# Politecnico di Torino

Department of Control and Computer Engineering (DAUIN)

Master Degree in
**Computer Engineering**



# Design and development of a Mixed Reality application in the automotive field


**Supervisors:**
Prof. Maurizio Morisio

**Author:**
Giovanna Galeano



December 2017

# Contents

# List of Figures

# Abstract

Advancements in computing have continued to enhance user effectiveness and efficiency in all aspects of home and work life. Mixed Reality technology is rapidly becoming one of those major technological progressions.
Mixed reality is the result of blending the physical world with the digital world. It is the next evolution in human, computer, and environment interaction and unlocks possibilities that before now were restricted to our imaginations.
The term mixed reality was originally introduced in a 1994 paper by Paul Milgram and Fumio Kishino, "A Taxonomy of Mixed Reality Visual Displays." Their paper introduced the concept of the virtuality continuum and focused on how the categorization of taxonomy applied to displays. Since then, the application of mixed reality goes beyond displays but also includes environmental input, spatial sound, and location.

The enterprise market for these technologies is growing rapidly. In particular, the market value for Augmented Reality applications is expected to significantly surpass that of Virtual Reality. Experts are predicting that “VR and AR will be a $150 billion market by 2020, with $120 billion of that dedicated to AR”. The reason for this is because Virtual Reality, although flashier, takes the user out of the real world. Augmented Reality instead is more applicable to interaction with the physical world around us. Developers and engineers have been progressively releasing more business productivity and enterprise applications with the goal of enhancing business value with this technologies. This thesis work is indeed an example of this trend.

It is a proposal of a headset for sport drivers that exploits the potential of this emerging technology. The main goal was the definition and development of an advanced head-mounted device that could allow the driver to see on the display of his helmet useful information.

This thesis is focused mainly on the first phases of the software/hardware development. After an initial analysis of the main requirements of this particular context and the state of art, a first possible architecture for the head-mounted device has been proposed. The next step was the choice of the best platform to be used for the development of this kind of application and the best engine that could support the design and coding of it. The comparison made showed that the best option was the Microsoft Hololens System, that is a holographic devices with a see-through display that allows the user to see the physical environment while wearing the headset. This technology should be adapted and incorporated into a driver headset. As far as concerns the computer engine, the comparison

made among the main engines on the market showed that Unity3D was the best option. The last part of the work done has been the development of a prototype that could give an idea to the user of the final result.

# Chapter 1

# Mixed Reality

The word "mixed reality" comes from the research paper by Paul Milgram and Fumio Kishino entitled "A Taxonomy of Mixed Reality Visual Displays" published in 1994. In that paper, the term "mixed reality" appeared in reference to a part of "the virtuality continuum" (later also called the Reality-Virtuality (RV) continuum).

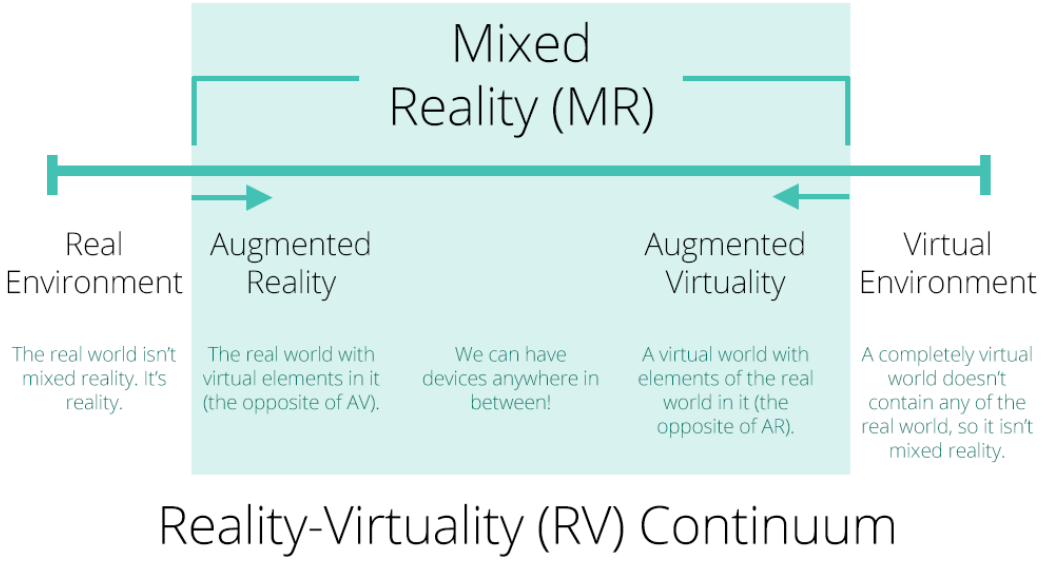


Figure 1.1: Reality-Virtuality Continuum

This spectrum basically shows a range of environments, from the real world with no virtual elements to a totally virtual environment with none of our reality visible. The space in between when real world elements start to mix with virtual ones to some degree or another is "mixed reality" in their original definition.
*"The most straightforward way to view a Mixed Reality environment, therefore, is one in which real world and virtual world objects are presented together within a single display, that is, anywhere between the extrema of the virtuality continuum."*, "A Taxonomy of Mixed Reality Visual Displays".

Since the publication of the above-mentioned document, the application of mixed reality goes beyond displays but also includes environmental input, spatial sound and location.

Now, the combination of computer processing, human input, and environmen-

tal input sets the opportunity to create true mixed reality experiences. Movement through the physical world can translate to movement in the digital world. Boundaries in the physical world can influence application experiences, such as game play, in the digital world. Without environmental input, experiences cannot blend between the physical and digital realities.

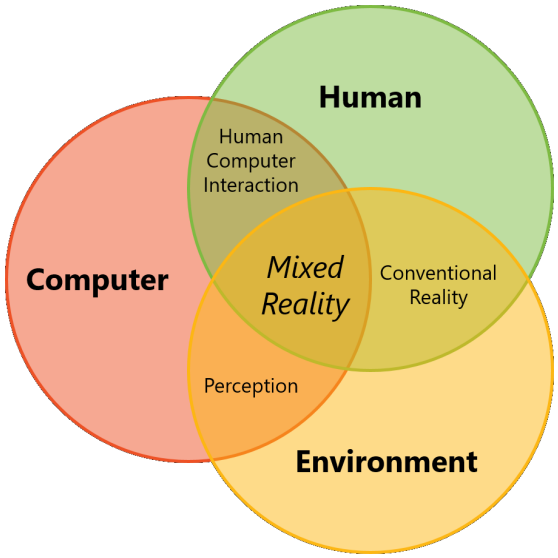


Figure 1.2: Mixed Reality Venn Diagram

## 1.1 Real Environment

The real environment (also called “natural environment”) refers to the natural world we consume everyday. This natural environment encompasses all living and non-living things occurring naturally on Earth. Consequently, most virtual environments are modeled after real environments.

## 1.2 Augmented Reality

A live direct or indirect view of a physical, real-world environment whose elements are "augmented" by computer-generated or extracted real-world sensory input such as sound, video, graphics or GPS data is called Augmented Reality (AR).

With the help of advanced AR technology the information about the surrounding real world of the user becomes digitally manipulable and interactive.

### 1.2.1 Augmented Reality (AR) Categories

- **Marker-based augmented reality** (also called Image Recognition)
  A camera and some type of visual marker, such as a QR/2D code are used

to produce a result only when the marker is recognized by a reader.
Marker based applications use a camera on the device to distinguish a marker from any other real world object. Distinct, but simple patterns (such as a QR code) are used as the markers, because they can be easily recognized and do not require a lot of processing power to read.

- **Markerless augmented reality** (also called location-based, position-based, or GPS)
  This type of AR application is one of the most widely used. It uses a GPS, digital compass, velocity meter, or accelerometer to provide data based on your location.
  It is used for location-centric mobile applications (ie finding nearby businesses or mapping directions).

- **Superimposition based augmented reality**
  In this case the original view of an object is either partially or fully replaced with a newly augmented view of that same object.
  In superimposition based augmented reality, object recognition plays a vital role because the application cannot replace the original view with an augmented one if it cannot determine what the object is.

Figure 1.3: Ikea Augmented Reality App

### 1.2.2 Key Components to Augmented Reality Devices

Key components to AR devices are:

- **Sensors and Cameras**
  Sensors are usually on the outside of the augmented reality device, and

gather a user's real world interactions and communicate them to be processed and interpreted. Cameras are also located on the outside of the device, and visually scan to collect data about the surrounding area. The devices take this information, which often determines where surrounding physical objects are located, and then formulates a digital model to determine appropriate output.

- **Projection**
  A projector can essentially turn any surface into an interactive environment. The information taken in by the cameras used to examine the surrounding world, is processed and then projected onto a surface in front of the user.

- **Processing**
  Augmented reality devices are basically mini-supercomputers packed into tiny wearable devices. These devices require significant computer processing power and utilize many of the same components that our smartphones do. These components include a CPU, a GPU, flash memory, RAM, Bluetooth/Wifi microchip, global positioning system (GPS) microchip, and more. Advanced augmented reality devices utilize an IMU system to provide for truly immersive experience.

- **Reflection**
  Mirrors are used in augmented reality devices to assist with the way the eye views the virtual image. Some augmented reality devices may have "an array of many small curved mirrors" (as with the Magic Leap augmented reality device) and others may have a simple double-sided mirror with one surface reflecting incoming light to a side-mounted camera and the other surface reflecting light from a side-mounted display to the user's eye. The reflection paths have the same objective, which is to assist with image alignment to the user's eye.

### 1.2.3 AR headsets categories

There are two categories for AR headsets: some AR headsets are based on conventional screens "video see-through" and some deviceses are based on semi-transparent screens "optical see-through". A "video see-through" device displays video from camera which films the real scene and which is outside the headsets. "Optical see-through" systems combine computer-generated images with the view of the real world through a semi transparent mirror.

## 1.3 Augmented Virtuality

Augmented virtuality describes the environment in which real objects are inserted into computer-generated virtual environments. It is best described as the inverse of augmented reality, where real world objects are layered over virtual environments. In the following figure is possible to see the example showed by the Intel company, they exploited the HTC Vive technology in combination with data received from sensors and cameras to "import" the real hands into the virtual environment.

Figure 1.4: Intel proposal for Augmented Virtuality

## 1.4 Virtual Reality

Virtual Reality (VR) is a computer technology that uses virtual reality headsets or multi-projected environments to generate realistic images, sounds and other sensations that simulate a user's physical presence in a virtual or imaginary environment.

A person using virtual reality equipment is able to "look around" the artificial world, and with high quality VR move around in it and interact with virtual features or items. The effect is commonly created by VR headsets consisting of head-mounted goggles with a screen in front of the eyes, but can also be created through specially designed spaces with multiple large screens.

### 1.4.1 Virtual Reality (VR) Categories

- **Non Immersive Virtual Reality**
  Non-immersive simulations are the least immersive implementation of vir-

tual reality technology. In a non-immersive simulation, only a subset of the user's senses are stimulated, allowing for peripheral awareness of the reality outside the virtual reality simulation.

- **Semi Immersive Virtual Reality**
  Semi-immersive simulations provide a more immersive experience, in which the user is partly immersed in a virtual environment.
  Semi-immersive simulations are powered by high performance graphical computing systems coupled with large screen projector systems or multiple television projection systems to stimulate the user's visuals in the proper way.

- **Fully Immersive Virtual Reality**
  Fully-immersive simulations provide the most immersive implementation of virtual reality technology.
  In a fully-immersive simulation all of a user's senses are stimulated.
  Fully immersive simulations are able to provide a realistic user experiences, this thanks to a wide field of view, high resolutions and update rates and high levels of contrast into the display.

Figure 1.5: Example of immersive Virtual Reality

### 1.4.2 Key Components in a Virtual Reality System

The virtual environment must look real but also be felt real by the human brain in order to be fully accepted.
Looking real can be achieved by wearing a head-mounted display (HMD) that displays a recreated life size, 3D virtual environment without boundaries that are usually seen on TV or a computer screen.

Feeling real can be achieved through handheld input devices such as motion trackers that base interactivity on the user's movements.

Some of the key components behind this system are:

- **PC/Console/Smartphone**
  Virtual reality content, which is the what users view inside of a virtual reality headset, is equally important as the headset itself. In order to power these interactive three-dimensional environments, significant computing power is required. PC, consoles, and smartphones act as the engine to power the content being produced.

- **Head-Mounted Display**
  A head-mounted display is a type of device that contains a display mounted in front of a user's eyes. This display usually covers the user's full field of view and displays virtual reality content. Some virtual reality head mounted displays utilize smartphone displays, including the Google Cardboard or Samsung Gear VR.

- **Input Devices**
  Input devices are one of the two categories of components that provide users with a sense of immersion. Some of the more common forms of virtual reality input devices include joysticks, controller wands, data gloves, trackpads.

### 1.4.3 Key Components Inside of a Virtual Reality Headset

Inside of each virtual reality head-mounted display (HMD) is a series of sensors, individual eye displays, lenses, and display screen(s), among other various components. Some of the key components inside of a virtual reality headset are:

- **Sensors**
  The three most common sensors in a virtual reality headset are magnetometers, accelerometers and gyroscopes. These sensors work together by measuring the user's motions and direction in space. Their ultimate goal is to covers all the degrees of motion for an object in space.

- **Lenses**
  Lenses lie between the eyes and pixels on the display screen(s). They focus and reshape the picture for each eye by angling two 2D images to mimic how each of our eyes take in views of the world (also called stereoscopic). This creates an impression of depth and solidity, which we perceive to be a three-dimensional image. Lenses on each virtual reality device are not

one-size-fits all and have to be adjusted for initial use as all devices have different lens properties.

- **Display Screens**
  Display screens show the images that user view through the lenses. They are typically LCD and receive video feed from the computer or smartphone. Depending on the headset, the video feed is either sent to one display or two displays (one per eye). This happens via wireless connection, smartphone connection, or HDMI. The most common types of virtual reality display technology is a Liquid Crystal Display (LCD) screen, similar to the kinds used in smartphones and computer monitors. An alternative display technology is an Organic Light-Emitting Diode (OLED) screen.

- **Processing**
  Virtual reality systems demand a substantial amount of power, even in comparison to notoriously power hungry gaming systems. The processing power required by virtual reality systems can be broken down into several categories:
  *Input Processor* – Controls the devices used to input information to the computer. They retrieve and distribute data to the rest of the system with minimal lag time. Examples include keyboards, mouses, 3D position trackers, and voice recognition systems.
  *Simulation Processor* – Takes the user inputs along with any other tasks that are programmed from the natural world and determines the actions that will take place in the virtual world. This is a core component of the VR system.
  *Rendering Processor* – Creates the sensations that are output to the user. These include visual, auditory, haptic and other sensory systems. Separate rendering processes are used for each sensory system.

### 1.4.4 Performance parameters

- **Field of View**
  Humans have an FOV of around 180 degrees, but most HMDs offer far less than this. Typically, a greater field of view results in a greater sense of immersion and better situational awareness. Simply put, field of view refers to how wide the picture is. Field of view is measured based on the degree of display (e.g. 360 degrees).Consumer-level HMDs typically offer a FOV of about 30-40 degrees whereas professional HMDs offer a field of view of 60 to 150 degrees.

- **Interpupillary distance**
  Interpupillary distance (IPD) is the distance between the two eyes, measured at the pupils, and is important in designing head-mounted displays.

- **Latency**
  Latency refers to the amount of time it takes for an image displayed in a user's headset to catch up to their changing head position. Latency can also the thought of as a delay, and is measured in milliseconds (ms). In order for an experience to feel real, latency usually needs to be in the range of 20 milliseconds (ms) or less. Low latency, or very little delay, is needed to make the human brain accept the virtual environment as real. The lower the latency, the better. The higher the latency, a noticeable and unnatural lag may set in.

- **On-board processing and operating system**
  Some HMD vendors offer on-board operating systems such as Android, allowing applications to run locally on the HMD, and eliminating the need to be tethered to an external device to generate video. These are sometimes referred to as smart goggles. To make the HMD construction lighter producers may move the processing system to connected smart necklace form-factor that would also offer the additional benefit of larger battery pack. Such solution would allow to design lite HMD with sufficient energy supply for dual video inputs.

- **Frame Rate**
  Frame rate refers to the frequency (rate) at which the display screen shows consecutive images, which are also called frames. Television shows run at 30 frames per second (fps) and some game consoles run at 60 frames per second (fps). In virtual reality, a minimum frame rate of approximately 60 frames per second is needed to avoid content stuttering or cause of simulation sickness.

- **Tracking**
  Tracking handles the vital task of understanding a user's movements and then acting upon them accordingly. Eye trackers, for example, measure the point of gaze, allowing a computer to sense where the user is looking. This information is useful in a variety of contexts such as user interface navigation : by sensing the user's gaze, a computer can change the information displayed on a screen, bring added details to attention, etc.

- **Resolution**
  HMDs usually mention either the total number of pixels or the number of

pixels per degree. Listing the total number of pixels (e.g., $1600x1200$ pixels per eye) is borrowed from how the specifications of computer monitors are presented.

# Chapter 2

# Context

Few industries can match the automobile industry in terms of scale, reach and international appeal. As one of the driving forces of the 20th century, the automotive industry is also at the forefront of innovation in this century.

The face of the automobile industry is changing with new players and new technologies from electric cars to automated vehicles are disrupting traditional heavyweights of the industry. While the industry is not yet in a crisis it is however at a critical point in its history with many manufacturers' profits squeezed and margins thinned.

The 2017 PriceWaterhouseCoopers Automotive Trends report stated that the automotive companies that stand out and emerge from this crisis point successfully will be "the companies that harness their limited capital resources in creative ways, to navigate a still-unfolding and unfamiliar landscape."

Mixed reality can pay a part in this creative innovation, helping to redress the balance and give manufacturers an opportunity to innovate across all aspects of their processes.
Talking about the customer driving experience, it can be improved by the introduction of additional information directly on the windshield or on a display next to the driver or even projected on glasses or headsets worn by the driver. We have distinguished several categories for the information that can improve the driving experience.

- **Safety**
  Thanks to sensors and cameras around the car information such as a potential danger on the road ahead or obstacles in the immediate surroundings can be immediately seen by the driver. Information about the speed limit or in particular weather conditions detections (fog, heavy rain or snow) the road signs can be perceived as projected directly on the road. In addition, information about the health of the driver, for example the heart-beat to detect a possible status of sickness or anxiety, the eye-blinking to detect sleepiness.

- **Car information**
  Information about the car such as the oil temperature, water temperature, speed, engine rpm and many others.

- **General information**
  Information about the weather conditions, temperature and forecast; the driving route with potentially additional information about nearby historical, architectural or natural points of interest, facts on sights, exhibitions, shows and events nearby; latest news from the web and social network updates, phone calls, text messages.

- **Passengers entertainment**
  Passenger can zoom into objects along the road and obtain information about them in real time; special glasses, cameras and an appropriate car structure could be used to "see through" the vehicle creating a fly-like feeling.

- **Training**
  Track lanes lines projection, cones, braking trace and acceleration trace could help novice driver at learning a new track (this information can be automatically calculated or set by an expert driver); lap times and ghost cars to compete with can also be used to practice.

## 2.1 Technological scouting

During the requirements analysis a research on the actual state of mixed reality devices, in the more general meaning, has been performed with special focus on the automotive field. It can be noticed that the main technology used for this devices is the augmented reality and the first attempt to introduce a simple form of augmented reality in this field was made in 2002 for a Formula 1 driver. In addition, it can be seen that the main AR systems designed and developed are three:

- **Head or Helmet Mounted Devices**
  A head-mounted display (or helmet-mounted display, for some applications), both abbreviated HMD,as said before, is a display device, worn on the head or as part of a helmet, that has a small display optic in front of one (monocular HMD) or each eye (binocular HMD). A HMD has many uses including in gaming, aviation, engineering, and medicine.

- **Windshield-mounted Displays**
  A windshield-mounted display system, projects the information directly on the car windshield through the usage of projectors and mirrors .

- **Head-up Displays**
  A head-up display or heads-up display, also known as a HUD, is any trans-

parent display that presents data without requiring users to look away from their usual viewpoints. The origin of the name stems from a pilot being able to view information with the head positioned "up" and looking forward, instead of angled down looking at lower instruments. A HUD also has the advantage that the pilot's eyes do not need to refocus to view the outside after looking at the optically nearer instruments.

- **Smart Glasses**
  Smart glasses are wearable computer glasses that add information alongside or to what the wearer sees. Alternatively smartglasses are sometimes defined as wearable computer glasses that are able to change their optical properties at runtime. The most famous example is Google Glasses.

In the following pages several examples of each type of system will be provided, some of these are only conceptual proposals that still need to be tested and that are not on the market yet, while some are already on the market place or at least prototyped.

### 2.1.1 Automotive field

- **BMW F1 HMD**


Figure 2.1: BMW F1 HMD

This headmounted display was designed for Formula 1 racing driver Ralph Schumacher. It uses a Kopin AMLCD and a free-form surface (FFS) prism to project a virtual image into user's eye. It is mounted in the lower right corner of the driver's field of view. It is the first helmet to be used by a professional driver on a track. It wasn't used regularly by the driver but

the Formula 1 team never talked openly about this topic.

- **BMW Mini Augmented reality glasses**

Figure 2.2: BMW glasses

Beside the typical HUD functionalities provided by the glasses, they allow an "x-ray view" through the car's pillars, doors and other visual obstructions. When parking, these glasses are also able to display live video from a camera mounted to the underside of the passenger's wing mirror. Navigation arrows can be displayed on the road ahead in order to keep the driver focused on driving. Available parking spaces near the destination and points of interest along the route can also be pointed out by these glasses. If the drivers wear these glasses outside of the car, they can make use of its first and last mile navigation feature, which guides them to their vehicle or to their final destination from their parking spot, respectively. This example is a conceptual only at the moment, but reflects the market trend toward mixed reality.

- **Toyota Augmented Reality Windshield**

  Toyota's patent works like a conventional HUD to display vehicle instruments on the windshield. Where this solution differs is that an ECU (Electronic control unit) analyzes the steering angle and speed; a front-mounted camera identifies the lane markings, and an interior camera finds the driver's viewpoint. By combining this data, the system moves the information around the windshield to be in the best location. For added safety, the image's size corresponds to the vehicle's width, so drivers also know their position in a lane. As speeds increase, the display moves up the screen and gets smaller because a person is looking further into the distance.

- **Jaguar Land Rover Virtual Windscreen** Jaguar Virtual Windscreen concept takes things to the next level to turn real-life driving into a video

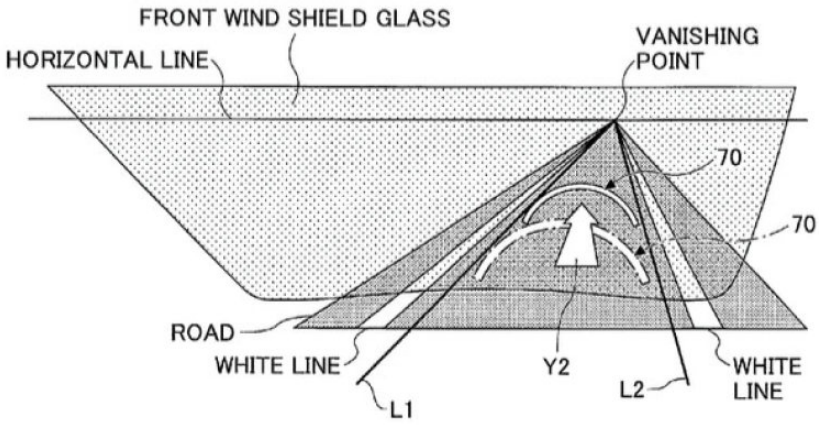


Figure 2.3: Toyota AR Windshield

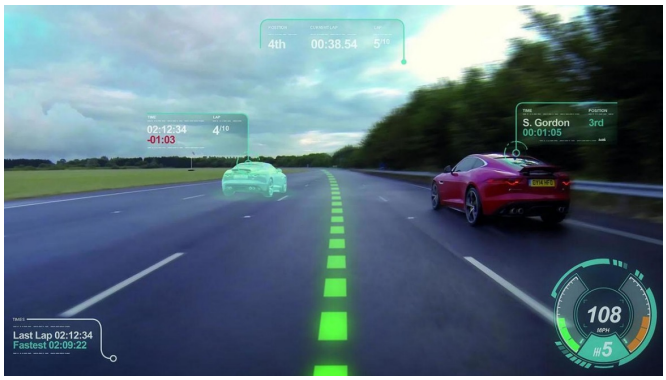


Figure 2.4: Jaguar Land Rover Virtual Windscreen

game-like experience with data like lap times, grid positions, virtual racing line and brake guidance, "ghost" cars from previous laps and virtual cones for simulated auto crossing. The concept system's features include virtual racing lines that change color to indicate optimum braking positions on treacherous curves, virtual cones placed along the road in real-time that could help novice drivers training. Jaguar Land Rover has developed an innovative 3D instrument cluster, which uses the latest head and eye-tracking technology to create a natural-looking, specs-free 3D image on the instrument panel. Cameras positioned in the instrument binnacle or steering column area track the position of the user's head and eyes. Software then adjusts the image projection in order to create a 3D effect by feeding each eye two slightly differing angles of a particular image. This creates the perception of depth which allows the driver to judge distance. The Jaguar Land Rover proposal is still conceptual only.

- **iSCOUT**


Figure 2.5: iSCOUT HUD

iSCOUT is a patent pending Head-Up Display (HUD) that projects all the information required for driving such as car speed, fuel level, GPS navigation as a floating virtual image in front of the vehicle. iSCOUT syncs the driver smartphone and displays incoming calls, messages, reminders, social media content and other notifications along with several other features such as blind-spot view and an integrated dashcam. iScout started as a project on Kickstarter, the famous website for crowdfounding.

- **Navdy**

Navdy's HUD comes with an inbuilt projector that displays all your information which could be anything ranging from navigation details to calls and social media messages on the driver's field of view thus giving him all the necessary information without losing his focus from the road. The display

Figure 2.6: Navdy HUD

is mounted on the dash while the Navdy Dial, the main tool for interacting with the phone and the Display, is fastened to one side of the steering wheel.

### 2.1.2 Other examples

- **DAQRI Smart Helmet**

Figure 2.7: DAQRI Smart Helmet

The helmet – which has a blue scratch-resistant visor – was specifically created for workers in industrial settings, such as oil rigs, water treatment plants and construction sites. Designed specifically for industrial augmented reality applications. The thermography technology delivers novel 3D thermal maps of equipment and infrastructure for industrial inspections.

- **F-35's Helmet Mounted Display System**

  This helmet incorporates LCD displays to take input from the aircraft's various sensors—radar, infra-red, and the electro-optical targeting system—and overlay them on the pilot's field of view. The F-35 has six such cameras, all forming the so-called *Distributed Aperture System*. The helmet visor can project other things, including digitally zoomed-in night vision from the

Figure 2.8: F-35 HMD

fighter's electro-optical targeting system, icons to depict threats and targets in the air and on the ground, and the aircraft's own weapons. The F-35's forward looking infra-red sensor can project sensor video in a "picture-in-picture" against the ground and sky.

- **Ukranian military headset**

Figure 2.9: Ukranian HMD

The headset, known as the Circular Review System, integrates a traditionally protective helmet with the Hololens. Through optical and thermal cameras located on the vehicle it would provide a 360 degrees review with high resolution in real-time. Other specifications: streaming video without delay, highlight allies and enemies position, automatic target tracking. This headset is still in the prototyping phase.

- **BMW Motorrad and DigiLens helmet head-up display**

The system presented by BMW employs a glass display over the right eye that projects a variety of useful data (speed, temperature, fuel level, gear, ... ). Cameras contained within the helmet can deliver a live feed from the

Figure 2.10: BMW Helmet

rear. The prototype BMW helmet is also equipped with an integrated mini-computer and speakers, wirelessly controlled via a multicontroller fitted at the left side of the handlebars. Power is provided by two batteries that are contained at the lower rear part of the helmet and can last for five hours on one charge. BMW foresees even more functions for its HUD, such as connectivity with vehicle-to-vehicle communication systems that will transmit road information in real time from one rider to the other. Such a system could also connect to a future road-to-vehicle communication infrastructure.

- **RideOn Ski Goggles**

Figure 2.11: RideOn Ski Goggles

The goggles combine aviation-style inertial sensors, GPS, and a built-in video camera with a see-through display. Together, these modules project virtual graphics and features into the eye and onto their accurate, real-world counterparts. The goggles are connected to RideOn's app on the smartphone. The app brings connectivity to friends, lets you download from hundreds of resort maps worldwide, and allows customization of RideOn's settings so that they fit personal ride styles.

- **HTC Vive**
  The HTC Vive must be tethered to a PC with high performance. It uses IR sensors mounted on walls to map the user location in the physical space,


Figure 2.12: HTC Vive

integrating this into the virtual space. (70 sensors with 37 sensors on the headset). (da finire)

- **Oculus Rift**


Figure 2.13: Oculus Rift

The consumer edition Rift uses a 2160 x 1200 resolution display, Pentile OLED, working at 233 million pixels per second, with a 90Hz refresh rate. Video is sent to the Oculus Rift via HDMI, with an optional DVI adapter for laptops and newer graphics cards. It has sensors for the head tracking like the accelerometer, gyroscope, magnometer. The field of view of this device is about 110 degrees, it can be tethered with HDMI, USB 3.0 and 2.0. The total weight is 470 gr. It is available and costs $450. In order to work it need to be connected to a PC.

- **Epson Moverio BT-2000**

Designed with an Android operating system, the new Moverio smart headset allows enterprises and other organizations to create and share applications that tailor the product to meet their precise needs. (da finire)

Figure 2.14: Epson Moverio BT-2000

- **Microsoft Hololens**

Figure 2.15: Microsoft Hololens

Microsoft HoloLens, known under development as Project Baraboo, is a pair of mixed reality smartglasses developed and manufactured by Microsoft. The Microsoft Hololens utilize an accelerometer (to measure the speed in which the head is moving), a gyroscope (to measure the tilt and orientation of the head), and a magnetometer (to function as a compass and figure out which direction the head is pointing) to provide for truly immersive experience. In the Microsoft Hololens, the use of "mirrors" involves see-through holographic lenses (Microsoft refers to them as waveguides) that use an optical projection system to beam holograms into your eyes. A so-called light engine, emits the light towards two separate lenses (one for each eye), which consists of three layers of glass of three different primary colors (blue, green, red). The light hits those layers and then enters the eye at specific angles, intensities and colors, producing a final holistic image on the eye's retina. This device properties will be better discuss in the final comparison.

- **Meta 2**

  The Meta 2 development kit enables you to create holographic apps, tools,

Figure 2.16: Meta 2

and experiences. The headset displays holograms and digital content, and comes with a software development kit (SDK) built on top of Unity, the most popular 3D engine in the world. It requires a desktop CPU Intel Core i7. It will be better analyze in the final comparison.

- **ODG R7**

Figure 2.17: ODG R7

The glasses run a custom version of Android KitKat, dubbed ReticleOS. It's easy to control, using either a trackpad on the glasses themselves, or a paired handheld controller. ODG's glasses are still aimed primarily at enterprise customers and developers.

# Chapter 3

# System design

The technological scouting on the automotive field, and not only on that, has been useful for better understanding the potentialities of mixed reality technologies through the analysis of them. In addition, it was oriented to the identification of a possible hardware component to be used in our project.

For this reason, in order to design a solution for our use case, a high level conceptual architecture of the AR system to be implemented has been defined, considering the two possible approaches: the HUD system and the HMD system. In both cases the focus must be on how the data communication can take place between the device itself and other entities (that can be the garage of the racing car company on the track, sensors on the track or any other external source), how the voice communication between the driver and the garage can be implemented during the training/race considering the possible noises that surely are present during a car competition, how to supply power to the device, if it has to be autonomous there must be a battery and it must be considered the capacity of that battery or if it is connected to the battery of the car itself, how the data is processed, if there is need of high processing capacity inside the device or outside of it, and of course the electro-optical part of the device. Then if we consider the hud system wa take into account the support system for it, that means where it has to be located, if it can be moved by the driver and other features. While if we consider the HMD option it must be considered the helmet part, that has to protect the driver but also must be a good platform for mounting the display optics.

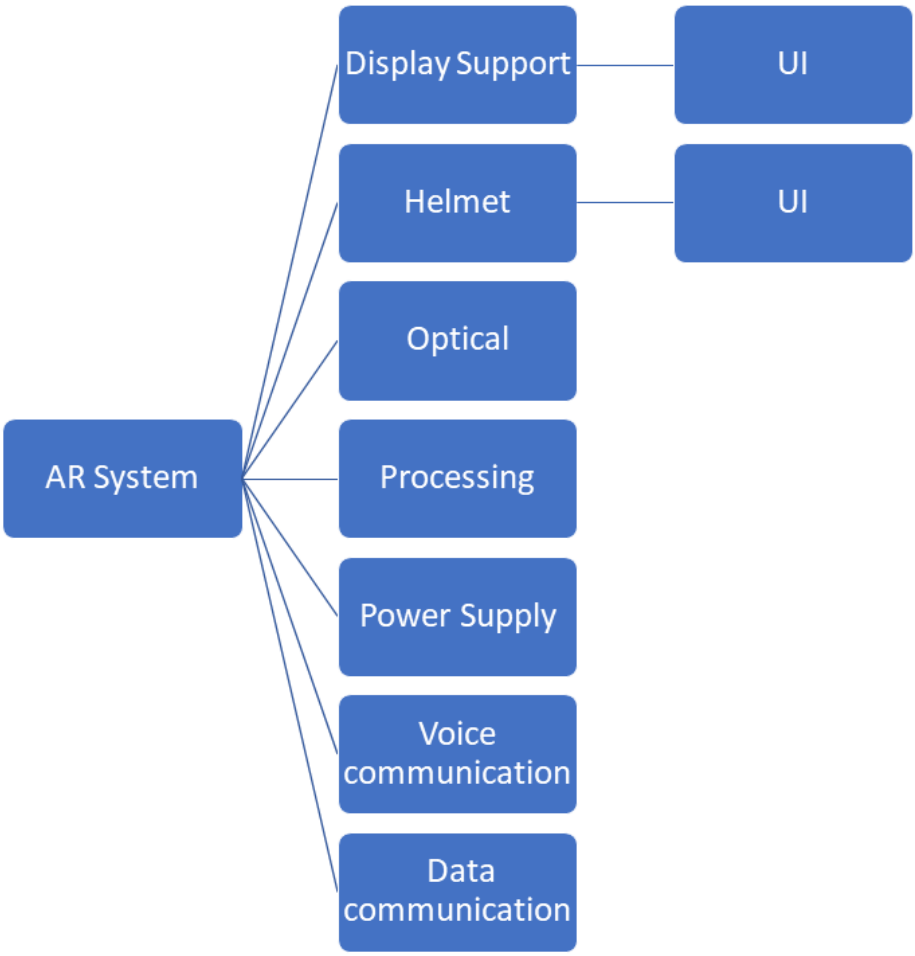


Figure 3.1: AR System

Since during a race the driver has already to wear an helmet is straight forward that the simplest and convenient choice is the development of a helmet-mounted display.

## 3.1 Helmet-mounted display design features

Helmet-mounted displays have been developed since 1960, since then technology has improved significantly. Their usage has become standard within the military community for flight application, as we have seen in the technological scouting performed the F-35 hmd is the most expensive device and the one with the highest number of functionalities. DEsignin a hmd system is not trivial, this mainly because every system where humans are key component is complex due to the human-machine interface that is really challenging.

In addition, it must be considered that the design of a HMD highly depends on it application and it is strongly environment-driven. Anyway, there are some common design features that are primarily driven by human perceptual and anthropometric limitations.

- **Ocularity**

  Ocularity refers to whether the HMD provides monocular biocular and binocular imagery. Monocular means that a single image source is viewed by a single eye, biocular means that a single image source is viewed by both eyes, binocular means that each eye views an independent image source. Biocular or binocular ocularity guarantees the widest field of view but is surely more complex. Viewing imagery with two eyes vs. one has been shown to yield improvements in detection as well as providing a more comfortable viewing experience (Boff and Lincoln, 1988; Moffitt, 1997).

- **Field of View**

  The fov concept has already been described before, briefly resuming fov describes how extensive the image apears to the user and is measured in degrees. When the HMD application is a full immersive virtual reality gaming for example a higher fov is raccomanded, to provide a more compelling sense od immersion. If the goal is a safety-of-flight-qualified HMD instead, a lower FOV reduces the head-supported mass/weight, which improves safety and reduces pilot fatigue. In this case FOV of 40 degrees horizontal by 30 degrees vertical can e enough.

- **Resolution**

Resolution refers to the apparent angular size of a displayed pixel or image element and the ability for the user to view and correctly interpret an object as imaged by that pixel (and others). Resolution contributes to overall image quality, but there is also a direct relationship with performance. The concept of resolution is directly connected to the FOV. Considering image 3.2, H = F tan theta , where F is the focal length of the collimating lens. If H is the size of the image source, then theta is the FOV, or the apparent size of the virtual image in space (which is desired to be large). If H is the pixel size, then theta is the resolution or apparent size of the pixel in image space (which is desired to be small). Thus, the focal length of the collimating optics simultaneously governs the FOV and the resolution. For a display with a single image source, the result is either wide FOV or high resolution, but not both at the same time. Generally, a larger FOV is preferred in order to provide a more immersive experience. But, also, high resolution (small pixels) is desired: how high depends on the user's task.

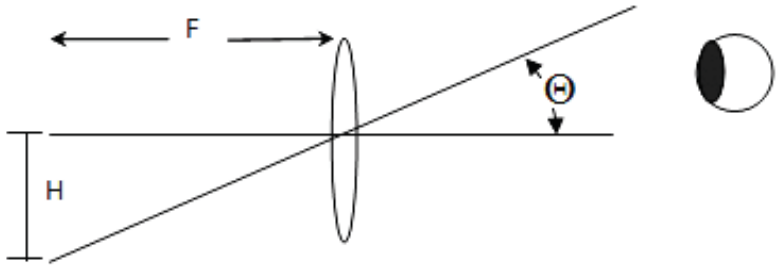


Figure 3.2: FOV- resolution relatioship

- **Optical design**

  In an HMD, the optics serve to: 1) collimate the image source (creating a virtual image, which appears to be farther away than just a few inches from the face), 2) magnify the image source (making the imagery appear larger than the actual size of the image source), and 3) relay the image source (creating the virtual image away from the image source, away from the front of the face). There are several optical design approaches common in HMDs in order to achieve one or more of these results. The optical design must provide a sufficiently large exit pupil or viewing eye box. For instance, a large exit pupil is important for a flight HMD, so the user does not loose the image if the HMD shifts on his head. A value of 12 to 15 mm has been deemed an acceptable value for these applications.

- **Optical distortion**

One of the important issues in an optical design is the control of residual optical aberrations such as focus, field curvature, and astigmatism. While this can be done with careful attention to the optical design, adding perhaps an additional lens or aspheric surface, distortion (defined as an off-axis image located at a different height than that expressed by paraxial equation) is more difficult to control, usually taking on a pincushion form in an imaging system.

- **Luminance and contrast**

  A critical optical issue is providing a good see-through transmission and at the same time an image with sufficiently high contrast against the high luminance background. A higher transmission requires higher image source luminance, which in-turn requires more power.

- **Helmet-mounted sensors**

  There is a large number of sensors that can be incorporated into a HMD. An example is an IMU sensor that provides information on the device's velocity, orientation and gravitational forces. Thanks to this sensor the rotation of the head can be tracked. Ambient light sensors enable the adjustment of the brightness of the display accordingly.

- **Visual vs. Auditory Mode for HMDs**

  The visual channel is the mode of choice for providing information at high rates. However, for certain tasks and situations an auditory display may be more effective. Auditory displays are best used for alerting, warnings, and alarms situations in which the information occurs randomly and requires immediate attention.

- **Safety role**

  The primary role of an helmet always has been to provide protection. This role has not changed and instead has been expanded with the introduction of HMDs to where the helmet is expected to serve as a mounting platform for the display without compromising the helmet's primary protective capability. These increasing demands means we must consider impact attenuation, head-supported weight, CM offset, frangibility, fit and comfort, retention and stability and their effects on head and neck biodynamics.

- **Frangibility**

  Frangibility refers to the ability of an HMD component to break free from the overall helmet-HMD system during a dynamic event. The purpose is to

"shed" mass from the HMD system, thereby reducing the risk of neck injury during a dynamic event such as a helicopter mishap or ejection. Frangibility often is desired and even required when the total head-supported weight and CM creates the potential for unacceptable risk of neck injury.

- **Helmet retention**

  Helmet retention refers to the ability of the helmet to stay in place on the wearer's head during dynamic events such as helicopter mishaps, ground vehicle accidents, or high speed ejections. This is critical because the helmet cannot perform its protective function if it has departed the wearer's head or it has rotated to a position that leaves the skull open to direct impact.

- **Perceptual and Cognitive considerations**

  All HMD components should earn their way onto the head because they reduce user workload and enable him to accomplish his mission. Information should not simply be a re-mapping of available information, but should be cognitively pre-digested to ease the transfer of information while not overloading working memory.

- **User adjustment**

  The selection and implementation of available adjustments must allow for individual differences while carefully avoiding complexity and minimizing the potential for misadjustment.

## 3.2 Devices Comparison

Once it is clear what features must be considered in the design of an helmet-mounted display and which are now the technological systems on the market, our attention was focused mainly on four headsets that could be adapted for our scope: the DAQRI Smart Helmet, the F-35's Helmet Mounted Display System, the Microsoft Hololens and the META 2.

The comparison made among these devices is focused on their optical features, their sensors, the processing and power capacity, on the connectivity and the operating system guested, the weight the availability on the market and least but not less important the price. In addition, it has been considered also the support to the main SDKs on the market.

#### 3.2.0.1 Final considerations

The F-35 helmet could have been a good reference for the analysis on the bio-dynamics point of view, its frangibility and retention are surely the best among them. But it is too warfight-oriented and is too expansive. The DAQRI smart Helmet, in their developer version, DAQRI Smart Glasses, offer a SDK for the Unity platform, but they have not a good resolution, that is important in our use case due to the fact that the driver needs to see objects on the long distance or detailed information that must be clear enough.

The Meta 2 and Microsoft Hololens comparison is the most interesting, so we will go deeper on this. One of the main difference is the fact that the Meta headset needs to be tethered to work, while Hololens do not. Hololens is a standalone computer running Windows 10 while Meta 2 allows you to move in a certain range of distance from the PC to which is connected, that must run, at least, a 64-bit Windows 8.1 OS. Of course a tethered approach gives more computing power to the device.

Talking about weight the Hololens weighs 579 grams, while Meta headset about 420 grams not considering the cables and head straps, in total would be 730 grams. On the development side, Meta 2 support Unity version 5 or higher. Hololens requires Visual Studio and Unity.

Meta 2 is definitely better on the resolution e field of view side, while it can not stand the competition on tracking and environment understand capabilities.

The biggest difference is that Meta 2 is not on the market yet, but it can be pre-ordered at the price of $949.00, while Hololens are already on the market at a price of $3000.00 and they are ready to the rapid increase of demand from developers for their device.

A solution with Microsoft Hololens has as advantages the fact that Microsoft's technology represent the state of art of our days. Drawbacks are the limited field of view of the device, problems with a slightly annoying reflection of violet light into display and the necessity of moving the BMS and processing power outside the headset.

A solution with the DAQRI Smart Helmet could be the best for an initial phase of testing due to the fact that the system is already inserted into an helmet, but also in this case the field ov view is quite limited.

The Meta 2 system has already the processing system and battery supply outside the device, has a good resolution and a display that should not cause problem of light reflection thanks to its higher location. Even though this advantages this solution can't be considered due to the fact that the system is not already on the market.

| | **DAQRI Smart Helmet** | **F-35 HMD** |
|---|---|---|
| Optical | True 4D, large field of view, high brightness | LCD Displays |
| Sensors | Tracking camera paired with a dedicated processor for AR applications, RGB camera, Stereo infrared cameras, Infrared light projector, Thermal camera, Intel RealSense LR200 Depth Sensor | Radar, Infra-red, Electro-optical targeting system, Six cameras (Distributed Aperture System) |
| Connectivity | USB | na |
| CPU and Memory | Intel Core m76Y75 | na |
| Power Supply | Rechargeable battery | n/a |
| Weight | na | 2,31 kg |
| OS | n/a | na |
| Development | na | na |
| Availability | For corporations only | Available |
| Price | $5,000 to $15,000 | $400,000 |

| | **Hololens** | **Meta 2** |
|---|---|---|
| Optical | See-through holographic lenses (waveguides), 2 HD 16:9 light engines, Automatic pupillary distance calibration, 35 degrees field of view | 2550x1440 resolution (60 Hz refresh rate), 90 degrees field of view |
| Sensors | Ambient light sensor, An inertial measurement unit (includes accelerometer, gyroscope, and a magnetometer), A depth camera, A 2MP photo / HD video camera, 4 Microphones | Sensor array for hand interactions and positional tracking, 720p front-facing camera |
| Connectivity | Wi-Fi 802.11ac, Micro USB 2.0, Bluetooth 4.1 LE | HDMI |
| CPU and Memory | Intel 32 bit architecture with TPM 2.0 support, Custom-built Microsoft Holographic Processing Unit (HPU 1.0), 64GB Flash, 2GB RAM | Not standalone - Requires a desktop CPU Intel Core i7 |
| Power Supply | 2-3 hours of battery life | HDMI cable for power |
| Weight | 579g | 730 g |
| OS | Windows 10 | Not standalone, Windows 8.1 64-bit or newer |
| Development | Visual Studio 2015 and Unity | Unity 3D version 5 or higher |
| Availability | Available | Not Available yet |
| Price | $3.000 | $ 949.00 |

## 3.3 Architecture design

Considering the car race context a first architecture, at a very high-level has been defined. The AR system incorporated into the driver helmet is located of course inside the car and has to communicate with it, it has also to communicate with the garage on the track to transmit information on the status of the car, the status of driver, the lap records and many other information that could be useful to define a strategy for the competition or to improve the driver and car performances.

In addition the system collects data that can be provided to the racing company headquarters where they can be analysed for statistics or other purposes.

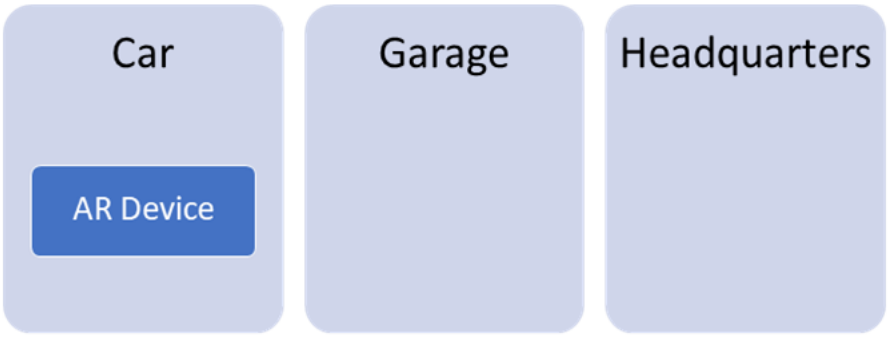


Figure 3.3: High-level Architecture


Figure 3.4: Headquarters internal architecture

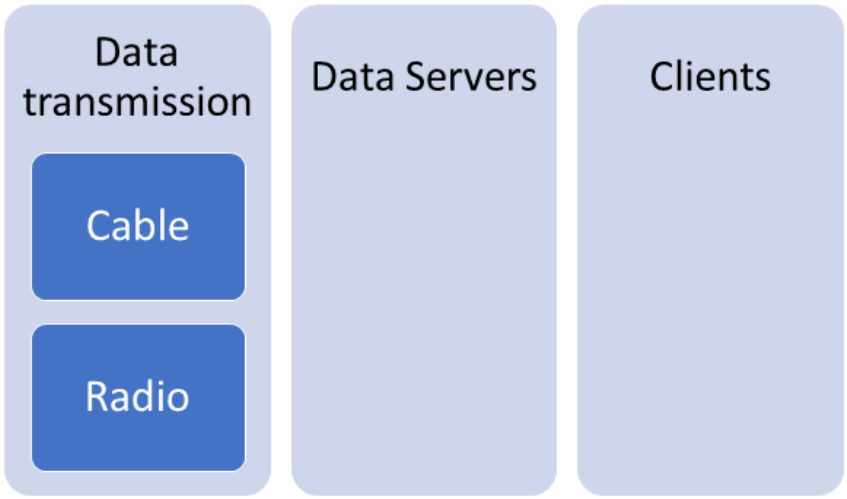


Figure 3.5: Garage internal architecture

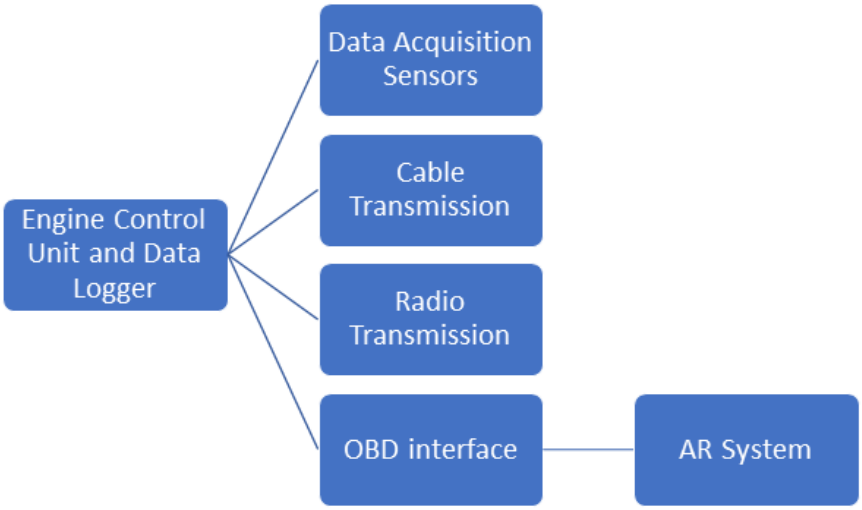


Figure 3.6: Car internal architecture

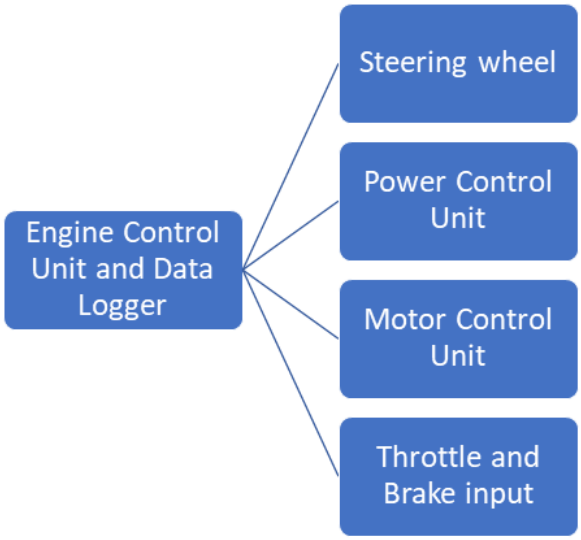


Figure 3.7: ECU possible configuration

The AR system incorporated into the helmet can be seen as composed by a battery management system, the power can be supplied by batteries inside of it but also by direct cables connected to the car, sensors, Wireless or Bluetooth connectivity, an holographic processor and a display. The car has internal and external sensors, a SOC processor, memory (RAM, SSD and DDR), Wifi connectivity and a I/O interface. The car internal information can be retrieved through a OBD (on-board diagnostic) interface.


Figure 3.8: AR System Architecture

# Chapter 4

# Development environment

A game engine is a software framework designed for the creation and development of video games, but not only for them. Despite the specificity of the name, game engines are often used for other kinds of interactive applications with real-time graphical needs such as marketing demos, architectural visualizations, training simulations, and modelling environments.

The core functionality typically provided by a game engine includes a rendering engine ("renderer") for 2D or 3D graphics, a physics engine or collision detection (and collision response), sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics.

The process of game development is often economized, in large part, by reusing/adapting the same game engine to create different games, or to make it easier to port games to multiple platforms.

In many cases game engines provide a suite of visual development tools in addition to reusable software components. These tools are generally provided in an integrated development environment to enable simplified, rapid development of games in a data-driven manner. Game engine developers attempt to "pre-invent the wheel" by developing robust software suites which include many elements a game developer may need to build a game. Most game engine suites provide facilities that ease development, such as graphics, sound, physics and AI functions. These game engines are sometimes called "middleware" because, as with the business sense of the term, they provide a flexible and reusable software platform which provides all the core functionality needed, right out of the box, to develop a game application while reducing costs, complexities, and time-to-market — all critical factors in the highly competitive video game industry.

Like other types of middleware, game engines usually provide platform abstraction, allowing the same game to be run on various platforms including game consoles and personal computers with few, if any, changes made to the game source code.

Often, game engines are designed with a component-based architecture that allows specific systems in the engine to be replaced or extended with more specialized (and often more expensive) game middleware components such as Havok for physics, Miles Sound System for sound, or Bink for video. Some game engines

such as RenderWare are even designed as a series of loosely connected game middleware components that can be selectively combined to create a custom engine, instead of the more common approach of extending or customizing a flexible integrated product. However extensibility is achieved, it remains a high priority for game engines due to the wide variety of uses for which they are applied.

## 4.1 Game engine components

Such a framework is composed of a multitude of very different components.

- **Main game program**

  The actual game logic has to be implemented by some algorithms. It is distinct from any rendering, sound or input work.

- **Rendering engine**

  The rendering engine generates 3D animated graphics by the chosen method (rasterization, ray-tracing or any different technique).

  Instead of being programmed and compiled to be executed on the CPU or GPU directly, most often rendering engines are built upon one or multiple rendering application programming interfaces (APIs), such as Direct3D or OpenGL which provide a software abstraction of the graphics processing unit (GPU).

  Low-level libraries such as DirectX, Simple DirectMedia Layer (SDL), and OpenGL are also commonly used in games as they provide hardware-independent access to other computer hardware such as input devices (mouse, keyboard, and joystick), network cards, and sound cards. Before hardware-accelerated 3D graphics, software renderers had been used. Software rendering is still used in some modelling tools or for still-rendered images when visual accuracy is valued over real-time performance (frames-per-second) or when the computer hardware does not meet needs such as shader support.

  With the advent of hardware accelerated physics processing, various physics APIs such as PAL and the physics extensions of COLLADA became available to provide a software abstraction of the physics processing unit of different middleware providers and console platforms.

  Game engines can be written in any programming language like C++, C or Java, though each language is structurally different and may provide different levels of access to specific functions.

- **Audio engine**

  The audio engine is the component which consists of algorithms related to sound. It can calculate things on the CPU, or on a dedicated ASIC. Abstraction APIs, such as OpenAL, SDL audio, XAudio 2, Web Audio, etc. are available.

- **Physics engine**

  The physics engine is responsible for emulating the laws of physics realistically within the application.

## 4.2 Game engines for Mixed Reality


Figure 4.1: Game Engines

### 4.2.1 Unreal Engine 4

Unreal Engine is an engine released from Epic Games, first showcased in the 1998 first-person shooter game Unreal. Although primarily developed for first-person shooters, it has been successfully used in a variety of other genres, including stealth, MMORPGs, and other RPGs. With its code written in C++, the Unreal Engine features a high degree of portability and is a tool used by many game developers today.

Epic Games for the use of Unreal Engine 4 under the free license agreement charges a 5% royalty based on gross revenue after the first $3,000 per product per calendar quarter,

The main programming language that can be used are C++ and Blueprints, a Visual Scripting system. Blueprints are authoring tools designed for non programmers so designers and other team members can help tweak and prototype.

UE4's Blueprint scripts resemble flowcharts where each box represents a function or value, with connections between them representing program flow.

Unreal allow the development of application for desktops, mobile devices, game consoles and VR systems

**Targets:**
*Desktop:* Windows, OSX, Linux, SteamOS, HTML5
*Mobile:* iOS, Android
*Console:* Xbox One, PlayStation 4, Nintendo Switch
*VR:* SteamVR/HTC Vive, Oculus Rift, OSVR, Google VR/Daydream, Samsung Gear VR

UE4 has some amazing graphical abilities including things like advanced dynamic lighting capabilities and a new particle system which can handle up to a million particles in a scene at one time.

UE4 gives full access to the C++ source code allowing editing and upgrading anything in the system.

Epic provides multiple official video tutorials, lots of free example projects and content, an extensive wiki and regular streams showing how to use latest features. It allows a texture/material artist or VFX artist to create amazing effects from the ground up.

On the other side, compared to other engines, UE4 may seem to perform various actions considerably slower. Actions like starting the engine, opening the editor, opening a project, rebuilding shaders, updating references, calculating lightmaps or saving projects, may take too long causing waste of development time.

Unlike Unity, there is poor documentation, causing a steep learning curve, especially for the beginners.

### 4.2.2 CryEngine

CryENGINE is an extremely powerful engine designed by the development company Crytek that was introduced in the first *Far Cry* game. The CryEngine software development kit (SDK), originally called Sandbox Editor, is the current version of the level editor used to create levels for CryEngine by Crytek. Tools are also provided within the software to facilitate scripting, animation, and object creation. The editing style is that of the sandbox concept, with the emphasis on large terrains and a free style of mission programming. The editor can also construct indoor settings.

CryEngine features, source code are completely free, with no royalities. Developers looking for optional access to additional training and support resources

beyond the community can consider a CRYENGINE membership for $50 to $150 per month.

The supported programming languages are C++, C# and Lua.

It allows the development of application for desktops, mobile devices, game consoles and VR systems.

**Targets:**
*Desktop:* Windows, Linux
*Mobile:* iOS, Android
*Console:* Xbox One, PS4
*VR:* Oculus Rift, HTC VIVE

Where Cry Engine really shines is with rendering scenes of nature: has realistic water effects that even simulate ocean physics, features allowing for realistic weather effects and an advanced volumetric cloud system. CryEngine has some C# templates and also a C# based system to write your function/ideas in to your game

Flow graphs resemble flowcharts where each box represents a function or value, with connections between them representing program flow. This provides a better at-a-glance indication of game logic than a simple list of events, and makes complex behaviours easier to accomplish. Cry Engine has a big marketplace where all users can buy and sell assets, 30% of all revenue generated from asset sales will go to CRYENGINE, with the remaining 70% going directly to the Asset Vendor. It has a good documentatioon, dozen of high-quality video tutorials and an active community.

Except for basic FPS games, develop a complete project requires solid knowledge of C++, Flash, ActionScript and Lua.

At the moment CryEngine doesn't support AR development but only VR Development on a limited number of devices.

### 4.2.3 Unity

Unity is a cross-platform game engine developed by the American video game development company Unity Technologies. It is primarily used to develop video games and simulations for computers, consoles and mobile devices. First announced only for OS X, at Apple's Worldwide Developers Conference in 2005, it has since been extended to target 27 platforms

Unity comes with four license options: Personal, Plus, Pro, Enterprise. Unity Personal is a completely free edition, there is no need for royalities or credit card but under certain limitations, the company must earn less than 100k dollars in

order to use this kind of license. With this edition the developer has access to all the engine features and platforms, to the continuous updates and beta releases, also to the core analytics, the cloud build and ads management. Unity Personal is a great place for beginners and hobbyists to get started. Plus version costs $35 per month, it enables the customization of the splash screen, the development on Pro Editor Skin UI, real time analytics to optimize your game monetization with Ads. It also enables the automatic capture of app errors in real-time across devices, platforms, and builds. This licence is allowed for company with a revenue capacity till $200k . Unity Pro has no limit on revenue, costs $125 per month, it enables concurrent cloud builds and all the other features already in the Plus version. Unity enterprise is customizable according to the magnitude of a company, is the only licence that allow access to the source code in order to customize Unity to the creation needs of the development team. Unity offers the support of its engineers to train the developers.

The main programming language that can be used is C#, *Boo*, a flavour of Python, was deprecated with the release of Unity 5 and *UnityScript* was deprecated this August.

The supported device platforms for this engine are both Windows and OSx. Unity allow the deploy of an application on many targets, from mobile to console systems or TV targets but also VR targets of course.

**Targets:**
*Desktop*: Windows, OSX, Linux
*Mobile*: Windows Phone, iOS, Android, BlackBerry 10, Tizen
*Console*: Xbox 360, Xbox One, Wii U, PlayStation 3, PlayStation 4, PlayStation Vita, Nintendo Switch, Nintendo 3DS
*Distributed targets*: Unity Web Player, WebGL
*VR targets*: Oculus Rift, Gear VR, Google Daydream, Cardboard, SteamVR/HTC Vive
*TV targets*: Android TV, Apple TV, Samsung Smart TV

Unity3D provides an exhaustive documentation where everything is given a full description supplied by a number of examples as well as video and text tutorials and live training sessions to understand the ins and outs of the engine.

In addition there's an ever-growing community that can offer advice to help resolve any situations that may arise. Along with the official Unity resources, there are many high quality (and often free) third party tutorials available.

Unity's modular system and usability allows for quickly developing a prototype of an idea. It has features like drag and drop editing, shaders, animation and other systems already in place to allow diving right into developing a game.

For those developers who can't afford an artist, or aren't skilled enough to create their own art, Unity features an Asset Store full of a wide variety of free and paid assets that can be easily added to a game.

The way the editor is structured, by setting scripts on objects, and the use of a high-level language, C#, makes it easy to learn.

The editor GUI is very powerful and intuitive. It allows pausing gameplay and manipulating the scene at any time as well as progress gameplay frame by frame.

Even-though the advantages pointed out, there are some drawbacks. Unity has a weak memory management and without an enterprise license the source code is not accessible.

Unity3D uses very unique approach for doing things. Most of the knowledge acquired while using it, would be completely non transferable to other engines.

For what concerns mobile development, even a Blank Project, Needs 18MB for the APK file (on Android).

Unity has a limited 3D modelling capabilities, but it allows the developer to import from other tools.

### 4.2.4 Unity as final choice

Nowadays Unity is the most popular and widely used game engine, especially for Augmented Reality applications.

The reason may be that it provides an exhaustive documentation where everything is given a full description supplied by a number of examples as well as video and text tutorials and live training sessions to understand the ins and outs of the engine. In addition there's an ever-growing community that can offer advice to help resolve any situations that may arise.

Along with the official Unity resources, there are many third party tutorials available. For our use case Unity is the best choice because it has the SDKs for the main AR engines that are Wikitude and Vuforia, and not only for them. In addition, as said before, since it is not the best tool for 3D graphic development it offers the possibility of importing materials from third party sources and many other graphic tools. As far as concerns the programming side, it is the best choice due to the possibility of develops in a great object oriented language such as C#, but also Javascript.


Figure 4.2: Augmented Reality SDKs for Unity

#### 4.2.4.1 ARToolkit SDK

ARToolkit is sponsored by DAQRI. Thanks to DAQRI's OpenSource release of ARToolkit, all of the features described below are available for free on a variety of platforms. The OS supported are iOs, Android, Windows Phone, Linux, Windows, MacOS X. The only digital Eyewear supported is Epson Moverio.

ARToolkit provides support for three general categories of tracker:

- **Natural Feature Tracking (NFT)**
  NFT supports freeform 2d images which may not have a clearly defined and consistent outside edge.

- **Traditional template square marker**
  These markers are generally a fairly simple icon with a mandatory solid black border around the periphery. These are best thought of as "designed" markers rather than images that you may already exist organically.

- **2D Barcode Markers**
  These markers are predefined (in the SDK itself) and are typically highly optimized for rapid detection and solid tracking in variable lighting conditions. Because they're so highly optimized, it's typically possible to have more of them detected and tracked simultaneously, and they'll be rapidly detected under circumstances that other types of tracker might not be detected at all. These look more like fiducial markers or low-resolution QR Codes than other types of marker, so they'll typically be obvious where you might want something more discreet.

There are a number of features that are somewhat unique to ARToolkit. Multiple cameras are supported and not only in stereo configuration. The presence

of overlapping areas, such as found in a stereo camera arrangement, provides extremely robust pose estimation for the camera array. It must be highlighted that ARToolkit is one of comparatively few SDKs that supports Windows Phone. In addition, it has a very robust tracking for markers over a range of distances Another interesting feature is the existence of a JavaScript-based version for integration into web frameworks. Thanks to the Open Source nature of the ARToolkit SDK (it's released under LGPLv3), it has Unreal Engine support that has benen developed with the contribute of one of its users.

A cncern about the use f ARToolkit is that there is no enterprise support, so a company could not want to take risks using this product.

#### 4.2.4.2 Vuforia SDK

It is perhaps the most well-known of the commercially available Augmented Reality SDKs. Vuforia was originally developed by Qualcomm and optimized for their chipsets. It supports many OS: iOS, Android, Windows, Windows Holographic. The digital eye-wear supported are the Hololens, ODG(R7), Epson Moverio (BT-200).

The target types identified by this SDK are:

- **Complex Objects**
  Objects scanned using an Android app that develops an internal model of the object in question
- **User-defined Images**
  They can be both locally stored and "cloud-based"
- **Cylinders**
  User-defined wrap-around imagery
- **Text**
  100,000 words or a custom vocabulary
- **Boxes**
  Simple cubes and rectangular boxes with unique user-defined side images Frame markers (QR-code like, limited number of unique ones, extremely fast recognition in challenging lighting conditions)

Pricing Structure and Licensing Vuforia source code is not open, their licences types are various. It is free if you use a watermarked version, it costs $499 per application with no watermark but without cloud recognition that is the service offered by Vuforia to manage online the Image Targets.

To have access to this service the subscription cost from $99 to $999 a month depending on recognition volume.

#### 4.2.4.3 Wikitude

Wikitude's cross-platform augmented reality SDK combines 3D Markerless Tracking technology (SLAM), Object Recognition and Tracking, Image Recognition and Tracking, as well as Geo-location AR for apps. The supported OS are : Android, IOs. The digital eyeware supported: Epson Moverio, Vuzix M100 and ODG R-7.

The main features offered are:

- **Object recognition**
  This technology allow real time and 360 degrees AR experiences around real world objects.

- **Instant Tracking** It is the first feature using Wikitude's SLAM technology (Simultaneous Localization and Mapping). It allows to easily map environments and display AR content without the need of a target image (markerless) for indoors and outdoors environments.

- **Multiple Targets**
  This feature enables recognition of several images simultaneously. Once the images are recognized, developers will be able to layer 3D models, buttons, videos, images and more on each target.

- **Extended Tracking**
  It allows developers to go beyond targets. Once the target image is recognized, users can continue the AR experience by freely moving their devices without the need of keeping the marker in the camera view.

- **3D Augmentations**
  The Wikitude SDK can load and render 3D models in the augmented reality scene.

- **Cloud Recognition**
  Wikitude's Cloud Recognition service allows developers to work with thousands of target images hosted in the cloud.

Wikitude offers different types of licences: SDK PRO and PRO 3D, Cloud and Enterprise. The SDK Pro license offers only the Geo-location feature, the 2D Image Recognition and the 3D Engine for a $2490 fee a year. In the SDK PRO 3D version the 3D Markerless Tracking technology and Object recognition

features are added for $500 more than the no-3D version. The other versions include Cloud recognition and costs $4490 per year or more.

# Chapter 5

# Prototype design and development

The main goal of the developed prototype was to give the user an idea of the final result. It is essentially a driving simulation in the case you are wearing the HMD with the additional informations projected on the headset display.

The development was carried on in three phases. A first phase was a first approach to the development environment, that is Unity 3D, with a simple simulation of the driving scene. A second phase was the implementation of a simulation using a library for Unity, Realistic Car, that helps the developer to simulate the real mechanic of the car. A third and last phase was the improvement of the graphic and the addition of other useful features.

Even though the final application will be an augmented reality application, it is still useful realize a prototype for the Microsoft Hololens that will be the final target.

The figure shows how the devices exist on the mixed reality spectrum, the focus is on the devices produced by Microsoft, among them of course the Hololens. It can be seen that today there is no device that can run experiences across the entire spectrum, however the Hololens can support a quite big range of it.

It is predicted that in the future holographic devices will become more immersive and immersive device will become more holographic. Over time more and more new devices will expand their range within the mixed reality spectrum.

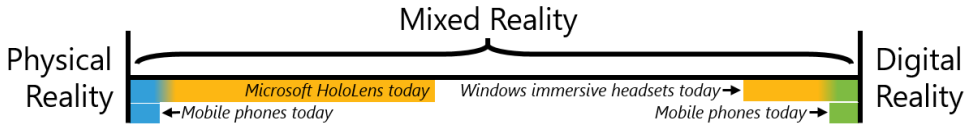


Figure 5.1: Devices on the mixed reality spectrum

## 5.1 HoloLens hardware review

HoloLens has see-through holographic lenses, three lenses laid one over the other allow the HoloLens to combine red, green, and blue images into color holograms. There are 2 HD 16:9 light engines. Hololens provides automatic pupillary distance calibration. Microsoft says that the holographic resolution of the device is 2.3M

total light points, while the holographic density is about 2.5k radiants (light points per radian). The more radiants and light points there are, the brighter and richer the holograms become.

Figure 5.2: The display

HoloLens has sensors for environment understanding and human understanding. There is an IMU (Inertial Measurement Unit) , an IMU works by detecting linear acceleration using one or more accelerometers and rotational rate using one or more gyroscopes. There are four environment understanding cameras and a depth camera. To record the surrounding environment there is also a 2MP photo / HD video camera. In order to capture the environment state there are four microphones and an ambient light sensor.

Figure 5.3: The sensor bar

The Hololens are capable of tracking the user gaze, some gesture inputs and they provide voice support. There are built-in speakers. It is possible to regulate the volume and the brightness of the device.

The possible connections to the device are through Wi-Fi 802.11ac, Micro USB 2.0 or Bluetooth 4.1 LE. It is predicted that the battery should last 2-3 hours if regularly used, up to 2 weeks of standby time.

Figure 5.4: The motherboard

The processing part is composed by an Intel 32 bit architecture with TPM 2.0 support and a custom-built Microsoft Holographic Processing Unit (HPU 1.0) to which is dedicated 1 GB of RAM. There are 2 GB of RAM and a 64 GB flash memory.

HoloLens weighs 579g, it has been tested and found to conform to the basic impact protection requirements of ANSI Z87.1, CSA Z94.3 and EN 166.

## 5.2 Hololens inputs

Gaze, gesture and voice (GGV) are the primary means of interaction on HoloLens.

Gaze is the first form of input and is a primary form of targeting within mixed reality. Gaze tells where the user is looking in the world. Like in the real world, when you intend to interact with an object you look at it, it is the same with gaze.

Mixed reality headsets use the position and orientation of your user's head, not their eyes, to determine their gaze vector, it is like a laser pointer that goes straight ahead from directly between the user's eyes. As the user looks around the room, the application can intersect this ray, both with its own holograms and with the spatial mapping mesh to determine what virtual or real-world object the user may be looking at.

On HoloLens, interactions should generally derive their targeting from the user's gaze. Once an interaction has started, relative motions of the hand may be used to control the gesture, as with the manipulation or navigation gesture.

A cursor (or other auditory/visual indication) can give the user confidence in what they're about to interact with. Typically the cursor is positioned in the world where the gaze ray first interacts an object, which may be a hologram or a real-world surface.

Interaction is built on gaze to target and gesture or voice to act upon whatever element has been targeted.

HoloLens currently recognizes two core component gestures that are *Air tap* and *Bloom*. These two core interactions are the lowest level of spatial input data that a developer can access.

- **Air tap**
  Air tap is a tapping gesture with the hand held upright, similar to a mouse click or select. This is used in most HoloLens experiences for the equivalent of a "click" on a UI element after targeting it with Gaze.

- **Bloom**
  Bloom is the "home" gesture and is reserved for that alone. It is a special

system action that is used to go back to the Start Menu. It is equivalent to pressing the Windows key on a keyboard or the Xbox button on an Xbox controller. The user can use either hand.

Apps can recognize more than just individual taps. By combining tap, hold and release with the movement of the hand, more complex composite gestures can be performed.

For gestures on HoloLens, the hand must be within a “gesture frame”, in a range that the gesture-sensing cameras can see appropriately. In the case of continuous gestures in particular, there is some risk of users moving their hands outside of the gesture frame while in mid-gesture (while moving some holographic object, for example), and losing their intended outcome. As the hand nears the edge of the gesture frame, it is provided a direction vector, which can be showed to the users so they know how to move their hand to get it back where HoloLens can see it.

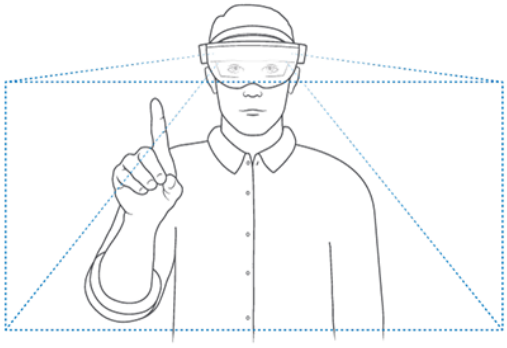

Figure 5.5: The Hololens gesture frame

Also the voice can be used to complete an interaction. Voice is a powerful and convenient way to control the system and apps. As a user targets any button through gaze or pointing, they can say the word "Select" to activate that button. "Select" is one of the low power keywords that is always listened for.

## 5.3 Hololens emulator

Microsoft offers an important tool for the developers who want to try the Hololens experience even without the physical device, that is the HoloLens emulator.

It allows you to test holographic apps on your PC and comes with the HoloLens development toolset.
The emulator uses a Hyper-V virtual machine. The human and environmental inputs that would usually be read by the sensors on the HoloLens are instead simulated using the keyboard, mouse, or Xbox controller. Apps don’t need to be


Figure 5.6: The Hololens Emulator

modified to run on the emulator and don't know that they aren't running on a real HoloLens.

Controlling the emulator is very similar to many common 3D video games. The emulator can be controlled by directing the actions of a simulated user wearing a HoloLens. The actions move that simulated user around and apps running in the emulator respond like they would on a real device.

The W,A,S, and D keys on the keyboard, or the left stick on an Xbox controller are used to walk forward, back, left, and right.
Clicking and dragging the mouse, using the arrow keys on the keyboard, or the right stick on an Xbox controller can be used to look up, down, left, and right.
Right-clicking the mouse, pressing the Enter key on the keyboard, or using the A button on an Xbox controller simulate the air tap gesture.
Pressing the Windows key or F2 key on the keyboard, or pressing the B button on an Xbox controller can be used to make the bloom gesture.
Holding the Alt key, holding the right mouse button, and dragging the mouse up / down, or in an Xbox controller holding the right trigger and A button down and moving the right stick up and down simulate the hand movement for scrolling.

## 5.4 Development basics

A developer creating a holographic application with Unity for Hololens need to switch between Unity and Visual Studio to build the application package that is deployed to the device.

Once the project is exported to Visual Studio is possible to change the scripts directly on Visual Studio selecting the option that enables the inclusion of all the C# code. In this way it is possible to use the same instance of Visual Studio

for writing scripts and building/deploying the project. However it is necessary to re-export the project in case it is needed to add or remove assets, to change any value in the Unity inspector tab, to add or remove objects in the scenes or to change any Unity project setting.

When developing an application for a mixed reality headset, you must know that the camera is the center of the holographic world.

The Unity Camera component automatically handles stereoscopic rendering and follow the head movement and rotation when the project has "Virtual Reality Supported" selected with "Windows Mixed Reality" as the device.

When running on an immersive headset, you are rendering everything the user sees, and so you can keep the skybox feature. When running on a holographic headset like HoloLens, the real world should appear behind everything the camera renders.

In this case there is no need to see the real world behind the lenses, so we can keep the skybox camera. The other Camera Features to be adjusted are its positioning and the clip planes.

Since the Main Camera is tracking movement of the user's head, the starting position of the user can be set by setting the starting position of the Main Camera that can be set as (X: 0, Y: 0, Z: 0) for simplicity.

As far as concerns clip planes, near and far clipping planes are imaginary planes located at two particular distances from the camera along the camera's sight line. Only objects between a camera's two clipping planes are rendered in that camera's view. Any parts of objects in the scene closer to the camera than the near clipping plane, or farther from the camera than the far clipping plane, are not rendered.

Microsoft developers recommand a clipping render distance at 85cm with fade-out of content starting at 1m. Content should be designed to minimize the need for interaction closer than 85cm from the user, but when content must be rendered closer than 85cm a good rule of thumb for developers is to design scenarios where users and/or holograms do not move in depth more than 25% of the time.

The key to holographic rendering is changing your app's view of its holograms each frame as the user moves around, to match their predicted head motion. Seated-scale experiences that respect changes to the user's head position and head orientation can be built using a stationary frame of reference.

Some content must ignore head position updates, staying fixed at a chosen heading and distance from the user at all times. The primary example is 360-degree video: because the video is captured from a single fixed perspective, it would ruin the illusion for the view position to move relative to the content, even though the view orientation must change as the user looks around.

This project doesn't need the user to walk or move in the surrounding space, so it is possible to build a seated-scale experience. In addition we can consider that a driver's field of view during a race is quite limited, mostly due to the high speed and the short reaction time needed. This can be seen in the many videos on-line about eye tracking during a race, the analysis of the gaze of a professional pilot shows how there is no rotation of the head but it is mostly the gaze that moves fast from a point to another on the track.

Expert drivers have trained their brain to speed up eye movements between focal points and understand new information more quickly, enabling them to take in and process more information than non-pro drivers.

A video made by Sky Sport in 2016 shows the F1 driver Nico Hulkenberg roaring out of a pit lane and checking his mirror in just a tenth of a second, "approaching the shortest amount of time a human can look at something" while processing the related information. Regular drivers typically need at least half a second to do the same thing.


Figure 5.7: Eye Tracking on an F1 car

It has been also interesting observe some videos on-line of people who tried to drive for some miles with the Hololens on. This tests are of course illegal, however they provide a good starting point for the development of the interface.

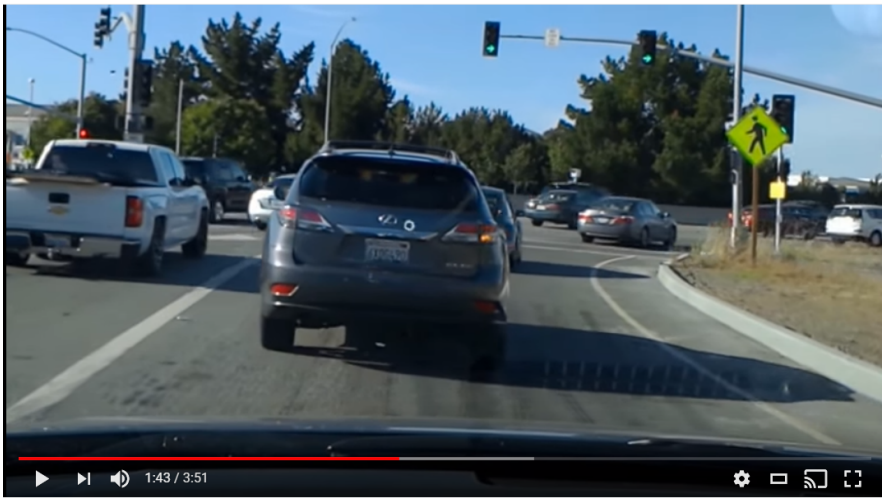


Figure 5.8: Test drive with Hololens

## 5.5 First phase

The first thing done is the creation of a new 3D project. When creating a 3D project on Unity any image imported is not assumed to be a 2D image, called Sprite in Unity, the Scene view is set to 3D but can be switched to 2D at anytime, the default game objects have real time directional light. The camera is set to be perspective, that means that the images are rendered with perspective intact, not uniformly.

In the first phase the camera is attached to a Game object, a sphere, that moves constantly forward. The speed and the number of rotation per minute of the engine are proportional to the space traveled by the object.

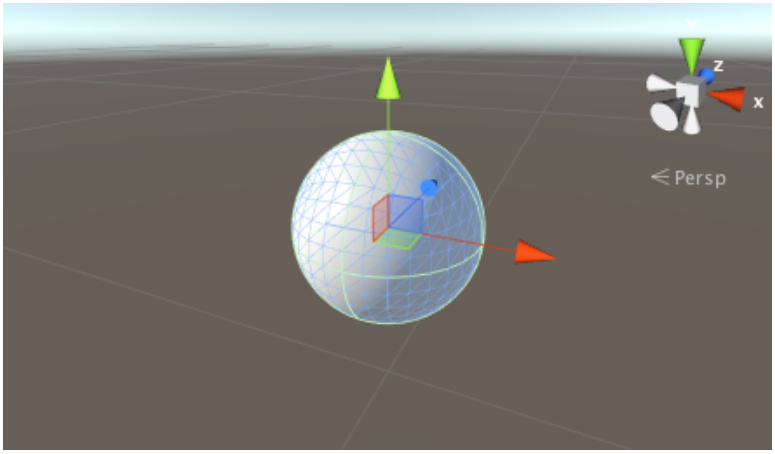


Figure 5.9: A primitive object

When the speed reached is 300 km/h the simulation ends. The landscape is composed by a two lanes street surrounded by some trees, the same surrounding is cloned and located immediately after the previous one as soon as the game

object exits the zone called *Collider*, this collision issues an action defined in the scripted attached to the object.

```
private void OnTriggerExit(Collider other)
{
   if (other.CompareTag("Player"))
   {
      component_transform.Translate(315, 0, 0);
      sceneObjects_transform.Translate(0, 0, 100);
   }
}
```

A *Collider* component define the shape of an object for the purposes of physical collisions. A collider, which is invisible, need not be the exact same shape as the object's mesh and in fact, a rough approximation is often more efficient and indistinguishable in gameplay, indeed the simplest and so called primitive collider has been used, the *Box Collider*.

The scripting system can detect when collisions occur and initiate actions using the *OnCollisionEnter* function. However, it is also possible to use the physics engine to detect when one collider enters the space of another without creating a collision. A collider configured as a *Trigger* (using the *Is Trigger* property) does not behave as a solid object and will simply allow other colliders to pass through. When a collider enters its space, a trigger will call the *OnTriggerEnter* function on the trigger object's scripts.

Event if it is not useful for the effective interface, the sphere object has been then substituted by a car skeleton object, following one of the tutorial proposed by the Unity manual. To the car object has been added a Physics 3D Rigidbody component with a mass set to 1500kg in order to make the object much heavier. After that a primitive object like a *Cube* is created as child of the car, with its collider. Then the wheels objects and colliders are created as children of the car object and positioned accordingly to it. To make the car drivable, a controller for it must be created and attached to it.

Then the project can be built and exported to visual studio and tested on the Hologram Emulator.

The information showed are the speed and the rotation per minute of the motor. In order to not occlude the vision of the driver the best position for the information is in the corner of the field of view.
The rpm value is on the top-right corner, while the speed on the top-left corner.

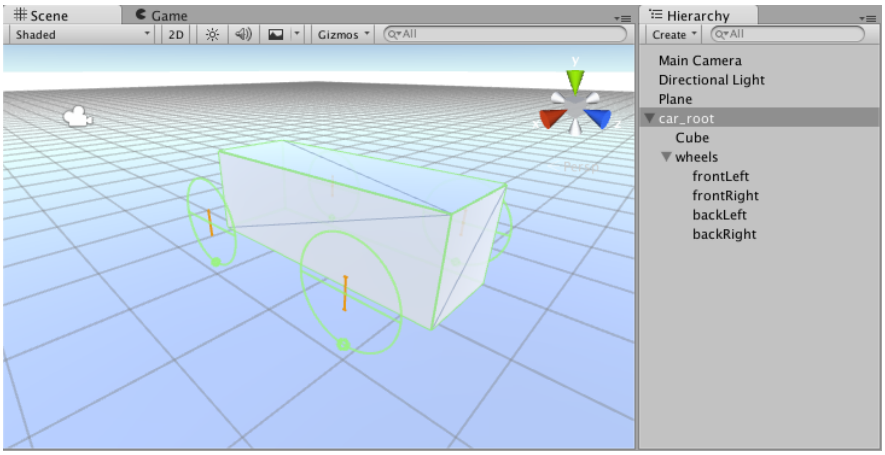


Figure 5.10: Car model implementation


Figure 5.11: Simple UI

## 5.6 Second phase

In the second phase a unity package retrieved in the asset store has been used. The usage of this package requires that a controller script must be attached to the car object. To improve the design and test the capabilities of the package an external 3D model of a sport car has been imported into the project.

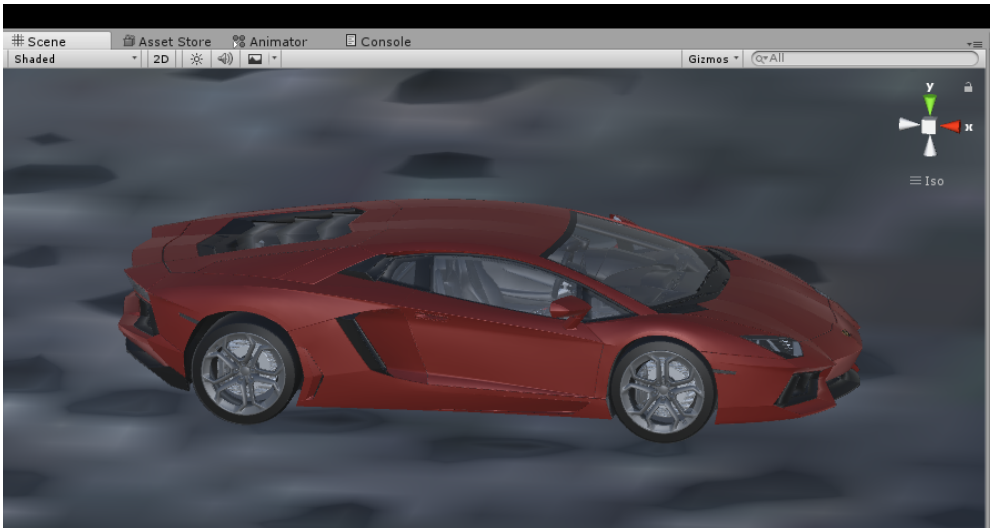


Figure 5.12: Sport car 3D model

In order to create a new vehicle within the project, first of all, the directions of the axises must be checked: Y must be up, Z must be forward, X must be right. Then the tool handles must be set in the object pivot point and not in global rotation.

As soon as the controller script is added to the car object the RigidBody Component is added. The mass is set as before to about 1500 kg. In addition the library automatically creates wheel colliders with the proper radius, suspension, friction, once that the wheel objects are set.

According to the shape of the car a collider for the main structure has been created.

The important parameter to configure is then the Center of Mass (COM). The positioning of it influences the behaviour of th car. In general it should be set just below the front seats, that is because usually the engine and transmission of a vehicle is at the front of the car, and they are the heaviest objects in it. This case is the RWD layouts, this implies that the driven wheels are located at the rear.

The surrounding environment is the same as in the phase one, with the street and tree object translated forward as soon as the car exits the previous zone. In order to make the interface just like the user is directly looking outside the

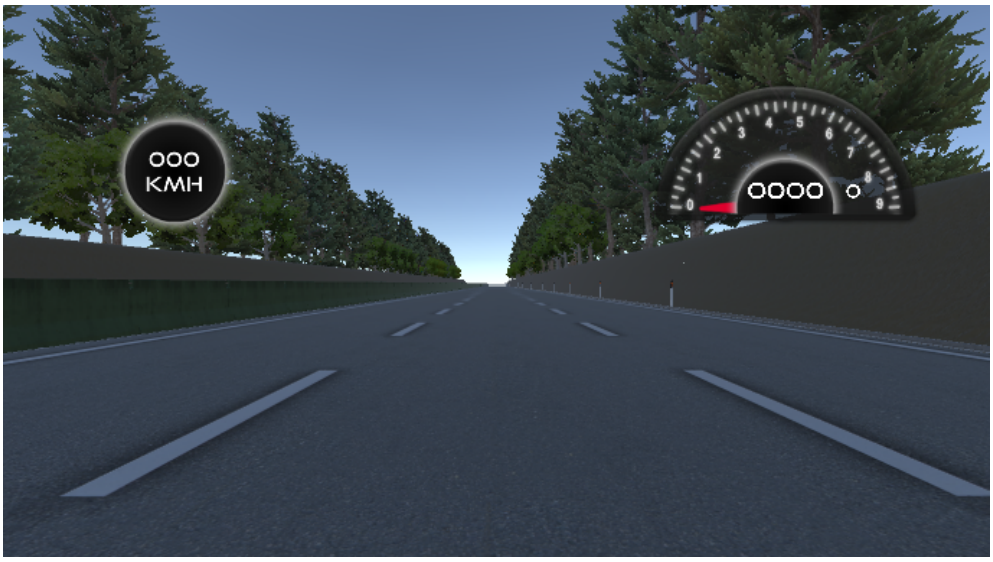


Figure 5.13: Phase 2 interface

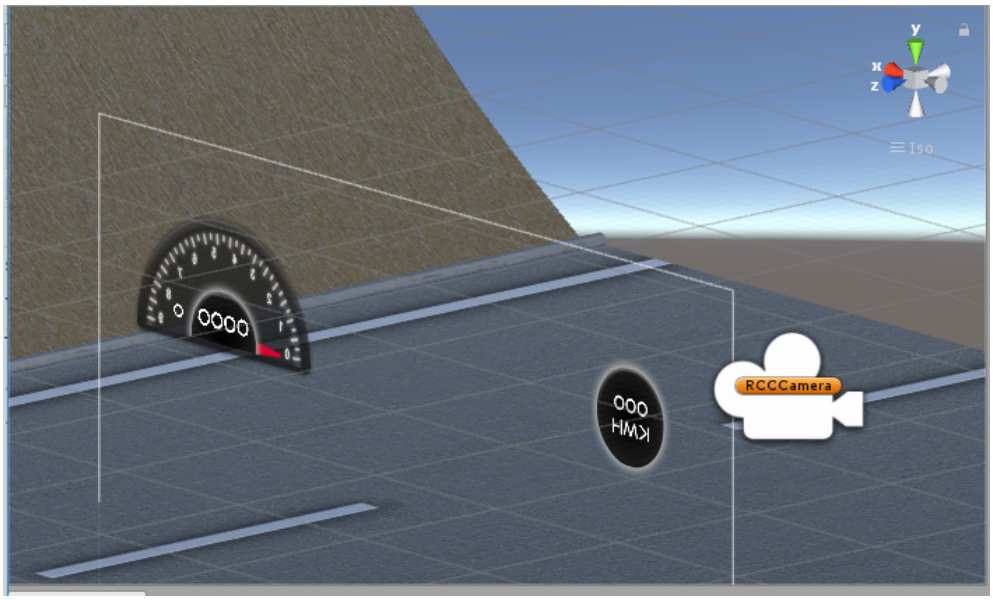


Figure 5.14: Phase 2 Scene View

windshield in front of him, every mesh render of each part that composes the car chassis has been disable. In few words the car in this way is "invisible".

In addition the interface has been improved. The color is lighter. The opacity of the objects in the information sections are lowered, in order to not occlude totally the real world behind of them.

## 5.7 Third phase

In the last phase, in order to make a realistic simulation of a track, from the access store a package has been downloaded for free. The *"Race track around the lake"* asset contains the models for each element in the scene, they are the track model, a bridge, the terrain with the vegetation, water, road signs, rocks and electric towers. In the final solution not every basic element has been used in order to make the Hololens processing and rendering faster.

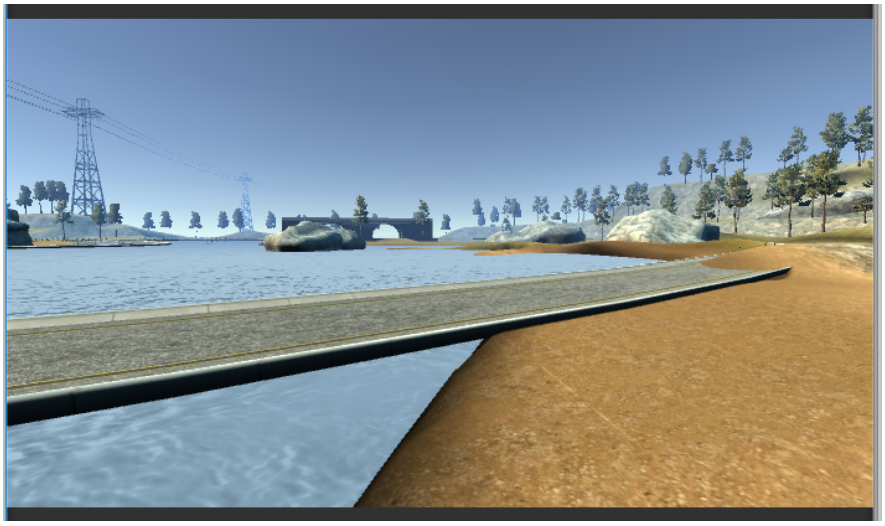

Figure 5.15: Race track around the lake

The car cantroller asset used in the previous phase was reused exploiting the AI feature.
The AI feature is based on the Unity's *NavMesh*, therefore the navigation mesh must be created and baked for the scene.

In Unity the *NavMesh* is a class that can be used to do spatial queries, like pathfinding and walkability tests, set the pathfinding cost for specific area types, and to tweak global behavior of pathfinding and avoidance. The process of creating a NavMesh from the level geometry is called NavMesh Baking. The process collects the Render Meshes and Terrains of all Game Objects which are marked as Navigation Static, and then processes them to create a navigation mesh that approximates the walkable surfaces of the level.

Unity supported mashes are triangulated or Quadrangulated polygon meshes. Every surface must be converted into polygons. Unity allows the importing of meshes from third-party 3D modeling softwares. Materials are used in conjunction with Mesh Renders, Particle Systems and other rendering components in Unity, they play an essential part in defining how an object is displayed.

The properties that a Material's inspector displays are determined by the

Shader that the Material uses. A shader is a specialized kind of graphical program that determines how texture and lighting information are combined to generate the pixels of the rendered object onscreen.

Textures are bitmap images. A Material may contain references to textures, so that the Material's shader can use the textures while calculating the surface colour of an object. In addition to basic colour (albedo) of an obejct's surface, textures can represent many other aspects of a material's surface such as its reflectivity or roughness.

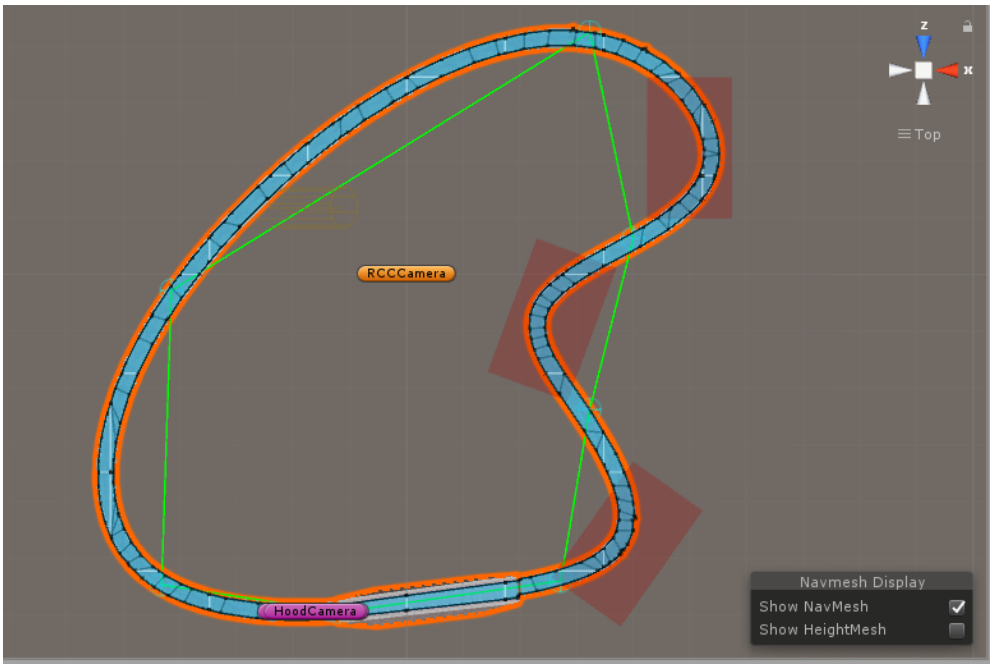


Figure 5.16: NavMesh baking

Once that the NavMesh baking is done it appears as in the figure 5.16.

An AI controller has been added to the car created in the previous phase. The car will use the *NavMesh Agent* based on the waypoints set on the track. The car will follow all the waypoints, there is no need to receive any input from the user.

The Brake zones objects of the asset have been exploited to simulate the detection of curves, when the car enters in the zone an arrow that indicates the turning side is projected in front of the driver.

Another feature to be underlined is the capability of the information displayed to be slowly relocated in the frame in order to be positioned further from the driver's eyes according to the speed of the vehicle. If the speed is higher the information are located further, if the speed decreases the information return to the previous location.

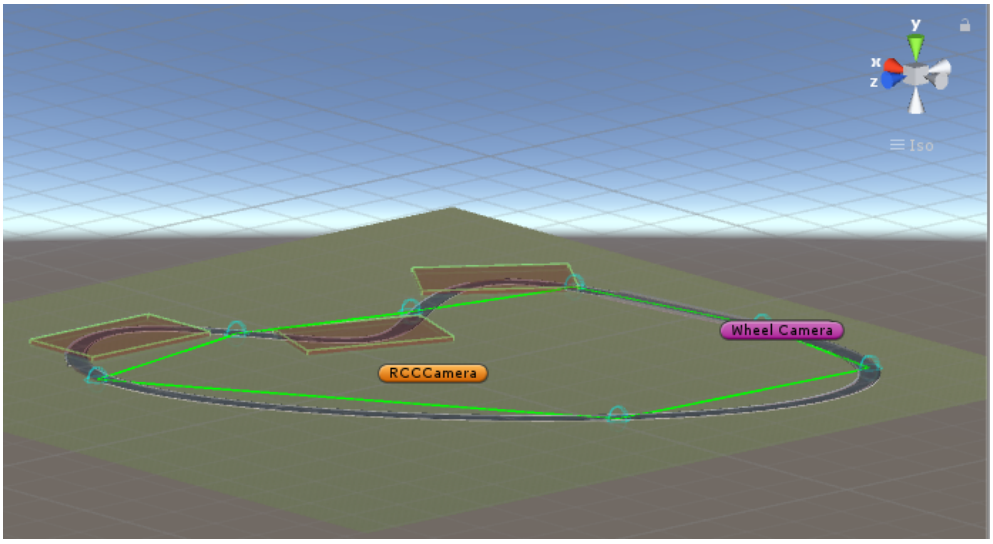


Figure 5.17: Curve zones and waypoints


Figure 5.18: Curve example

A person driving at higher speed looks further into the distance, so the driving information get smaller and put higher in the frame.

In the final interface has been added also the information about the driving style. Indeed there are few possible driving styles according to the car manufacturing company, in this case the options are Corsa and Strada.

To the information about the rotation per minute of the engine are added also the information about the gear number, that is the figure located to the right of it. Under them there is the driving style label. In the simulation, the driving style simply changes at each lap, but there is not an effective difference in the driving performance.

Another feature developed in the prototype is the presence of a "ghost car" to compete with, it could be useful for training. The driver would race against this "ghost car" that drives according to the best time and following the best path calculated.


Figure 5.19: Ghost car example

In the last figure 5.20, there is an alternative representation of the simulation that shows not only the street but also the internal part of the car.

Indeed the real vision may be different according to the driver physical appearance, his height for example, and the position of the seat while he is driving. As it is showed in the experiment in fig 5.8, a person of medium height that is driving has that field of view, that is mostly the street and partially the lower part of the windshield. That is why the final setting for the position of the camera in the AI car controller has been more like the example just quoted.


Figure 5.20: Alternative interface

# Chapter 6

# Conclusions and future work

## 6.1 Conclusions

In this last section it is possible to summarize and highlight the results obtained in this thesis work.

The first phase was about the analysis of the state of art in the field of mixed reality technologies, focusing on the automotive field.
Starting from that, the following possible use case has been analyzed: thanks to this new technology a race car driver can see the information that are usually showed on the display located on the steering wheel, or next to it, directly on the display of his helmet.
A quick analysis of the most famous and common game engine used for mixed reality applications development has been performed, this has led to the final choice of Unity 3D. Unity provides a good documentation and many tutorials that make the learning curve not too steep, its community is active and provides useuful advices in case of problems in the development. In addition this game engine would be the best choice because it has SDKs for the main AR engines on the market. Last but not least, it is free under certain usage conditions.
In conclusion, a prototype application has been developed for the Microsoft Hololens device using as development environment Visual Studio and Unity 3D.

In first instance has been analyzed the mixed reality context, underlining how complex and how many possibilities of deployment for applications of this type there are nowadays. It has been highlighted that there are different performance features to consider when selecting or developing a device for mixed reality applications and that they are slightly different according to which is the final goal. For example, if augmented reality is the field of your application you should pay more attention to the sensors and cameras incorporated into your device, because they can gather information from the real world and through the processing of them create the perfect combination of the real elements with the digital ones. In the case of Microsoft Hololens, for instance, specific cameras perform specific duties, such as depth sensing. These cameras work in tandem with two "environment understanding cameras" on each side of the device. Other two important factors for AR are the projection and reflection, the latter one is implemented

through mirrors that assist the device in the images alignment to the user's eye. In the VR case, the IMU sensors are more important than other sensors to detect the position and movements of the user. The processing power required is higher, because there is the need of processing audio, video, inputs from the user and then they must be combined all together. The lenses and the display are important too, everything must be configured in order to give the best experience to the user and not cause sickness, that means a good frame rate, a low latency and an adequate field of view.

After this analysis, a technological scouting has been performed, focusing especially on some aspects:

- The processing power
- The optical part
- The weight
- The power supply system
- The compatible SDKs
- The availability on the market
- The price

Taking in consideration these aspects and considering the guidelines for HMD design, Microsoft Hololens resulted as the best device for AR application deployment.
Microsoft Hololens indeed represents the state of art for AR technologies. It was chosen also because it provides not only the integration with one of the most used game engine all over the world, that is Unity, but also a Hololens Emulator that exploits the virtualization power of Hyper-V to simulate the real device.

The application development has been divided in three phases.
The first phase was useful to become familiar with the development environment, the programming language and the Hololens tools. It was very simple and with a minimal interface.
The second phase took advantage of an asset available in the Asset Store of Unity, it makes easier the development of applications that needs an advanced implementation of the mechanics and physics of a car. During this phase the focus on the interface increased, the information showed were the car speed, the rpm of the engine and the gear number.
In the last phase the information elements were rendered with a lower opacity, in order to not occlude completely the real world and new features have been

added such as the "ghost car" for the driver to compete with and the curve zones signaled by an arrow projected directly on the road.

It can be considered that the fundamental goals of the thesis work have been reached.

The technological scouting led to an adequate hardware solution and the architectural proposal is simple and feasible.

Unity 3D has proved to be a good and easy-to-use development environment, especially combined with Visual Studio that provides a good interface and a simple way to change the scripts attached to the objects in the Unity editor.

The final application gives a general idea of the mixed reality technology's potential in the real world and not only in the game market.

## 6.2 Future work

Surely it is possible to improve the application in its interface and functionalities.

It could be useful to create an icons system to be showed on the top of the display. To each parameter monitored by the sensors on the car (such as the brakes' pressure, the wheels' pressure, the fuel load, the engine oil and water pressure or the temperature and pressure of the water for the cooling system) may correspond an icon that can signal to the driver a problem related to it, for example by blinking for a certain interval of time.

It could be useful also to have a system to enable and disable the information visualization on the display.

An option could be using a voice command but it could not be recognized in case of louder noises from the car.

Another option is to use the Hololens Clicker.

The HoloLens Clicker is the first peripheral device built specifically for HoloLens and is included with the HoloLens Development Edition.

The HoloLens Clicker allows a user to click and scroll with minimal hand motion as a replacement for the air-tap gesture. It could be incorporated into the steering wheel and easily used by the driver.

At last, a testing phase should be performed in order to evaluate whether the system has complied with all the requirements.

# Bibliography


[1] Unity User Manual https://docs.unity3d.com/Manual/

[2] Introduction to mixed reality technology http://inbound.business.wayne.edu/blog/an-introduction-to-mixed-reality-technology

[3] Windows Mixed Reality - Learn to build mixed reality experiences for HoloLens and immersive headsets https://developer.microsoft.com/en-us/windows/mixed-reality

[4] Web resource for the field of reality technologies http://www.realitytechnologies.com

[5] Eye-Tracking of F1 driver https://www.youtube.com/watch?v=zjkUUMZnTnU&index=12&list=WL

[6] What is Mixed Reality https://devdiner.com/augmented-reality/what-is-windows-mixed-reality

[7] Driving in Virtual Reality, B. Blissing, Linkoping Studies in Science and Technology, 2016

[8] How augmented reality can help safety, S. Sybenga, Study Tour Pixel 2010 - University of Twente, 2010

[9] A taxonomy of mixed reality visual displays, IEICE Transactions on Information Systems, Vol E77-D, No.12 December 1994

[10] Guidelines for HMD design, James E. Melzer, Frederick T. Brozoski, Tomasz R. Letowski, Thomas H. Harding, Clarence E. Rash, 2009

[11] Augmented SDKs for Unity, link https://www.linkedin.com/pulse/dozens-more-augmented-reality-sdks-than-you-think-here-offermann

[12] 2017 Automotive Industry Trends, link http://www.strategyand.pwc.com/reports/2017-automotive-industry-trends

# Ringraziamenti

Vorrei cogliere questa occasione per esprimere un sincero ringraziamento al relatore di questa tesi, il Prof. Morisio, che mi ha sempre dimostrato comprensione e disponibilità.

Ringrazio le compagne di questo viaggio, Roberta e Lucia, con le quali le ore di studio sono sicuramente passate più velocemente, ma non meno istericamente.

Ringrazio le amiche lontane, quelle di sempre, sulle quali so di poter contare oggi così com'era ieri e come sarà sicuramente domani.
Grazie Alessandra, Giulia, Maria Teresa e Rossella.

Ringrazio Sabrina, la mia amica più cara, al mio fianco sin dai banchi di scuola, che forse neanche noi dieci anni fa ci saremmo immaginate qui, lontane km da casa, e più amiche di prima.

Ringrazio Fabio, compagno di vita, supporto costante e, ahilui, valvola di sfogo per le mie ansie.

Un infinito grazie va alla mia famiglia che anche a distanza sa sempre essere vicina e di supporto, spero di deludervi raramente.
Grazie a mio fratello, che è sempre stato per me modello e punto di riferimento.
Grazie a mia madre e mio padre, perché ciò che sono oggi e i miei risultati li devo soprattutto a loro.