/windows/mixed-reality/gestures`, 2019.

[20] Kari Pulli. Immersive optical-see-through ar with meta 2. `http://on-demand.gputechconf.com/gtc/2017/presentation/s7757-pulli-immersive-optical-see-through-vr.pdf`, 2017.

[21] XinReality: Virtual Reality and Augmented Reality Wiki. Meta 2. `https://xinreality.com/wiki/Meta_2`, 2017.

[22] Metavision. Sdk features. `https://docs.metavision.com/external/doc/latest/sdk_features.html`, 2018.

[23] Metavision. Meta hands. `https://docs.metavision.com/external/doc/latest/meta_hands`, 2018.

[24] Metavision. Meta mouse. `https://docs.metavision.com/external/doc/latest/meta_mouse`, 2018.

[25] Metavision. Meta gaze. `https://docs.metavision.com/external/doc/latest/meta_gaze.html`, 2018.

[26] Virtual Reality Society. What is virtual reality? `https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html`, 2017.

[27] Oculus. What's in the box. `https://www.oculus.com/rift/#oui-csl-rift-games=robo-recall`, 2016.

[28] Wikipedia. Unity (game engine). `https://en.wikipedia.org/wiki/Unity_(game_engine)`, 2019.

[29] Unity. Scenes. `https://docs.unity3d.com/Manual/CreatingScenes.html`, 2019.

[30] Unity. Gameobjects. `https://docs.unity3d.com/Manual/GameObjects.html`, 2019.

[31] Unity. Scripting. `https://docs.unity3d.com/Manual/ScriptingSection.html`, 2019.

[32] Unity. Asset workflow. `https://docs.unity3d.com/Manual/AssetWorkflow.html`, 2019.

[33] Unity. Prefabs. `https://docs.unity3d.com/es/current/Manual/Prefabs.html`, 2018.

[34] Wikipedia. Microsoft visual studio. `https://en.wikipedia.org/wiki/Microsoft_Visual_Studio`, 2019.

[35] Robert Plutchik. The nature of emotions: Human emotions have deep evolutionary roots, a fact that may explain their complexity and provide tools for clinical practice. *American scientist*, 89(4):344–350, 2001.

[36] Evi Joosten, Giel Van Lankveld, and Pieter Spronck. Colors and emotions in video games. In *11th International Conference on Intelligent Games and Simulation GAME-ON*, pages 61–65, 2010.

[37] Unity. Vuforia. `https://docs.unity3d.com/Manual/vuforia-sdk-overview.html`, 2019.

[38] Sandra G Hart and Lowell E Staveland. Development of nasa-tlx (task load index): Results of empirical and theoretical research. In *Advances in psychology*, volume 52, pages 139–183. Elsevier, 1988.

[39] Ben Lang. Microsoft reveals hololens 2 with more than 2x field of view & 47 pixels per-degree. `https://www.roadtovr.com/microsoft-hololens-2-announcement-2x-fov-47-pixels-per-degree/`, 2019.

[40] Microsoft. Hololens2 - a new vision for computing. `https://www.microsoft.com/en-us/hololens/hardware`, 2019.

[41] BBC News. A point of view: Chess and 18th century artificial intelligence. `https://www.bbc.com/news/magazine-21876120`, 2013.

[42] Leontxo García. Los trucos de ibm contra kaspárov. `https://elpais.com/deportes/2017/06/08/la_bitacora_de_leontxo/1496908568_067804.html`, 2017.

[43] Erik J. Larson. A brief history of computer chess. `https://thebestschools.org/magazine/brief-history-of-computer-chess/`, 2015.