

UNIVERSIDAD AUTONOMA DE MADRID
ESCUELA POLITECNICA SUPERIOR SUPERIOR



DEVELOPMENT OF MIXED REALITY APPLICATIONS USING THE MAGIC LEAP ONE DEVICE

Mercedes Soto Ramos
Director: Juan Carlos San Miguel Avedillo

-MASTER THESIS-

Electronics Technology and Communications Department
Escuela Politecnica Superior
Universidad Autonoma de Madrid
September 2019



université
de **BORDEAUX**



PÁZMÁNY PÉTER CATHOLIC UNIVERSITY
Faculty of Information Technology and Bionics

UA
UNIVERSIDAD AUTONOMA
DE MADRID

DEVELOPMENT OF MIXED REALITY APPLICATIONS USING THE MAGIC LEAP ONE DEVICE

Mercedes Soto Ramos

Director: Juan Carlos San Miguel Avedillo



Video Processing and Understanding Lab
Informatics Engineering Department
Escuela Politecnica Superior
Universidad Autonoma de Madrid
September 2019

This work has been partially supported by the company Conexiono thanks to
Fundación of the Universidad Autónoma de Madrid (FUAM)



Resumen

El objetivo de este Trabajo fin de Máster es el desarrollo de una aplicación de realidad mixta que simule un entorno de compra virtual.

La aplicación se ha realizado para el dispositivo Magic Leap One, aprovechando alguna de sus funcionalidades, como el reconocimiento de gestos o el seguimiento de la posición de la cabeza. La aplicación presenta un entorno de compra virtual con un lineal en el cual se exponen distintos productos con los que el usuario puede interactuar y, si lo desea, comprar.

Una vez terminado el demostrador de la aplicación se han realizado múltiples pruebas para evaluar su rendimiento y comprobar que cumple con los requisitos deseados. Finalmente, se ha solicitado a un grupo de personas la realización de diversas acciones dentro de la aplicación con el objetivo de obtener una valoración subjetiva por parte de posibles futuros usuarios.

Palabras Clave

Realidad mixta, Magic Leap One, Unity, modelos 3D.

Abstract

The main objective of this Master Thesis is the development of a mixed reality application simulating a shopping virtual environment.

The application has been developed for the Magic Leap One device, taking advantage of some of its functionalities, such as gesture recognition or head pose tracking. The application is a virtual shopping environment with a shelf containing different purchasing items which the user can interact with and, if desired, buy them. Once the demo application has been finished different tests have been carried out to evaluate its performance and verify the fulfilment of the desired requirements. Finally, a group of people have been asked to perform several actions within the application in order to obtain a subjective assessment from possible future users.

Keywords

Mixed reality, Magic Leap One, Unity, 3D models.

Acknowledgements

En primer lugar, agradecer al equipo de Conexiono y al VPULab la oportunidad de realizar este proyecto y a mi tutor Juan Carlos por su apoyo.

A mis compañeros de zulo, a Paula y a Sara, por ser mi familia y por todas las experiencias de estos dos últimos años, sin vosotros nada habría sido igual. Y a todos los amigos que, aunque no me hayan acompañado en esta pequeña aventura, han estado ahí.

Pablo, qué decirte... gracias, gracias y mil gracias.

Un especial agradecimiento a Beltrán, por dedicar un poco de su tiempo a leer, al menos, parte de las páginas que vienen a continuación.

Y por último, gracias a mi familia por su constante apoyo, en especial a ti mamá. Muchísimas gracias por todo, de verdad.

Mercedes

Contents

Resumen	v
Abstract	vii
Acknowledgements	ix
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	2
1.3 Thesis Structure	2
2 State of the Art	5
2.1 Introduction	5
2.2 Reality technologies	5
2.2.1 Virtual reality (VR)	5
2.2.2 Augmented Reality (AR)	6
2.2.3 Mixed Reality (MR)	6
2.3 Mixed reality applications	7
2.4 Mixed Reality Platforms	8
2.4.1 Magic Leap One	8
2.4.2 Hololens	11
2.4.3 Comparison	13
2.5 Game Engines	14
3 Application Design	17
3.1 Requirements	17
3.1.1 Functional needs	17
3.1.2 Non-functional needs	18
3.2 Application Overview	18
3.2.1 Start screen	19
3.2.2 Positioning of objects	19
3.2.3 Shopping items	20
3.2.4 Checking items	21
3.2.5 Interaction with objects	21

4	Development	23
4.1	Development environment	23
4.1.1	Unity	23
4.1.2	Magic Leap Package Manager	24
4.1.3	Magic Leap Remote	24
4.2	Demonstrations of functionalities	25
4.2.1	Scene modelling	26
4.2.2	Control interaction	27
4.2.3	Hand tracking	28
4.2.4	Detect and tracking images	30
4.2.5	Other functionalities	31
4.3	Database	32
4.3.1	Database source	33
4.4	Demo application	33
4.4.1	Positioning stage	34
4.4.2	Purchasing stage	36
4.4.3	Checking stage	38
4.4.4	Software architecture	41
5	Experiments	43
5.1	Conditions for optimal working	43
5.2	Demonstrations of functionalities	43
5.2.1	Scene modelling	43
5.2.2	Control interaction	45
5.2.3	Hand tracking	45
5.3	3D modeling	46
5.4	Retail application	47
5.4.1	Software testing	47
5.4.2	User experience	51
6	Conclusions and Future Work	55
6.1	Conclusion	55
6.2	Future Work	55
	Bibliography	57

List of Figures

2.1	Virtual reality, augmented reality and mixed reality	6
2.2	North America mixed reality market revenue by application, 2015 - 2024	7
2.3	MLO hardware. From left to right: Lightwear, Lightpack and Control	9
2.4	MLO Control: buttons and raycast	10
2.5	Magic Leap software overview	10
2.6	Hololens 2	12
2.7	Field of view comparison between human vision, VR devices, MLO, Hololens 2 and Hololens 1	14
2.8	Game engines structure	15
3.1	Proposed design for the application	19
3.2	Start scene	20
3.3	Positioning scene	20
3.4	Purchasing scene	21
3.5	Checking scene	22
4.1	Unity Window	24
4.2	Magic Leap Remote	25
4.3	Scene modelling application in Unity	26
4.4	Initialize control	28
4.5	Hand tracking	29
4.6	Script <i>MImageTrackerBehavior</i> options	31
4.7	Database elements	32
4.8	State diagram of the retail application	34
4.9	State diagram of the position stage with manual mode	35
4.10	State diagram of the position stage with manual mode	37
4.11	State diagram of the checking stage	40
4.12	Hierarchy application structure	41
5.1	Scene modelling example	44
5.2	Hand tracking example	45
5.3	3D modeling	46
5.4	Visualization scheme of the software testing images	47
5.5	Visual examples of the positioning stage	48
5.6	Visual examples of the purchasing stage	50

5.7	Visual examples of the checking stage	51
5.8	Subjective answers about specific functions of the application	53
5.9	Subjective answers about the global application	54

List of Tables

2.1	Magic leap One Creator edition and Hololens 2 hardware and software comparison.	13
4.1	Button input behavior across Magic Leap platform.	27
4.2	Gestures and actions linked to the positioning stage	36
4.3	Gestures and actions linked to the purchasing stage	38
4.4	Gestures and actions linked to the checking stage	40
5.1	Subjective questions about the difficulty and clarity of the different modules of the application, as well as specific preferences of the tester.	51
5.2	Subjective usability survey.	52

Chapter 1

Introduction

1.1 Motivation

Virtual reality (VR), augmented reality (AR) and mixed reality (MR) technologies have highly increased, providing us with new experiences and new ways of seeing the world. These technologies are expected to experience exponential growth in the coming years and very interesting applications and devices have emerged to satisfy the needs of the industry.

The most recent of those three technologies is mixed reality, which gives the possibility of creating new spaces where people interact with virtual and real objects. Potential MR applications can benefit a wide variety of sectors such as education, military training, healthcare, the entertainment industry and the retail industry, which is expected to be the most relevant sector in mixed reality technologies. In this context, there are two main devices in the market that implement the concept of mixed reality through the use of glasses. Microsoft's HoloLens glasses, that already have version 1 and 2 released in 2016 and 2019 respectively, and the Magic Leap One (MLO) of the Magic Leap company, which is currently only available in its creator edition from August 2018.

Due to the need to cover the market relating to the mixed reality, Conexiono, together with the VPULab, has decided to carry out a project whose objective is to develop a retail application for the MLO device.

The motivation of this Master Thesis is to gather information about the MR devices, their functionalities, how they work and which needs they can cover. With this information and the collaboration of other VPULab members, it will be possible to give the users a shopping mixed reality experience.

1.2 Objectives

The aim of this work is to develop a mixed reality application for the Magic Leap One device based on its Software Development Kit (SDK) and Unity 3D. This application is a prototype which aims to demonstrate functionality with a use case oriented to the retail sector. The application simulates a shopping environment where shelves and products are virtual objects with which the user can interact for visualization, purchase or get extended information about them.

It should be borne in mind that this Master Thesis is part of a bigger project in which more people collaborate in the development of several parts of the application. The specific objectives of this Master Thesis are:

- Study mixed reality state of the art. Investigate MR technology, its devices and which software is needed to develop an application for them.
- Investigate the potential of the Magic Leap One glasses. Through tutorials and creating demo applications, we considered the functionalities that can be used with this device. Specifically, this work is focused on the evaluation of the remote, gestures, mesh and image tracking modules. Other members of the project have investigated other modules like the raycasting or the video player.
- Generate a database with different objects contained in the virtual scenario.
- Design and develop the retail application with the help of other members of the project. How we want to interact with the real and virtual world, which items would it have and where are they positioned, which kind of extra information is shown to the user, etc.
- Analyze the user experience, evaluating the interaction, functionalities, naturalness, etc. on the final application.

1.3 Thesis Structure

This Master Thesis is divided as follows:

- Chapter 1: this chapter contains a brief introduction of the project, the motivation, the objectives and the structure of this report.
- Chapter 2: this chapter presents the virtual, augmented and mixed reality technologies. Reviewing MR applications and providing an overview of its devices. Moreover, it includes a brief explanation about 3D engines.

- Chapter 3: in this chapter the requirements to be met by the project and the design desired for the final application are presented.
- Chapter 4: this chapter includes the development procedure followed to obtain the final retail application. Starting from the selection of the development environment, the research of the MLO device functionalities, obtaining of the database and, finally, developing the application.
- Chapter 5: this chapter contains the experiments and the results obtained from the tests carried out with MLO modules and from the final application.
- Chapter 6: in this final chapter, the conclusions drawn from this work are presented and improvements to be made as future work are discussed.

Chapter 2

State of the Art

2.1 Introduction

In the current market we can find three different types of reality technologies: virtual reality, that creates a digital environment shutting down the real world, augmented reality, which introduces digital content on top of the real world, and finally, mixed reality in which the added digital content interacts with the real world and with the user [1]. Nevertheless, there is a great controversy between the differences between augmented reality and mixed reality, while some experts define both terms as a same technology sold separately to increase profits, others see a clear difference between the two technologies [2].

This chapter is an overview of these technologies, the current mixed reality devices and the game engines used to design 3D applications.

2.2 Reality technologies

Magic Leap device, based on mixed reality, has been used for developing this Master Thesis, therefore, this section introduces MR technology as well as virtual reality and augmented reality technologies. In Figure 2.1 is shown a brief comparison between the three technologies.

2.2.1 Virtual reality (VR)

Virtual reality replaces the world you are standing with a virtual one, everything you see and hear is replaced with a 3D environment computer-generated that seems real for the user experience. This technology needs a device, generally a head-mounted display, that allows the user to see stereoscopic 3D scenes. Virtual reality offers the

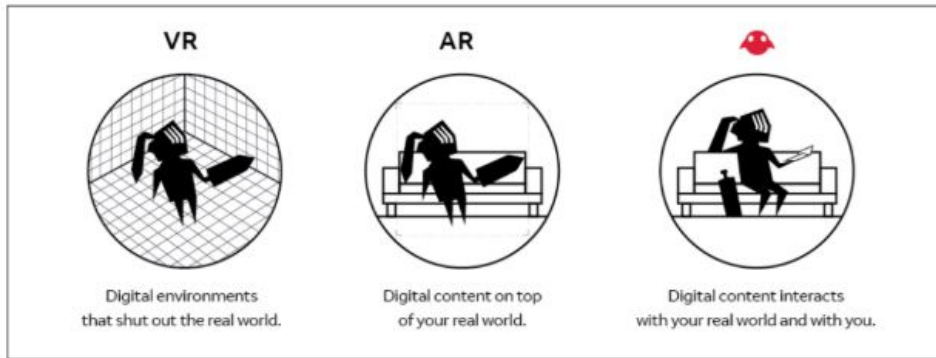


Figure 2.1: From left to right: virtual reality, augmented reality and mixed reality from Magic Leap's creators point of view [1]

user a fully immersive experience where he can interact with the virtual environment. Some popular virtual reality devices are Facebook Oculus, Samsung Gear VR or Google Cardboard.

2.2.2 Augmented Reality (AR)

Augmented reality is a live direct or indirect view of a physical-real world environment whose elements are augmented by computer-generated or extracted real-world sensory. AR can add any kind of digital objects to the real world, which means that the user look at the real world and it is augmented with additional information or graphics in his view. The key concept of AR is that objects are indexed to a fixed location, occupying a relevant place in the user view, and in many cases, interact with the user. An application that clearly shows this technology is *Pokemon Go* [3].

Augmented reality is implemented in many phone apps, and for higher performance, it is possible can use dedicated augmented reality glasses like Vuzix or Google Glass.

2.2.3 Mixed Reality (MR)

Mixed reality produces new environments and visualizations where physical and digital objects coexist and interact in real time, it is a mix between VR and AR. In these environments, the user can interact with digital objects in the same way as with real ones, whether through gestures, a control command or any other input method that the application and the main device allows. The MR devices are similar to the VR headsets but it is possible to see through the display. The most famous MR devices are the Microsoft Hololens and the Magic Leap One.

In short, Mixed Reality shows computer graphics in the field of view of the user who can see a virtual world while he can continue visualizing the real world, interacting

with both of them.

Moreover, Magic Leap gives added value to the definition of mixed reality, giving it the new name of Spatial Computing. Spatial Computing in addition to fulfilling the features of mixed reality creates a reconstruction of the real world and gives rise to a great spatial sensing of all sounds. It also have information about the position of the head, eyes, hands and gestures of the user.

2.3 Mixed reality applications

Mixed reality technology is growing and increasingly used by professionals and ordinary users. As with all the previous technologies the first target for mixed reality was gamers, consumers very interested in realistic and interacting 3D environments, but this technology is used also on many other purposes; as shown in Figure 2.2 the mixed reality market is expected to grow mainly in the automotive and aerospace industries. The main mixed application types are:

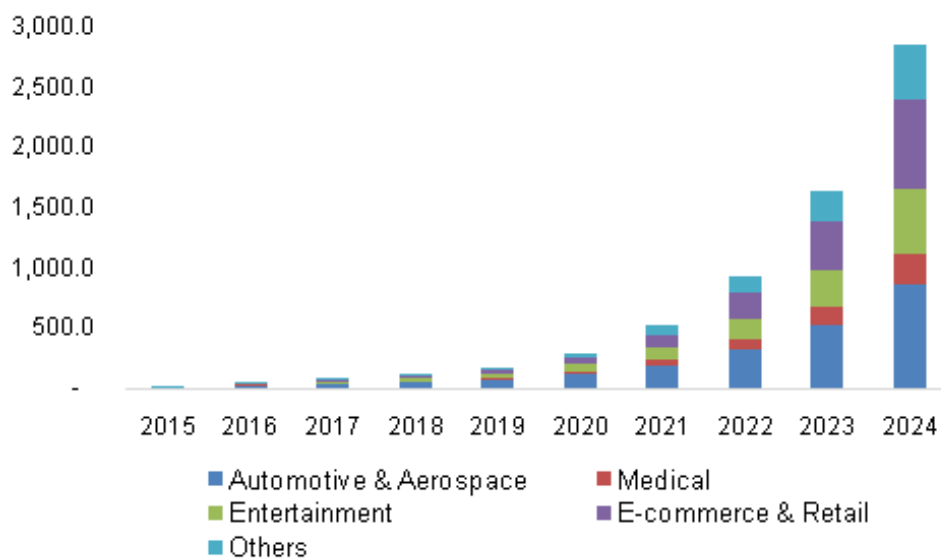


Figure 2.2: North America mixed reality market revenue by application, 2015 - 2024 (USD Million) [4]

- **Automotive and Aerospace.** Use to help workers to build, repair or design automotive and aerospace equipment.
- **Entertainment.** Gaming is the main entertainment industry of mixed reality technology. As entertainment we can also consider the creation of augmented images in film making or the apps based on the fashion industry.

- **Medical.** MR has a great potential in health and medicine. This type of applications are used to display augmented 3D models of human bones and organs for medical diagnosis and to help in the study of human anatomy. Moreover, the projection of patient information or organs is very useful to provide assistance during surgeries.
- **E-Commerce and retail.** These industries use mixed reality technologies to enhance appeal during purchasing. Specially focused on advertising and promotional purposes.
- **Communication.** Chat applications like Skype. With these apps the user can see the hologram or the video of the person he is chatting within the room.
- **Visualization.** Allows to visualize a screen with information, either a text and images with a recipe, a movie or a television channel. The user can choose the position and size of the screen.
- **Others.** Mixed reality technology also involves military and defense industry, architectural, industrial, education or logistic applications, among others.

2.4 Mixed Reality Platforms

2.4.1 Magic Leap One

Magic Leap One is the first device created by Magic Leap [5]. The MLO device is currently only available to application developers (creator edition) from August 2018 and is expected to be commercially available in early 2020 (public edition). MLO combines technology of diverse nature like scene modeling, gesture recognition, object tracking, virtual object design, high processing services... that requires multi-disciplinary professional profiles. This device stands out for its good specifications, having a high processing capacity, a wide field of view, realistic design of virtual objects, a development environment based on open source and a lower cost than other devices with similar characteristics.

2.4.1.1 Hardware

Magic Leap One is a three-part device. The head-mounted display, called Lightwear; a small wearable computer, the Lightpack; and a wireless controller [6]. These elements can be seen in Figure 2.3.



Figure 2.3: MLO hardware. From left to right: Lightwear, Lightpack and Control [6]

- **Lightwear.** The head-mounted glasses allows the user to see at the same time the real and the virtual world. They work with a photonic chip, a processor that can redirect the light to the eye of the user in the right way producing the feeling of projecting digital objects at different distances. Comparing to classical lenses this technology allows a reduction on the size of the device. The field of view of the Lightwear is the $40^{\circ} \times 30^{\circ}$.

Moreover, the Lightwear have multiple cameras and infrared (IR) sensors to detect the environment, gestures or position of the controller and 6 audio channels for a greater spatial sensation of all sounds.

- **Lightpack.** It is a small computer connected to the Lightwear. This computer works with a powerful CPU and GPU, allowing the correct execution of any application and generating quality graphics thanks to having installed the graphic APIs OpenGL and Vulkan. It also has 128 GB of internal memory and 8GB of RAM memory[7]:
- **Control.** One of the input methods of the Magic Leap One device. It is connected wirelessly to the Lightwear so that at all times it is known its position and rotation relative to the glasses as well as if you are pressing any button. The control (Figure 2.4) has two binary (pressed/not pressed) buttons the home and the bumper, on the other hand, it has the trigger that can be press with different intensity. Moreover, it has a touchpad that is able to recognize the gestures and the pressure that is made on it.

The control is tracked on three axes of translation and three axes of rotation by the glasses thanks to the 6 degrees of freedom (6DoF), being able of freely move and rotate the digital content and control raycasting.

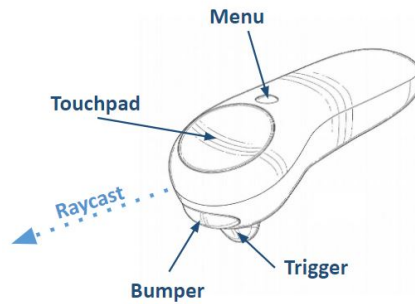


Figure 2.4: MLO Control: buttons and raycast

2.4.1.2 Software

Magic Leap One is based on a software developed specifically for this device: Lumin OS [8] a custom operating system derived from open source components such as the Android and Linux Open Source Project (AOSP). This Operating system is specifically designed to ride out the high-performance requirements of spatial computing, that means, to work fluently in an application in which is needed to analyze and combine real and virtual worlds at the same time the user interacts with them. Magic Leap software has many layers (Figure 2.5):

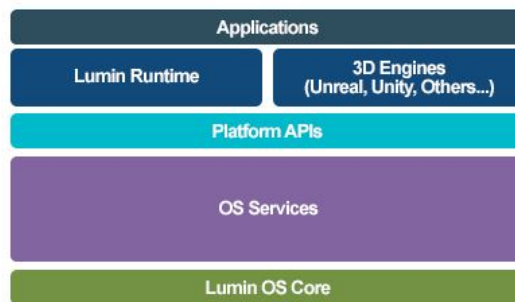


Figure 2.5: Magic Leap software overview [8]

- **Lumin OS Core.** The low-level Lumin software. Is a Linux-based kernel with custom drivers compatible with Magic Leap One
- **OS Services.** Lumin OS has a wide wide variety of OS services. These services cover traditional functions of a modern operating system like bluetooth, wifi connectivity, multimedia, power management, applications lifecycle management, input management, and much more. It also includes native services for logging capabilities and debugging. Lumin OS also covers the specific functionalities needed by MLO, having a complete set of Perception services to manage

the environment that surrounds the device and detect the user position and movements. Moreover it contains high performance and dedicated graphics and audio systems.

- **Platform APIs.** Is a combination of specialized APIs and standard POSIX APIs granting to the apps access to the functionalities of spatial computing. Using a blinder we will allow to the APIs interfacing with the Lumin OS services. Lumin OS works as a gatekeeper for sensitive features. The user will be asked if it is okay to let a specific app allow access to a feature like the camera or the microphone, being able to give or remove access to such functionality.
- **Lumin Runtime.** Provides a set of APIs and a toolkit to offers the user high fidelity graphics and audio. The Lumin Runtime framework allows the user to experience a coherent experience executing at the same time different apps. This is achieved by operating in a client-server model.
- **3D Engines.** 3D engines have already integrated Lumin SDK and also can be used the Lumin SDK C API to integrate the selected 3D engine. Unlike Lumin Runtime, working with 3D engines lead to immersive applications that generate the entire wearer's experience, that is, a single application running at the same time. Most common 3D engines are Unity and Unreal Engine 4.
- **Applications.** The applications can be *Immersive* or *Landscape* depending on the development system 3D engine or Lumin Runtime correspondingly and the purpose of the developer. In an immersive application you interact with the new universe created by your app while in landscape apps can coexist with others at the same time, providing the home view, app launcher settings and notifications given by default; furthermore, landscape applications can become a part of the environment and persist between sessions.

2.4.2 Hololens

Hololens is Microsoft mixed reality device and the first mixed reality glasses launched on the market, in 2016. Early 2019 its second version was released, the Hololens 2, a product that has a clearer market to focus on: Microsoft has decided to sell its new mixed reality glasses only to companies and further ahead, on May 2019, to start selling the developer version. This device is characterized for being more compact than its previous version and has an incredible increase the field of view (FoV) from 30°x17.5° to 43°x29°. They also have a great processing power and

software enhancements such as recognition of more gestures, eye tracking or a better stabilisation of the content.

2.4.2.1 Hardware



Figure 2.6: HoloLens 2 [9]

The HoloLens 2, as well as the HoloLens 1, hardware is only composed by the headset (Figure 2.6) which includes the display, sensors a five channel microphone array and spatial speakers [10].

The HoloLens 2 has a powerful computer integrated on the back part of the device, which works with the second-generation custom-built holographic processing unit. The display of the headset is composed by holographic lenses with a resolution of 2k and an holographic density of more than 2.5k radiants, that is, more than 2.5k points per radiant. The lenses optimize the rendering of the graphics according to the 3D eye position. The headset has multiple sensors that allow tracking both the position of the head and the eyes of the user, an accelerometer, gyroscope, magnetometer and also a depth sensor and a 8MP camera to recognize the world-space.

2.4.2.2 Software

The HoloLens are based on Microsoft's software and Windows 10 SDK, which gives the libraries and metadata required to compile all the applications over the HoloLens 2. In order to develop applications, 3D engine Unity should be used, moreover, the multiplatform kit MRKT can be added to improve the development of the applications and to make easier the creation of mixed reality applications.

2.4.3 Comparison

If we look Table 2.1 we can see that the MLO and Hololens 2 devices have quite similar features, both of them have good hardware specifications, maybe the MLO hardware is more powerful but in exchange it has the small computer outside the headset.

Characteristic	MLO Creator edition	Hololens 2
Price	\$2295	\$3500
System on Chip	NVIDIA Parker	Qualcomm Snapdragon 850
Memory	8GB RAM	4GB LPDDR4x system DRAM
Storage	128GB	64GB
Power	Up to 3 hours continuous use	2-3 hours of active use
Connectivity	WiFi 802.11ac/b/g/n Bluetooth 4.2	WiFi 802.11ac/b/g/n Bluetooth 4.1 LE
Audio	Onboard speakers 3.5mm jack	3D Audio speakers 3.5mm jack
Ports	USB-C	Micro USB 2.0
FoV	40°x30°	43°x29°
Control	Yes	No
Gestures	8	2+5
Hand tracking	Yes	Yes
Eye haze	Yes	Yes
Voice input	Yes	Yes
Spatial mapping	Yes	Yes
Spatial sound	Yes	Yes

Table 2.1: Magic leap One Creator edition and Hololens 2 hardware and software comparison.

The field of view of both are very similar, wider for Hololens and higher for MLO, but they are still far from being able to cover the FoV of the virtual reality devices and much more to cover the entire the field of view of the human being as can be seen in Figure 2.7.

About the implemented software in the devices, both of them includes the main core blocks to create applications easily. They are able to recognise the environment and create a mesh around it and they have modules to recognize the voice and track the eyes and the hands. The main differences are that Hololens voice recognizer includes Cortana, having more voice commands than MLO. On the other hand, MLO is able to recognize eight gestures while Hololens 2 recognizes only two gestures and four combinations of the previous two depending on the type of interaction with the

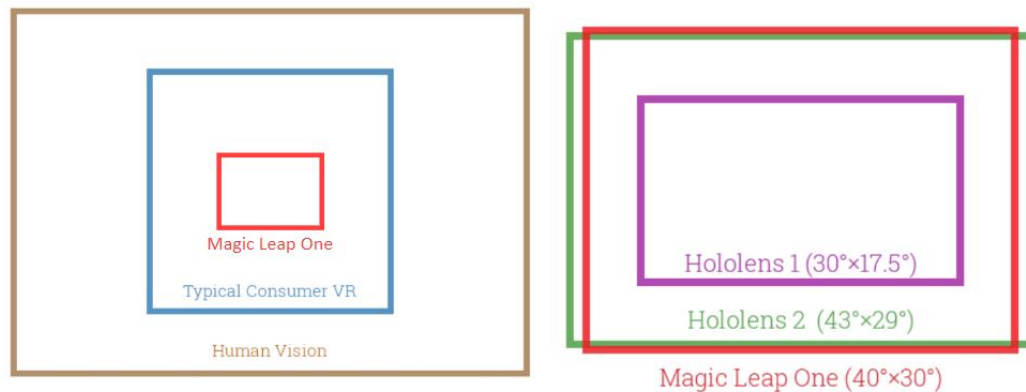


Figure 2.7: Field of view comparison between human vision, VR devices, MLO, Hololens 2 and Hololens 1

application.

2.5 Game Engines

In the last years, video games and mixed reality industry, as well as virtual and augmented reality applications, have grown substantially, leading to the creation of software specially designed for developing 2D and 3D environments. These specific software are the game engines, capable of generating real environments without the need of high performance computers, using a standardized framework where games and virtual simulators can be designed with greater flexibility optimization and ease of use. This is thanks to the modular simulation code, a code written specifically for a particular game but which can be used for similar games. Game engines consist of a set of modules that, without specifying the logic of the game or its environment, give generic information about the physics and dynamics of the game worlds and manage the inputs and outputs of the application [11]. The general structure of the game engines is shown in Figure 2.8.

The game engine is built from a specific operating system, connected to the other modules by the graphics drivers which translate the requests done from the rendering engine. As the drivers are open source they can be easily modified to be compatible with the hardware on which the application is used. The rendering engine is the main part of the structure and can not be modified. This black box has the necessary code to identify and render correctly the 3D model of the game.

Parallel to the graphics drivers and the rendering engine is the networking code that supports a connection code allowing several users to remotely access the same

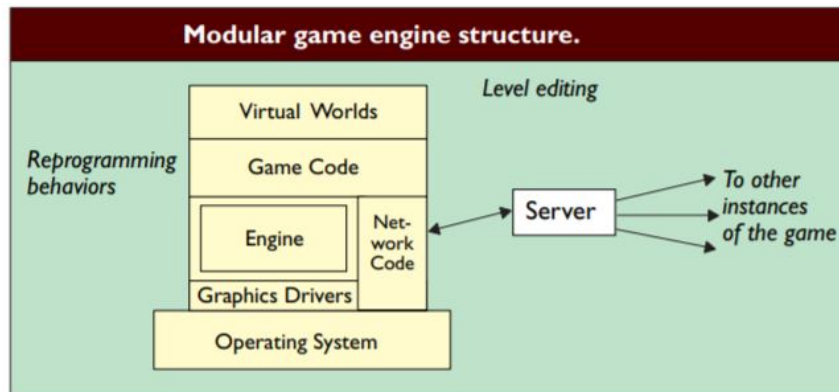


Figure 2.8: Game engines structure [11]

virtual environment. The networking code is connected to the server, a process that works separately, usually on another machine, which maintains information about the 3D environment, constantly updating it for all users who are within the shared virtual environment.

Above the previous levels is the game code, that handles the basic features of the game such as display parameters, animations or networks. To make modifications to this level is possible although a great knowledge of the internal components of the game and the knowledge of the specific language of the game is required.

Finally, at the top level, there are the virtual worlds with which the player interacts. This level allows the simple modification of the game, even the game itself sometimes includes an advanced development environment so that users can change the appearance of the game or the rules of interaction.

Depending on the application you want to develop and the device for which it is the application, we can find different 3D engines such as Amazon Lumberyard, Panda 3D, Unity 3D and Unreal Engine. As we want to work with mixed reality applications, is highly recommended game engine that supports the SDK of the device we are going to work with, so we have Unity, supported by both MLO and Hololens or Unreal Engine, only supported by MLO.

Both, Unity and Unreal are free license and can be written in C#. Moreover, Unity 3D can be also written in JavaScript and incorporates other services such as performance reports, ads or multiplayer.

Chapter 3

Application Design

This chapter outlines the requirements that the application must fulfill as well as the design of its different modules.

3.1 Requirements

The developed application have to simulate a shopping environment. This section describes the functional and non-functional needs that must cover the application. In the experiments chapter, in section 5.4.1, a series of tests will be made to evaluate if all the requirements are completed.

3.1.1 Functional needs

The application should have the following characteristics.

- FN1: The user can interact with the objects through gestures to select, take, move, rotate, buy and discard.
- FN2: There will be a shopping cart represented by a virtual object that can be positioned on surfaces that have been previously included during the stage of scene modelling. This shopping cart will be used to select the objects the user wants to buy.
- FN3: The products to select from the shelf will be high quality virtual representations from multiple views of the actual products.
- FN4: The objects will be presented in a shelf that will be designed virtually. The positioning of the shelf can be changed according to user preferences.

- FN5: The objects can be positioned in different places of the environment where the user is.
- FN6: The positioning options will depend on the type of object being considered.
- FN7: In addition, selecting an item will display a floating screen with information about product (i.e. price, characteristics...).

3.1.2 Non-functional needs

In order for the developed application to be considered to work correctly the following requirements must be covered.

- NFN1: **Performance.** The application must be designed to work in real time. With each user interaction, affected virtual objects must perform the action assigned to them in the shortest possible time.
- NFN2: **Reliability.** The application must run for a long time without failures.
- NFN3: **Maintainability.** The software must be properly commented and distributed in modules. In addition the software will be documented and tutorials will be included to use each one of the tools in an appropriate way.
- NFN4: **Scalability.** The software must be clearly divided in different modules in a way that it would be easy to implement new ones and add new windows to the application.
- NFN5: **Usability.** The different interactions between the user and virtual objects must be intuitive and easy to carry out. It must be clearly shown (by sound or image) with which element is the user interacting or which action is he doing.

3.2 Application Overview

The objective of this Conexiono project is to develop an application that simulates a shopping environment. This application has five main states as shown in Figure 3.1.

- First the application must be selected to open and a welcome message appears.
- A mapping of the environment that surrounds the user will be made. With this information the application will be able to find an optimal place to establish the shopping environment.

- Once the virtual shopping environment has been established in the real world, the user can select, visualize and buy or discard the different items displayed on the shelf.
- The user will also be able to view the items added to the shopping cart and edit them if desired.
- The user may navigate between the screens of shopping and checking until he wants to finish the purchase.

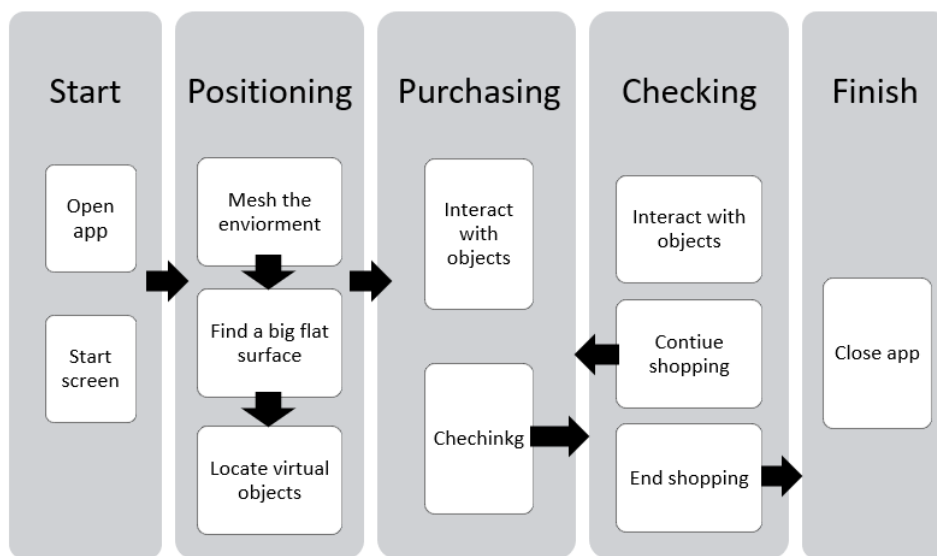


Figure 3.1: Proposed design for the application

3.2.1 Start screen

Each time the application is opened the first thing that will be shown is a welcome message (Figure 3.2). This screen serve in the first instance to inform the user about the nature of the application: an online shop. In addition, the start screen can show logos of the store or other commercial advertising.

3.2.2 Positioning of objects

Once the application has started, the positioning objects scene will be shown (Figure 3.3), as in other scenes, the user will be able to consult the necessary instructions to interact properly with the elements of the scene. It is needed to recognize the room in which the user is located, mapping of the scene is done by building a mesh around

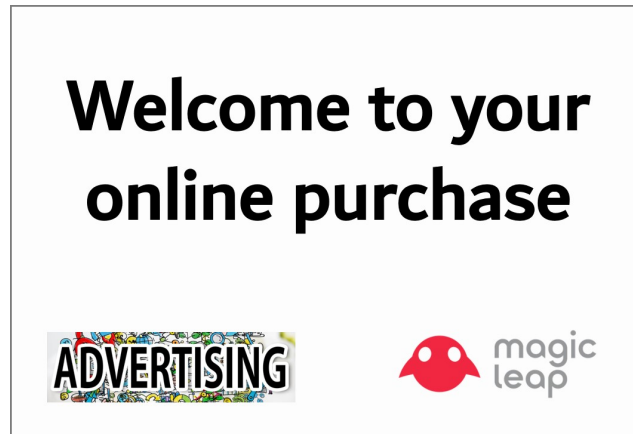
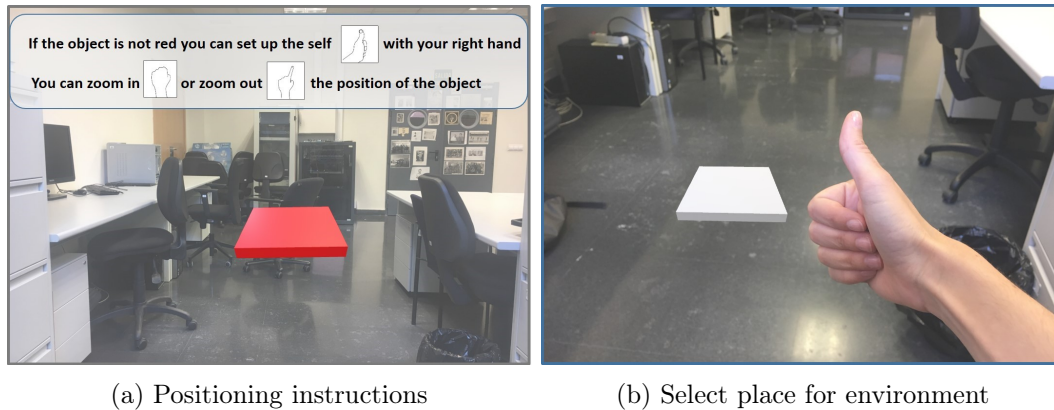


Figure 3.2: Start scene

all the objects that are near to the user. This information is used to know where the virtual shopping environment can be located. For this purpose a virtual object is used in this scene, this object changes its color every time the shelf and the shopping cart have enough space to be positioned without colliding with real-world objects, each time this condition is met, if the user agrees with the placement he have to do a gesture to set the shopping environment.



(a) Positioning instructions

(b) Select place for environment

Figure 3.3: Positioning scene

3.2.3 Shopping items

The shopping scene shows the shelf with the items that can be purchased, a shopping cart to introduce the objects that the user wants to buy and a floating button to switch to the checking scene (Figure 3.4d). In this scene you can interact with the items located in the shelf (Figures 3.4b and 3.4c), picking them up, rotating them

and looking at additional information about them or similar products. Each gesture will be associated with a specific action depending on the version of the application.

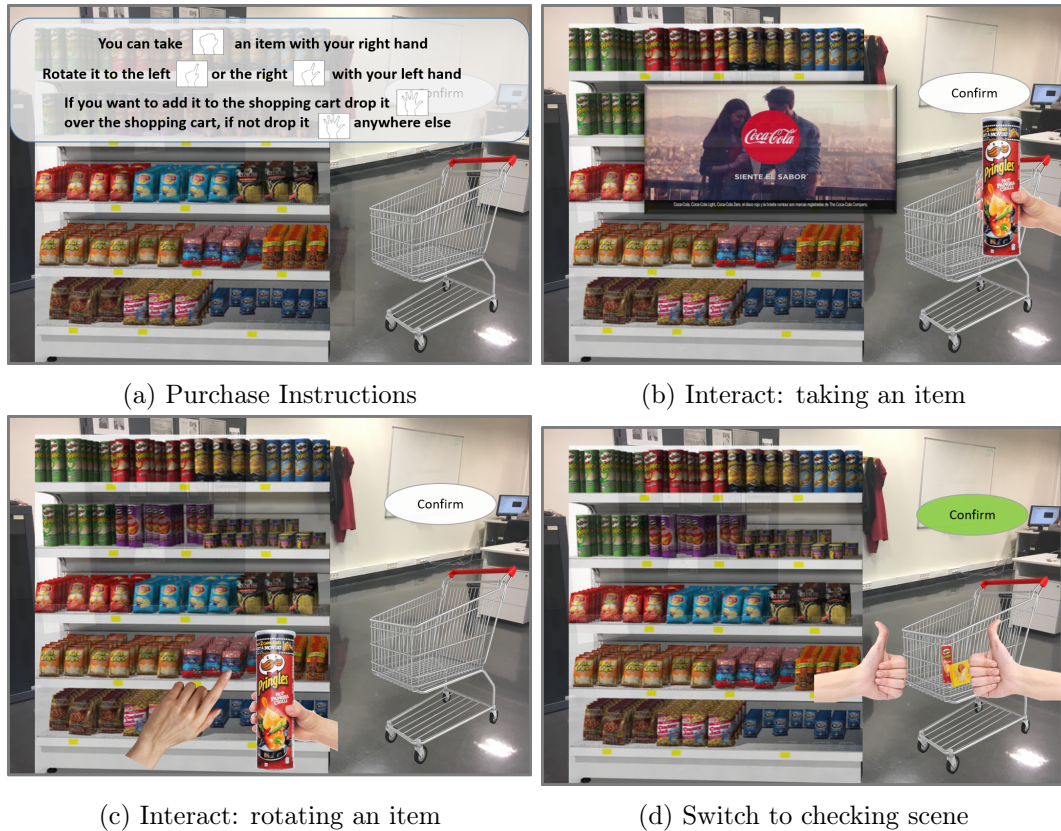


Figure 3.4: Purchasing scene

3.2.4 Checking items

In the checking items stage we have once again the shelf, but only with the purchased objects on it. On the right will be shown a bin in order to discard the objects, it also contains two buttons: the first one to continue shopping and the second one to complete the purchase and close the app (Figure 3.5). This scene works in a very similar way to the purchase, being able to interact with the different items that are observed, discarding them or leaving them back on the shelf for purchase.

3.2.5 Interaction with objects

The user will be able to interact with the items on the shelf through gestures. Taking and turning them to observe them from different points of view, and moving the items to the shopping cart or the bin. Moreover, while the user is interacting with an item

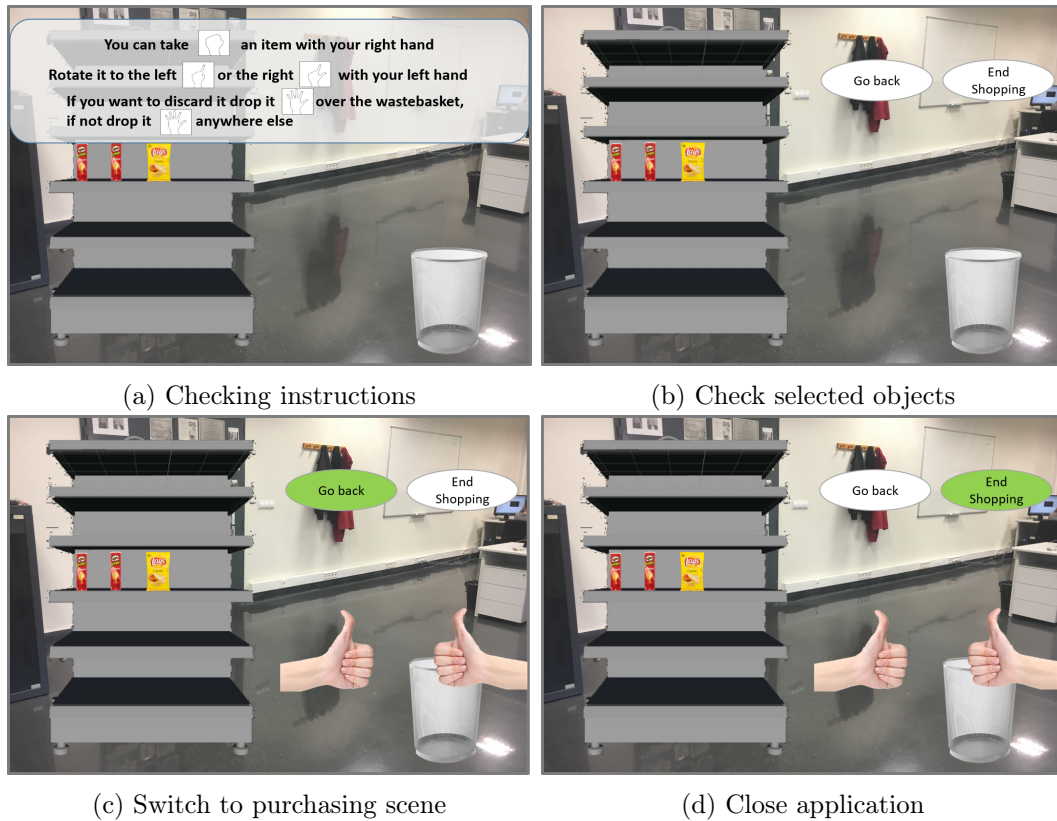


Figure 3.5: Checking scene

it would appear extra information about it. Furthermore, the user can also switch between scenes by selecting the buttons on the screen with the gaze and a gesture.

The different gestures will be adapted to be simple and natural, so that the user interacts in the most intuitive possible way with the different objects.

Chapter 4

Development

The main objective of the Conexiono and VPULab project is to develop an application which fulfills all the requirements exposed in Chapter 3. For this purpose, together with the rest of the members of the project, three main steps have been followed: First, small demos have been created and tested. in order to explore the possibilities and restraints of the device. Then, the 3D objects required for the application will be searched, creating a database. Finally, the application will be developed by integrating all the needed components.

4.1 Development environment

The proposed application has been developed for the Magic Leap One device requiring a development platform. This environment must integrate the Lumin SDK to easily access the APIs that incorporate the glasses, having access to its frameworks and libraries such as hand tracking.

The main 3D engines that incorporates Lumin SDK are: Unity, Unreal and Lumin Runtime. For this project we have selected Unity. Moreover, it is needed the Magic Leap Package Manager to integrate the 3D engine in the SDK, this software also is used to open the simulator of the Magic Leap device, with ML Remote it is possible to test the application on the computer or on the device, but without charging the application on the device.

4.1.1 Unity

Unity is a 3D engine that supports C# and java programming languages, as the Lumin SDK is configured to program only with C# we will use this language. In Figure 4.1 Unity is composed of several windows. The Hierarchy window, where the

different objects that we want on the scene are incorporated and can be observed in the right window (Scene). In the Scene window you can change to Game view or purchase assets in the Asset Store. In the Inspector window you can see, modify and add characteristics to the selected object. Finally you have the Project and Console windows, in the first one you can see all the objects and scripts that you can incorporate to your project while in the second one is shown the errors and warnings shown when the application is running.

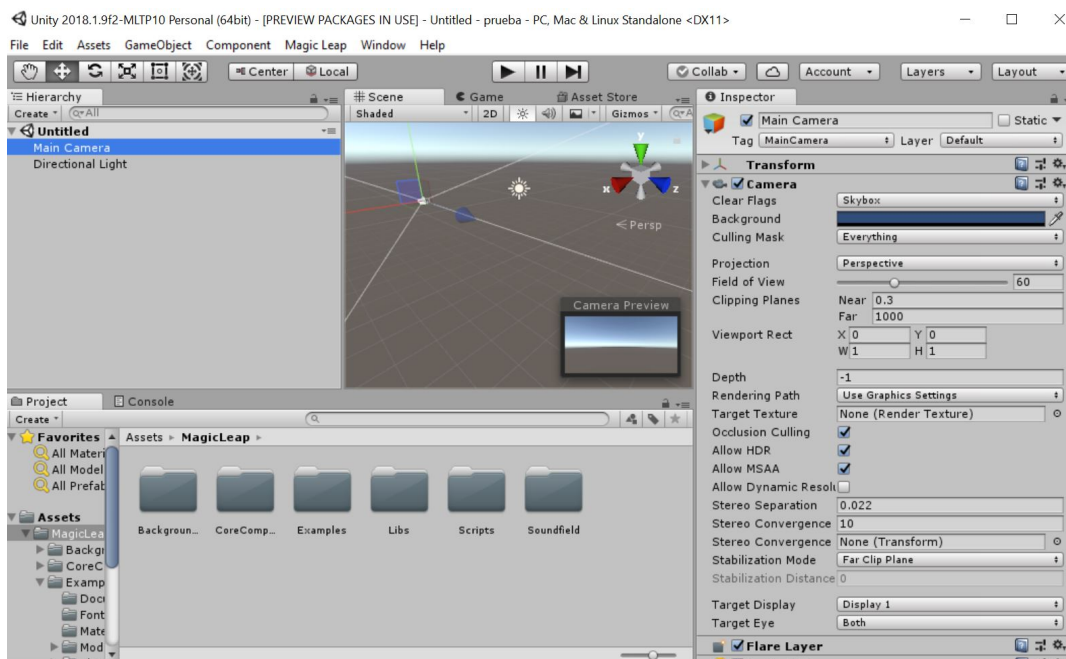


Figure 4.1: Unity Window

4.1.2 Magic Leap Package Manager

This software is used to update the Lumin SDK version and the corresponding Magic Leap package for the used Game Engine. Also you can download documentation about the SDK or other extensions if it is needed.

4.1.3 Magic Leap Remote

Lumin SDK features Magic Leap Remote, which allows remote execution of applications run from Unity. This software is very useful for testing applications, since it allows to see how they works in the computer or in the glasses without building the application. This method is much faster than building the application in the device and also support almost all the functionalities given by the device: Headpose, eye

gaze, gestures, hand tracking, meshing, raycasting, plane extraction, image tracking, control and audio. The only functionalities not supported by the simulator are the LED's and the persistence between frames. If the developer is executing the application on the computer it is possible to change some functionalities to simulate human head or hands movement among others, this can be found and modify in the properties window of the simulator (Figure 4.2). Furthermore, to perform more realistic simulations is possible to add virtual rooms that can be modified, editing the objects shown by default.

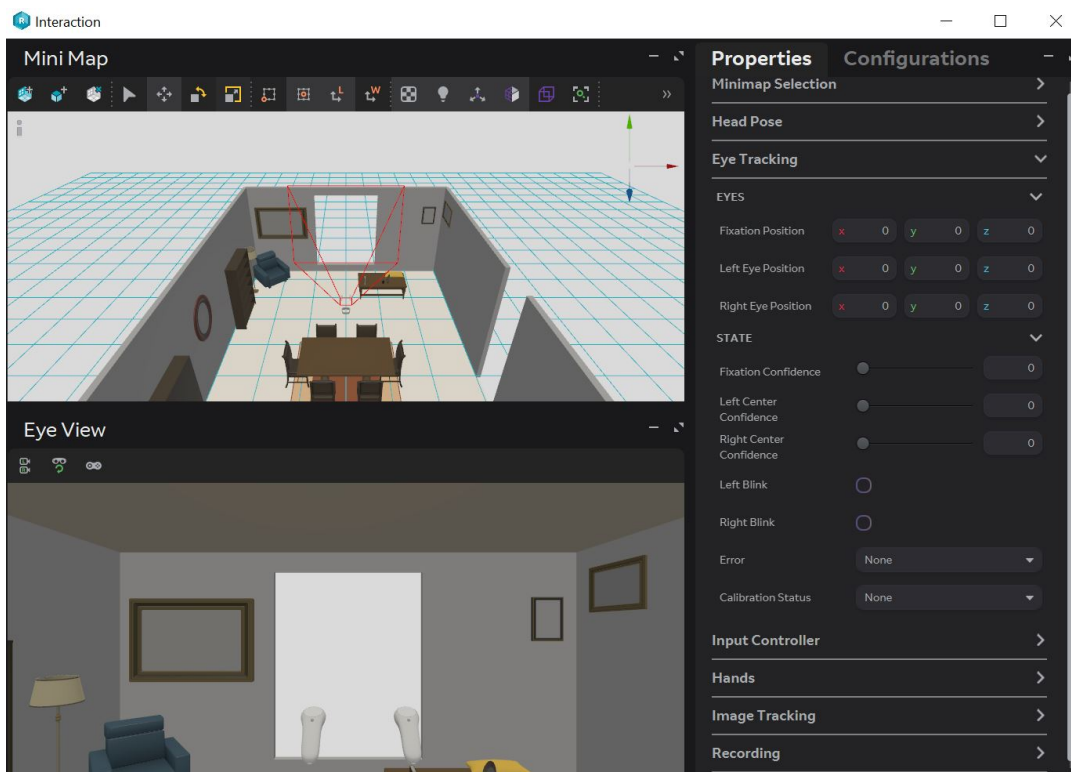


Figure 4.2: Magic Leap Remote

4.2 Demonstrations of functionalities

This work focuses on the research of the modules based on the scene modelling, remote control, gestures and image tracking. Besides, to correctly complete the project, other members have investigated the raycasting or the video player modules.

4.2.1 Scene modelling

4.2.1.1 Description

The modelling of scenes consists of identifying the real objects that are in the scene and creating virtual objects that allow us to simulate the limits of these real objects but in the virtual field, so there can be an interaction between real and virtual world. To do this, we use this functionality provided by Magic Leap, which allows us to create a mesh of the real scene so that we obtain a series of virtual objects (triangles) that define the mesh. Meshing is especially important in the context of real-time scene modelling and collision detection.

4.2.1.2 Applying functionality

For allowing the application to model the user's environment, firstly we have to add to the Hierarchy window the Magic Leap library that allows the meshing, *MLSpatialMapper*, we can find it in the project folder *Assets > MagicLeap > Examples > Prefabs*. Then, it is necessary to create an empty object that will be used as the parent object of all the triangles, that are part of the mesh and which has been identified by the *MLSpatialMapper*. In order to do this we right click on the scene and select Create Empty and rename it to *MeshingNodes*. After that, we have to set the *Mesh Parent* of the *MLSpatialMapper* to be the *MeshingNodes* object. To do this we drag the *MeshingNodes* object to the Mesh Parent attribute of the *MLSpatialMapper*. The final scene should look as shown in Figure 4.3.

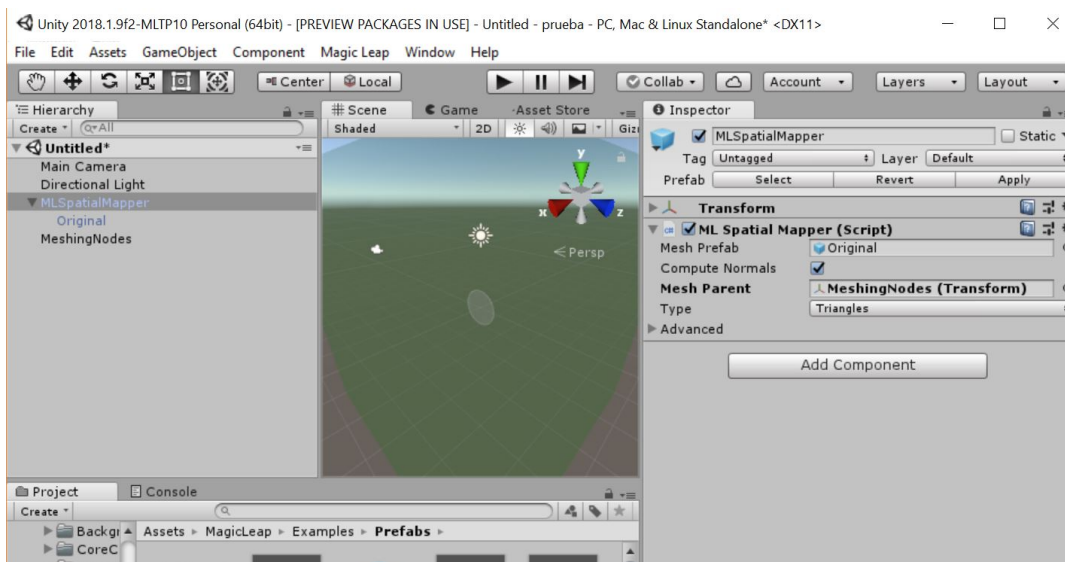


Figure 4.3: Scene modelling application in Unity

4.2.2 Control interaction

4.2.2.1 Description

The control (Figure 2.4) is a very useful way to interact with the Magic Leap device [12]. It has four buttons:

- **Touchpad.** Pressure range between 0.0 and 1.0 plus force touch functionality. XY touch locations in a -1.0 to 1.0 range detecting events.
- **Bumper.** Binary button (pressed/not pressed).
- **Trigger.** Pressure range between 0.0 and 1.0.
- **Home.** Binary button (pressed/not pressed). Specially use for system and power commands.

These buttons have a series of interactions that are common to any application, these actions and their outputs are shown in Table 4.1. Also new outputs can be defined for a particular action within our application, so that the user can interact with the various elements

Button	Action	Input/Output Behavior
Touchpad		
Bumper		
Trigger	Pull to Select	
	Long squeeze and hold	Extract and Move; releasing will usually place an object
Home	Double-press	While in a Landscape app, it will open the Icon Grid
	Single-press	Creators may use this functionality to let users move back within the app's own menus
	Hold Trigger and Home button for 3 seconds	Powers down the Control
	Press and Hold for 1 second	Powers on the Control

Table 4.1: Button input behavior across Magic Leap platform [12].

Moreover, the user can interact with the applications changing the position and the orientation of the control, thanks to its six degrees of freedom and the raycasting.

4.2.2.2 Applying functionality

To interact with the control it is necessary to initialize it in the script where we are going to call it by adding the *Start()* routine to the code shown in Figure 4.4.

```

1 // Define controller as global variable
2 private MLIInputController _controller;
3 // Use this for initialization
4 void Start () {
5     MLIInput.Start ();
6     MLIInput.OnControllerButtonDown += OnButtonDown;
7     _controller = MLIInput.GetController (MLIInput.Hand.Right);
8 }

```

Figure 4.4: Initialize control

Then the control functions can then be accessed via the following attributes of the controller object:

- *controller.Bumper* for the bumper
- *controller.HomeTap* for the home button
- *controller.TriggerValue* for the trigger
- *controller.Touch1PosAndForce.x*, *controller.Touch1PosAndForce.y*, *controller.Touch2PosAndForce.x*, *controller.Touch2PosAndForce.y* for the position and force exerted on the touchpad.
- *controller.Position* for the control position.
- *controller.Orientation* for the control orientation.

By checking the events occurring in the control and the instructions indicated above, it is possible to associate an action in the application to a specific interaction with the control.

4.2.3 Hand tracking

4.2.3.1 Description

Hand tracking includes recognizing gestures and tracking the key points of the hands in every frame. Each hand is tracked with fifteen key points which help to estimate the eight defined hand poses (Figure 4.5).

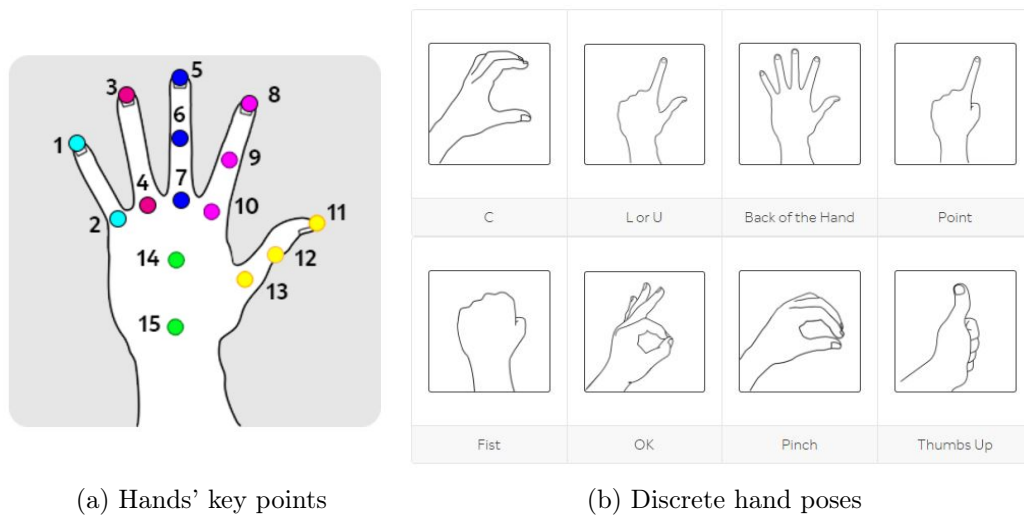


Figure 4.5: Hand tracking [13]

4.2.3.2 Applying functionality

To know the gesture and position of the hands the first step is to access both hands through the following sentences:

```
MLHand rightHand = MLHands.Right
```

```
MLHand leftHand = MLHands.Left
```

To identify the gesture being made with either hand it is necessary to access the *KeyPose* attribute of the corresponding hand. Using the *ToString()* method, the name of the gesture can be obtained in order to execute a response depending on it.

```
string keyPose = MLHands.Left.Keypose.ToString();
```

The names that the previous line of code can return correspond to the eight discrete hand poses indicated in Figure 4.7d from right to left and from top to bottom are: *C*, *L*, *OpenHand*, *Finger*, *Fist*, *OK*, *Pinch*, *Thumb* and in case there is no gesture detected *NoHand*.

In addition, it is possible to obtain the accuracy of the gesture that has being carried out with a value between 0 and 1 by means of the following instruction:

```
float keyPoseConfidence = MLHands.Left.KeyposeConfidence();
```

Finally you can find out the spatial position of each hand by knowing the position of one of its key points, for example its center point:

```
Vector3 position = MLHands.center
```

4.2.4 Detect and tracking images

4.2.4.1 Description

Magic Leap has an image tracking system [14], which, using the device's camera, can detect two-dimensional images from a custom-defined target set. With this information, it detects and tracks the location and rotation of the targets while you are moving around. You should keep in mind that this functionality can not be used with QR codes or arUko markers because they are poor target images. The best conditions to detect an image properly is an image with high contrast, no symmetries or repeating patterns and identifiable differences between the target images. On the other hand, if the image has reflections, is distorted or the viewing angle is too high, probably the system will not detect the target.

You can detect at the same time up to 25 images, but only one copy of each image target at the same time. If you have an application that uses image detection it is strongly recommended deactivate the function when is not needed because it has a high computational cost and can affect the overall performance of the application.

4.2.4.2 Applying functionality

To include image detection and tracking in a Unity application, a specific Magic Leap library based on Machine Learning, *MImageTrackerBehavior*, is used. This library is obtained by importing Magic Leap Custom Packages when creating the project and adding it to an empty game object. This script is located in *Assets > MagicLeap > Libs > LuminUnity > MImageTrackerBehavior*. Once added the script, its variables must be defined. To do this, first, we include the image of the object to track, and then, according to our preferences, we set the rest of the variables (Figure 4.6). Where *Image* is the 2D texture of the object to detect and track, *Is Stationary* must be set to true if the object to follow is assumed fixed in the real world and its surroundings are planar (i.e. wall, table, floor, etc.), *Auto Update* must be checked if the behaviour should automatically move the attached game object and finally the value of *Longer Dimension in Scene Units* indicates the longest dimension of the printed image in scene units, if the width is greater than the height the width will be indicated, height otherwise, the smaller this parameter the application will be able to recognize the image on a smaller scale.

If you want to point out the location of the objects, you have to add a visualization script. In this case, we have used the example that comes by default in the Unity package: the *ImageTrackingVisualizer* script.

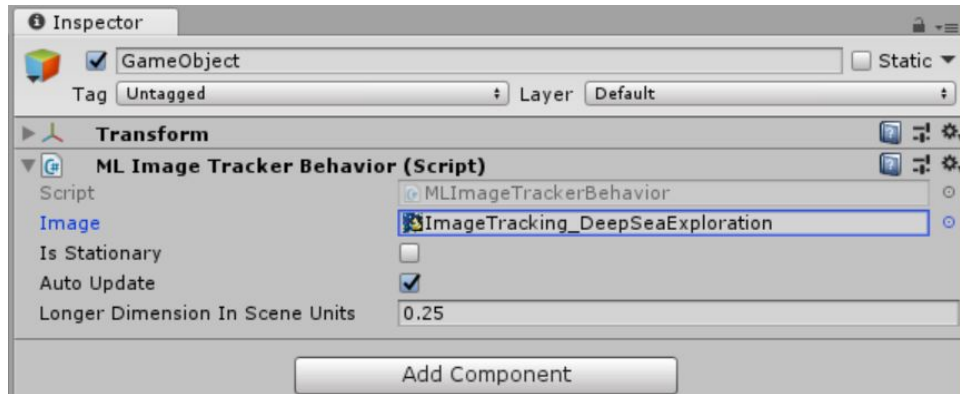


Figure 4.6: Script *MLImageTrackerBehavior* options

4.2.5 Other functionalities

The device has other features built in like raycasting and media player. Even though these functionalities have been studied by another member of the project, I would like to introduce them as they are used in the final application.

4.2.5.1 Raycasting

Raycasting [15] is a technique through which an invisible beam is projected from a certain position in a direction. The ray collides with real and virtual objects. In that way, we can detect or interact with them. Raycast usually is produced by:

- **Control.** Where the control is pointing.
- **Head.** Where the center of the lightwear is pointing.
- **Eyes.** Where your eyes are fixated
- **Virtual object.** Where a specific part of the virtual object is pointing.

Raycast reconstructs the environment of the user and save the surrounding is a radius of 5 meters truncating the ray if it does not collide in this area. If the application needs information of more distant elements to works properly it is recommended to map previously the room and save the mesh, due to the collision ray-mesh is not subject to the distance limit of 5 meters, this method also improves the performance if it is used raycasting constantly.

4.2.5.2 Media player

This feature is used to show the user a video in the application. With this methodology, you can add videos that are on a server or charging them to the Magic Leap device.

To add you have to add a *Video>Video Player* object in the *Hierarchy*, set the variables of the object and activate it when is wanted to be shown.

4.3 Database

This section describes the current database of the application and its elements.

This application considers a database containing the different objects to be displayed. These objects should be as realistic and of the highest possible quality. For the application we need objects of four types (Figure 4.7):

- The shelf on which the shopping items are placed.
- A shopping cart to add the purchased items.
- A trash bin to discard items previously added to the shopping cart.
- The different shopping items. The purchasing products must be diverse, and in many occasions personalised. In the created database the objects can be classified into cans, boxes, bottles, vegetables, fruits and books. Moreover, there are some models created by our own. The database contains at least one product from each of these purchase items, being of satisfactory quality and containing products of well-known brands such as *CocaCola* or *Coronita* or more general such as a box of cereals or different kinds of fruits.

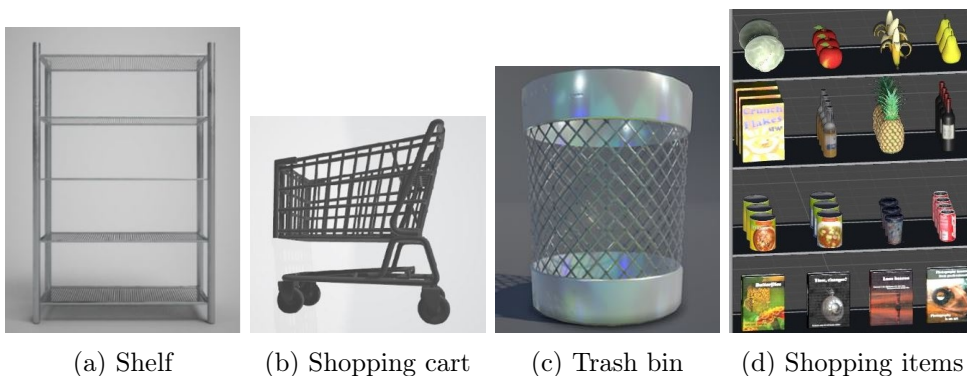


Figure 4.7: Database elements

4.3.1 Database source

The items of the database must be mainly files with obj extension, since it is the format known to be supported by Unity and widely used in the creation of elements 3D. We have three main sources for obtaining 3D objects:

- **The Unity shop.** It consists of both free and paid items, and the option to add different textures to the same object.
- **Web pages.** In the same way, they have both free and paid objects. We can find created models in web pages like Free3D¹, Turbosquid², cgtrader³ or Hum3D⁴. The content available at the Internet is very large and the previous pages are just a small sample. Depending on the quality you are looking for and the price you are willing to pay you can select between different models of the same object.
- **Creation of own objects.** In the case you do not find any model similar to the real object you desired you can create a 3D model of it. For this purpose you can use 3D scanners or images. In the context of this work, we have investigated the procedure to generate 3D models using the free software Colmap and Meshlab and created some objects.

In order to create a 3D model from a real object the first step is to take photos of the object from different points of view. As more different details the object has the better. Colmap processes those photos to relate the images developing feature matching and obtaining the relative location of each image and a dense point cloud of the object. To have a reconstructed object in an importable format for Unity reconstruction object we need the mesh (.obj file) and the texture (.jpg file), for that, Colmap's output is processed with Meshlab. Cleaning the dense point cloud and applying post processing filter we finally obtained the required files with the 3D model reconstruction of the photographed object.

4.4 Demo application

Once the different modules have been studied and the database created we proceed to the development of the demonstrator. This part of the project has great support from the rest of the team.

¹<https://free3d.com>

²<https://www.turbosquid.com>

³<https://www.cgtrader.com>

⁴<https://hum3d.com>

This section explains the operating characteristics of the demonstrator from a functional and structural point of view, indicating the modules and the actions and gestures integrated in each stage.

For the development of our application, most of the modules described above are used. Scene modelling for meshing the surrounds of the user and locate the environment, hand tracking and head raycasting to interact with the digital objects and the media player to show a video to the user when an object is selected.

The proposed demonstrator is based on a state machine with 3 main states: positioning, purchasing and shopping (Figure 4.8). The first state is Positioning, from which the Purchasing stage starts and after that, the user can check the purchased items in the Checking stage. Once in Checking the user you can go back to continue buying (and then go back to Checking) or finish the application.

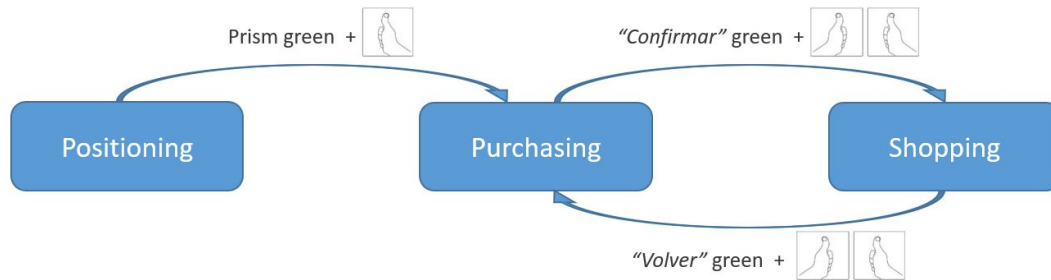


Figure 4.8: State diagram of the retail application

4.4.1 Positioning stage

Positioning stage is opened when the application starts. It includes a prism which color changes from red to green when there is enough space to locate the shopping environment (the shelf, the products and the shopping cart).

4.4.1.1 Modules

This stage has the following modules implemented:

- **Text.** All the messages are in front of the user, using the head pose information. When the application starts, in the scene is shown a welcome message to the user. Moreover, performing a gesture, the user can look up the instructions for use of the positioning stage.
- **Scene modelling.** We need to create a mesh around the objects surrounding the user. With this mesh it will be checked if the prism is located or not on a

flat surface and it will be calculated whether it is possible to locate the shopping environment without colliding with any real object.

- **Gesture recognition.** Recognize four different gestures done by the user.
- **Head raycasting.** A raycast is thrown from the centre of the lightwear to determine if there is any object that collides with the beam. If the beam does not collide with any object, the prism will be located one meter away and follow the headpose.

Additionally, it is implemented a manual mode (deactivated by default) in which the user can modify the distance between the glasses and the prism between 0.6m and 4m. With gestures. In this case, if the raycast strikes an object in the scene and the distance between the glasses and this object is less than the defined distance to the prism, the prism will be placed on the surface of that object.

4.4.1.2 Gestures and states

Figure 4.9 shows the state diagram of actions that can be done in the manual model. On the other hand, Table 4.2 shows an overview of the gestures and their corresponding outputs in the Positioning stage.

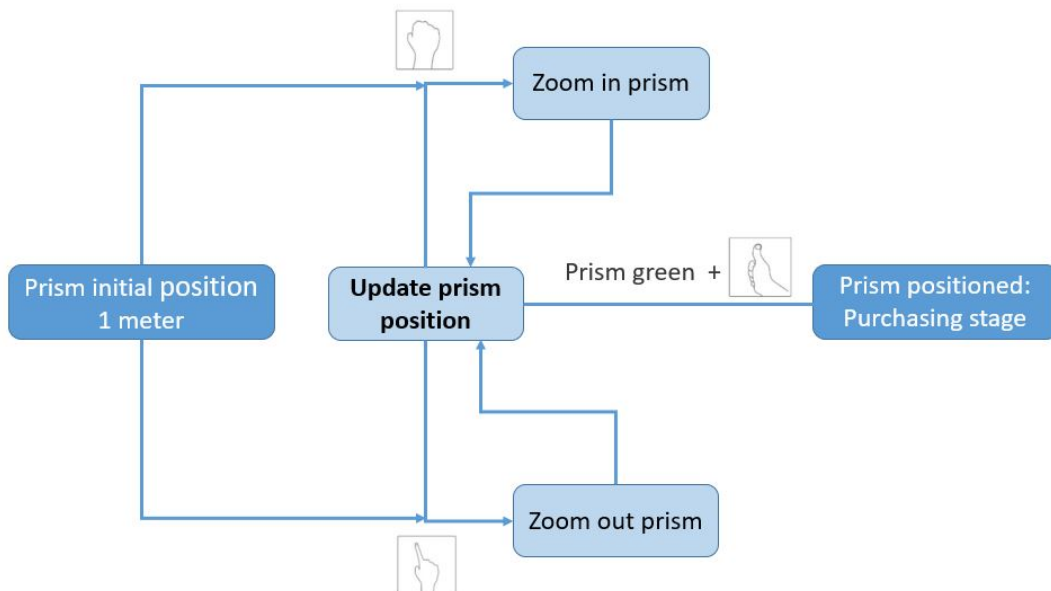


Figure 4.9: State diagram of the position stage with manual mode





Gesture	Hand	Action
	Right	If the prism is green, in a flat surface with enough space for the shelf and the shopping cart, making this gesture the purchasing state will start.
	Left	Configurable. Decrease the distance between the user and the prism.
	Both	The general instructions for the use of the application are displayed for 8 seconds.
	Left	Configurable. Enlarge the distance between the user and the prism.

Table 4.2: Gestures and actions linked to the positioning stage

4.4.2 Purchasing stage

Once the user have located the shopping scene the user will see a shelf with 20 different items and a shopping cart on the right with a *Continuar* button over it to continue with the shopping and check the purchasing objects in other scene. This objects can be taken, observed from different angles, added to the shopping cart or discarded with gestures. Moreover the user can change the position of the shopping cart and additional information will appear when an item is selected.

4.4.2.1 Modules

This stage has the following modules implemented:

- **Text.** All the messages are in front of the user, using the head pose information. These messages are warnings about the constraints of the Magic Leap One device or the application. They will appear when the object is too near to the glasses, or the user is too near to the shelf. Also, there is information about the price and the weight of all the products on the shelf and when an object is selected and positioned in front of the user, next to the object, will appear textual information about it.
- **Media player.** When an object is selected and positioned in front of the user, next to the item will appear a video with related information about the selected object or a similar one.
- **Advertisement.** In the shelf can be observed advertising of different companies. It can be easily change with other images.

- **Head raycasting.** An invisible beam is projected all the time from the head. When it collides with an object on the shelf it is illuminated with a blue halo, that means that this object is preselected, also, when the ray collides with the *Continuar* button its colour changes to green.
- **Hand tracking.** We track the centre point of the hand to know its position. If the hand is near a product or the shopping cart the user can take it, also with this information we can know if the user's hand is above the shopping cart or not.
- **Gesture recognition.** Recognize seven different gestures done by the user the result action depends on the position of the hand and where the user is looking.

4.4.2.2 Gestures and states

Figure 4.10 shows the state diagram of actions that can be done in the manual model. While Table 4.3 shows an overview of the gestures and their corresponding outputs in the Purchasing stage.

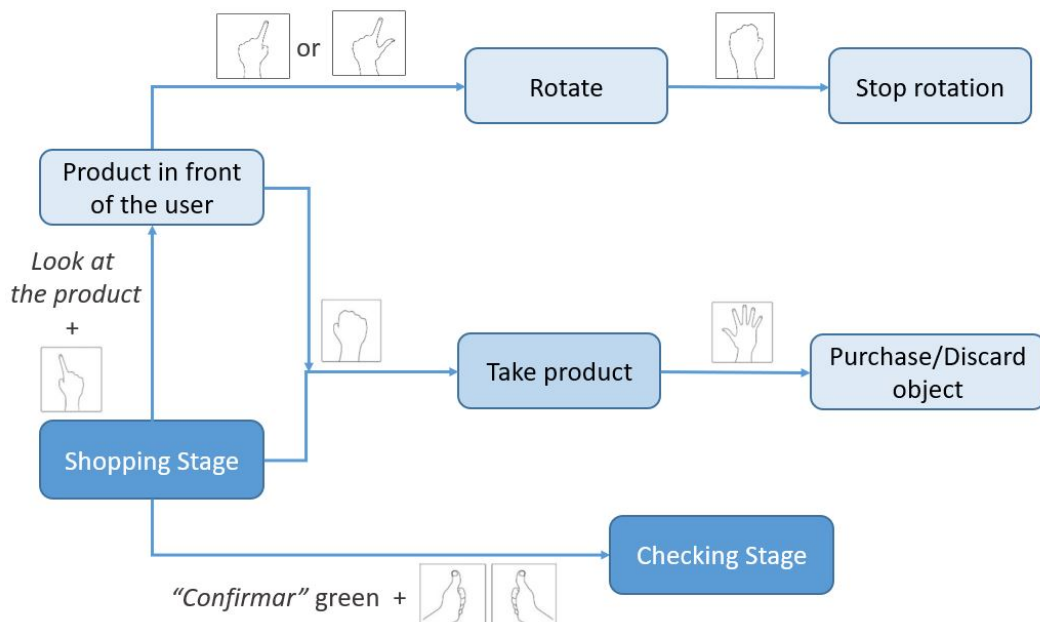


Figure 4.10: State diagram of the position stage with manual mode

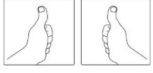









Gesture	Hand	Action
	Both	To change to the Checking stage. The user have to look to the <i>Confirmar</i> button and when it is green make the gesture.
	Left	Stop the rotation of the previously selected object at distance.
	Right	Take an object that has previously selected object at distance doing this gesture near to it.
	Right	Take an object from the shelf doing this gesture near to it.
	Right	Take the shopping cart in the middle of the handlebar and move it
	Right	Select an object at distance. If the user is at a certain distance from the shelf (1.5 meter minimum) position the object he is looking at, denoted by a blue halo, 1 meter from the user.
	Left	Start rotating to the left the previously selected object at distance.
	Right	Leave the shopping cart in the current place after moving it with the fist gesture with the right hand
	Right	Leave the shopping cart in the current place when you are moving it
	Left	Start rotating to the right the previously selected object at distance.

Table 4.3: Gestures and actions linked to the purchasing stage

4.4.3 Checking stage

In this stage, the user can see the shelf with the previously bought items and in the position of the shopping cart a bin. Taking and dragging the products on the bin means to remove them from the shopping cart. In this scene we have two floating buttons, on the left, *Volver* to return to the shopping scene and on the right *Terminar* to close the application. Comparing to the purchasing stage in this one we decided to reduce

the number of gestures and the possibilities to interact with the objects to see which interaction modes the users prefer.

4.4.3.1 Modules

This stage has the following modules implemented:

- **Advertisement.** In the shelf can be observed advertising of different companies. they can be easily change with other images.
- **Head raycasting.** An invisible beam is projected all the time from the head. When the ray collides with the *Volver* or the *Terminar* buttons their colours change to green.
- **Hand tracking.** We track the centre point of the hand to know its position. If the hand is near a product the user can take it, also with this information we can know if the user's hand is above the bin or not.
- **Gesture recognition.** Recognize five different gestures done by the user.

4.4.3.2 Gestures and states

Figure 4.11 shows the state diagram of actions that can be done in the manual model. To finish, Table 4.4 shows an overview of the gestures and their corresponding outputs in the Checking stage.

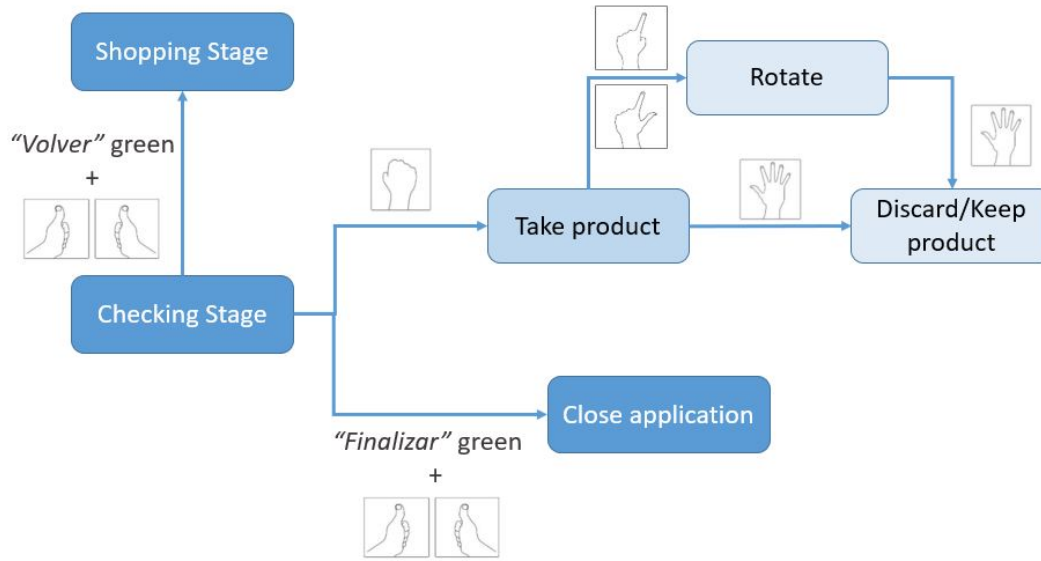


Figure 4.11: State diagram of the checking stage






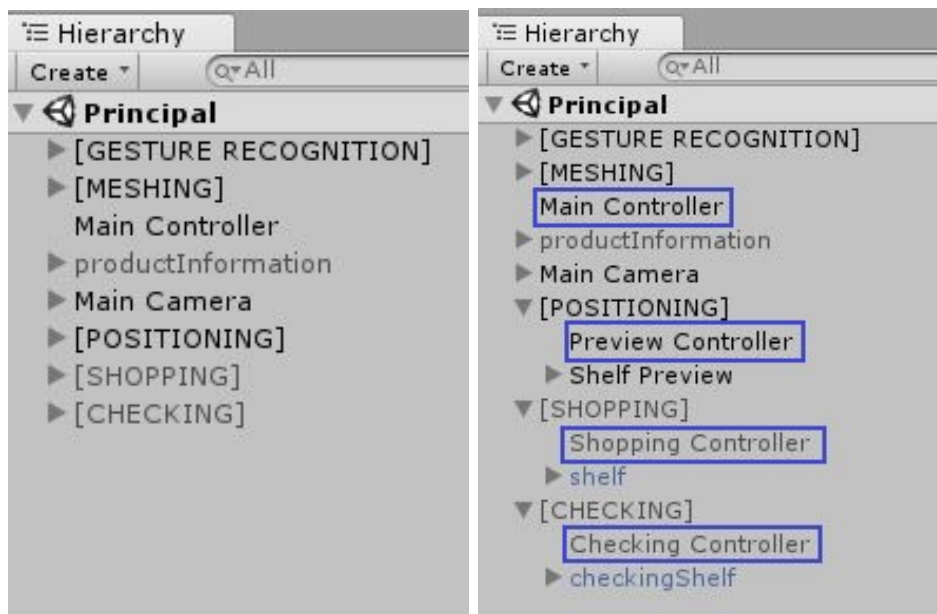
Gesture	Hand	Action
	Both	To change to the Checking stage. The user have to look to the <i>Confirmar</i> button and when it is green make the gesture.
	Right	Take an object from the shelf doing this gesture near to it.
	Left	When the user has a product in his hand doing this gesture the object rotate to the left.
	Right	When the user has a product in his hand, the item is discarded if he opens the hand over the bin (the bin is green) if the hand is opened in another position the item go back to the shelf.
	Left	When the user has a product in his hand doing this gesture the object rotate to the right.

Table 4.4: Gestures and actions linked to the checking stage

4.4.4 Software architecture

4.4.4.1 Modules

The structure that has been followed to organize the different objects and script within the application follows the concept of modules. A module is the set of objects and script that have a common function. In the application these modules have been defined by an empty object that contains all the objects and scripts involved. These modules have their name in square brackets ([NAME]) as shown in Figure 5.3a.



(a) Application modules

(b) Controllers. Inside the blue boxes

Figure 4.12: Hierarchy application structure

Each module has a functionality or a application stage associated:

- **Gesture Recognition.** Module in charge of detecting the gestures carried out with both hands and to provide the level of accuracy associated with them.
- **Meshing.** This module contains the functionality associated with the realization of the mapping of the real environment in which the application is running.
- **Positioning.** Module in charge of managing all the functionality associated with the positioning application scene.
- **Shopping.** Module in charge of managing all the functions linked to the purchasing scene.

- **Checking.** Module in charge of managing the functionality associated with the purchase scene.

4.4.4.2 Controllers

The application has an empty object (Main Controller) that contains the script that works as the main controller for the entire application. This script manages the operation of the application state machine and iterates on it calling the different modules. Each of the modules is defined in an empty object that contains a script with the corresponding functions. All the controller scripts are positioned as the first child of the modules (Figure 5.3b).

Chapter 5

Experiments

This chapter shows the results obtained, both in the tests carried out with the different modules and with the final application. These results are mostly subjective and intend to demonstrate the usability of the application by users

5.1 Conditions for optimal working

Magic Leap detects and combines both virtual and real worlds, so to add correctly virtual objects to the real world that surrounds the user we recommend a series of tips.

To scan properly the environment it is recommended to be in a room with around 50 lux evenly distributed, in case the device receives direct light or there are poorly lit areas may not detect the room correctly. Reflective and transparent surfaces can also lead to detection errors. Furthermore, the infrared rays only detect the objects located less than 5m from the device and if there are moving objects will generate a false mesh, so it is recommended not to use the device in open or crowded spaces.

The device have the restriction that it not allows the representation of those virtual object that are located less than 0.45m from the lens of the device. This have to be taken in mind, trying to get the user to see objects from a distance and in case they cases the user can have them too close showing a warning.

5.2 Demonstrations of functionalities

5.2.1 Scene modelling

This module of the device is the most sensitive to complex scenarios: poorly illuminated, very illuminated or with elements that do not reflect infrared light properly,

either because of reflective or transparent surfaces. In addition, the theoretical scope of scene mapping is 5 meters.

The tests carried out on the modeling of scenes have been as expected. Under ideal conditions (well-lit room, with limited extensions and non-reflective objects) the room is modeled correctly, being able to detect both the floor and the walls and the various objects. However, if we include in the scene computer screens or televisions, or reflective floors, those areas are no longer detected by the device, giving rise to unmapped areas. Also, in very large rooms or open environments we find the problem that the farthest areas cannot be detected. We can observe some of those events in Figure 5.1.

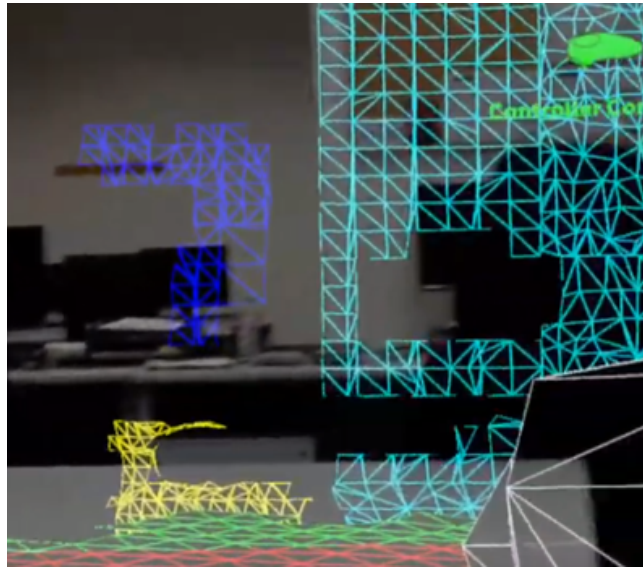


Figure 5.1: Scene modelling example

Other problems detected are:

- After the continued use of an application with the scene modelling functionality working, if abrupt changes in the position of the glasses are made (by removing them, putting them on etc.) the initial mesh changes, moving away upwards, downwards, to the right or to the left, in such a way it no longer matches with the elements of the scene. In this case it is necessary to restart the application and sometimes the device.
- Problems are also seen when there are moving objects in the scene (i.e. people), in this case, it is created a mesh around the person and it will be updated with the movement. The problem arises when this update is not performed correctly, leaving a fake object identified in the scene.

If the modeling of the scene is a very important component of the application is recommended that reflective objects are covered or removed from the room, as well as large rooms or moving objects.

5.2.2 Control interaction

The results obtained with the control are very good. It has been observed that the control is continuously located and both its position and its rotation are very sensitive. Besides, it could be said that all information is sent and received in real time to the lightwear. Regarding the buttons, the results are similar. It is easy to interact with the buttons and in real time you can see the output of pressing them. The Touchpad has a large sensitivity to see the pressure being exerted on it and where and as well with the trigger. It can be configured different outputs depending on how much pressure is exerted on them.

5.2.3 Hand tracking

The tests carried out to see how the hand tracking works consist of showing in a floating window by means of images and text the gestures and level of confidence of them, that is, the security that the device has that the detected gesture is the one indicated (Figure 5.2). The aim of this test is to check the speed with which the device recognizes gestures, whether it can easily recognize both hands and with which accuracy it recognizes each of the gestures.

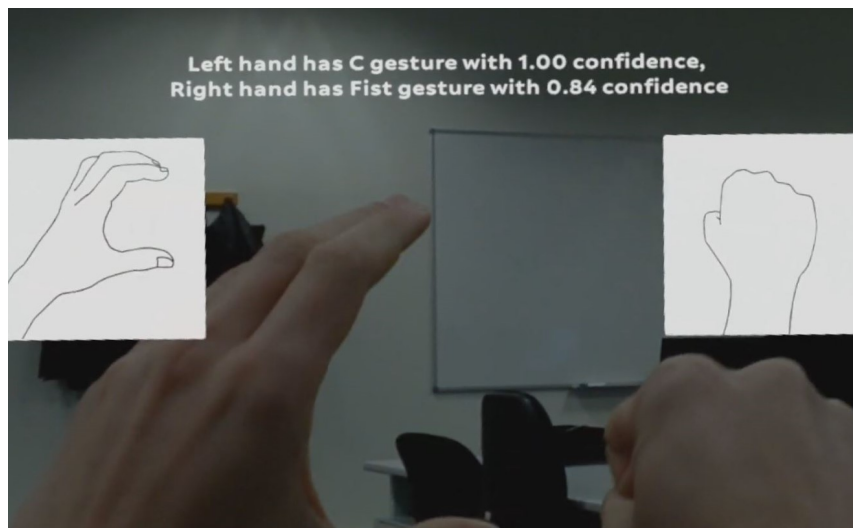


Figure 5.2: Hand tracking example

In addition, the position of the hands to the lightwear is also known. Specifically,

the position of the centre of the hands is obtained, although the library of Magic Leap allows to obtain the position of different points distributed throughout the fingers of each hand. It has been observed that the device obtain the gestures very quickly being able in certain cases to suppose a problem when it is made a change of gestures. This can be seen when the hand is closed and is opened to release an object. In this transition, the glasses often detect other gestures with a high level of confidence such as the *Thumb Up* gesture. The implemented solution to this failure is to maintain a gesture for a certain period (e.g. one second) eliminating possible "false gestures" detected in the transitions or actions. Another error detected in the test performed has been that the *OK* gesture is not detected in either of the two hands, so for the approval or confirmation actions have been used the *Thumb Up* gesture.

5.3 3D modeling

For this project we wanted to investigate the possibility of create 3D models of real objects with free software, in this case Colmap and Meshlab. Taking pictures (around 50 images) of the desired objects and following the necessities steps we obtained good 3D representations of three real objects a coffee bag, a lime box and a cup (Figure 5.3)



(a) Images of real objects



(b) 3D reconstruction models of the above real objects

Figure 5.3: 3D modeling

As can be seen the obtained models are not the best, but they are quite good. We have found problems with the reconstruction when we have an object with two very similar faces, but developing a more complex reconstruction process and taking more

specific photos this has been solved. On the other hand, as we do not have taken photos from the bottom of the objects, there is no information about this part of the objects.

5.4 Retail application

In order to evaluate the demo application two types of tests have been carried out. First, a subjective test by the developers, checking that the requirements set out in Section 3.1 are met. On the other hand, a series of experiments have been carried out by a group of people who have evaluated the application according to subjective parameters.

5.4.1 Software testing

In this section are explained and shown all the functionalities of the final demonstrator, and which requirements have been fulfilled. For easier comprehension, this section is divided into the three stages: positioning, purchasing and checking.

Throughout this section, visual examples are shown of the gestures made in the different phases of the demonstrator. Three views are provided for these examples. A front recorded view with an external camera, another side view recorded with another external camera and a recording of the content displayed by the user using the MLO device. Figure 5.4 shows the display schema to be used in the rest of the section.

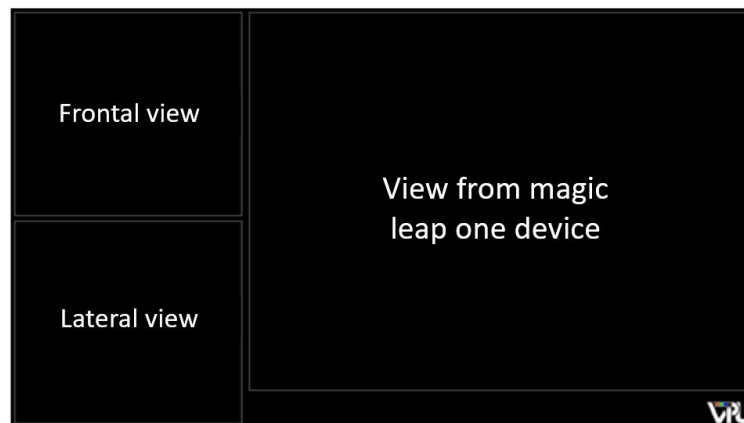


Figure 5.4: Visualization scheme of the software testing images

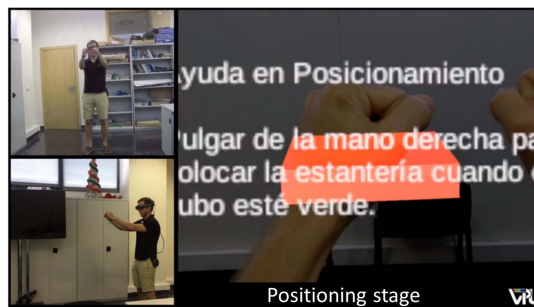
Moreover, in *YouTube*, can be found two videos demonstrators of the different actions that can be done in the developed application. A complete version¹ with all

¹<https://youtu.be/iN4YPbomJsY>

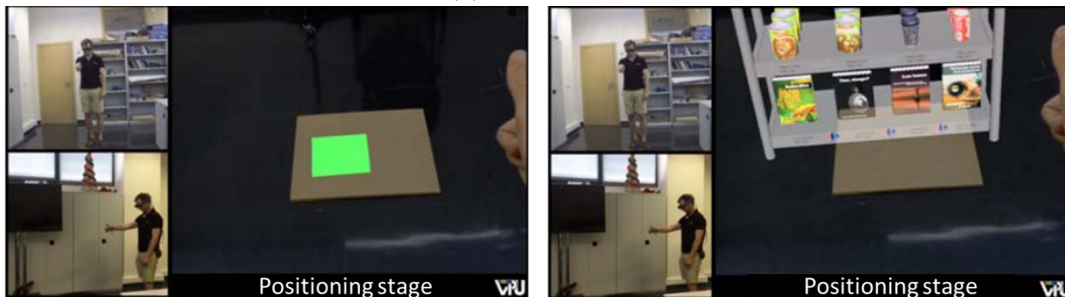
the steps carried out by the testers and a shorter version of it².

5.4.1.1 Positioning

After starting the application, the positioning stage begins. In this first part of the application, the modelling of the room is carried out with the purpose of being able to locate the shopping environment (Figure 5.5b). Due to the environment of tests has a reflecting floor the experiments have been carried out putting an opaque object on the floor to facilitate the detection of the floor, thanks to this, the detection of a place in which to locate all the environment is usually carried out in a fast way. Also, in this scene, the user can consult the instructions for use through a gesture (Figure 5.5a).



(a) Instructions



(b) Correct positioning of the purchasing environment

Figure 5.5: Visual examples of the positioning stage

5.4.1.2 Purchasing

In the purchase stage, the shelf with the different objects and a shopping cart appears. The virtual objects are of good quality, although they could be improved in cases such as the pear or the cereal box. The environment also includes one of the reconstructed objects, the mug. Moreover, to build a more realistic shopping environment multiple

²<https://youtu.be/VJsX6p6cs2c>

copies of each object have been added, although, the user can only interact with the first of each of these objects.

To interact with the objects the user can take the different products of the shelf closing his right hand near them (*Fist* gesture) and the product will remain in his hand until he opens it, as can be seen in Figure 5.6a. While the object is in the user's hand if he puts his hand too near to the lightweight warning advice will appear, indicating that he must keep the object further away so that it can be rendered by the device, being partially cut off or disappearing completely. The user can also select the products from distance, looking and then the product will be illuminated with a blue halo and the user can put it closer doing *Point* gesture with the right hand. This product is located 1 metre to the user appearing extra information: a video and text about the selected item (Figure 5.6b). With the object selected in front of the user he can rotate it to see it from different points of view. In this stage the user only has to do the rotate gestures once and the object will rotate until the user does the stop rotating gesture.

The user can discard a product, making it go back to the shelf with two different gestures depending if the object is in front of him or on his hand. On the other hand, if he wants to add the product to the shopping cart he has to take it with the *Fist* gesture and open his hand above the shopping cart (Figure 5.6c).

Finally, as can be seen in Figure 5.6d, there is panel information where the user can easily see how many products he has added to the shopping cart and the money spent accordingly. Below this is the *Confirmar* button which allows the user to go to the checking stage.

5.4.1.3 Checking

In this phase the shelf is shown with the products purchased in the previous stage and there is a bin instead the shopping cart. As most of the changes for the demonstrator have been developed in the purchasing stage, in terms of interaction with objects, this stage is a basic version of the previous one, in which the user can only interact with the products going to the shelf to take them. Besides, to rotate the item, the user has to keep the rotating gestures as long as he wants the object rotates. These differences are useful in subjective evaluation for users to decide which modes of interaction they prefer.

To discard an object the user has to take and drag it in the bin as shown in Figure 5.7a. Finally, the user also has the panel information about the total purchase and two buttons, one to go back to the Shopping stage (*Volver*) and another to finish the purchase and close the application (*Terminar*), as can be seen in Figure 5.7b.

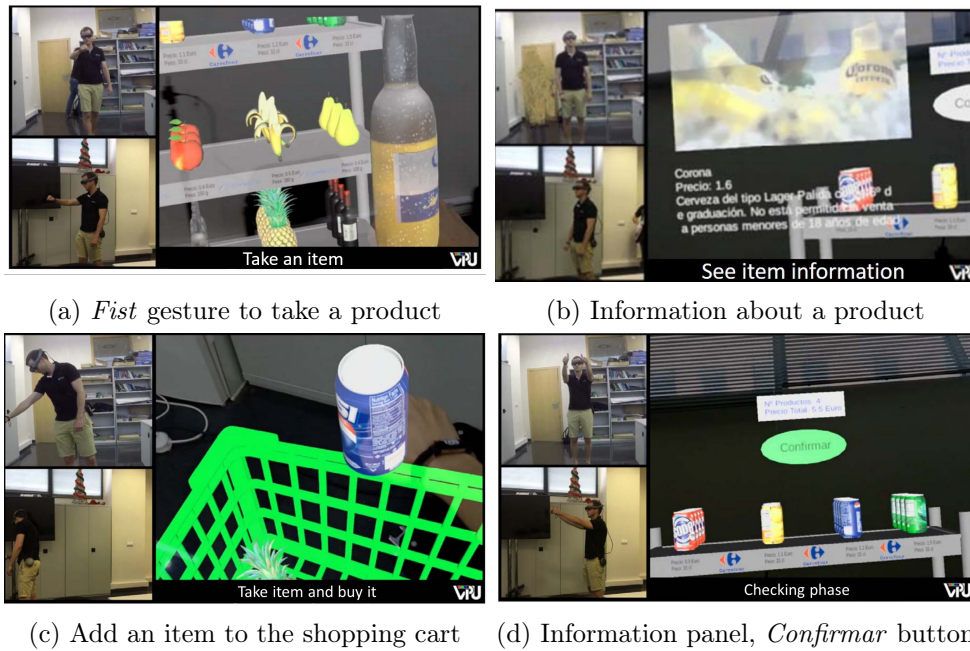


Figure 5.6: Visual examples of the purchasing stage

5.4.1.4 Requirements analysis

From the previous analysis can be said that all the functional needs are fulfilled. In the different stages of the application it is possible to interact with the objects by the use of gestures, hand tracking and head raycasting. The shopping items are high quality virtual representations of real objects and, in the case some of them wanted to be improved, solutions such as obtaining payment models or creating new models have been shown. Moreover, extra information about the objects is displayed in the purchasing scene.

The non-functional needs are also met. It works in real time, the implemented code is well structured and scalable. The most problematic NFN is the reliability, when the application is used for too long (more than an hour), it starts coming up some bugs.



(a) Remove an item from the purchase (b) Information panel and buttons

Figure 5.7: Visual examples of the checking stage

5.4.2 User experience

5.4.2.1 Testing procedure

Questions relating to specific functions of the application
Q1: Do you read the instructions well? (1: Bad - 3: Good)
Q2: Are the instructions understandable? (Yes/No)
P3. Was it difficult for you to place the prism? (1: Difficult - 3: Easy)
Q4: How easy do you think it is to approach the products from distance? (1: Difficult - 3: Easy)
Q5: How easy do you think it is to pick up objects up close (1: Difficult - 3: Easy)
Q6: What do you prefer, taking objects from afar or approaching the shelf? (Near/Far)
Q7: Which rotation mode do you prefer? The purchase or the checking one? (Purchase/Checking)
Q8: Is it easy for you to vary from article to article? (1: Difficult - 3: Easy)
Q9: Do the warning messages seem clear to you? (1: Clear - 3: Not clear)
Q10: Would you purchase online with this device if a similar application were implemented? (Yes/No)
Q11: Would you buy books with this device? (Yes/No)

Table 5.1: Subjective questions about the difficulty and clarity of the different modules of the application, as well as specific preferences of the tester.

In order to evaluate the demonstrator, 14 people have been asked to test the application. For this purpose, they have been given a brief description of the usefulness of the application and information about the gestures and the related actions, anyway, as the ways of interacting with objects are wide it used to be remembered what gesture to perform depending on the action required during the experiments.

To evaluate the application all the testers have to follow the same steps doing the same actions. The procedure includes almost all the possibles actions that the application allow, from buying multiple objects to forcing the appearance of the warning messages.

During the test specific questions about the functionality and preferences are made to the user, these questions are the ones in Table 5.1. Each time a person finishes testing the application they will be asked to answer the questions in Table 5.2, giving values from 1 to 5.

Category	Questions
Easy to use	How accessible did you find the application to be? (1: very difficult - 5: very easy)
Easy to learn	How complicated did you find the instructions to remember? (1: difficult to learn - 5: easy to learn)
Naturalness	How intuitive did you find the application to be? (1: very contrived - 5: very natural)
Fatigue	How fatigue where you after using the application? (1: very fatigued - 5: not fatigued at all)
Speed	How fast did you feel you were able to complete the task? (1: very slowly - 5: very fast)
Attractive	How attractive did you find the application to be? (1: unattractive - 5: very attractive)

Table 5.2: Subjective usability survey assessing the easy of use, easy to learn, naturalness, fatigue, speed and attractive [16] [17].

5.4.2.2 Results

For the evaluation of the results, it is important to point out that the testers are divided into two groups, those who had never used the MLO device before and those six people who used it before, even four of them have tried an old version of the

developed application. However, if you divide the results into the previous groups the difference of the answers are not relevant. For example, inexperienced users find it easier to pick up close objects up, while experienced users find it easier inspect several elements of the same shelf, varying from one to the other and in the case of approaching a specific object from the shelf, both groups consider the difficulty to be the same. It can therefore be deduced that there are no major changes in the level of usability between the two groups of testers; it depends more on the person rather on their experience with the device.

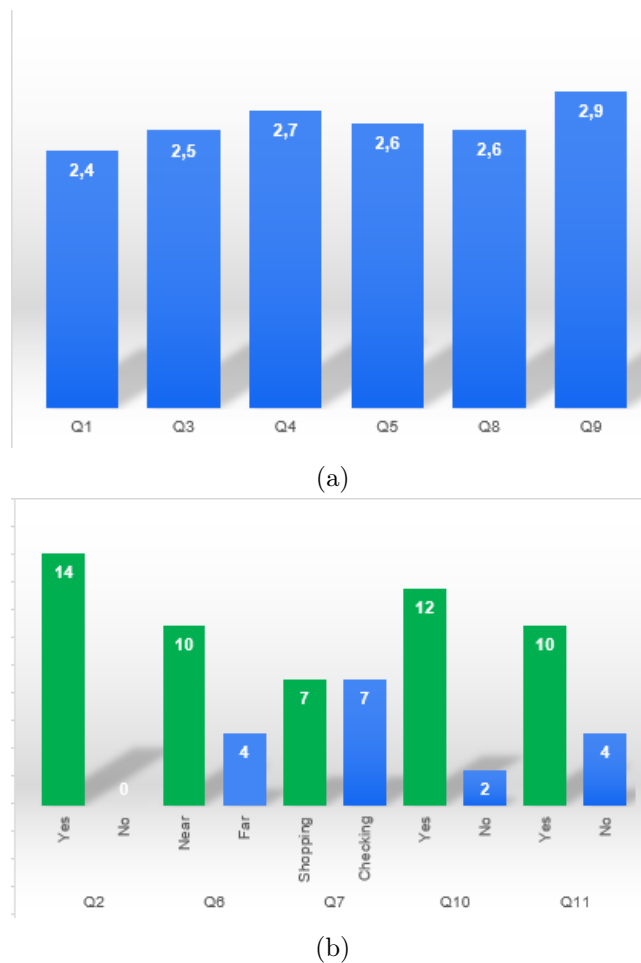


Figure 5.8: Subjective answers about specific functions of the application. Corresponding questions in Table 5.1

Figure 5.8 reflects the results obtained from the subjective evaluation of the different modules of the application. In the Figure 5.8a is shown the numeric result, from 1 (worst case) to 3 (best case) answering the question of Table 5.1, where the

questions Q9, related to a warning message and the easiest to implement, is the best evaluated, and been also important that question Q4, Q5 and Q8, related to the interaction between the user and the object, are good evaluated. Figure 5.8b reports users answers to binary questions, comparing the different ways of interacting with the object: rotating or taking them, where most of the users prefer to objects from near. This results also shows that the instructions are very clear and most of the testers will use an application like this, for both food and books, if it is available to them in the future. Those users reluctant to use it, is because they prefer prefer to go to the physical store for the purchase of some products, such as fruits and vegetables.

The last experiment proposed is showed in Figure 5.9, where we assume that users enjoy the application and see it as very innovative, nevertheless, the velocity and the difficulty to learn the instructions still being a problem, obtaining the lower evaluations.

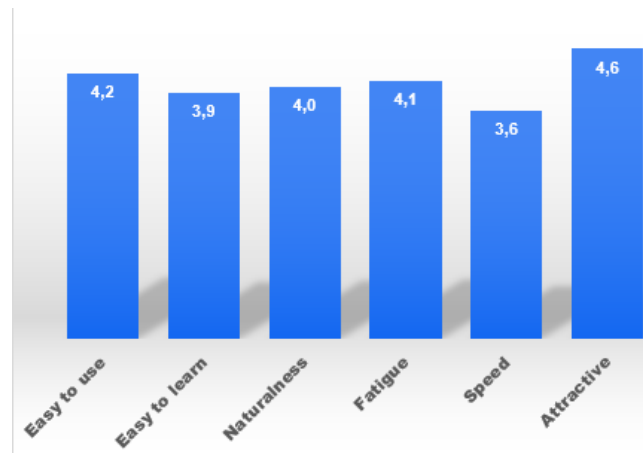


Figure 5.9: Subjective answers about the global application. Corresponding questions in Table 5.2

Chapter 6

Conclusions and Future Work

6.1 Conclusion

In this Master Thesis, we have developed a mixed reality application, based on Unity 3D, for the Magic Leap One device. The main idea of this application is to simulate a three dimensional shop where the user, by making different gestures and interactions, can perform his daily shopping. The first part of the project, consists of a research of the state of the art of the mixed reality technology and other similar ones, such as virtual reality and augmented reality, where we can conclude that MR, by using techniques from both of them, is the most innovative one. Afterwards, the different functionalities from the MLO device have been studied and tested. Moreover, a database of 3D virtual objects has been created and the process of generating 3D models from images of a real object has been researched. Lastly, with all this information the application has been created.

The experiments and results show that the requirements to be met for the application have been satisfactorily completed. In addition, the tests with users have been very satisfactory, most of them would use an application like this if it is implemented in the market.

6.2 Future Work

The developed application is just a demo, for the generation of a real application it would be necessary to scale it, that is, to add more products, videos, etc.

To improve the demo it is important to focus on the comments made by users, from which we can draw:

- Taking into account users' preferences to homogenize the way the user interacts

with the objects in the purchasing and checking stages.

- Make the interaction with objects more natural, improving at the same time the speed of the application.

Other improvements to be made would be:

- Include instructions in all stages.
- Include sound in the videos.
- Improve the database.

It should be borne in mind that the device on which the application has been made is a developer's version, so it is expected that the commercial version has improvements in both software and hardware, allowing the development of the proposed future work more easily.

Bibliography

- [1] M. Leap, “What is spatial computing?.” <https://creator.magicleap.com/learn/guides/design-spatial-computing>, 2019. Last accessed June 2019.
- [2] M. Speicher, B. D. Hall, and M. Nebeling, “What is mixed reality?,” 2019.
- [3] Niantic, “Pokemon go.” <https://www.pokemongo.com>, 2019. Last accessed August 2019.
- [4] I. Grand View Research, “Mixed reality market,”
- [5] M. Leap, “Magic leap home page.” <https://www.magicleap.com/>, 2019. Last accessed June 2019.
- [6] M. Leap, “Magic leap one overview.” <https://creator.magicleap.com/learn/guides/magic-leap-one-overview>, 2019. Last accessed August 2019.
- [7] M. Leap, “Magic leap one creator edition.” <https://shop.magicleap.com/en/Categories/Devices/Magic-Leap-One-Creator-Edition/p/M9001>, 2019. Last accessed June 2019.
- [8] M. Leap, “Lumin os overview.” <https://creator.magicleap.com/learn/guides/design-spatial-computing>, 2019. Last accessed June 2019.
- [9] Microsoft, “Hololens 2. remote assist.” <https://dynamics.microsoft.com/es-es/mixed-reality/remote-assist/>, 2019. Last accessed August 2019.
- [10] Microsoft, “Hololens 2. hardware specifications.” <https://www.microsoft.com/en-us/hololens/hardware>, 2019. Last accessed August 2019.

- [11] M. Lewis and J. Jacobson, “Game engines,” *Communications of the ACM*, vol. 45, no. 1, p. 27, 2002.
- [12] M. Leap, “Magic leap learn guide. control.” <https://creator.magicleap.com/learn/guides/control>, 2019. Last accessed June 2019.
- [13] M. Leap, “Magic leap learn guide. hand tracking.” <https://creator.magicleap.com/learn/guides/lumin-sdk-handtracking>, 2019. Last accessed June 2019.
- [14] M. Leap, “Magic leap learn guide. image tracking.” <https://creator.magicleap.com/learn/guides/lumin-sdk-image-tracking>, 2019. Last accessed June 2019.
- [15] M. Leap, “Magic leap learn guide. raycasting.” <https://creator.magicleap.com/learn/guides/raycast-overview>, 2019. Last accessed June 2019.
- [16] M. Schrepp, A. Hinderks, and J. Thomaschewski, “Design and evaluation of a short version of the user experience questionnaire (ueq-s).,” *IJIMAI*, vol. 4, no. 6, pp. 103–108, 2017.
- [17] D. Jo and G. J. Kim, “Iot+ ar: pervasive and augmented environments for digital shopping experience,” *Human-centric Computing and Information Sciences*, vol. 9, no. 1, p. 1, 2019.