UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Barcelona Est

DEGREE DISSERTATION

Mechanical engineering degree

DESIGN OF VR APP APPLIED TO COGNITIVE TRAINING

Project Memory and Annexes

Authors: Marc Rodríguez Reus

Rubén Tetuá Sanchez-Rey

Director: Jordi Torner Ribé

Department: EGE

Co-Director: Francesc Alpiste Penalba

Announcement: June 2019

Resum

Aquest projecte pretén desenvolupar una aplicació de realitat virtual per a millorar els tractaments actuals per pacients que pateixen algun tipus d'infermetat degenerativa a nivell cognitiu. Això és possible gràcies al que ofereix aquesta tecnologia, que ha tingut un creixement notable en els darrers anys, i que ofereix al pacient una possibilitat d'immersió alhora que una experiència sensorial molt alta. El fet que el pacient estigui treballant en l'entorn de la realitat virtual fa que no hagi d'estar pendent del que passa al món real. La tecnologia necessària per a desenvolupar aquesta aplicació ha estat proporcionada per l'empresa "Vysion360".

Aquest projecte és la continuació dels diversos projectes anteriors on s'ha desenvolupat una aplicació de realitat virtual ambientada en un supermercat, gràcies el motor de joc "Unity". Aquesta conté diferents modes de joc que permeten l'obtenció de dades i seguir el progrés que experimenta cada pacient. En aquest projecte, el principal objectiu va ser la reducció del pes de l'aplicació, alhora que millorar el rendiment d'aquesta i fer un supermercat més realista i uniforme. Tot això ajuda a la immersió del pacient i al seu confort.

Al llarg d'aquesta memòria es detallen les passes seguides a l'hora de desenvolupar l'aplicació, les eines utilitzades, la metodologia, el funcionament de cada mode de joc, l'explicació dels controls a utilitzar, les millores introduïdes i les valoracions dels pacients al testejar l'aplicació.

Per acabar, la col·laboració establerta amb l'*Hospital Clínic* va permetre portar a la pràctica l'aplicació en projectes anteriors. En aquest projecte, a més, l'aplicació s'ha provat amb diversos pacients de l'*Associació de Disminuïts Físics d'Osona*, i que ens ha permès identificar algunes de les línies futures que hauria de seguir l'aplicació per tal que es segueixi desenvolupant i optimitzant atenent a les valoracions del pacients.

Resumen

En este proyecto se pretende desarrollar una aplicación de realidad virtual para mejorar los tratamientos actuales de pacientes que sufren algún tipo de enfermedad degenerativa a nivel cognitivo. Esto es posible gracias a lo que ofrece esta tecnología, que ha tenido un notable crecimiento en los últimos años, y que ofrece al paciente una posibilidad de inmersión a la vez que una experiencia sensorial muy alta. El hecho que el paciente esté trabajando en un entorno de realidad virtual hace que no tenga que estar pendiente de lo que pasa en el mundo real. La tecnología necesaria para el desarrollo de esta aplicación ha estado proporcionada por la empresa "Vysion360".

Este proyecto es la continuación de varios de los proyectos de final de carrera anteriores donde se ha desarrollado una aplicación de realidad virtual ambientada en un supermercado, gracias al motor de juego "Unity". Ésta contiene distintos modos de juego que permiten la obtención de datos y seguir el progreso experimentado por cada paciente. En este proyecto, el objetivo principal ha sido la reducción del peso de la aplicación, a la vez que mejorar el rendimiento de ésta y hacer un supermercado más realista y uniforme. Todo esto ayuda a la inmersión y confort del paciente.

A lo largo de esta memoria se han detallado los pasos a seguir a la hora de desarrollar la aplicación, las herramientas utilizadas, la metodología, el funcionamiento de cada modo de juego, la explicación de los controles a utilizar, las mejoras introducidas y las valoraciones de los pacientes al probar la aplicación.

Para acabar, la colaboración establecida con el *Hospital Clínic* permitió llevar a la práctica esta aplicación en proyectos anteriores. En este proyecto, además, la aplicación se ha probado con varios pacientes en la *Associació de Disminuïts Físics d'Osona*, y que nos ha permitido identificar algunas de las líneas futuras que debe seguir la aplicación de manera que siga desarrollándose y optimizándose atendiendo a las valoraciones de los pacientes.

Abstract

This project pretends to develop a virtual reality application to improve actual treatments for patients who suffer from a cognitive degenerative disease. This is possible thanks to what this technology offers to us, and which have had a "boom" in recent years, offering the patient to have a fully immersion and high sensorial experience. The fact that patient works in a virtual environment makes them to not be aware of what happens in real world. The required technology for the development of this application has been offered by "Vysion360" company.

This project is the continuation of several end-of-degree projects in which a virtual application set in a supermarket has been developed, thanks to "unity" software. This VR application contains different game modes that allow data collection and to follow patients' progress. In this project, the main goal was to reduce application's weight, as well as improve its performance and make a more realistic and uniform supermarket. All what is explained before help patient's immersion and comfort.

Throughout this project's memory, all steps done while developing the application has been explained, as well as the tools we had, the methodology followed, how does each game works, an explanation on game controls, the improvements made and patients' assessments.

At last, previous projects had collaboration with *Hospital Clínic* which allowed the application to be tested with patients. In this project, the VR app has been tested with several patients from *Associació de Disminuïts Físics d'Osona*, and that allowed us to take into account some of their appreciations to be the future lines that the application has to follow in order to be developed and optimized.

Acknowledgements

During the realization of this project have been some people that, in different degree, have helped or have collaborated with us to carry it out. In this part of the project we want to thank all of them because without their help would be much more difficult.

First of all, we want to thank the directors of this project. Jordi Torner and Francesc Alpiste. They always had a moment to answer any question we had, to give us any tool we need to carry out the project or to organize meetings with real patients.

We also want to thank all of the members of Associació de Disminuits Físics d'Osona (ADFO) for letting us test our application inside their installations and, of course, all the patients that came to the test with lots of motivation and made our visit to the association very comfortable and productive.

Thanks to the company "Visyon 360" for providing the HTC Vive glasses and the powerful computer to create, develop and run the application with any difficulties. We also want to thank them for letting us contact with them if we had any doubt and showing us their showroom sited in Barcelona.

We want to thank the student of the previous project for helping us to understand the application, solve some doubts and give us some advices.

Finally, we want to thank our families and friends for the support they have given through the construction of this project because they know the effort we have made more than others.

Glossary

- **AD:** Acronym for Alzheimer's Disease.
- **ADAS-Cog:** Alzheimer's Disease Assessment Scale-Cognitive Subscale, is a scale of evaluation of Alzheimer's disease.
- **ADFO:** Acronym for Associació de Disminuïts Físics d'Osona.
- **App:** Abbreviation for application.
- **Asset:** In Unity software context, is the representation of any item that can be used in a Unity project, such as images, audios or any other files that the program supports.
- **Bug:** When an error occurs in videogame's environment.
- **C#:** Programming language used in Unity.
- **Game object:** Fundamental objects in Unity's software that represent characters, props and sceneries.
- **HMD:** Acronym for Head-mounted display.
- **Lag:** In a computer game, it is the delay between an action that the user does and the game's reaction.
- **LCA:** Acronym for Life cycle assessment.
- **MCI:** Acronym for mild cognitive impairment.
- **MD:** Acronym for mild dementia.
- **Prefab:** In Unity's software, is a reusable asset that allows you to create, configure and store a complete game object with all its components and properties.
- **Scene:** The scenes, in Unity's software context, contain the environments and menus of the game. Each scene is a unique level.
- **Script:** A script is where all the code is written. In Unity, once it is saved in the project, it appears as a new asset that can be configured.
- **Unity:** It's a popular computer software for the development of 3D video games.
- **UPC:** Acronym for Universitat Politècnica de Catalunya.
- **VR:** Acronym for Virtual reality.

Index

Figures index

Tables Index

1. Preface

1.1. Project backgrounds

This project pretends to change and modernize current methods that are used for training people with mild cognitive problems by the creation of a virtual reality application (VR app). This Virtual Reality application will not be the cure for the disease, but it turned out to be a good workout to decelerate the process of the increasing effects that appear during the evolution of the illness. Because of that, teachers Jordi Torner and Francesc Alpiste, in cooperation with Alzheimer's unit of Hospital Clínic of Barcelona, decided to carry out this project.

This application has been designed and developed because of the different end-of-degree projects that were carried out by several students. Each of the projects lasts an entire semester, and these students that contribute to the design and development of it were Adrian Mora, Jesús Maria, Sergio Sanjorge, Óscar Ribas, Isarn Andanuy, Xavier Riera, Miquel Pedragosa and Rubén Tetuá, who was also involved in this project.

It is known that research studies say that this kind of exercises, in which the patient is immersed in the activity, considerably improve the results in comparison with current treatments. So, the development of this application pretends to be very beneficial for people with mild cognitive problems.

1.2. Motivation

1.2.1. Medical motivation

With the soar of virtual reality in recent years, it was necessary to update cognitive training exercises by using this technology, as it allows patients to be immersed in an activity that takes place in a different location.

As this was not possible to do it with the old cognitive exercises, it was necessary to introduce these modern techniques. As a result, they have had a great impact in cognitive training field, as several research studies explain and demonstrate that virtual reality is a very useful tool to face dementia or cognitive problems. Because of that, professionals from Hospital Clínic in Barcelona decided to talk with UPC to create and develop a VR application that can be used for their patients.

Not only Hospital Clínic was interested in testing and developing this application. In fact, an association located in Vic, ADFO, has also been interested in the application developed throughout this project and it was tested there at the end of May.

As it was said before, the design and development of this application does not mean the end of the disease. The main focus is to help reducing the effects of the disease for a while and so, improve patient's quality life.

1.2.2. Personal motivation

As it was said before, virtual reality has been growing in recent years and will continue to do it as it will probably be generalized in the near future. It is very useful for us to learn about it and work with this technology, and with important Virtual Reality companies such as “Vysion360”.

Also, and the most important part, is that we knew that this project could help in the future to improve some people’s quality of life, while this technology changes the way cognitive training has been done. This is a unique feeling.

Another point is that we knew we would be collaborating with “Hospital Clínic” or other Associations which would allow us to test the project and face real problems that would have been faced in the future.

And finally, to do this project in English was also an opportunity to learn more of it.

For all these reasons, we chose this project as the end-of-degree one.

1.3. Previous knowledge

In order to develop the VR application of this project it was necessary some previous knowledge about:

- **Body representation knowledge in 2D and 3D.** Vital for creating and developing any virtual reality application.
- **Unity software.** Thanks to several tutorials that were seen, the knowledge required to continue developing the application was successfully learnt: how to create objects and prefabs, how to create textures and modify scene resolutions, how to download objects from the asset store, what cameras and controls exist, etc.
- **C# language.** As Unity scripts work with C# language, several tutorials were seen to learn how it works.
- **Current status of the application.** In order to continue developing the application, it was necessary to analyze the steps taken before and identify what things should be done to improve it.

2. Introduction

2.1. Project goals

2.1.1. Medical goals

The main goal of this project is to create a virtual reality application that can help patients who suffer degenerative mental illness to step up their abilities and, if possible, slow down this degenerative process of the disease by improving their quality of life. It is important to say that user can only decelerate the process by therapy, drugs or using virtual reality, as cognitive problems do not have a cure yet.

VR technology allows the immersion of the patients in a virtual world where they can work out their mental abilities by doing different exercises in a comfortable environment without any danger.

2.1.2. Project goals

Before starting a project, it is very important to set the main goals of it. The idea is to be able to complete them throughout the project and, at the end, see if we achieved all of them. This also helps in the organization of the project. Our objectives are:

- Learn about VR technology.
- Learn about mild cognitive impairment treatments.
- Learn how to use Unity software.
- Learn how to program in C# language.
- Reduce application size on disk by working on all assets of every scene.
- Reducing time of execution when starting the app by adjusting textures, resolutions, etc.
- Leave the application uniform, with all products ready to be used, without bugs, realistic and as intuitive as possible for the patient.
- Leave the application ready to be tested with real patients as well as test it with them.
- Find ways to improve the VR app in the future.
- Find if it is possible to run it on a portable computer.

2.2. Scope of the project

The main goal of this project is the development of a VR application applied to cognitive training for patients suffering a mild cognitive impairment. The application includes several games in which there are different levels of difficulty, so the doctor has several ways to evaluate the patient. In addition to that, the VR app has also a database in which all successes and errors are stored while the patient is using it. The results can be consulted by the doctor. With this data collected, the doctor can follow patient's degenerative disease and decide whether one level or another is the most suitable one.

Focusing on the patient, the application must be as intuitive as possible, so all audios, signs, texts or game controls have to be very simple and easy to use. Also, all objects must be easy to identify.

It is important that the patient stays comfortable through all the activity and keeps motivated when doing it. That is why the application has to be as realistic as possible and create a great immersion in the virtual environment.

3. State of art

3.1. Virtual Reality

3.1.1. Definition

Virtual reality is a realistic three-dimensional image or artificial environment that is created with a mixture of interactive hardware and software, and is presented to the user in a way it is accepted as a real environment, giving the possibility of creating an experience not possible in our physical reality. User has total immersion in this virtual world, which means that the sensory experience feels so real, that player forgets it is a virtual-artificial environment and begin to interact with it as we would naturally in real world as well as the user obtains answers from the program. So, the player it is not just a spectator.

So, VR is the most immersive type of reality technology that can convince the human brain that it is somewhere it is really not.

3.1.2. History

Virtual Reality is not a concept created on the early ages, it started before the 50s of the last century, first with illustrations and texts referred to an alternative reality and then with machines that simulated a costumer's journey to unknown worlds. The technique limited these experiences a lot and it is not comparable with actual systems.

In 1930, the first flight simulator was created in order to train inexperienced soldiers to pilot the latest models of aircraft, since otherwise it would not turn out well either of the two parts. This aroused the interest of researchers and designers to develop more and more simulators increasing the quality and interaction capacity.

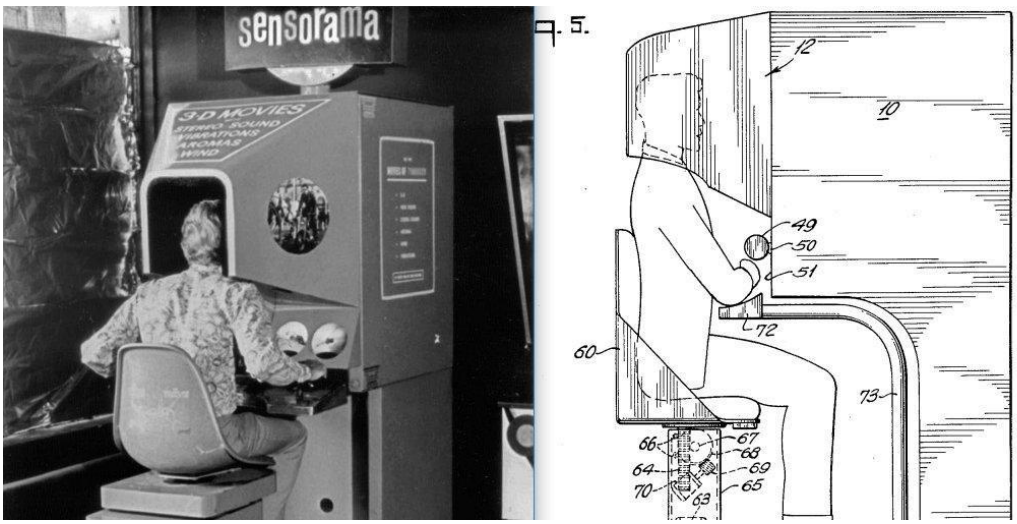


Figure 3.1. The simulator Sensorama (Source: Google Images)

In 1950, Morton Heilig developed a simulator called Sensorama. He saw the theatre as an activity that could encompass all the senses in an effective way, thus preparing the viewer for the activity on the screen. The viewer could visualize 5 small movies while simulating the other senses. He called it "Theatrical Experience". This simulator combined 3D images along with sound, wind and smells to create an illusion of reality.

The first truly virtual reality helmet would arrive in 1961. It was built by Corneau and Bryan, employees of the company Philco Corporation. The helmet, called "Headsight", incorporates a screen and has a position control of the head. The project was used for military training. Although the element was not connected to a computer, "Headsight" pioneered the practice of leveraging virtual reality technology and training purpose.

In 1965, Ivan Sutherland described the concept of virtual reality with the publication of an article called "The Ultimate Display" where he explained the concept that scientists had been trying to establish for years. A couple of years later, he and his team at MIT (Massachusetts Institute of Technology) developed a virtual reality device which consists of a helmet attached to a computer.

The helmet and computers used at that time are so big and heavy that the helmet is hung from the ceiling, earning the device the nickname of "Sword of Damocles".

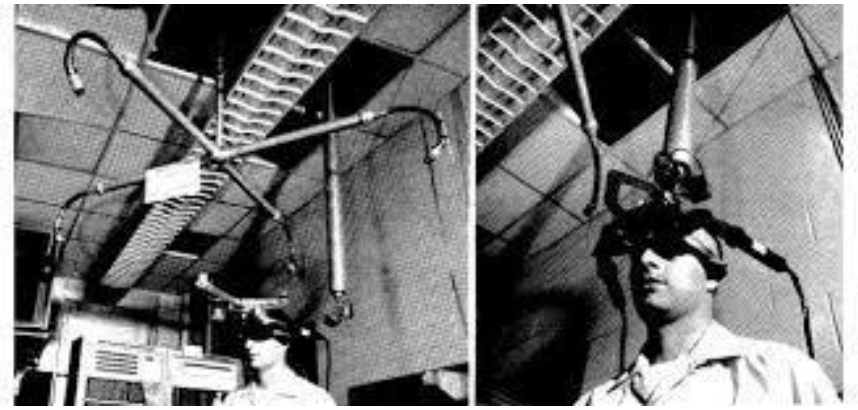


Figure 3.2. The "Sword of Damocles" (Source: Google Images)

In 1982, Jaron Lanier develops the Data Gloves, gloves with sensors capable of recognizing the movements and position of the fingers. This same year the company SEGA presents the first video game on the market with stereoscopic image, the SubRoc-3D, with some glasses and recreational machine.

It was the NASA, in 1984, who first created a low cost virtual reality glasses called VIVED (Virtual Visual Display System). It was developed by a research group led by Michael McGreevy. With these glasses demonstrations were made oriented to the industry and the university.

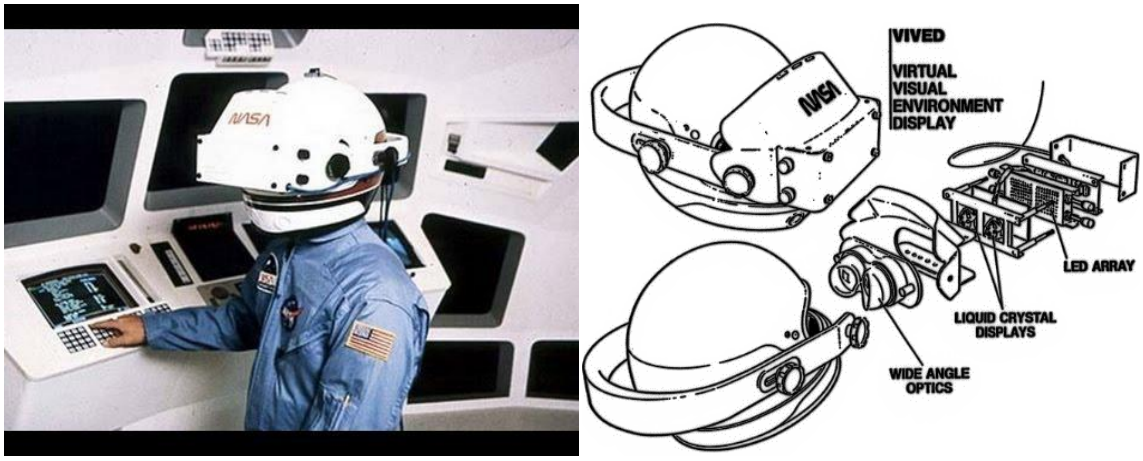


Figure 3.3. The VR glasses VIVED (Source: Google Images)

In the 90s, the first commercial launches on Virtual Reality were seen. In 1991, SEGA announced the Sega VR for arcade video games and the Mega Drive console. It used the LCD screen in the viewfinder, stereo headphones and displacement sensors that reacted to the user's head movements. The model was exposed in several videogames fairs but it never got commercialized due to different problems.

Nintendo, an expert company in the field of innovation, launched "Virtual Boy" in 1995. It was a pair of glasses that used a projector inside to show monochrome 3D through a stereoscopic effect. Although it reached the market with great expectation due to the rise of virtual reality, the console was a true commercial failure, so much that it was not even commercialized in Europe and many consider it one of the worst consoles that Nintendo has always created since its inception.

Figure 3.4. The glasses "Virtual Boy" (Source: Google Images)

Palmer Luckey, in 2010, tries to recover almost 20 years after failed projects the idea of a helmet/ virtual reality glasses. Luckey start developing a device that will be the prelude of the Oculus Rift. He made a campaign to get the necessary funding and the campaign was a success. Later, Facebook made a large outlay of 2.000 million dollars and bought the whole project and the Oculus Company. After the birth of Oculus, the big companies from different areas of the technological world started a career and began to develop prototypes of virtual reality glasses. In April 2016, HTC ordered the first HTC VIVE Steam VR helmet units, marking the first commercial launch of a virtual reality system with sensor-based position tracking, allowing free movement of users within a space definite.

3.1.3. Virtual reality types

Virtual reality's main goal is to immerse the user in a virtual environment. However, there are different types and the differences between them are explained below:

- **Non-immersive reality.** Virtual reality experiences where the user does not experience the sense of being in a virtual environment (VE), as it is viewed through desktop or laptop screens. An avatar represents the user and can interact with VE by keyboards, computer mouse or other devices.

- **Semi-immersive reality.** Allows the users to experience virtual environments while being aware of the real world surroundings as well as keeping control over physical objects. However, the basic goal is providing users with a maximal sense of presence through special hardware and high-end software that renders a digital 3D image.

- **Fully immersive reality.** The user is totally immersed in the virtual world by using glasses, headsets or other devices. The user can explore and interact in a virtual environment as if it was the reality. In this case, it is not possible to be aware of the real world outside the virtual environment.

3.2. Possible applications of virtual reality

The Virtual Reality was highly developed in the field of video games and entertainment in general, but due to its almost infinite possibilities has spread to fields and sectors very varied. Here are some of the applications that VR has:

- Medicine: In the medicine field, the uses of VR can be multiple. Currently, it is the sector where their advances are bigger and most effective. Within medicine, the most representative applications are carried out in the following areas: simulations for medical training, treatment of psychological traumas, surgery operations and pain management through distractions.
- Military: VR allows to train military professionals in a virtual environment where they can improve their skills and abilities without the risks of training in a battlefield or using real

vehicles or weapons. This environment generated by computer applications is also used in flight simulation for the air army where people train to be pilots. In the same way, virtual driving simulations are used to train trucks, tanks and all kinds of vehicle drivers.

- Leisure and entertainment: Is in the field of leisure and entertainment where Virtual Reality applications are more developed thanks mainly to the video game industry. The industry moves millions of Euros and their public is willing to spend large amounts of money to get all the peripherals to enjoy a total immersion in their favorite games. The immersion and the interactivity with the environment generated are virtually clear differences with regard to traditional video games.
- Architecture: The great advantage of virtual reality in architecture is that through a virtual reality viewer it is possible to immerse in the projects created. It is very helpful because you can perceive proportions or sizes in a realistic way. The tendency to apply for projects in VR is beginning to grow because when customers put on the glasses they understand the space as they were there.
- Education: The possibilities of virtual reality in the field of education are endless and bring many advantages to students of all ages. The technology has enormous potential in education since it is proven that students process content better when there is a motivational component and virtual reality applications have all the attractiveness and tools to achieve it.

On the next image we can see a prediction of the sectors that the VR applications will be more important in the future:

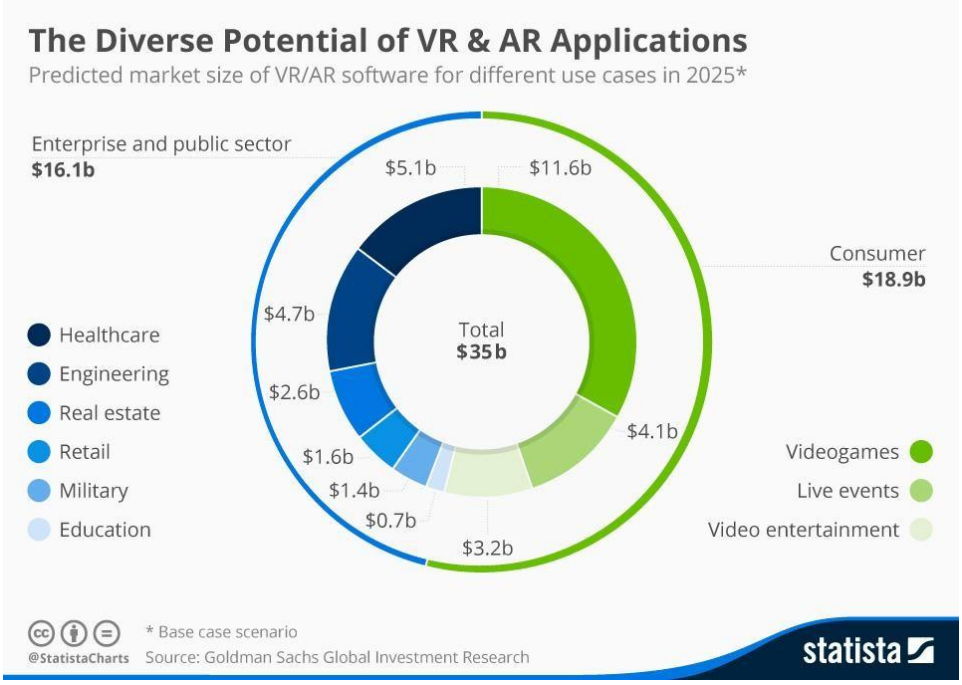


Figure 3.5. The predicted market of VR in 2025 (Source: Statista)

4. Current situation on mild cognitive impairment treatment

Thanks to the evolution of medicine and the multiple studies carried out today, life expectancy has increased considerably over the years. People live longer, and that is one of the reasons why the number of people with dementia has increased during the past decades and is increasing continuously and the main risk factor is old age (Winblad et al., 2016). In fact, studies show that the numbers of people affected by dementia almost double every twenty years and, by 2050, it would reach 131.5 million people (Prince, 2013). Alzheimer's disease (AD) is the most common cause of dementia, as it accounts for an estimated 60% to 80% of cases diagnosed (Valladares-Rodriguez et al., 2018).

There exists a transitional phase between normal aging and dementia. First stage of this transition is the mild cognitive impairment (MCI), in which the subjects start experiencing subtle cognitive deficits with activities of daily living (Petersen and Morris, 2003) as well as response inhibition, cognitive flexibility, and attention switching (Yeh, Chen, Tsai, and Rizzo, 2012). Medical research is focused on early diagnose AD, so it is recommended to track all the MCI process to detect problems early. However, there is not exist any specific treatment for this disease yet. Currently, there are two types of treatment to try to prevent AD:

- Pharmacological method (using drugs)
- Therapies and activities (Individual or group therapies)

It is important to say that no existing treatment can cure this type of disease. The idea is to slow down the process of the different problems that appear in the transition from MCI to AD (Requena et al., 2006). The first mentioned method is the pharmacological method. This method enhances and improves cognitive abilities but there are a lot of factors that can affect it such as the age, the sex of the person, the psychological state of the subject, etc. In addition, there is the possibility that various side effects appear, so it could become counterproductive. That is the reason why the second method, activities and group or individual therapies, is more common used (Sitzer, Twamley and Jeste, 2006) despite their lower efficacy, as shown in several studies (Baldelli et al., 1993; Clare and Woods, 2003). These results could be due to the fact that this method can be boring and stressful for the subjects.

In this context, videogames and virtual reality (VR) have been gaining strength thanks to different studies in which they are analyzed as a new treatment for MCI. As it has been seen in the previous section of the project, the steadily evolution of technology has made both VR and video games being currently used in different fields at a professional level such as architecture, engineering or as in this case, medicine.

Several clinical studies reflect positive results when using video games or VR in therapies for the treatment of MCI (Fernández-Calvo et al., 2011; Hill et al., 2016; Lampit, Hallock and Valenzuela, 2014; Optale et al., 2010; Valladares-Rodriguez et al., 2018). Thanks to this new treatment

method, cognitive skills are trained at the same time as the subject is having fun (Jensen and Konradsen, 2018; Rosenthal et al., 2008). All this, together with the fact that it can become a relaxing activity, makes this method the most suitable for patients with MCI.

In the following figure it can be seen the results of one the treatments with the videogame "Big Brain Academy" done in the study "*Eficacia del entrenamiento cognitivo basado en nuevas tecnologías en pacientes con demencia tipo Alzheimer*" (Fernández-Calvo et al., 2011), in English: "Efficacy of cognitive training based on new technologies in patients with Alzheimer's dementia".

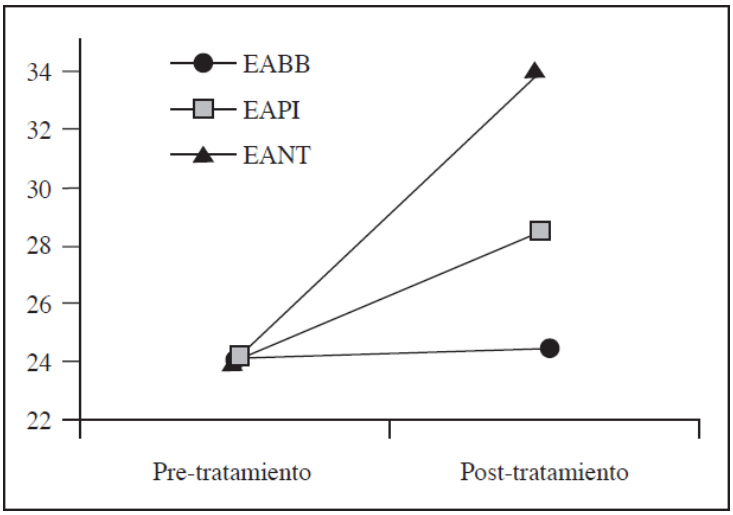


Figure 4.1. Evolution of MCI patients in a test with different treatments (Source: Fernández-Calvo et al., 2011)

The variables of the graphic correspond to:

- **EABB:** Group with the new treatment of videogames.
- **EAPI:** Group with the traditional therapy.
- **EANT:** Group without treatment.

As can be seen in the figure, subjects without treatment (EANT) follow the fast process of loss of cognitive faculties once the treatment has been completed. The subjects who follow the traditional treatment through activities and therapies (EAPI) show a better evolution than the previous ones since the slope of their line is smaller, so they lose less cognitive faculties. Finally, as can be clearly seen, the subjects submitted to the new treatment of video games (EABB), after treatment have barely lost cognitive faculties.

In conclusion, this study showed that videogames provide beneficial results for the prevention of dementia thanks to the differential factors that these can bring. That is why, even better results with the VR are expected, since the possibility of immersion takes the game to another level. In addition, considering the increasing availability of low-cost VR devices and the increasing need for self-administered and personalized rehabilitation approaches, VR applications are expected to play an important role in improving rehabilitation outcomes (Cao, 2016; Rizzo, 2017).

5. Definition of the project

5.1. Scale of the project

This project is the continuation of a fellow-up of previous projects based on the design of a VR application applied to cognitive training. This application has been developed in order to offer the maximum immersion of the user and to feel as if they were inside a supermarket doing daily activities. Because of that, during the project we were focused on the following objectives:

- Reduce application's weight by converting all products from models to prefabs. Prefab system allows the developer to create and configure a GameObject complete with all its components, properties, etc… as a reusable asset.
- Leave the application in the most uniform way possible, that is: make the inside of the supermarket to be the same in all games and levels, as until now the vegetables and fruits section was different depending on the game.
- Make the game more realistic by making all products grabbable, with gravity and redoing some of colliders (spheres and capsules) so that when the app is running and the gravity effect is over the products, fruits and vegetables sections now look better.
- Trying to reduce the start time of the game when the user runs the application by having a look at textures and resolution of the products and all elements of the scenes.
- Organize all elements in an intuitive way by creating new folders and deleting all the elements that are or will not be used.
- Test the application and verify that everything works well and clearly so that it is ready to be tested with patients.
- Test the application with patients.

These is what make up the whole project. However, the main goal is to make the application lighter, optimized and realistic, so that the patient does not feel bored when doing the exercise because of lag or anything else.

Throughout the entire project, meetings with project coordinators have been organized in order to arrange what improvements in the application had to be made. Also, patients' appreciations were taken into account for the future lines of this project, as the test was done some days before the delivery of this project.

The most important thing is that patients do not feel as if they were examined, so they must be comfortable and fully immerse in the virtual environment. Everything what has been done in this project is to achieve that.

5.2. Planning and methodology

Our planning to achieve the previous goals in this project was divided into different points:

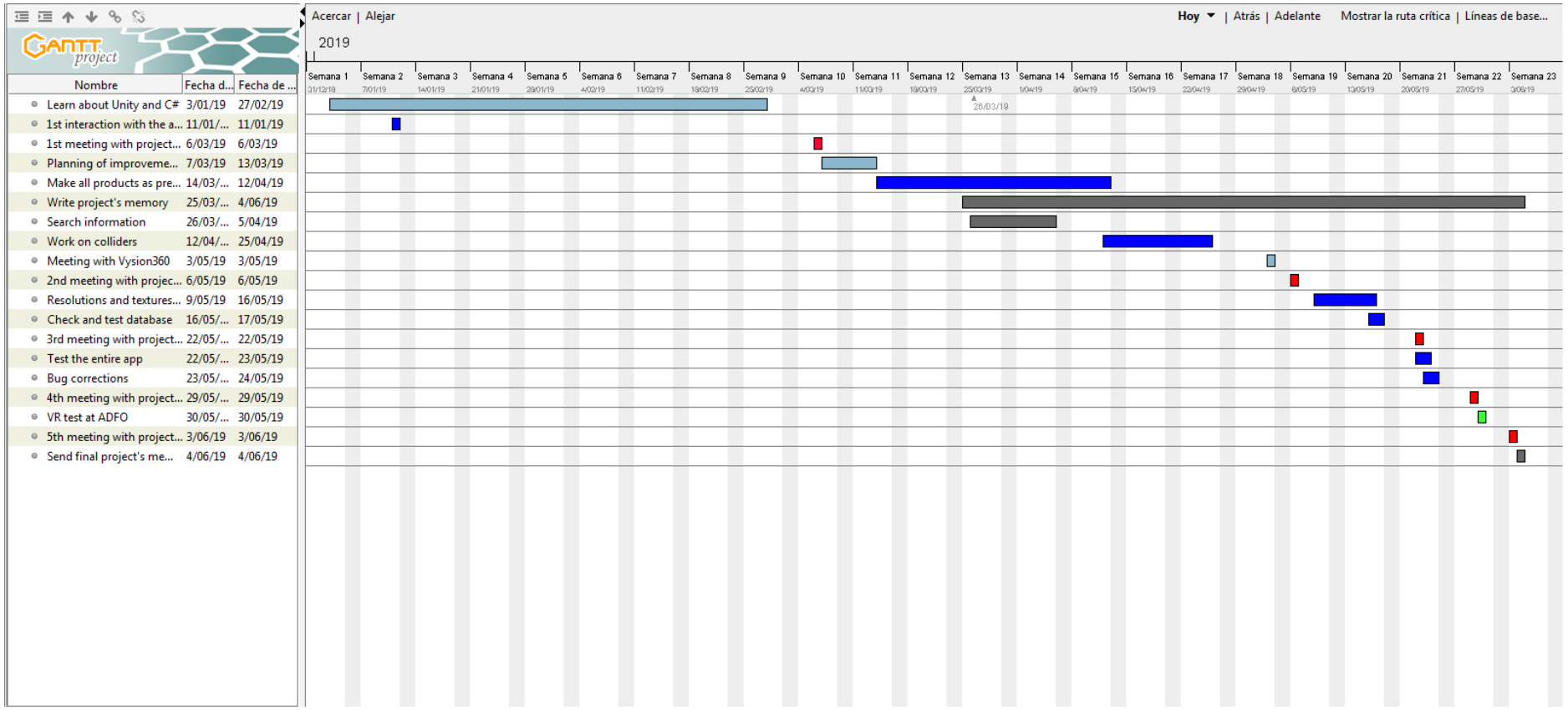
- Learn about "Unity" and its interface.
- Search information about cognitive treatments and virtual reality.
- Meetings (with coordinators, "Vysion360", app test in ADFO).
- Write project's memory.
- Plan the improvements for this project.
- Do all products as prefabs.
- Change resolutions and textures.
- Update colliders.
- Organize and clean the application.
- Test the application with ADFO patients.

Before starting the project, some tutorials were seen to learn about "Unity" and all its interface. Once this was achieved, we tried the application which we would have to develop and so, we got used to it.

After all "secrets" of the application were known, and with the agreement of the project coordinators, it was the time to start the development of the application. So, all the products were converted from models to prefabs. That was a major change and, after each of the updates, the game was tested to see if some bugs had to be fixed or the application was working well. Once the previous part was completed, it was the time to do the other changes.

Finally, a test was done in ADFO. It is a shame that patients' suggestions for improving the application cannot be done in this project because the short period of time between the test and the delivery.

5.3. GANTT diagram



Task name	Start date	Finish date
Learn about Unity and C#	03/01/2019	05/03/2019
1st interaction with the app	11/01/2019	11/01/2019
1st meeting with project coordinators	06/03/2019	06/03/2019
Planning of improvements	07/03/2019	13/03/2019
Make all products as prefabs	14/03/2019	12/04/2019
Write project's memory	25/03/2019	03/06/2019
Search information	26/03/2019	05/04/2019
Work on colliders	12/04/2019	25/04/2019
Meeting with "Vysion360"	03/05/2019	03/05/2019
2nd meeting with project coordinators	06/05/2019	06/05/2019
Resolution and texture improvements	09/05/2019	16/05/2019
Check and test database	16/05/2019	23/05/2019
3rd meeting with project coordinators	22/05/2019	22/05/2019
Test the entire app	22/05/2019	23/05/2019
Bug corrections	23/05/2019	24/05/2019
4th meeting with project coordinators	29/05/2019	29/05/2019
VR test in ADFO	30/05/2019	30/05/2019
5th meeting with project coordinators	03/06/2019	03/06/2019
Send final project's memory	04/06/2019	04/06/2019

5.4. Application requirements

If the developer wants to make an application as useful and optimal as possible, some requirements must be established at the start of the project. They have to take into account from software to quality attributes. It has to be said that the developer has to be sure all works properly because this application will be used by patients, so all has to be focused on patients' experience.

These requirements can be found below:

- Application fully compatible with VR HTC Vive glasses and controls.
- Scripts with a robust, fail-safe structure.
- Use ID at the start of the application to identify the patient.
- Data collection in a database. (Consulted only by the doctor)
- No lags when patient is in the game.
- Use prefabs to improve applications' performance and reduce application weight.
- Easy and intuitive controls to move throughout the game.
- Uniformity in supermarket's environment.
- Keep all menus and exit scenes with the same interface.
- Improve application's performance when a game starts.
- Remove all useless elements (such as products that are not used, etc.)
- Final version with no bugs.

6. Chosen technology

6.1. Hardware

To create and run a VR application we need to know the special hardware which is necessary to run a virtual reality app and have a full experience.

The most important hardware is the virtual reality glasses, used to improve the immersion into the game. It is also very common the use of special commands to able the ability to grab objects or shoot.

Currently, this type of technology is more advanced and modern, making it cheaper and even more realistic. Nowadays there are lots of glasses in the market, with a big amount of prices and every day with higher resolution.

For this reason, we can find different virtual reality glasses to choose depending on our necessities and our computer or video game console.

In our case, we did not have to choose the technology based in what we were working because this project is the continuation of others. During the project we have used the VR HTC VIVE glasses for the application.

As we have said previously, in the market there are different types of virtual reality glasses. Therefore, the next market study shows the characteristics of the different hardware:

The Google Cardboard is a virtual reality platform developed by Google made of folding cardboard that creates a mounting to put a smart mobile phone with Android or IOS. If you move your head, the game also moves, creating the illusion of immersion. The lenses give the feeling of depth. The fields of vision for the eyes are delimited by a strip of cardboard in the center of the glasses. The crystals create a magnifying effect, so it is quite important that the phone has a rather high concentration of pixels per inch to use Google Cardboard.

Its graphics depends on the mobile phone and immersion is not very sophisticated at all however, it has some advantages over its competition. Google Cardboard is impressively cheap. This low price helps spreading the concept of virtual reality among the population. Secondly, it is very easy to use and you always have the screen (your telephone). You only need to put your device into the cardboard mounting. Google Cardboard also has more than 1000 applications in Play Store.

Figure 6.1. Google Cardboard (Source: Google Images)

The Google Cardboard is a very simple and cheap hardware, it only cost 13€, and it does not have any controllers or movement sensors. These factors make the Google Cardboard an accessible hardware for everyone, even having a low immersion.

The Samsung Gear VR is wireless, lightweight and has a low price compared with other products available in the market nowadays. Its interface is intuitive and easy to use. It needs a smartphone to work and only some Samsung phones are compatibles with these glasses.

The virtual reality experience that it offers is not as sophisticated as the ones used on net systems of computer and video game consoles. The field of view is only of 96 degrees. However, the ecosystem still lacks of quality and variety of content to really take advantage of it.

Figure 6.2. Samsung Gear VR (Source: Google Images)

The Samsung Gear VR has the same problem that the Google Cardboard, it only works in mobile platforms. It has a simple controller for playing and his structure is more robust. His price is 129€ and it has low immersion too.

This hardware was the one we planned to use if we could have developed the mobile application because is the most used in tutorials of Unity mobile apps. Despite being more developed than the Google cardboard is still very simple for our game and this could generate problems with polygons because all the products in the supermarket have big meshes and this will reduce the resolution and make it slower.

Despite these problems, this hardware is the most developed one for mobile applications and, until March, is impossible to import our game to mobile, when new hardware are released with

the kinect function included. This function will give us the opportunity to run our game in a mobile in a good resolution and speed.

The PlayStation VR (PS4) is more expensive, it cost 249€ but offer the user more immersion thanks to a camera that captures all the movement. Furthermore, it has special controllers to improve the game experience. However, it is limited because it only can be used in PlayStation platform.

Figure 6.3. The PlayStation VR (Source: Google Images)

The Razer OSVR HDK2 let the user use it in different platform without limitation and has a similar price that the PlayStation VR, about 210€ and offer more immersion. On the other hand, it does not have any controller so it is difficult to play with it.

Figure 6.4. The Razer OSVR HDK2 (Source: Google Images)

The Oculus Rift is a virtual reality helmet developed by Oculus VR. Rift's integrates 360 degrees spatial audio making immersion even higher than any other hardware named before. The graphics are better but a computer with minimum requirements is required to use it properly.

The Oculus Touch controls incorporate hand and finger movements as well as physical buttons like traditional game controllers. Motion tracking lacks the full room scale of the HTC Vive.

Figure 6.5. The Oculus Rift (Source: Google Images)

The Oculus Rift is considered one of the best virtual reality hardware. It has good immersion thanks to his movement sensors and let the user play with universal controllers from any video game console. It can be used in PC platform and his price is about 399€. However, Oculus Rift was developed to play sat down.

The HTC VIVE is also one of the best virtual reality glasses in the market. It has two laser towers that capture your movements and special controllers to make games and applications even more immersive. However, it needs high computer specifications and his price is higher than the others, it costs about 639€.

The HTC Vive is a virtual reality glasses manufactured by HTC and Valve. A powerful computer is needed for the correct functioning of HTC Vive. The device is designed to use space in a room and immerse in a virtual world where the user is allowed to walk and use drivers to give the user the possibility of interact with virtual objects. Hand tracking and physical movement makes the user experience a full immersion into virtual world.

The front camera allows the glasses to detect any object, static or moving, in the area. This function also becomes a security system preventing users from colliding with objects.

Figure 6.6. The HTC VIVE (Source: Google Images)

The HTC Vive Pro is the last virtual reality glasses released in the market earlier in 2018. These glasses are also developed by HTC and Valve as the previous ones we have talked about. The HTC Vive Pro uses the same technology as the HTC Vive but with several changes. As we said before, it need high PC specifications and has an elevated price (1020€).

This hardware improves resolution, includes headphones with noise cancelling, improves comfort, includes an advice if we are close to a physic object thanks to the Chaperone system and does not include wires what makes it completely wireless allowing the user to have more immersion.

Figure 6.7. The HTC VIVE Pro (Source: Google Images)

6.2. Engine development

As we said previously, this project is the continuation of others. This is the reason why the engine development was already chosen.

We have used Unity3D, but there are other engines like Game Maker or Unreal Engine. In the following table we can compare the characteristics of these three engines:

Properties	Unity3D	Game Maker	Unreal Engine
Free version	YES	NO	YES
Compatible with windows	YES	YES	YES
Compatible with HTC VIVE	YES	NO	YES
3D design program integration	YES	NO	NO
3D support	YES	NO	YES
VR support	YES	NO	YES
VRTK (virtual reality toolkit)	YES	NO	NO
Help forums	YES	YES	YES

Table 1. Engine development

As you can see, the Unity3D engine is the best option to create a 3D application. It is compatible with windows and HTC VIVE, both needed to create the application. Also, using this engine we can contact with Visyon360 to obtain support.

The engine Game Maker is obviously discarded because it does not have compatibility with HTC VIVE and it is not a free engine.

On the other hand, Unity3D and Unreal Engine are both compatible and free engines, but Unity takes advantage because it has lots of repositories and documentation for learning and solve problem for beginning users.

Other important factor is the coding language, needed to create all the script of the application. While Game Maker uses C++, Unity3D use C# which is an easier language for beginner users.

Finally, all these factors make Unity3D the best option for our project because of his characteristics and the design of the application which is easier for users that never have created a 3D project.

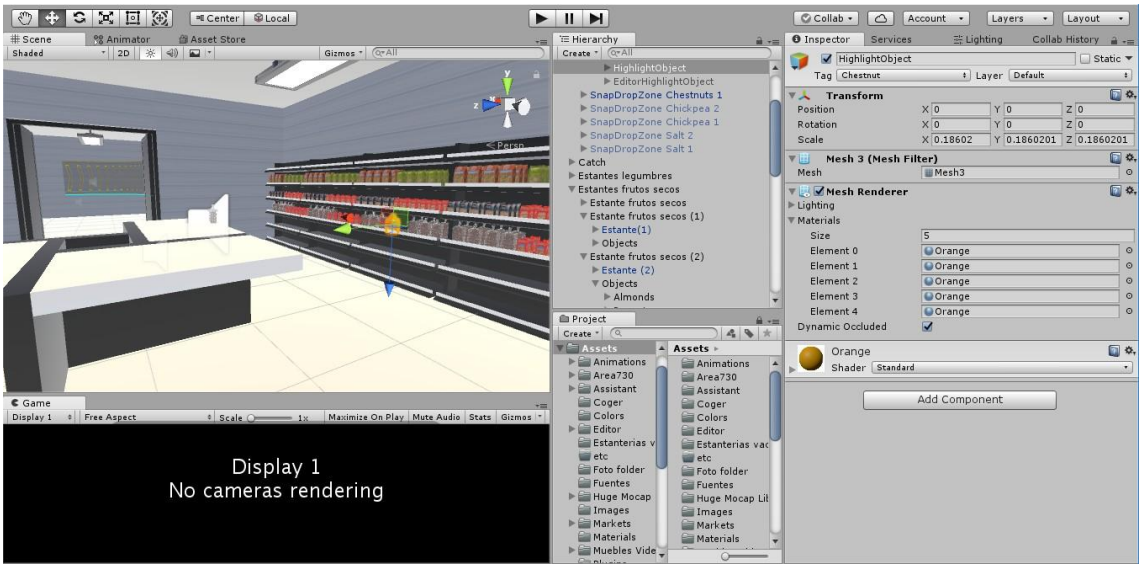


Figure 6.8. Unity interface (Source: Prepared by the authors)

6.3. Programming software

To carry out the programming of all the scripts needed to develop our application we decided to use the Microsoft Visual Studio software, although Unity uses by default the MonoDevelop we decided to work with Visual Studio.

Visual Studio is an integrated development environment (IDE) developed by Microsoft, it is a very versatile application with which you can edit almost any code and for different platforms such as iOS, Windows, Android or web, it has been decided to use this IDE because it is a more advanced development environment, offering aids that MonoDevelop does not offer as intelligent

autocomplete; it also offers syntax coloring help, code schematization or pop-up descriptions of the code element you select, among other things.

This programming software admits different coding languages like C, C++, C#, Python or others.

For this reason, we use Visual Studio because we can program in C# so, as we explained before, is easier to use for beginners.

6.4. Extra resources

This project needs to use other resources in addition to Unity, HTC VIVE or Visual Studio. Since the beginning of this project we have been using different tools to carry it out. All of them are explained in this part:

SKANECT: It is a scanning program that let us capture new textures and structures to do new objects in the application.

GANNT PROJECT: This application helps to organize and structure our objectives and goals during the time that we are doing the project. It also simplifies the fact of making a Gannt Diagram, very useful to help us to organize the work.

MYSQL: It is a database management program. We have used this database to save all data during the game to make easier to the doctor extract the results.

VOICE RECORDING: We use our mobile phones to record the audio used in the application. After recording the audio, we need to convert these types of files (MP4 format) to MP3 format.

7. Design

In order to get the sense of fully immersion in the supermarket environment, one of the main goals of these projects was to make the entire supermarket as realistic as possible. That is why the design of products and the structure of the supermarket are so important.

Most of the supermarket environment has been developed throughout previous end-of-degree projects, so it is important to mention the whole procedure that previous students had followed to understand application's development.

The different tools that have been used before to design the application are explained in the following sections.

7.1. "Unity tools"

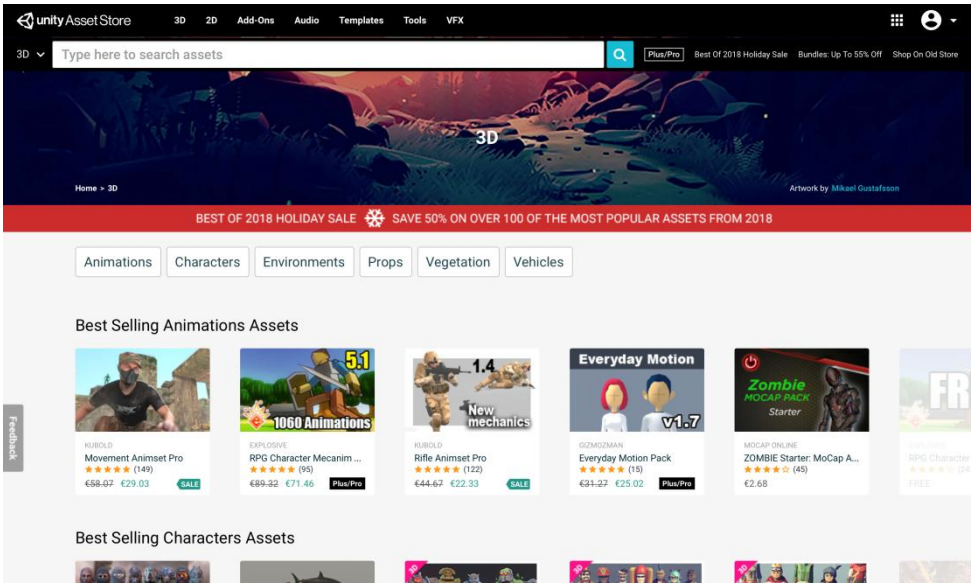


Figure 7.1. "Unity Asset Store" (Source: Prepared by the authors)

This program also allows the developers to create any of the desired objects from several basic forms that can be seen below:

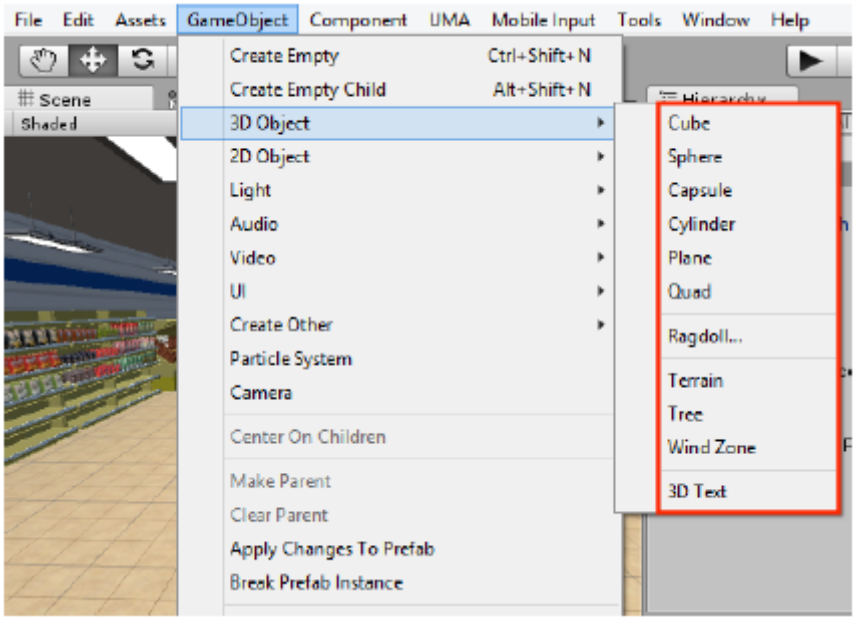


Figure 7.2. "3D Object" creation menu (Source: Prepared by the authors)

This procedure was followed, for example, in the previous project to create the automatic cashier, banknotes and coins. That was because these previous objects were not in the asset store or were not free.

Once this is created, the program allows the developers to add its color or image by creating materials to which the desired texture is assigned. This is how created objects seems real.

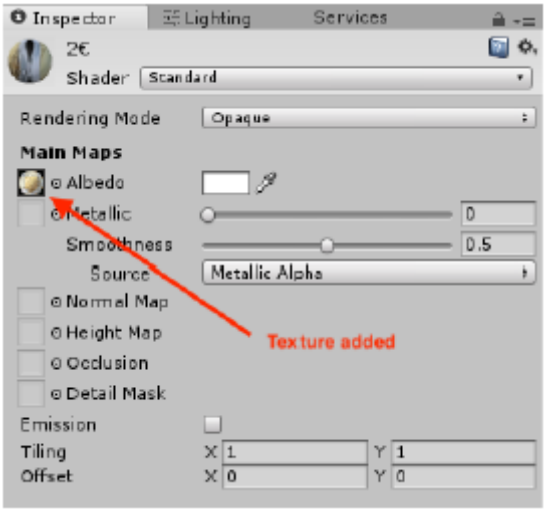


Figure 7.3. Material and texture for the 2€ coin (Source: Prepared by the authors)

7.2. “Paint”

“Paint” program was used in previous projects for the creation and modification of certain textures, as it is easy and intuitive to the user. This is because the main goal was to create the objects as realistic as possible.

When downloading an object from the “Unity Asset Store”, the texture sometimes was lost and it was necessary to create another one. This texture had to be correctly adapted to the shape of the project.

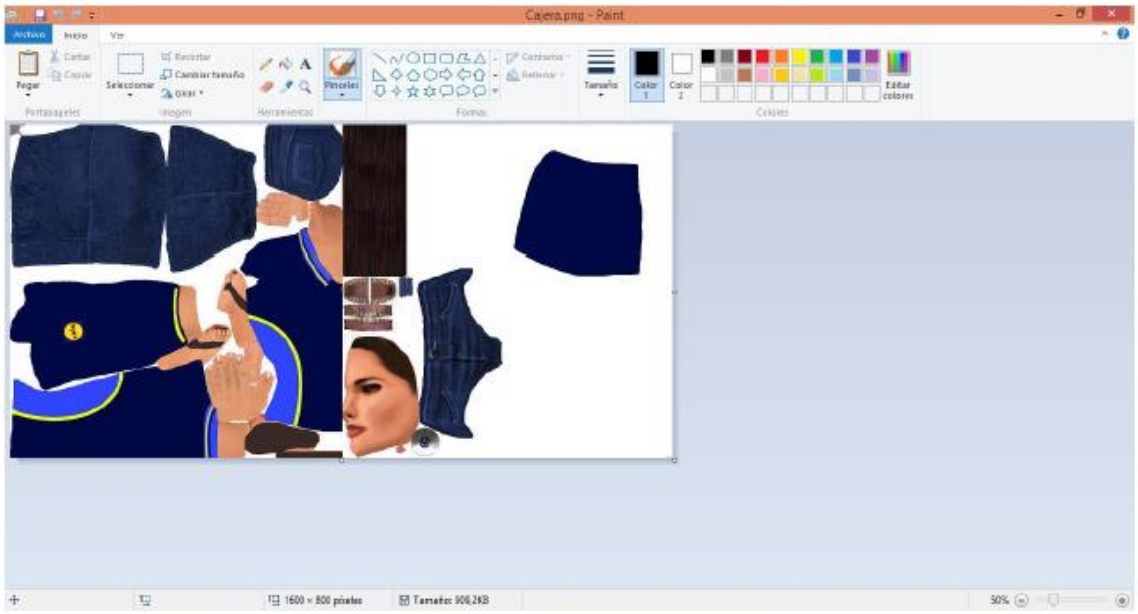


Figure 7.4. Texture for the supermarket cashier modified with “Paint” (Source: Prepared by the authors)

Many of the images for the textures were obtained from the internet and later modified with “Paint”.

7.3. Websites

Most of the assets were found in the “Unity Asset Store”. However, other assets were downloaded from other websites, as they were not available in “Unity Asset Store” or were not free.

In previous projects, several pages similar to the previous mentioned store were used: “3dwarehouse.sketchup.com” and “Archive3d.net”. Both of them allows the user to download and import to “Unity” the desired object.

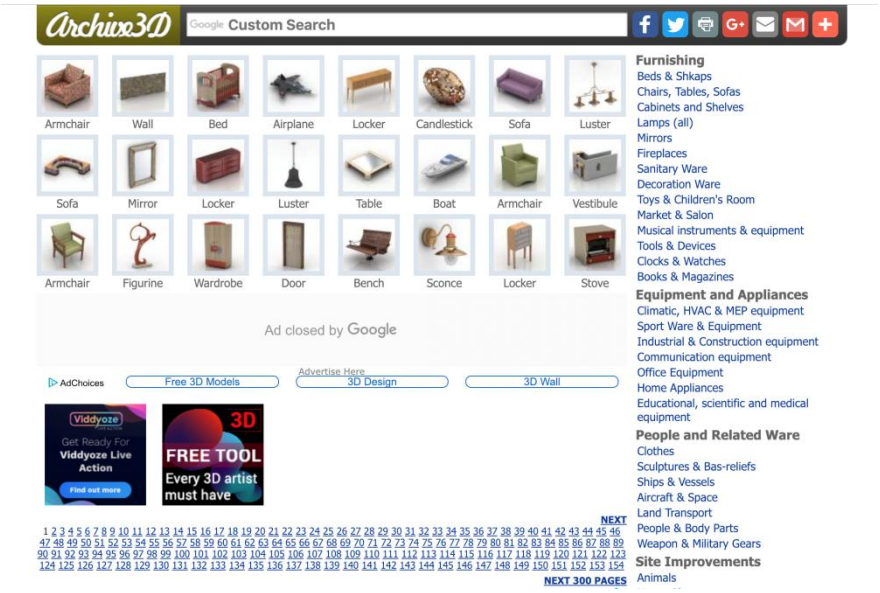


Figure 7.5. "Archive3d.net" website (Source: Prepared by the authors)

7.4. "Skanect"

It is a very useful program which uses a "Kinect" device (a motion sensing input device) where you can scan objects or people to create their mesh or shape, as well as its texture, and import it to "Unity". This is how all supermarket products' shapes were created.

Figure 7.6. ID Menu (Source: Prepared by the authors)

The initial videos of the tutorial, in which there were some instructions for the patient to move their head, were also recorded with "skanect", as it allows recording and saving movements. If an audio is supported with a video, the patient can learn easily what has to do.

Figure 7.7. Movement animation with "Skanect" (Source: Prepared by the authors)

8. VR Supermarket Experience

"VR Supermarket Experience" is the name of the application developed throughout this project and the previous ones. This development has been possible thanks to "Vysion360", Hospital Clínic, and ADFO, as in previous projects some students were able to test the app with patients in the mentioned hospital and, in this project, in Associació de Disminuïts Físics d'Osona.

The main goal of this application is to create a new way of cognitive training by immersing the patient into a virtual environment in which the patient works out doing a quotidian activity, forgetting about the real world while they are being examined by the doctors.

In this part of the project all the application is explained, from the first menu, where the doctor has to introduce the ID of the patient to collect the data, to the last game. Also, there is the explanation of the game controls.

8.1. Menus

The Menus are important in the application, as they distribute and organize it. It is a tool for the doctors to be controlled from a PC, as they are the ones who supervise the patients. That is why all menus are in 2D. The patient never sees the menus, is the doctor who chooses what activity the patient is going to perform.

8.1.1. ID Menu

The "identification menu" or "ID Menu" is the first menu of the application. The doctor will see on the computer screen what is observed in Figure 8.1.

Figure 8.1. ID Menu (Source: Prepared by the authors)

In "Escriba aquí" section, the doctor must enter a number that will be the identification number of the patient who will proceed to train. This was decided in order to maintain patient's privacy. By entering this ID, all data of the patient is identified and stored in the database.

While the doctor is entering the ID number, the patient, with VR HTC Vive Glasses, is seeing and scene where the shop assistant in the supermarket adding fruits to the fruit shelf.

Once the doctor has written the identification number, the doctor must press "Siguiente" button to proceed to the Main Menu. If, otherwise, the doctor presses "Salir" button, the menu and the application will be closed.

Figure 8.2. ID Menu waiting scene by the patient (Source: Prepared by the authors)

8.1.2. Main menu

Once the doctor has clicked the "Siguiente" button in the previous menu, the "Main Menu" appears in doctor's computer screen.

In this menu, doctor can choose between five games (the tutorial plus the four games in which data can be collected) to start the activity. There is also a "Cambiar Usuario" option in case the doctor wants to change the patient when finishing an activity without having to restart the application.

Also, there is the "Salir" button to be used in case the doctor wants to close the application at the end of an activity.

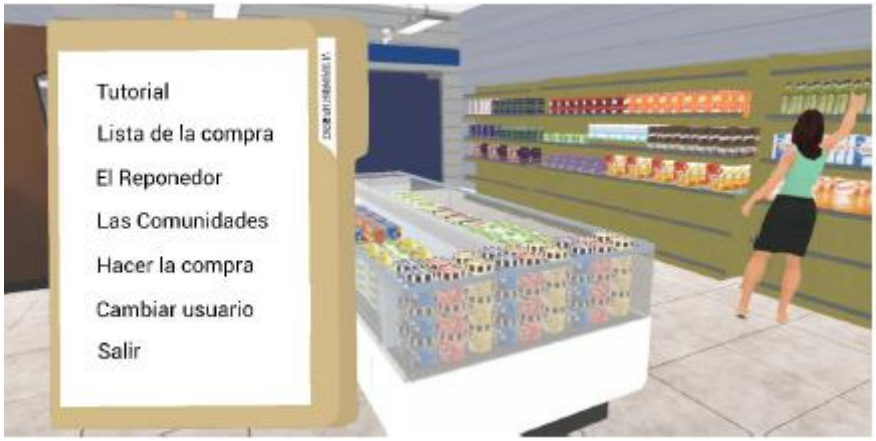


Figure 8.3. Main menu (Source: Prepared by the authors)

If the doctor clicks the "Tutorial" button, this game mode will start automatically. Otherwise, if any other game mode is selected, another menu in which doctor can choose the level of the selected game mode will appear. In this menu, three levels per mode are available. There is also "Volver" button which will allow the doctor to return to the "Main Menu".

In the following figure it can be seen this menu with each level:

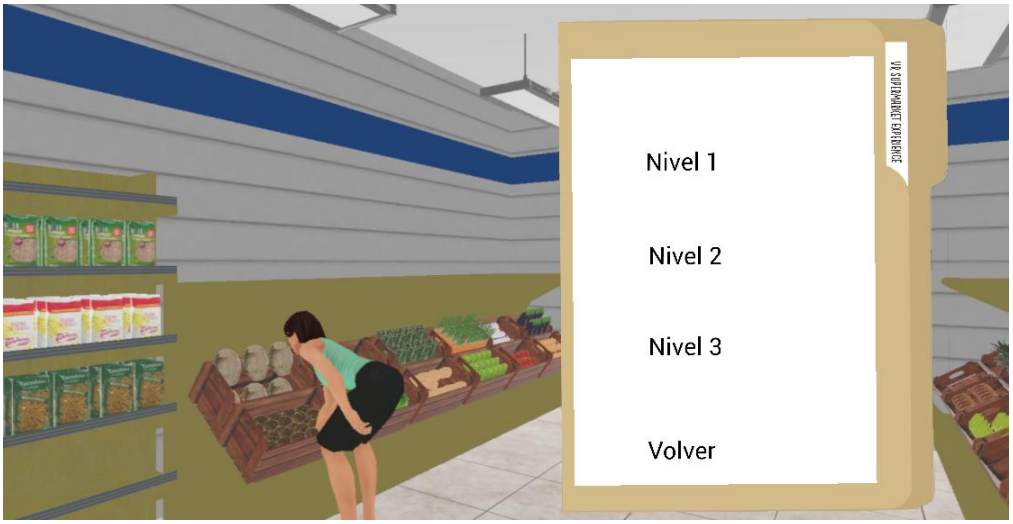


Figure 8.4. Levels' menu (Source: Prepared by the authors)

As it happens in the "ID Menu", while the doctor is selecting the game mode or the level in the "Main Menu", the patient is seeing a scene in which the shop assistant is adding oil bottles to the shelf.

8.1.3. Final menu

The "Final Menu" appears when the patient finishes any level of any game mode. Doctor can choose between two options: click "Repetir" button and restart the level, or return to the "Main Menu" by clicking the "Salir" button.

Figure 8.5. Final menu. (Source: Prepared by the authors)

While this menu appears in the computer screen and doctor is choosing whether to repeat the level or leave, the patient is seen the shop assistant walking around the supermarket.

8.2. Tutorial

As expected in most of applications, there is a tutorial in which the user can learn how the application works, which are game controls, learn to use them and deal with the virtual environment.

As this application will be used by people between forty and sixty years, this tutorial is crucial as it can be difficult to get used to a VR application.

8.2.1. Description

The patient will start in a small room with four. An audio will welcome them and will explain what has to be done. The first thing user will learn is to move his head to look around. To show how VR allows the user to have a 360-degree vision, a red sphere will appear in 3 of the four walls. As soon as the user follow the audio and looks at them frontally, they will change its red color to green.

Figure 8.6. First step of the tutorial (Source: Prepared by the authors)

Once all the spheres are green, the user will have to look at the reminder wall, where there is no sphere but a door. At this point, with the help of an audio, users will learn how to use game controls to move around the virtual environment. Once the audio finishes, the user has to go outside the room through a door which will open once they are near to it. Behind the door there is a corridor that has to be completed by moving around using game controls.

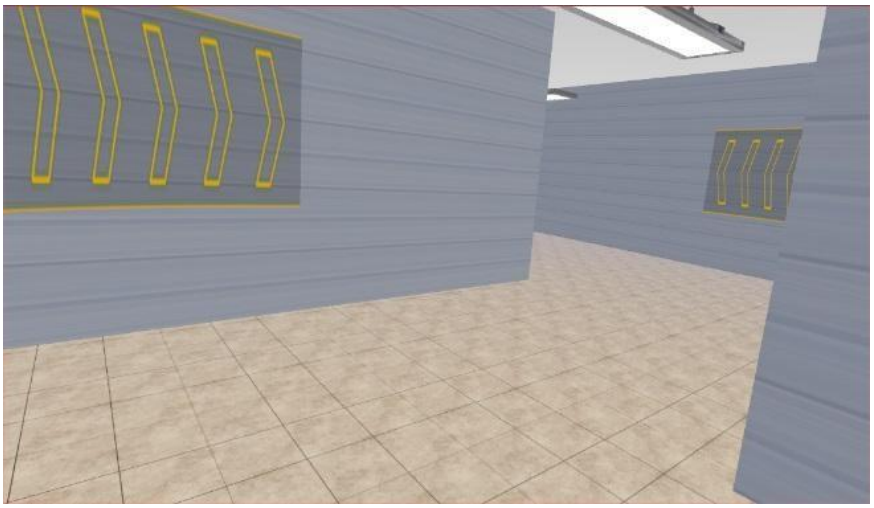


Figure 8.7. Corridor of the tutorial (Source: Prepared by the authors)

Throughout the corridor, the patient has to follow the arrows on the wall. At the end, the user has learned how to look and move. However, the corridor ends in a door that will open once the user approaches. At the other side, there is the final room in which the user must learn how to interact with supermarket products. Inside the room, there is a great table in the middle where three products are on it. In all walls, except in the door's one, there are shelves with typical products of the supermarket.

Figure 8.8. Final room of the tutorial (Source: Prepared by the authors)

Inside the room, an audio explains to the user how they can grab the objects on the table and that, once they take a product from the table, a green silhouette will appear a shelf. Then, the user has to leave the product in the silhouette that is located at the appropriate shelf. The tutorial finishes once all three products are correctly placed.

Figure 8.9. Green silhouette where the product has to be placed (Source: Prepared by the authors)

When the tutorial finishes, the patient appears in the supermarket, the environment where all learned before has to be practiced. Once there, the user is able to take any product and put them in the basket. This scene is the final preparation for the patient in order to get used to the place where will work later.

Figure 8.10. Supermarket environment test (Source: Prepared by the authors)

8.2.2. How does the tutorial work?

Some of the objects that conform the "Tutorial" scene have attached scripts that allow the user to interact with them. Some of these scripts will also be explained or even showed if necessary later on. All the scripts of the "Tutorial" can be found in the section "C1. "Tutorial" programmed codes" of Annex C. Program Codes".

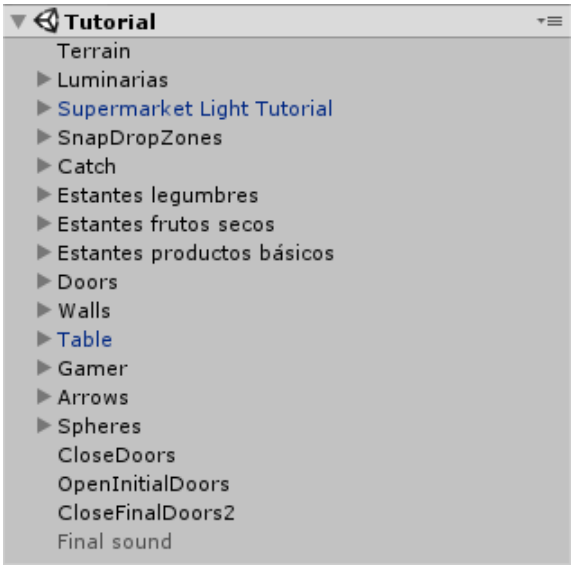


Figure 8.11. Tutorial objects (Source: Prepared by the authors)

The objects that make up the scene, and that can be seen on figure 8.11, are listed and explained below.

- "Terrain"
- "Luminarias"
- "Supermarket Light Tutorial"
- "SnapDropZones"
- "Catch"
- "Estantes Legumbres"
- "Estantes Frutos Secos"
- "Estantes Productos Básicos"
- "Doors"
- "Walls"
- "Table"
- "Gamer"
- "Arrows"
- "Spheres"
- "CloseDoors"
- "OpenInitialDoors"
- "CloseFinalDoors2"
- "Final sound"

Following lists' order, "terrain" is the floor that can be seen in the scene. For doing it more realistic, this game object has been made with a determinate texture.

"Luminarias" are the different lights that can be seen in the ceiling of the different parts of the tutorial. These objects make the scene more realistic, but they do not give light, their function is only decorative.

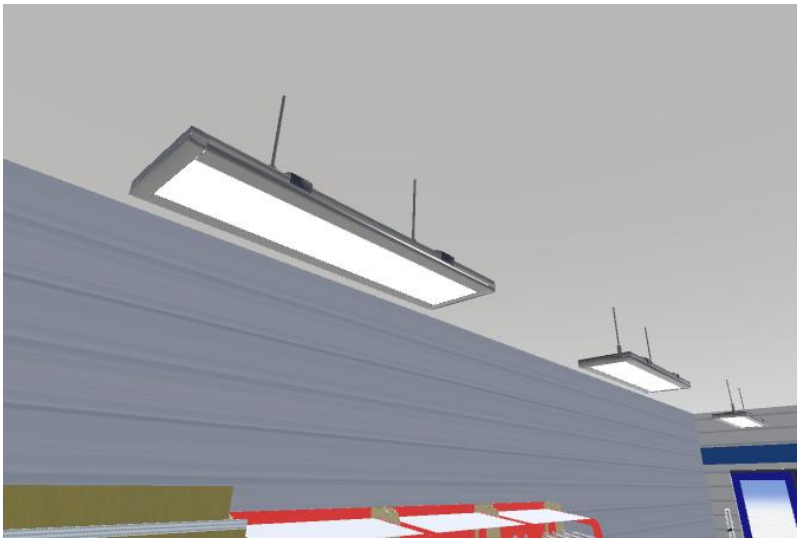


Figure 8.12. "Luminarias" in the supermarket environment. (Source: Prepared by the authors)

The "Supermarket Light Tutorial" contains different points of light in order to illuminate the scene, so the room is not dark and affect patients' visibility in the virtual environment.

Figure 8.13. "Supermarket Light Tutorial" game object (Source: Prepared by the authors)

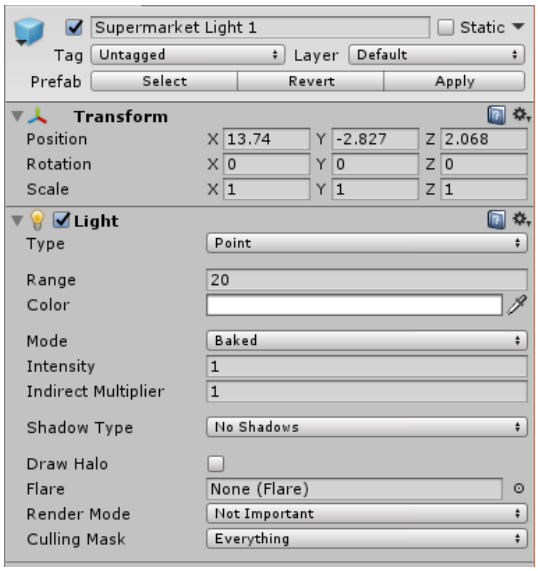


Figure 8.14. "Supermarket Light 1" characteristics (Source: Prepared by the authors)

As it can be seen in Figure 8.14 for each light, type, color, intensity and others can be changed.

Next up are the "SnapDropZones". This object contains the different areas that the three selected products occupy in the final room of the tutorial.

Figure 8.15. "SnapDropZones" game object. (Source: Prepared by the authors)

SnapDropZones" 1 are those situated on the table, where the products appear once the user arrives to the room. "SnapDropZones" 2 are the highlighted spaces on the different shelves where the product has to be placed.

The characteristics of the two "SnapDropZones" objects of the chestnuts can be seen below. This product is one of the three that are used to train at the final room of the tutorial.

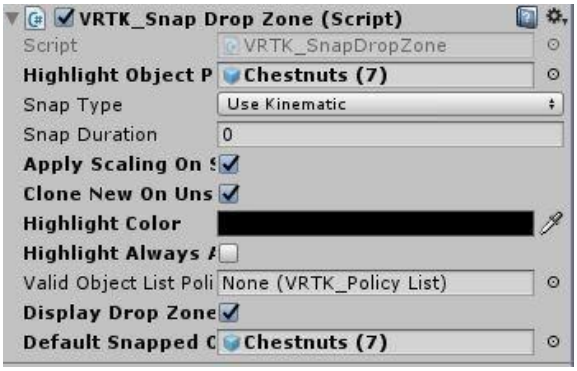


Figure 8.16. Chestnuts "SnapDropZone" 1 (Source: Prepared by the authors)

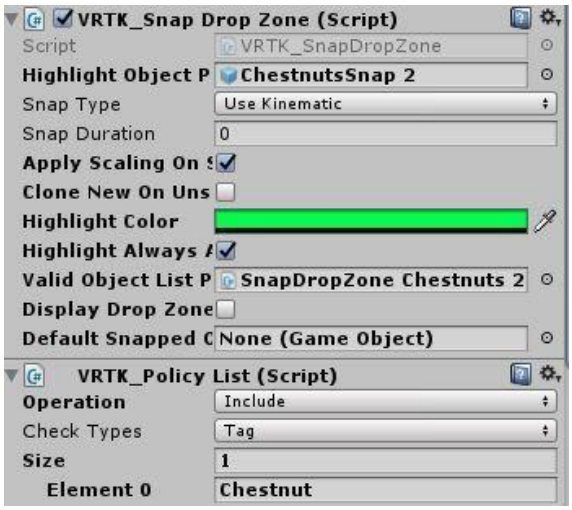


Figure 8.17. Chestnuts "SnapDropZone" 2.1 (Source: Prepared by the authors)

As it can be seen in Figure 8.17, the script "Snap Drop Zone" is the one that allows the zone where product has to be placed to be highlighted with the associated color (in this case, green). In addition to that, thanks to another script called "Policy list", it is possible to see if the chestnut is placed correctly.

There is also another script related to this game object. This one generates a "Unity" event, which makes the next product appear on the table once the previous product has been successfully placed in the correct Snap Drop Zone. This can be seen in figure 8.18:

Figure 8.18. Chestnut "SnapDropZone" 2.2 (Source: Prepared by the authors)

Following the order of the tutorial objects, next object is "Catch". This one has the three game products that are on the table in the final room, and these are: chestnuts, salt and chickpeas.

The three shelves on 3 of the walls (the reminder wall has the door in which the user enters in the room) are the next game objects to be explained. First one is the legumes shelf, "Estantes legumbres". This is where the patient has to place chickpeas, but there are other products such as lentils, beans, rice, flour or macaroni.

Figure 8.19. Legumes shelf (Source: Prepared by the authors)

The second shelf is "Estantes frutos secos", where the patient can find almonds, pistachio, peanuts, chestnuts and hazelnuts. The patient must drop the initial product that appear (chestnuts) in the appropriate zone of the shelf.

Figure 8.20. Nuts shelf (Source: Prepared by the authors)

The third of the shelves is the basic products' shelf, "Estantes productos básicos". In this one, vinegar, salt, pepper, eggs, sugar and oil can be found. User must place salt in the correct "SnapDropZone" to complete the tutorial.

Figure 8.21. Basic products shelf (Source: Prepared by the authors)

The "Doors" object is the one that contains the characteristics (structure, texture, material, etc.) of the two doors that appear in the tutorial.

There is another object for the walls. There are lots of them throughout the tutorial, but all of them are built with the same material to make it realistic.

The table that appears in the centre of the final room has another game object called "Table", where the different products appear.

The most important object is the “Gamer” one. It allows the patient to control the camera and the commands for the game. Having a look inside of it, there are five more game objects.

Figure 8.22. “Gamer” game object (Source: Prepared by the authors)

“VRTK_SDK Manager” is where we can control the camera and game controllers and relate them to each other. Inside of it there is “SteamVR” which contains “CameraRig” and “VRSimulatorCameraRig”.

Figure 8.23. Extended “Gamer” game object (Source: Prepared by the authors)

In the figure shown before, we can observe that the left and right controllers are inside “CameraRig”, as well as the camera’s head. In “Camera (head)” we can find several game objects which contain important scripts, such as the control of explanatory audios and the detection of the spheres at the first step of the tutorial in “Camera (eye)”, the ones that contain the audio source and ambient music, “Camera (ears)” and “Menu follower”, which allows the user to see the explanatory videos and instructions that are shown on screen during the tutorial.

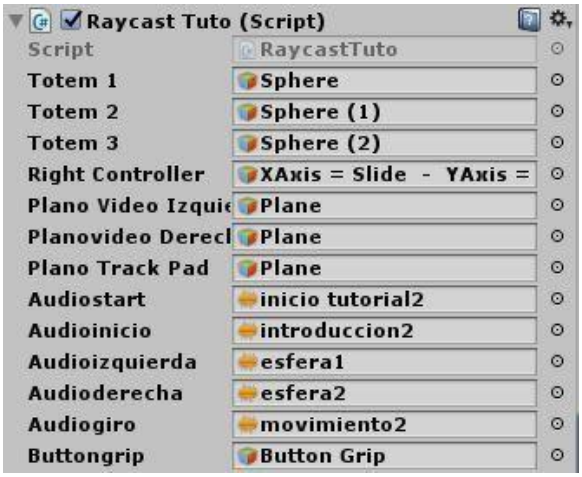


Figure 8.24. The most important script in "Camera (eye)" game object (Source: Prepared by the authors)

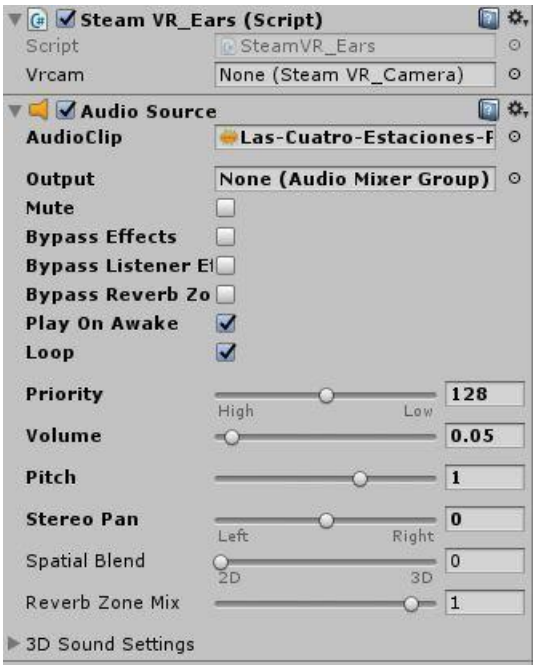


Figure 8.25. "Camera (ears)" game object characteristics (Source: Prepared by the authors)

"VRTK_Scripts" contains right and left controllers. In this game object there are different options which can change the way the patient moves with controllers and vary the speed of movement. The configuration here is as simple as possible, with low speed, as this game is thought for people who are not familiar to videogames or virtual reality.

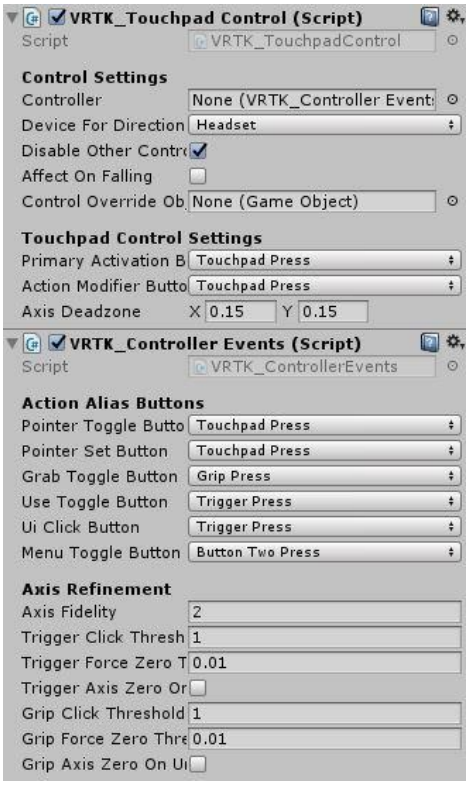


Figure 8.26. "Right controller" game object characteristics (Source: Prepared by the authors)

In the previous figure it is shown that it is easy to assign actions to buttons with "VRTK_ControllerEvents" script.

Next up in tutorial's game object list is "Arrows". This object contains the arrows that can be seen in the corridor and that lead the patient into the final room while training how to move through the virtual environment.

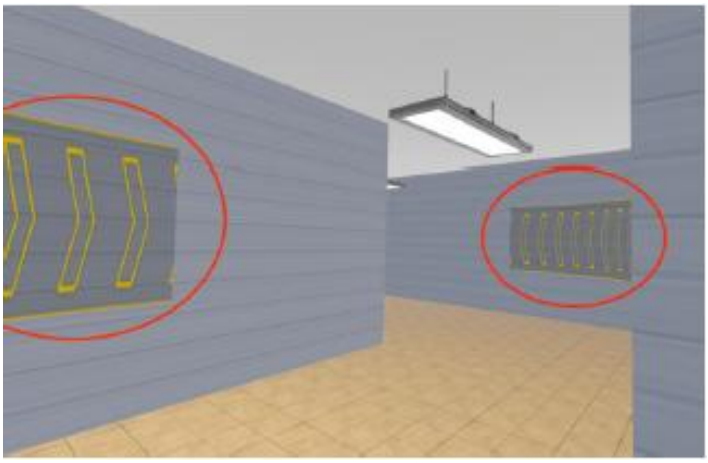


Figure 8.27. Arrows in the corridor (Source: Prepared by the authors)

As explained before, three spheres appear in the first part of the tutorial and they are inside “Spheres” game object.

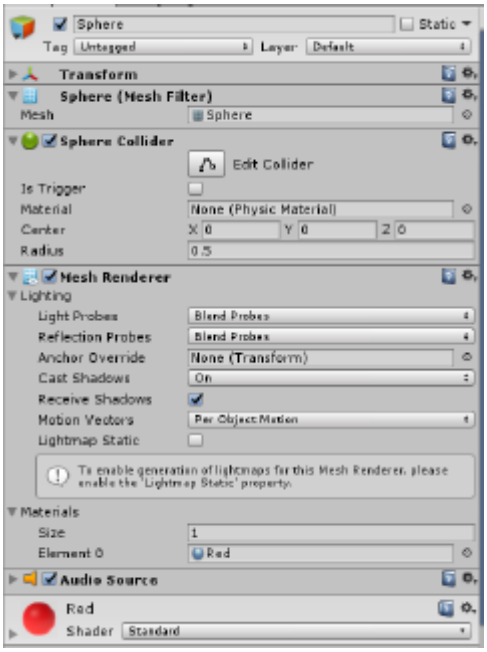


Figure 8.28. “Sphere” characteristics (Source: Prepared by the authors)

“OpenInitialDoors” is an object that contains a script with the same name that allows the user to open the door once all the spheres are green and the user approaches to the exit of the initial room.

Figure 8.29. “OpenInitialDoors” game object (Source: Prepared by the authors)

There is another object similar to the previous one, “CloseDoors”, which contains the script “CloseInitialDoors” that is activated once the patient has gone outside the room and it is at the beginning of the corridor. To avoid the patient going backwards, the room, when the script is activated, an animation closing the door starts.

Figure 8.30. "CloseInitialDoors" script (Source: Prepared by the authors)

"CloseInitialDoors2" object works as the previous one but it refers to the doors at the end of the corridor. So, when the patient enters in the final room, the animation closing the door starts.

At the end of the tutorial, an audio congratulating the patient when placing the last object properly on the correct shelf starts because of a script located at "Final sound" game object.

8.3. Shopping list

The "Shopping list" is the first game mode of the virtual application. The patient will listen an audio at the beginning in which the instructions of the game are given. Then, a list with a certain number of products will be shown. The user must take all this products and place them in the basket in the same way as in the final room of the tutorial.

In the supermarket, all products can be grabbed, but the main goal of this game is to take the specific products on the list.

User will always see a list of objects introduced in the basket, "Productos Cesta". This is done for the patient to be aware of the products placed correctly or incorrectly in the basket.

Figure 8.31. "Shopping list" game mode scene (Source: Prepared by the authors)

Once the patient has finished collecting all items from the list, an audio starts explaining to them that the level has finished and they must go to the exit door. There are three levels of difficulty, as in the other games. Level 1 is the easiest one, and level 3, the most difficult one.

8.3.1. Description

As in the tutorial and all levels of the games, there is an audio at the beginning explaining the instructions for a correct performance. The audio of the first level of the game is different to the other which explains level 2 and 3.

- First level: A list of three products will be shown when the initial audio finishes. This list will be permanently visible. The patient has to pick up the products placed in different parts of the supermarket and place them inside the basket. Once all products are correctly placed, user can go to the exit door and finish this level.

- Second level: The list shows five products instead of three. In addition to that, the list is visible only for a period of time, which is stipulated. However, the patient can have a look at the list by pressing the appropriate button of the controller (and that is explained in the initial audio). The number of requests that user does to have a look at the list, in order to complete the level, is controlled and collected in the database. Also, to increase difficulty, a distraction has been added. It consists of a supermarket replacement person who will go to the user and offer something to him.

Figure 8.32. "Shopping list" level 2 and 3 distraction (Source: Prepared by the authors)

- Third level: The list shows 7 seven products. The list disappears after a stipulated time, sooner than in level 2. Also, a second distraction was added: the first one is exactly the same as in the previous level and the new one is near fruits section and works in the same way as the old one.

Figure 8.33. "Shopping list" level 3 distraction (Source: Prepared by the authors)

8.3.2. The supermarket environment

Supermarket's environment is the same in all games except for the "Autonomous communities" due to its different characteristics. In order to make the supermarket the most realistic, products are organized by sections, and these sections have several shelves in which products are classified depending on their type.

The first section that the patient can see when starting the level is the bakery section. Only bread loaves and Bimbo bread can be found there.

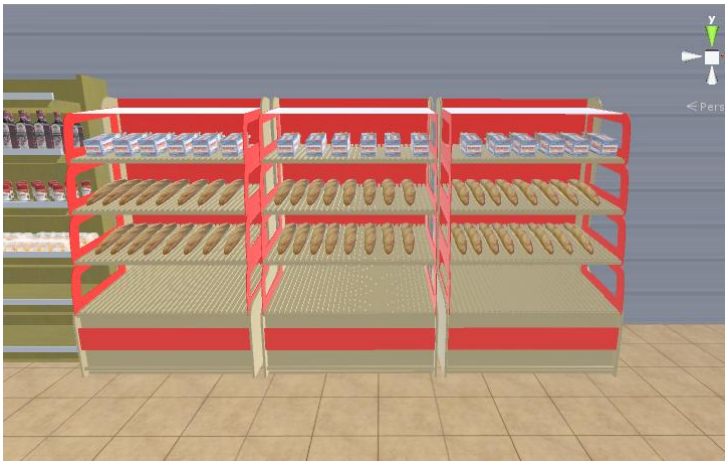


Figure 8.34. Bakery section (Source: Prepared by the authors)

Next to the bakery section, the user finds the basic products section. These are: oil, vinegar, salt, pepper, sugar and eggs.

Figure 8.35. Basic products section (Source: Prepared by the authors)

Following the previous section, there is the candy one. The patient will find several products such as: cereals, chocolate, coffee, milk, tea, marmalade and honey.

Figure 8.36. Sweets products section (Source: Prepared by the authors)

The last section of the right wall of the supermarket is the personal hygiene one. Tampons ("Tampax"), shaving gel, toilet paper, toothpaste and deodorant can be found there.

Figure 8.37. Hygiene products sections (Source: Prepared by the authors)

In the middle of the supermarket there is the freezer. The patient can find several products such as: ice cream, sweet corn, fish sticks, prawns, pizzas, frozen potatoes, lasagna and peas.

Figure 8.38. Freezer section (Source: Prepared by the authors)

Next to the freezer, two refrigerators can be found. The first refrigerator is the one that has, in the right side, chicken, steak and fish, and in the left side, mushrooms, cherries and strawberries.

Figure 8.39. First refrigerator (Source: Prepared by the authors)

The second refrigerator has, on the right side, lettuce, grapes and blackberries. On the other side, there are yogurts, cheese and butter.

Figure 8.40. Second refrigerator (Source: Prepared by the authors)

On the opposite side of the wall, there is the fruit section. The user can find a great variety of fruits, such as: coconuts, watermelons, radishes, red peppers, lemons, oranges, melons, pears, pineapples, peaches, bananas and apples.

Figure 8.41. Fruits section (Source: Prepared by the authors)

In front of the fruit section, there is the vegetables section. In this section, cauliflowers, kiwis, avocados, cucumbers, asparagus, onions, carrots, pears, garlic, tomatoes, eggplants and potatoes can be found.

Figure 8.42. Vegetables section (Source: Prepared by the authors)

The final part of the supermarket includes three shelves, the legumes, nuts and chips ones. Following this order, in the first one the patient can find lentils, chickpeas, beans, rice, macaroni and flour. In the second one, chestnuts, pistachio, hazelnuts, almonds and peanuts can be found. Finally, the products located in the last shelf are a varied assortment of chips.

Figure 8.43. From left to right, chips, nuts and legumes sections (Source: Prepared by the authors)

A general view of all sections of the supermarket can be seen below:

Figure 8.44. General view of the supermarket (Source: Prepared by the authors)

We have talked about the sections where products are located. Turning around, looking at the exit of the supermarket, the patient can see the entrance at the left, two checkout counters in the middle (the right one is not usable), and the exit door on the right. To avoid patient's confusion when trying to finish the level, left checkout counter has a sign which indicates it is closed ("Caja Cerrada"). This can be seen in the figure below:

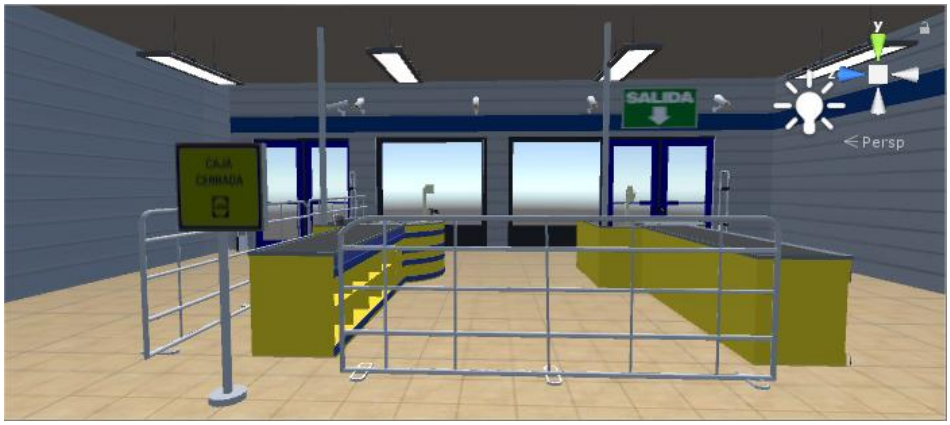


Figure 8.45. Supermarket checkout and doors (Source: Prepared by the authors)

8.3.3. How does it work?

Next up is the explanation of the game objects that make up this game. Each scene represents a level of the game and has its own objects. However, as level 3 is the most difficult one, level 1 and 2 game objects will not be explained as they can be found in the mentioned level. Some of the objects have attached scripts that allow the user to interact with them. Some of these scripts will be explained or showed if necessary. All of them can be found in section "C2. "Shopping list" programmed codes" of "Annex C. Program Codes".

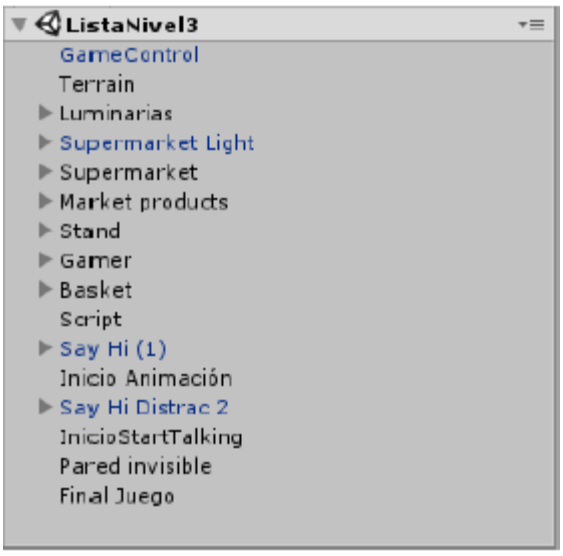


Figure 8.46. "Shopping list" level 3 game objects (Source: Prepared by the authors)

Objects that made up the scene are listed below and explained later:

- "Game control"
- "Terrain"
- "Luminarias"
- "Supermarket light"
- "Supermarket"
- "Market products"
- "Stand"
- "Gamer"
- "Basket"
- "Script"
- "Say Hi (1)"
- "Inicio animación"
- "Say Hi Distrac 2"
- "InicioStartTalking"
- "Pared Invisible"
- "Final Juego"

"Game control" is an object found in each scene of the application. This is where all data of the patient is collected.

"Terrain" and "Luminarias" game objects are the same as the ones explained at the tutorial in "8.2.2. How does it work?" section.

"Supermarket Light" object contains the different lights distributed in different sections which illuminate the scene (Supermarket lights 1 to 4). These lights have the option to change its type, color and intensity. This game object does not change in any of the games. Also, there is "Directional light" which is the light that corresponds to the sunlight, and it is located outside the supermarket.

Figure 8.47. "Supermarket light" game object (Source: Prepared by the authors)

"Supermarket" object is the next to be explained. Inside of it there are all the elements that form the supermarket, except products and their shelves. These elements are: divisors, cashiers, checkout zone, detectors, office objects (the office is out of the environment in which the patient works), doors and windows, security cameras, walls and others.

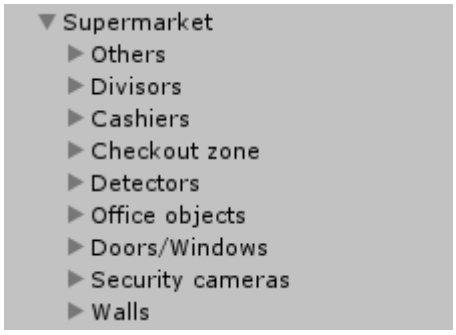


Figure 8.48. "Supermarket" game object (Source: Prepared by the authors)

"Market products" contains all the products and shelves of the different sections explained before in section 8.3.2.

Figure 8.49. "Market products" game object (Source: Prepared by the authors)

"Stand" object is where the shop assistant is located in the second distraction of the level showed in Figure 8.33.

"Gamer" object is almost the same as the one explained before in 8.2.2. section. However, there are some differences. The first of them is in "Camera (ears)" object, as audios and objects at the beginning are different compared to the tutorial.

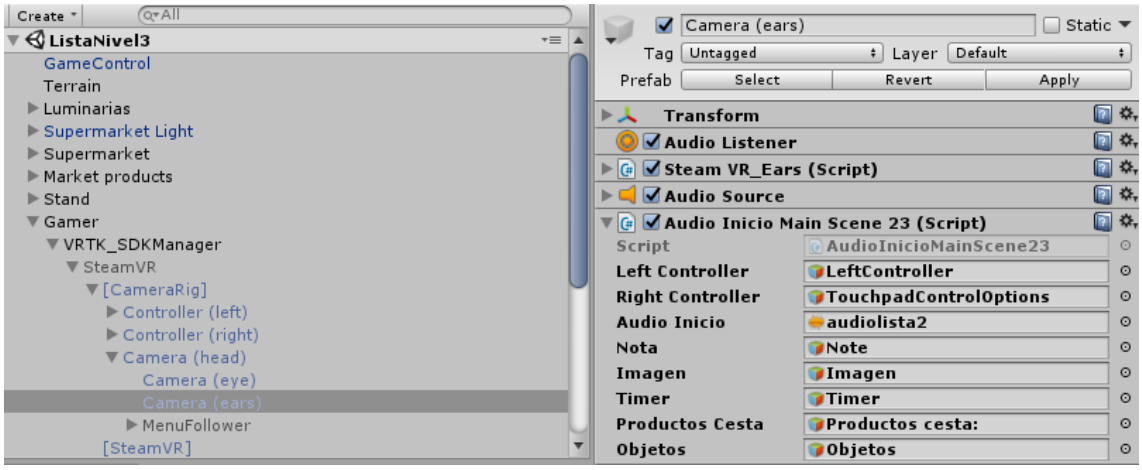


Figure 8.50. "Camera (ears)" object characteristics (Source: Prepared by the authors)

Second difference is in "MenuFollower". This game object contains the objects that the patient that have to be shown on user's screen which are: the "Note", where the products to be collected are listed, and the products introduced in the basket ("Productos cesta:" and "Objetos").

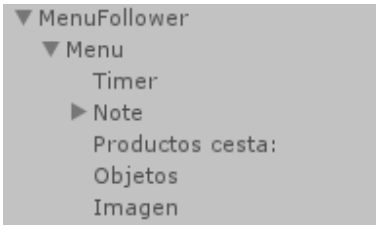


Figure 8.51. "Menu follower" game object (Source: Prepared by the authors)

The other differences are in game controllers. In "left controller", the basket has been attached and, thanks to "VRTK_Object Auto Grab" script, it always appears throughout the scene.

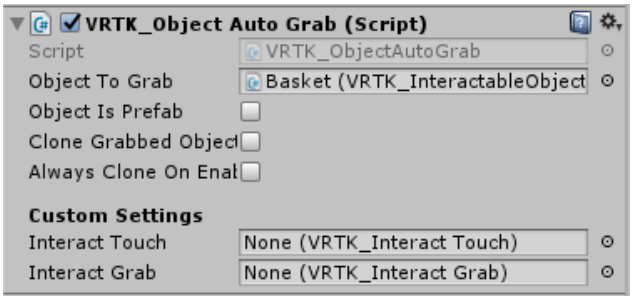


Figure 8.52. "VRTK_Object Auto Grab" script of the left controller (Source: Prepared by the authors)

In "Right controller", the script "Contador Lista Consulta" has been added in order to know how many times the patient has asked the controller to check the products' list. This only happens in levels 2 and 3 and the time it takes to disappear can be changed in "Tiempo Vista".

Figure 8.53. "Contador Lista Consulta" script of the left controller (Source: Prepared by the authors)

Next up is "Basket" object, which is the blue basket on the left controller and where the products must be introduced.

Figure 8.54. "Basket" game object (Source: Prepared by the authors)

"Mesh2" is the game object in which the shape and color of the basket is defined.

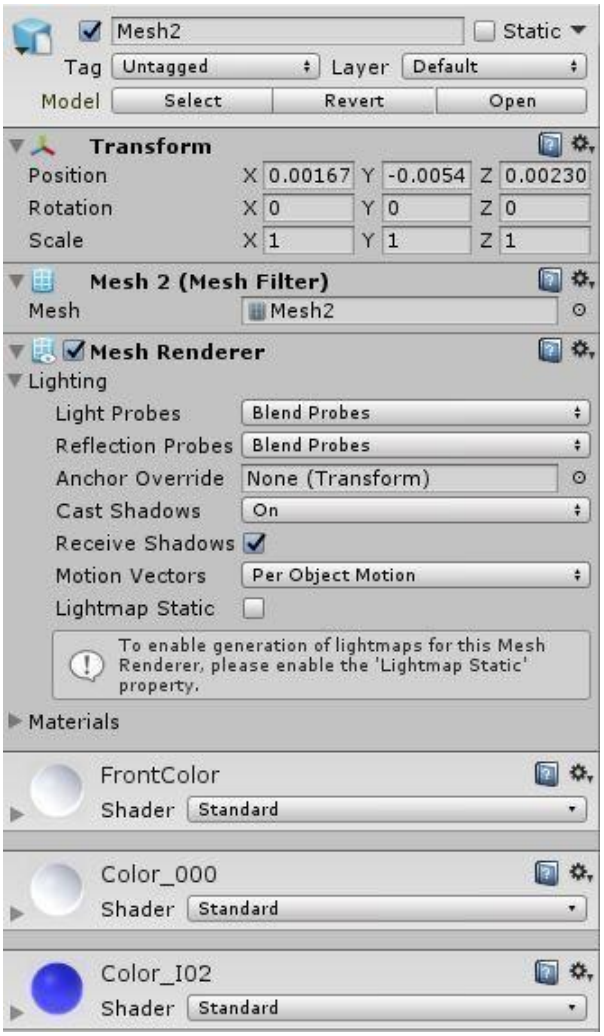


Figure 8.55. "Mesh2" game object of the basket (Source: Prepared by the authors)

The object "DentroCesta" is an object that contains the "StopGravity7" script which controls the products that enter properly or incorrectly in the basket. For level 1, the script is called "StopGravity3", and for level 2, "StopGravity5". This script also makes a list on screen with the products that are inside the basket.

Figure 8.56. "StopGravity7" script (Source: Prepared by the authors)

```
void OnTriggerEnter (Collider other) {

    if (other.tag != "Untagged") {

        other.GetComponent<Rigidbody> ();
        //other.attachedRigidbody.isKinematic = true;
        other.gameObject.transform.parent = transform;

        if (pet2 == 0) {

            if (other.tag == "Garbanzos") {
                V1 = 1;
                print (V1 + ("V1"));
                print (Valid+("Valid"));
            }
            if (other.tag == "Aceite") {
                V2 = 1;
                print (V2 + ("V2"));
                print (Valid+("Valid"));
            }
            if (other.tag == "Sal") {
                V3 = 1;
                print (V3 + ("V3"));
                print (Valid+("Valid"));
            }
            if (other.tag !="Garbanzos" && other.tag !="Aceite" && other.tag !="Sal")
            {
                NoVal = NoVal + 1;
                print (NoVal+("NoVal"));
            }

        }
```

Figure 8.57. "StopGravity3" script code (Source: Prepared by the authors)

The figure above shows the script associated to the first level. It checks if the object that collides with the basket is one of the listed ("pet2") and notes if it is correct or not.

The list of products is randomly chosen, as the following script shows:

```
public class ListaTres : MonoBehaviour {

    public Text Lista;
    private string[] listas = {
        "Garbanzos\nAceite\nSal",
        "Leche\nPan\nPlátano",
        "Pollo\nGambas\nManzana",
        "Bistec\nPizza\nSandía",
        "Lentejas\nMiel\nTomate",
        "Pepino\nVinagre\nMelocotón",
        "Azúcar\nMantequilla\nHelado",
        "Arroz\nPan bimbo\nGuisantes"
    };
    public int pet;

    // Use this for initialization
    void Start () {

        pet =Random.Range(0,listas.Length);
        print (listas[pet]);

        Lista.text = listas [pet];

    }
```

Figure 8.58. "ListaTres" script code, associated to the first level (source: Prepared by the authors)

As we can see, there are 8 lists of products, and each one have a number associated. The system chooses randomly one number and appears as "pet2" in the script of the figure 8.57. All the programmed code of these scripts can be found in section "C2. "Shopping list" programmed codes" of "Annex C. Program Codes".

"Script" is an object that contains a script in order to control de time.

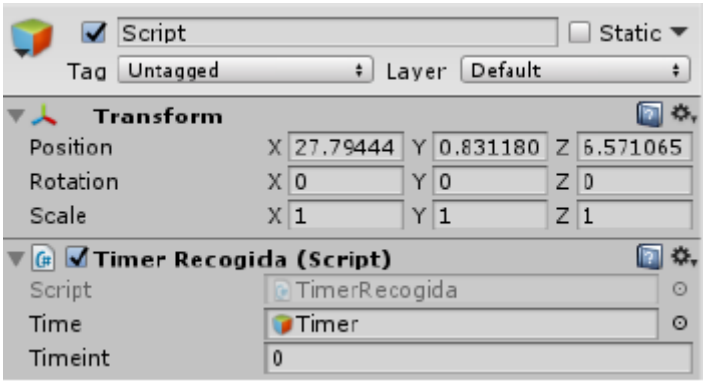


Figure 8.59. "Script" game object (Source: Prepared by the authors)

"Say Hi (1)" is an object that generates the first distraction. As we have mentioned before, this is only for levels 2 and 3. It consists of a shop employee avatar that starts talking (that is the audio source) and an animator generate the movements of the avatar to make it more realistic.

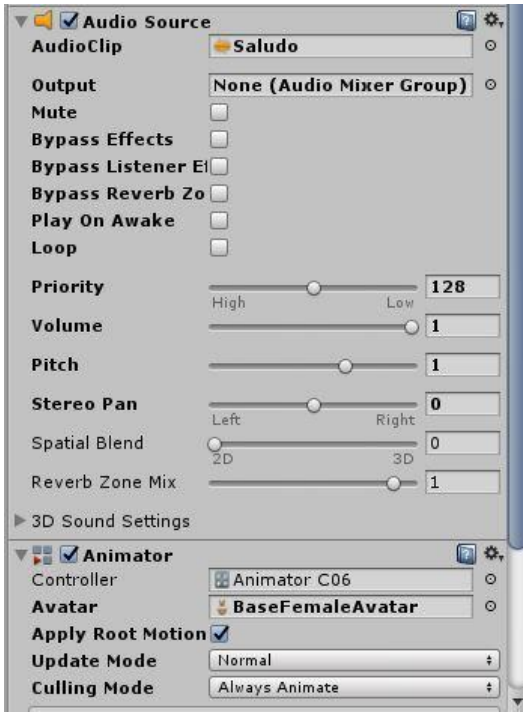


Figure 8.60. "Say Hi (1)" game object (Source: Prepared by the authors)

"Inicio Animacion" object is related to the previous object. It contains the script "StartWalking" which indicates when the shop assistant has to start the animation. The settled time for the avatar to start walking is 15 seconds, audio is activated 3 seconds and the animation lasts 6 seconds.

Figure 8.61. "Start walking" script (Prepared by the authors)

"Say Hi Distract 2" is the object that generates the second distraction, which only occurs in level 3. Is the same animation as in "Say Hi (1)" game object, only varies what the shop assistant says and the animation.

The game object "Inicio Start Talking" is the same as the object "Inicio Animacion". It contains a script that indicates when the shop assistant has to start talking and its animation. It is programmed to start when the patient is near the shop assistant.

Figure 8.62. "Start talking" script (Source: Prepared by the authors)

"Pared invisible" is an invisible wall that allows the shop assistant to do the animation without the user being bothered in the middle of her trajectory.

"Final Juego" is the last object of the game. It contains an invisible wall, located at the exit door, which has a script that activates once the patient collides with it. It automatically closes the scene and the final menu appears.

8.4. Shop assistant

The "Shop assistant" is the second of the games of the application. In this case, the patient becomes the supermarket worker instead of being the supermarket customer.
The patient starts at the entrance of the supermarket. Then, the user must go to a table where there are several products that have to be placed in the correct shelf, as if they were the shop assistant.

When the patient places the last product, a congratulating audio starts and indicates that the game has finished. There are 3 difficulty levels, as in all games, going from the easiest one (level 1) to the most difficult (level 3).

8.4.1. Description

As in all games, at the beginning there is an audio explaining what the patient has to do.

In the first level, the user will find three different products in a table next to both checkout counters. Then, the patient has to identify and place all products in their correct shelf. Once all the products are placed, the level automatically finishes.

The main difference between levels is the number of products in the table that have to be placed. In level 2, instead of three products, there are four. In level 3, the number of products increases up to six. All products of the levels are shown below:

- Level 1: Pizza, milk and “Bimbo” bread.

Figure 8.63. “Shop assistant” level 1 products (Source: Prepared by the authors)

- Level 2: Pineapple, steak, eggplant and flour.

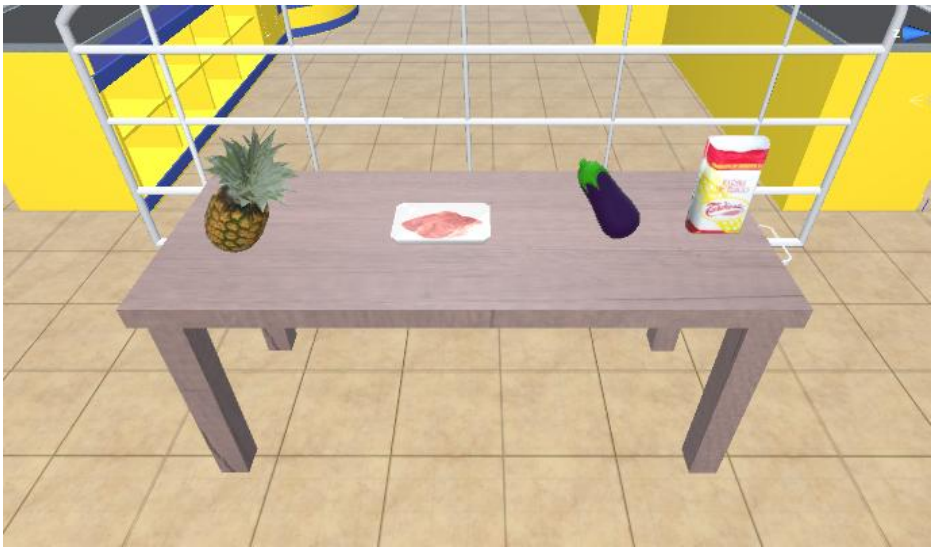


Figure 8.64. “Shop assistant” level 2 products (Source: Prepared by the authors)

- Level 3: Yogurt, chips, peas, toothpaste, watermelon and endive.

Figure 8.65. "Shop assistant" level 3 products (Source: Prepared by the authors)

The only thing that is studied in this game is the time it takes the patient to place all products from the table to their correct shelf. As all of the data collected in the application, it can be consulted in applications' database.

8.4.2. The supermarket environment

As mentioned in 8.3.2. section, the supermarket environment is the same in all games except for the "Autonomous Communities". That means, all products, shelves, lights, checkout counters, etc. are at the same place. Game controls are the same, except for the left controller, where there is not a basket as it is useless in this game. However, a table with the products to be placed has been added in front of the checkout area.

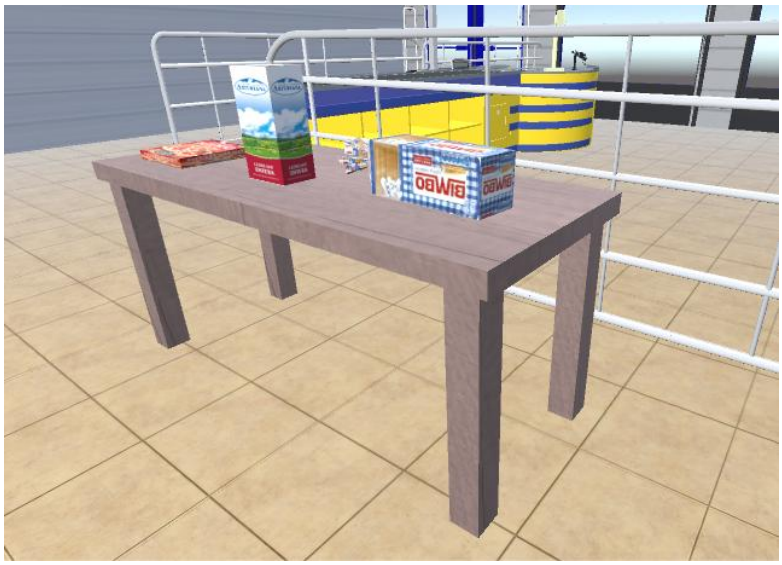


Figure 8.66. The table where level 1 products are at the beginning (Source: Prepared by the authors)

8.4.3. How does it work?

This section lists and describes the different game objects that make up this "Shop assistant" game. Many of the objects are the same as in previous games. Reusing them is useful to save working time and create a more compact application.

As in all games of the application, there is a scene per level. Each scene has its own objects. However, it is only necessary to explain one of the scenes because the only changes between levels are the number of products to place.

The list of game objects that make up level 3 is shown below:

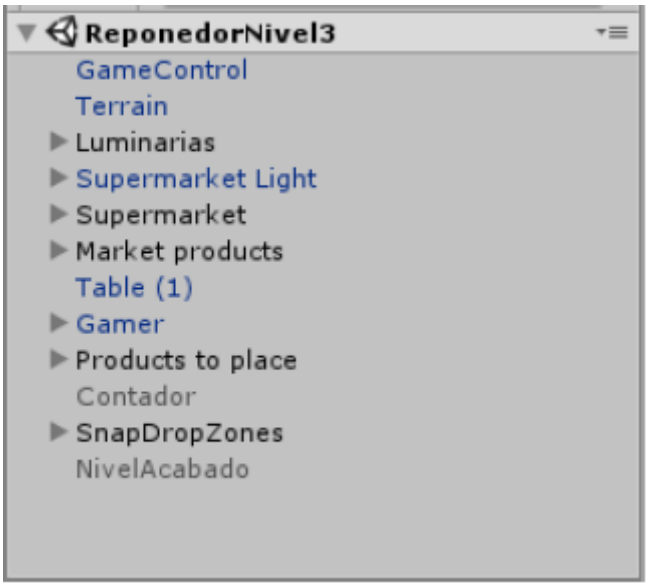


Figure 8.67. "Shop assistant" level 3 game objects (Source: Prepared by the authors)

So the objects that make up the scene are:

- "Game control"
- "Terrain"
- "Luminarias"
- "Supermarket light"
- "Supermarket"
- "Market products"
- "Table (1)"
- "Gamer"
- "Products to place"
- "Contador"
- "SnapDropZones"
- "Nivel acabado"

As in all games, "Game Control" is where all data of patient's performance is collected. In this game the only thing that is monitored is the time the patient takes to complete the game.

The "terrain" object is the same as the other games.

"Luminarias" object is also the same object as in the other games.

The “Supermarket light” is a game object for all games which includes light points to illuminate the scene.

“Supermarket” and “Market products” are all the same as in the other games, as part of the uniformity of the supermarket.

All four previous game objects are explained in 8.2.2. and 8.3.3. sections.

The object “Table (1)” is the table located at the centre of the supermarket, near checkout zone, where there are the products that the patient must place. In figure 8.66. this table can be seen.

Next up is the “Gamer” object. It is necessary in all scenes of the games in the application. However, there are few differences between them as part of the variance of the games. So, the first difference is in “Camera (ears)”. The initial audio is different for every game, so the script which activates it also changes. The details for this game object are shown before:

Figure 8.68. “Camera (ears)” game object characteristics (Source: Prepared by the authors)

Another slight difference that can be found in “Gamer” object is that this games’ one does not have the “MenuFollower” object inside “CameraRig”. This is because there is no object or text shown in the screen during the activity.

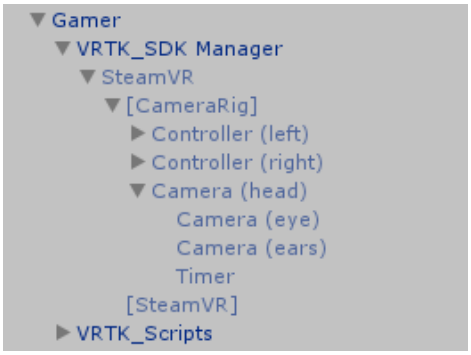


Figure 8.69. “Gamer” object of “Shop Assistant” (Source: Prepared by the authors)

Inside “Products to place” there are the different products that appear on the table and that have to be placed in their appropriate shelf.

Figure 8.70. “Products to place” game object (Source: Prepared by the authors)

Also, “Products to place” contains the script “ProductosCestaController”, which is the responsible of the count of the correct products that are placed. Once the patient has placed all products properly, the script stops the time and activates the object “NivelAcabado” to finish the level.

Figure 8.71. “ProductosCestaController” script (Source: Prepared by the authors)

The count of the products correctly placed is made by a counter (the object “Contador”) that contains a script.

```
public class Contador : MonoBehaviour {

    private float contador = 5.0f;
    int aciertos = 0;
    string escena;
    string levelname;

    public void Aciertos(int valor)
    {
        aciertos = valor;
    }

}
```

Figure 8.72. “Contador” script code (Source: Prepared by the authors)

Returning to the object “Products to place”, it has been said that inside of it there are the products to be placed. These objects vary depending on the level. The characteristics of these products will be shown below, and are the same as all the supermarket products.

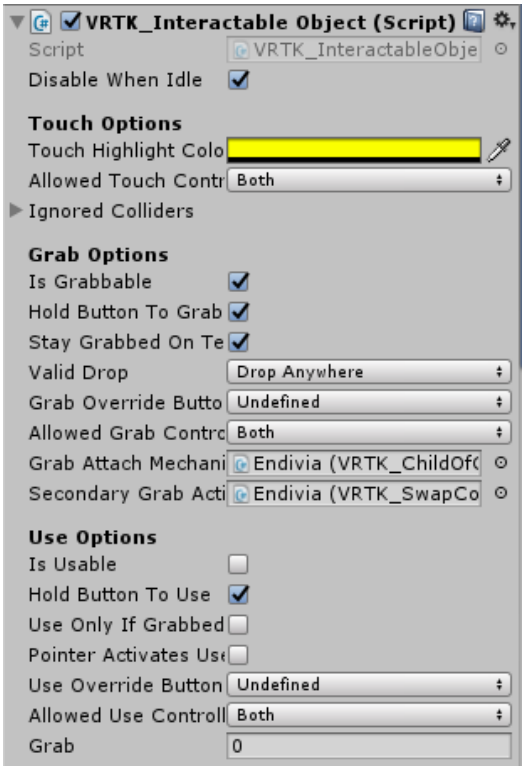


Figure 8.73. "VRTK_Interactable Object" script (Source: Prepared by the authors)

This is the component that makes the object be grabbable and highlighted when the patient approaches the controller to grab it.

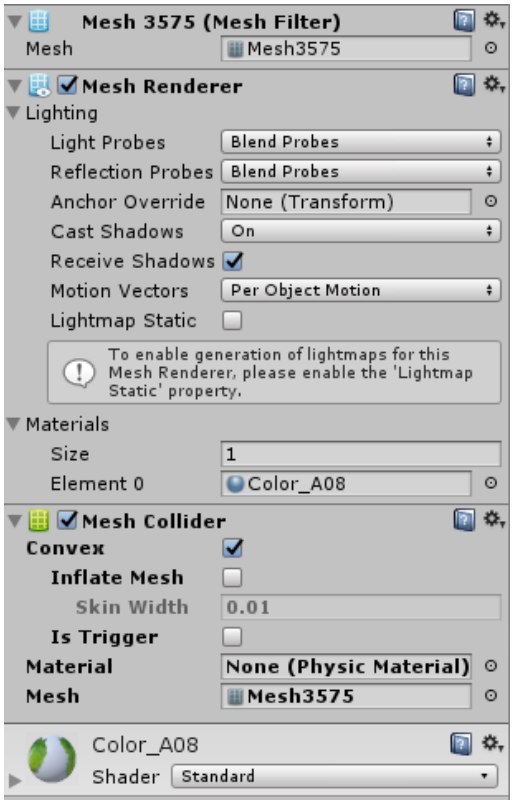


Figure 8.74. Texture and mesh of a supermarket product (Source: Prepared by the authors)

This object is the responsible for the shape of the object. The texture is what gives the color and makes it look realistic.

Finally, "Rigidbody" component allows the developer to decide the mass the object has and if it follows gravity laws or not.

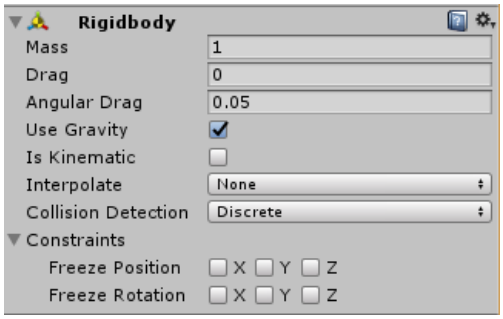


Figure 8.75. "RigidBody" component (Source: Prepared by the authors)

Next one is an object that has appeared also in the tutorial. "SnapDropZones" generates the space where the patient must place the product.

Figure 8.76. "SnapDropZones" game object (Source: Prepared by the authors)

Figure 8.77. "SnapDropZones" in the supermarket environment (Source: Prepared by the authors)

There is a "SnapDropZone" for each product to place in their correct shelf. This game object was explained in "8.2.2. How does it work?" section of the tutorial.

There is one final object in the game, "NivelAcabado". This one contains a script ("Salirnivel6") that finishes the activity once the patient has placed all the products correctly.

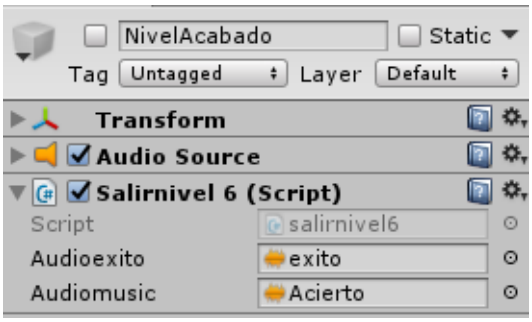


Figure 8.78. "NivelAcabado" game object (Source: Prepared by the authors)

```
public class salirnivel6 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Reponedor 3");
    }
}
```

Figure 8.79. "Salirnivel6" script code (Source: Prepared by the authors)

When the patient finishes the activity, two audios start thanks to the previous "Salirnivel6" script: one is a success sound and another is a message congratulating the user. Once the second audio finishes, the scene closes and the final menu appears.

All scripts of this game can be found in ""C3. "Shop assistant" programmed codes" of "Annex C. Program Codes".

8.5. Autonomous communities

The "Autonomous communities" game is the third of the games that make up the application. It is the one that has more similarities with the "Shop assistant" game.

The game elapses in one of the rooms of the supermarket, which is only used for this game. The patient is in the middle of the room and sees a certain number of stands with a flag of an autonomous community of Spain in each one. Behind the patient, there is a table with the same number of products as flags are in the stands. The user has to identify and place the products on the appropriate stand. The products have been thought to be easy to match with their autonomous community. Also, the names of the autonomous community have been added below the flags to avoid identification errors.

As in previous games, once the patient has placed all products on the stands, a congratulating audio sounds and the game automatically finishes. There are also 3 levels, level 1 is the easiest and level 3 is the most difficult.

8.5.1. Description

As always, at the beginning of the game there is an audio with instructions to let patient the patient know what to do.

In the first level, 3 products will be on the table. As it has been explained before, the patient has to place them in their corresponding stand on the room. The level is finished once the patient places all products correctly.

The level of difficulty increases at level 2 and 3. In level 2, instead of 3 products there are 4. In the most difficult level, there are a total of 6 products. Also, levels 2 and 3 have an extra stand that does not correspond to any product and its only goal is to try to mislead the patient. All products of the levels can be seen below:

- Level 1: Gazpacho (Andalusia), bananas (Canary Islands) and “fuet” (Catalonia).

Figure 8.80. Autonomous communities” level 1 products (Source: Prepared by the authors)

- Level 2: “Mojo picón” (Canary Islands), Asturian “fabada” (Asturias), olives (Andalusia) and cava (Catalonia).

Figure 8.81. "Autonomous communities" level 2 products (Source: Prepared by the authors)

- Level 3: Oranges (Valencian community), cherries (Castilla y León), "txakoli" (Euskadi), mussels (Galicia), red wine (La Rioja) and blood sausage (Extremadura).

Figure 8.82. Autonomous communities" level 3 products (Source: Prepared by the authors)

The only thing studied in this game is the time it takes the patient to identify and place all products in the appropriate stand. Database store all data collected from this game.

8.5.2. The supermarket environment

This games' environment is different from the other games. It takes place in a room, in which there is a table next to the wall that has a door. This is the same table as the one next to checkout zone in "Shop Assistant" game.

Figure 8.83. Table with level 1 products to be placed (Source: Prepared by the authors)

On the opposite wall there are the stands with their flags in each one. In the other two walls, there are shelves with products from the supermarket, but their only function is decorative (to make it more realistic, so objects cannot be grabbable or does not have gravity.

Figure 8.84. Stands with autonomous communities' flags (Source: Prepared by the authors)

All walls, ceiling, terrain and shelves are the same as the ones on the supermarket, as well as their textures and colors, in order to give uniformity to the application.

8.5.3. How does it work?

As the other games, this one has also 3 scenes, one per level. It is necessary to explain only one of them because the only difference between levels is the number and type of products that the patient has to match. All scripts of "Autonomous communities) can be found in ""C4. "Autonomous communities" programmed codes" of "Annex C. Program Codes". Below there are the different game objects that make up level 3 on "Autonomous communities" game:

Figure 8.85. "Autonomous communities" level 3 game objects (Source: Prepared by the authors)

So, the objects that will be explained, following this order, are:

- "Game control"
- "Terrain"
- "Luminarias"
- "Supermarket light Comunidades"
- "Supermarket"
- "Decoration products"
- "Table"
- "Shelves"
- "Flags"
- "Gamer"
- "Products to place"
- "SnapDropZones"
- "Contador"
- "Nivel acabado"

As in all games, "GameControl" is the game object where the data of patience's performance is collected. As in "Shop assistant" game, only the time that it takes the patient to match the products with their correspondent flag is studied.

"Terrain" and "Luminarias" game objects are the same as the ones explained in this section in previous games (for example, in "8.2.2. How does it work?").

"Supermarket Light Comunidades" is the same as the previous games "Supermarket Light" game object. It contains different light points to illuminate the scene, so the patient has no problem to see in it.

"Gamer" object is almost the same game object as in the other games. However, the difference between them is always at "Camera (ears)", because the initial explanatory audio varies. So, the only thing that change is the script that makes the audio start.

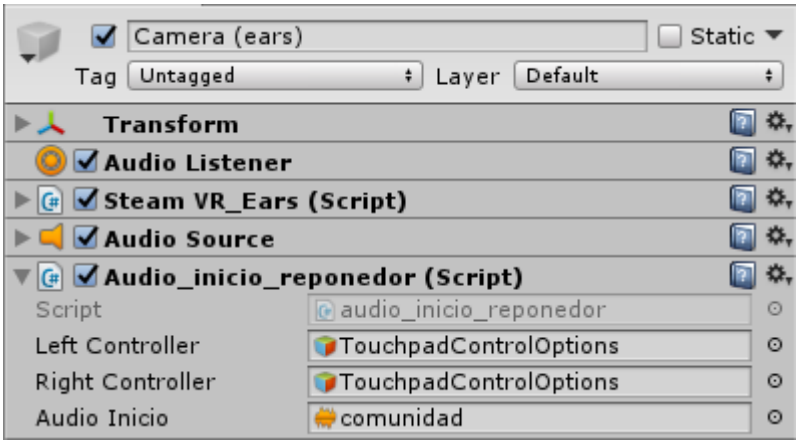


Figure 8.86. "Camera (ears)" game object characteristics for level 3 (Source: Prepared by the authors)

Next up is the "Supermarket" game object. This one is the same for all games, so it has been explained in the previous two games. It defines the structure (walls, ceiling, etc.). This structure includes the whole supermarket, that is: all tutorial rooms and corridor, the supermarket where "Shopping list" and "Shop assistant" games take place, office rooms and the room in which this game takes place. That is why this game object is the same in all games.

Figure 8.87. "Supermarket" game object overview (Source: Prepared by the authors)

"Decoration products" is a game object which includes all products on the two shelves in the room and that are only decorative, the user cannot interact with them (they are not grabbable) as there are no scripts that allows the patient to do that. These products are some of the ones we can find in the supermarket. This can be seen below:

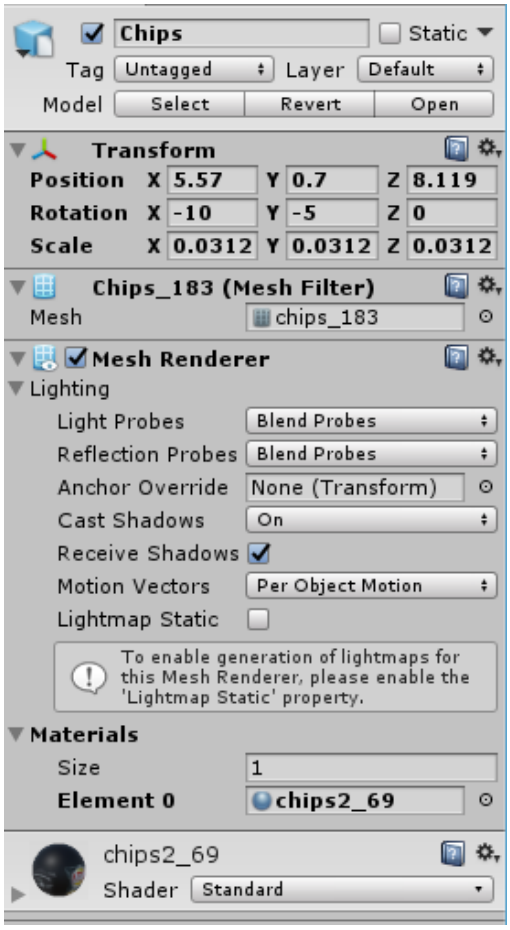


Figure 8.88. "Decoration products" characteristics (Source: Prepared by the authors)

"Table" is the same game object as the "Shop assistant" game one. There is where the patient finds the products to be matched.

In this game we can find a game object called "Shelves". It contains all shelves that are in the room, including the different stands that they contain.

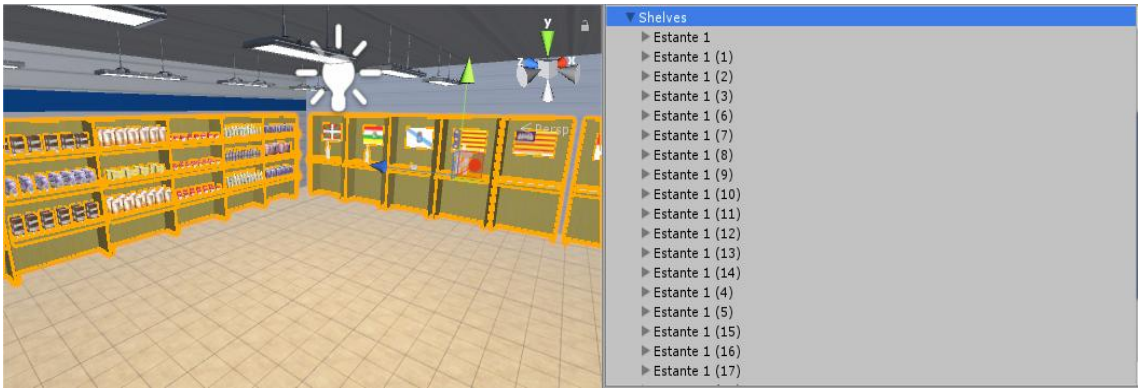


Figure 8.89. "Shelves" game object (Source: Prepared by the authors)

"Flags" game object contains the different flags of the communities. They vary depending on the level.

Figure 8.90. "Flags" game object (Source: Prepared by the authors)

"Products to place" and "Contador" game objects are the same as the ones explained before in "8.4.3. How does it work?" section from the previous game. The only variation here is the variation of products, which change depending on the level.

These games where the patient must place or match products with the corresponding shelf or flag require the game object "SnapDropZones". This object changes because of the number of products, as in the first one there are less SnapDropZones than in the most difficult level. The operation and scripts are exactly the same as the ones explained before in 8.2.2. and 8.4.3. sections.

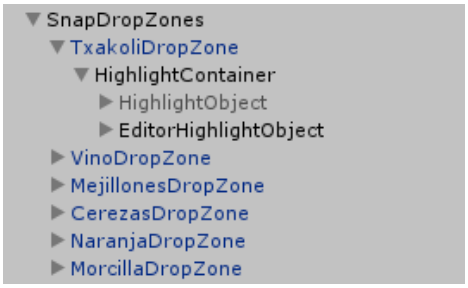


Figure 8.91. "SnapDropZones" game object (Source: Prepared by the authors)

The last object is "NivelAcabado", and has the same characteristics as the one explained in the previous game, in section "8.4.3. How does it work?". The script is the responsible for finishing the level once the patient has matched all products.

8.6. Go shopping

This is the last game of the application. The patient is, as in the first game, the customer.

In this game the patient must take a certain number of products (which depends on the level). In this occasion, the user decides which of the products wants to take. As in the "shopping list" game, all products taken must be placed inside the basket. Each product has a price associated to it. So, once the patient has introduced a product in the basket, a list is shown on the screen with the product and its price. That is made for the patient to know if the product has been properly introduced or not.

Figure 8.92. "Go shopping" level 1 scene (Source: Prepared by the authors)

Once the patient has taken enough products, the payment must be made in the cashier. When been there, basket and list will no longer be on the screen. But, instead, the ticket with the products, their price and the total price of the purchased products will be shown on the screen. The patient, then, can proceed with the payment.

Bills and coins available to make the payment are located at the left part of the desk. The patient must enter in the indicated area of the automatic cashier and take the necessary bills and coins for the payment.

In this game, the levels' main difference is the payment which, depending on the level, must be exact or not. A list with the money introduced in the cashier is shown on the screen once the patient introduces the first bill or coin.

Figure 8.93. "Go shopping" payment level 1 scene (Source: Prepared by the authors)

Once the payment is completed, the patient must go to the exit door and the level will be completed. There are also 3 difficulty levels, going from 1 (the easiest) to 3 (the most difficult one).

8.6.1. Description

An explanatory audio is also included in this game to explain the patient what has to do and the number of products that has to take.

In addition to that, there is also a short summary shown on the screen along with a button to omit it. This is to make the game more dynamic and not bore the patient if the game has been done before. This can be seen in the figure below:

Figure 8.94. Audio summary and skip button for the initial audio (Source: Prepared by the authors)

In the first level, the patient is required to take three products from the supermarket and do the payment in the cashier. When the patient introduces the third product in the basket, an audio starts explaining that the payment is ready to be made. There is also a short summary on the screen when entering at the checkout zone.

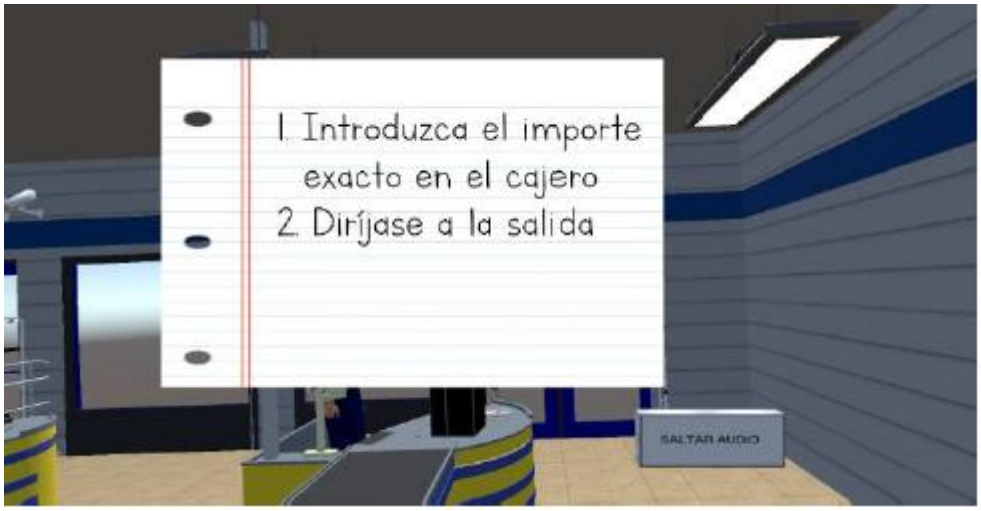


Figure 8.95. Summary for the second audio and skip button (Source: Prepared by the authors)

The patient, in the first level, has to pay the exact price of the purchase and that is shown in the ticket. This is why the user has available all coins and bills. Once the patient believes the payment in the cashier is properly done, they must go to the exit door to finish the level.

In the second level, the main changes are: number of products increased from 3 to 5 and a change in the way the payment has to be done. Coins will disappear and only banknotes will be available to the user. The consequence is that, most of the times, the payment will not be exact, as there are banknotes of 5, 10 and 20 Euros. That is why the change will be shown on the screen. However, it has been programmed to not always be correct, as sometimes there is more money, less money or the exact change. The patient has to decide whether the change is correct or not.

Figure 8.96. Change shown on the screen in level 2 (Source: Prepared by the authors)

As it can be seen, the patient must choose between the green button if change is correct and red button if, instead, thinks that it is wrong. Once one of the buttons is pressed, the final audio will start saying to the patient that has to go to the exit and that the level is completed.

Finally, in the third level, the patient must take also five products and go to the checkout area, where there will only be banknotes to pay. Until now, this level is the same as the second one. The main change comes now. When the change is showed in the screen, the user can interact with banknotes and coins. If there is more change than the correct one, the patient must grab and drag the excess of money to the red button. When patient thinks it is correct, the green button must be pressed and the final audio will sound explaining the patient that can go to the exit and this level is completed.

Figure 8.97. Change shown in the screen in level 3 (Source: Prepared by the authors)

This game is the most complete one as there is a lot of data collected. As in most of the games, it is supervised the time it takes the patient to complete the level. Also, for each level there is different data collected:

- First level: It is taken in consideration the ability of the patient to pay the exact price, if there was a mistake by paying less money or if the mistake was because the patient paid more.
- Second level: Data collected shows if the patient has pressed the correct button, if the patient has paid less than the total price of the purchase, if the patient has pressed the green button when change was incorrect or vice versa.
- Third level: The data indicates if the patient has paid less than required or if there was a mistake when correcting the change.

8.6.2. The supermarket environment

As we have been saying throughout this project, all games have the same environment in order to avoid confusions to the patient when changing games. This has been explained in previous sections such as “8.3.2. The supermarket environment”. Game controls are also the same. However, in this game there are some exclusive objects:

- A cashier desk sign which indicates that is closed. It is located in front of the left cashier desk so that the patient goes straight to the other checkout counter.

Figure 8.98. Cashier desk closed sign (Source: Prepared by the authors)

- A cashier. To make the scene more realistic, there is a cashier behind the checkout counter desk where the patient has to do the payment.

- An automatic cashier. It is located above the desk. It has a sign that indicates where the patient has to enter the money, so there is no confusion on where to introduce it.

Figure 8.99. Cashier and automatic cashier. (Source: Prepared by the authors)

8.6.3. How does it work?

As in all of the games, there are 3 scenes, one per level. In this section there will be the explanation for game object of level 3 scenes, as almost all of the objects appear in all scenes.

All scripts that appear or are mentioned in this section can be found in the section ““C5. “Go shopping” programmed codes” of “Annex C. Program Codes”. These scripts allow the patient to interact objects such as products, banknotes, coins…

In figure 8.100. all game objects of level 3 are shown:

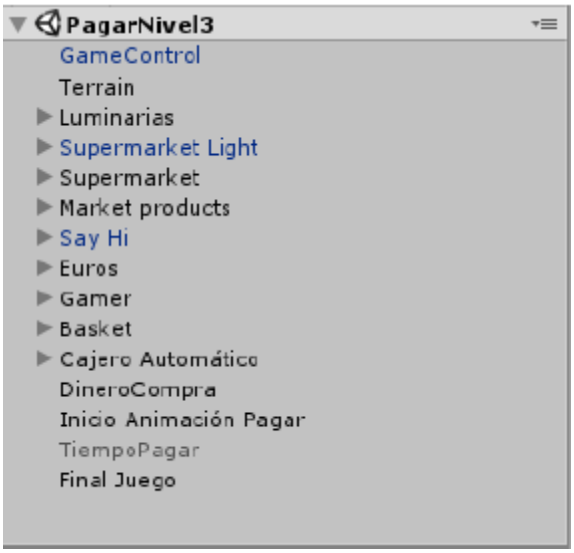


Figure 8.100. “Go shopping” level 3 objects (Source: Prepared by the authors)

The objects that make up the scene are:

- "GameControl"
- "Terrain"
- "Luminarias"
- "Supermarket light"
- "Supermarket"
- "Market products"
- "Say Hi"
- "Euros"
- "Gamer"
- "Basket"
- "Cajero Automático"
- "DineroCompra"
- "Inicio Animación Pagar"
- "Tiempo Pagar"
- "Final Juego"

As in every game of the application, "Game Control" object is where all data related to patient's performance is collected. The results can be seen in the Database, which is explained later in "9. Database" section.

"Terrain" and "Luminarias" game objects are the same as the ones explained before in "8.2.2. How does it work?" section of the tutorial.

"Supermarket Light" is also the same game object as in the other games. It contains the different lights points that illuminate the scene. Again, it was explained in several sections before.

"Supermarket" and "Market products" game objects are the same, as we have explained throughout the project, to make the application the most uniform as possible. However, there is a slight difference in "Supermarket" in relation to the other games that is the closed checkout signal.

Next up is "SayHi" object, which contains the cashier that appears in the checkout desk. This avatar is the same as the one used in "Shop assistant" game. However, this one does not move from the original place, it only moves its arms and head. Also, since it does not speak, the script which contains the audio is eliminated so that the avatar is only a decorative element as does not interact with the patient.

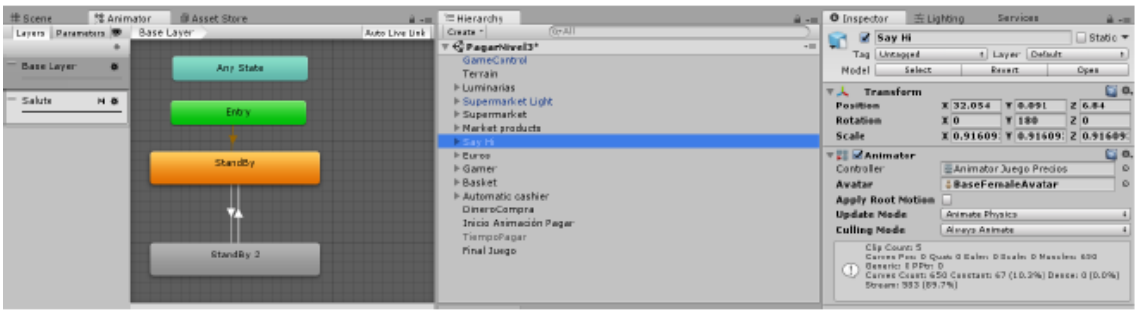


Figure 8.101. "SayHi" object animation (Source: Prepared by the authors)

The previous figure shows the sequence of movements the avatar does throughout the game.

"Euros" is the game object which has the banknotes and coins available for the patient to pay. The figure below shows the level 1 game object as in level 2 and 3 there are no coins when doing the payment:

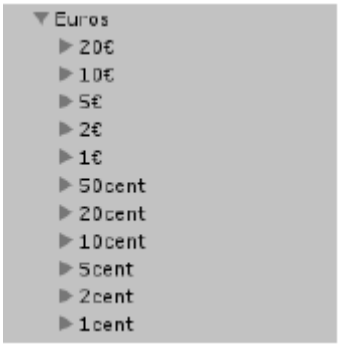


Figure 8.102. "Euros" level 1 game object (Source: Prepared by the authors)

"Gamer" game object is almost the same as the ones explained in previous games. Camera and controls are the same. As in all games, the main difference is found in "Camera (ears)". The fact that is a different game makes the audios and objects to be different. In the following picture this can be seen:

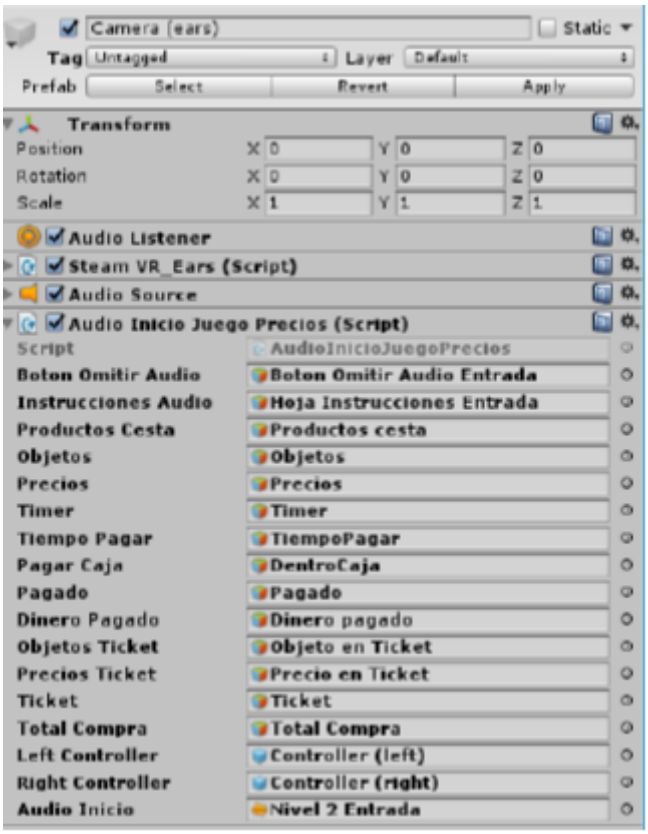


Figure 8.103. "Camera (ears)" object characteristics (Source: Prepared by the authors)

"MenuFollower" is also different. It contains the objects that patient sees on screen throughout the game.

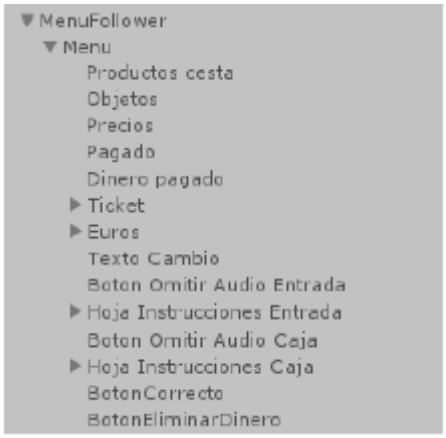


Figure 8.104. "MenuFollower" game object (Source: Prepared by the authors)

All objects from the figure above are briefly explained:

- "Productos Cesta", "Objetos" and "Precios" are the objects introduced in the basket with their corresponding price.
- "Pagado" and "Dinero pagado" are objects that contain the text that appear on the screen once the patient starts to introduce money in the automatic cashier.
- "Ticket" is the object that contains the ticket which appears once the patient enters at the payment zone. It shows the products, its price and the total price.
- "Euros" and "Texto cambio" are objects that only appear in level 2 and 3. The first one contains the banknotes and coins that appear once the patient has paid. The second one contains the text that is shown once the change appears on screen.
- "Boton Omitir Audio Entrada", "Hoja Instrucciones Entrada", "Boton Omitir Audio Caja" and "Hoja Instrucciones Caja". These objects are audio summaries and skip buttons from the initial and pre-payment audios.
- "BotonCorrecto" is the object in level 2 and 3 which contains the green button to accept the change.
- "BotonEliminarDinero", only in level 3, is the red button where the patient must drag the extra money from the change.

There are some differences in "Euros" and "BotonEliminarDinero" between levels 2 and 3. In level 2, "Euros" object has banknotes and coins as images so that they cannot be grabbed, while in level 3 these objects can be grabbed. In the following Figures this can be seen:

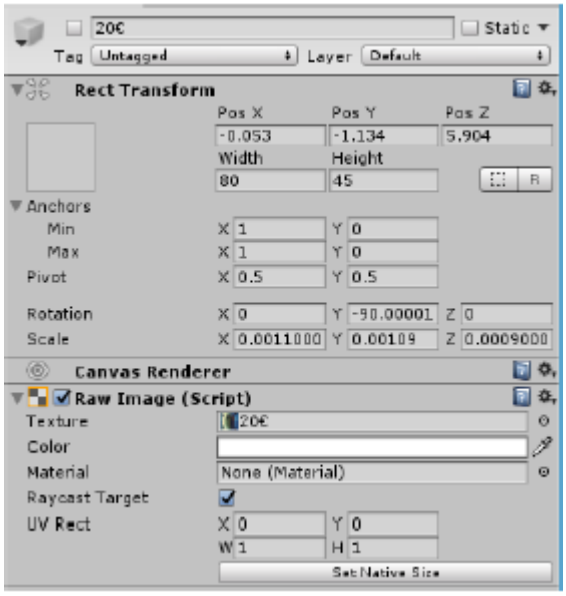


Figure 8.105. "Euros" characteristics of level 2 (Source: Prepared by the authors)

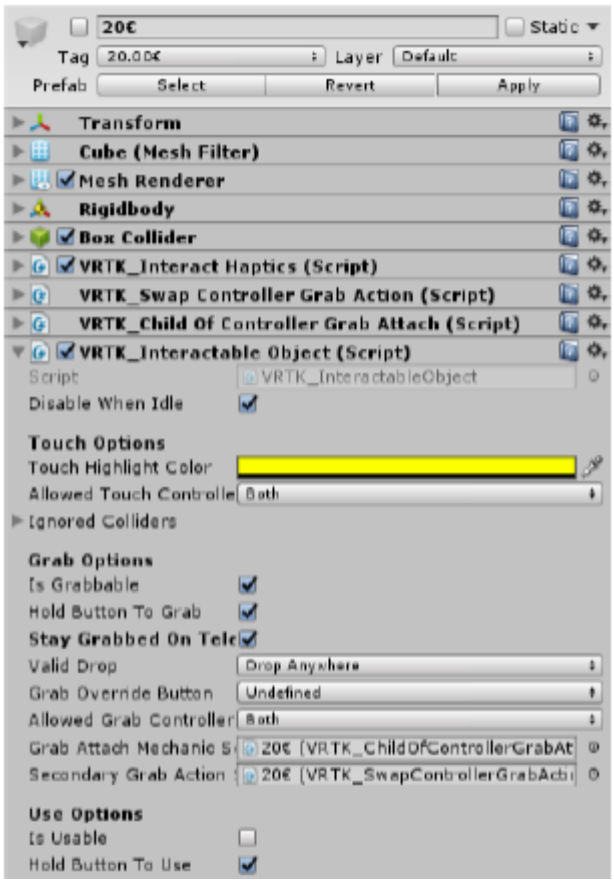


Figure 8.106. "Euros" characteristics of level 3 (Source: Prepared by the authors)

"BotonEliminarDinero" is different in level 2 and level 3. In the first of these two levels, this button has to be grabbed to activate it, while in the second of the two previous mentioned levels,

the extra money from the change has to be dragged there and it disappears from the screen. These differences can be seen below:

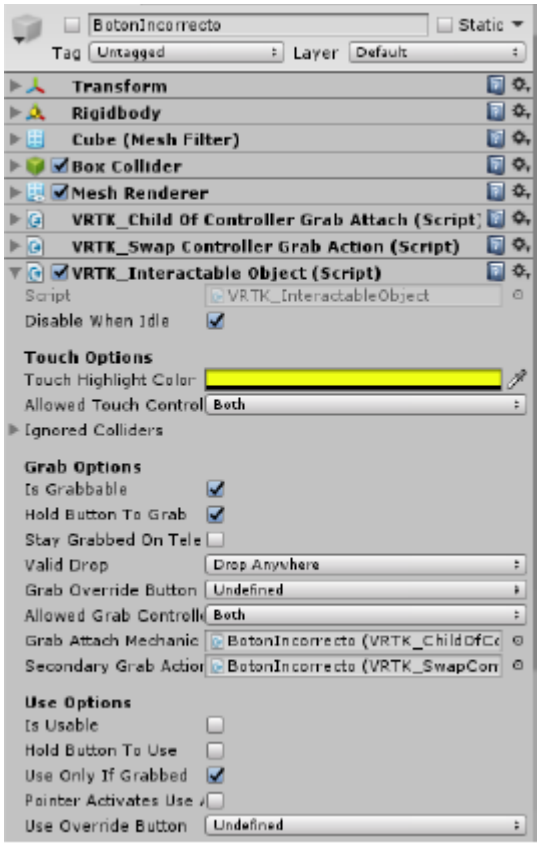


Figure 8.107. "BotonIncorrecto" characteristics of level 2 (Source: Prepared by the authors)

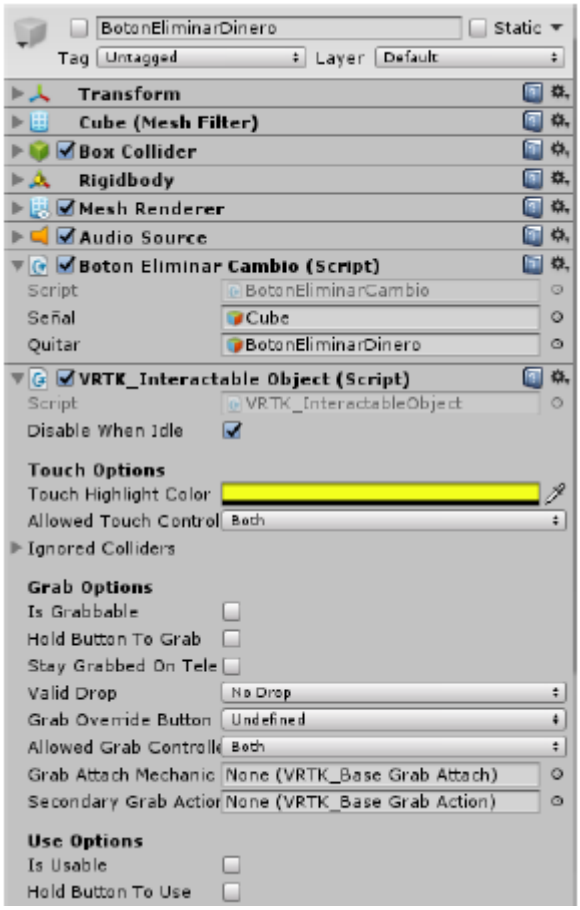


Figure 8.108. "BotonEliminarDinero" characteristics of level 3 (Source: Prepared by the authors)

```
void OnTriggerEnter(Collider other)
{
    if (other.tag != "Untagged")
    {
        other.GetComponent<Rigidbody>();
        //other.attachedRigidbody.isKinematic = true;
        other.gameObject.transform.parent = transform;
        {
            {
                if (other.tag == "20.00€")
                {
                    Valor = 20.00m;
                }
                if (other.tag == "10.00€")
                {
                    Valor = 10.00m;
                }
                if (other.tag == "5.00€")
                {
                    Valor = 5.00m;
                }

                        ...

                if (other.tag == "0.02€")
                {
                    Valor = 0.02m;
                }
                if (other.tag == "0.01€")
                {
                    Valor = 0.01m;
                }
            }
        }
        other.gameObject.SetActive(false);

        Diferencia = Diferencia - Valor;
    }
}
```

Figure 8.109. "BotonEliminarDinero" script code fragment (Source: Prepared by the authors)

The script shows the identification of the banknotes and coins once they are introduced in the automatic cashier. "Diferencia" is the variable whose value is the random additional money added to the correct change. "Valor" is the sum of all money introduced and is subtracted to "Diferencia", and when this last variable equals to 0, the patient has correctly finished the patient.

"Basket" game object is the same as the one explained in "8.3.3. How does it work?" section of this project. This object contains "Mesh2", which is where basket's shape and colors are defined.

Figure 8.110. "Basket" game object (Source: Prepared by the authors)

We can notice one difference between "Basket" object from the first game and this one. In this game we can find the script "CestaJuegoPreciosN2" located inside "DentroCesta", which includes the prices for the products and the sum of all prices. All this is shown on the screen so the patient can check the products introduced in the basket and their prices.

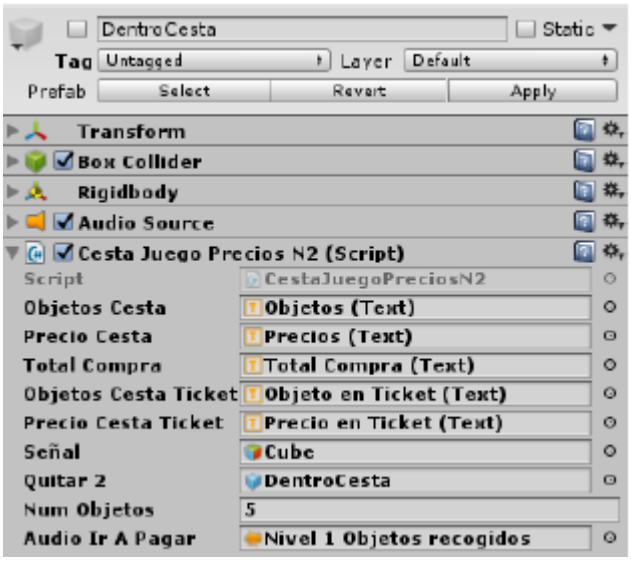


Figure 8.111. "DentroCesta" game object characteristics (Source: Prepared by the authors)

```
void Start ()
    {
    TotalCompra1 = 0.00m;
    }
    void OnTriggerEnter (Collider other) {

    if (other.tag != "Untagged")
    {
        other.GetComponent<Rigidbody>();
        //other.attachedRigidbody.isKinematic = true;
        other.gameObject.transform.parent = transform;
        {
            {
                if (other.tag == "Bistec")
                {
                    Precio = 4.50m;
                }
                if (other.tag == "Pizza")
                {
                    Precio = 2.50m;
                }
```

Figure 8.112. "CestaJuegoPreciosN2" script code (1) (Source: Prepared by the authors)

```
            if (other.tag == "Del")
            {
                Precio = 1.40m;
            }

        }
    }
    TotalCompra1 = TotalCompra1 + Precio;
    other.gameObject.SetActive(false);

    ObjetosCesta.text = string.Format("\n" + other.tag + ObjetosPrevios2);
    ObjetosPrevios = ObjetosCesta.text;
    ObjetosPrevios2 = ObjetosPrevios;

    Precio2 = Precio.ToString() + "€";
    PrecioCesta.text = string.Format("\n" + Precio2 + PreciosPrevios2);
    PreciosPrevios = PrecioCesta.text;
    PreciosPrevios2 = PreciosPrevios;

    ObjetosCestaTicket.text = ObjetosCesta.text;
    PrecioCestaTicket.text = PrecioCesta.text;
  }

}
```

Figure 8.113. "CestaJuegoPreciosN2" script code (2) (Source: Prepared by the authors)

From the previous Figure we can observe some parts of the programmed code: how prices are assigned to some of the products, how the total sum is calculated and how texts that are shown on screen are written. All the script can be found in ""C5. "Go shopping" programmed codes" of "Annex C. Program Codes".

"Cajero Automatico" object is the automatic cashier where the patient has to introduce the money to pay. There are two objects inside of it, as it is shown below:

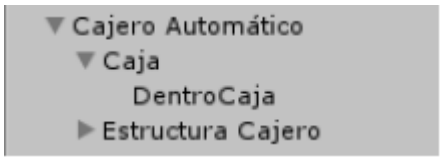


Figure 8.114. "Cajero Automático" game object (Source: Prepared by the authors)

"Estructura cajero" object is the one that defines the shape and colors of the automatic cashier.

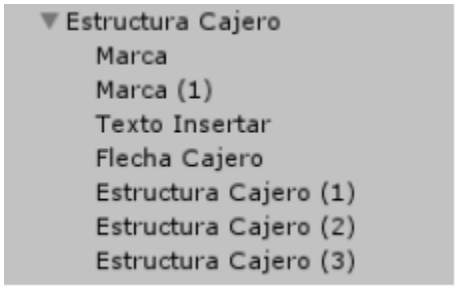


Figure 8.115. "Estructura cajero" game object (Source: Prepared by the authors)

One of the most important scripts is inside "DentroCaja". "PagarCaja3" is the one that controls which coins and banknotes enter the cashier and if the payment is correct or not. For levels 1 and 2 these scripts are "PagarCajaN1" and "PagarCajaN2" respectively. Thanks to these scripts, the

performance of the activity can be controlled. They also have the function of writing on the screen what coins or banknotes have been introduced in the automatic cashier as well as to show the random change in levels 2 and 3, which can be correct or not.

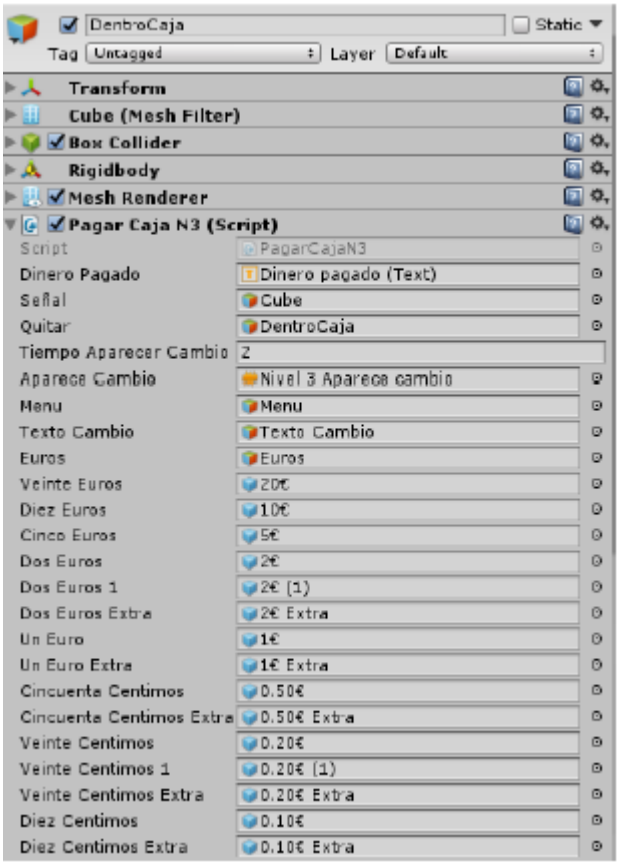


Figure 8.116. "DentroCaja" game object of the automatic cashier (Source: Prepared by the authors)

"DineroCompra" object and "Inicio Animación Pagar" game objects are two related objects that work in a similar way. Both are invisible walls, one next to the other, that are before the payment area.

Figure 8.117. "DineroCompra" and "Inicio Animacion Pagar" game objects (Source: Prepared by the authors)

The first of these objects, “DineroCompra”, has a script in charge of saving the total price of the purchase that is obtained from the script “CestaJuegoPreciosNX” (where “X” is the number of the level), contained in the “basket” object. This is because the “Basket” disappears once the patient enters to the payment area. This price is used in “PagarCajaNX” scripts (“X” is the number of the level) to know if the patient is doing the exercise good or not.

The second of the objects shown of the previous figure is “Inicio Animación Pagar”. It contains also a script that is responsible for the basket to disappear and activate the second explanatory audio along with the skip audio button and its summary. The ticket is displayed on the screen once the audio finishes or is omitted.

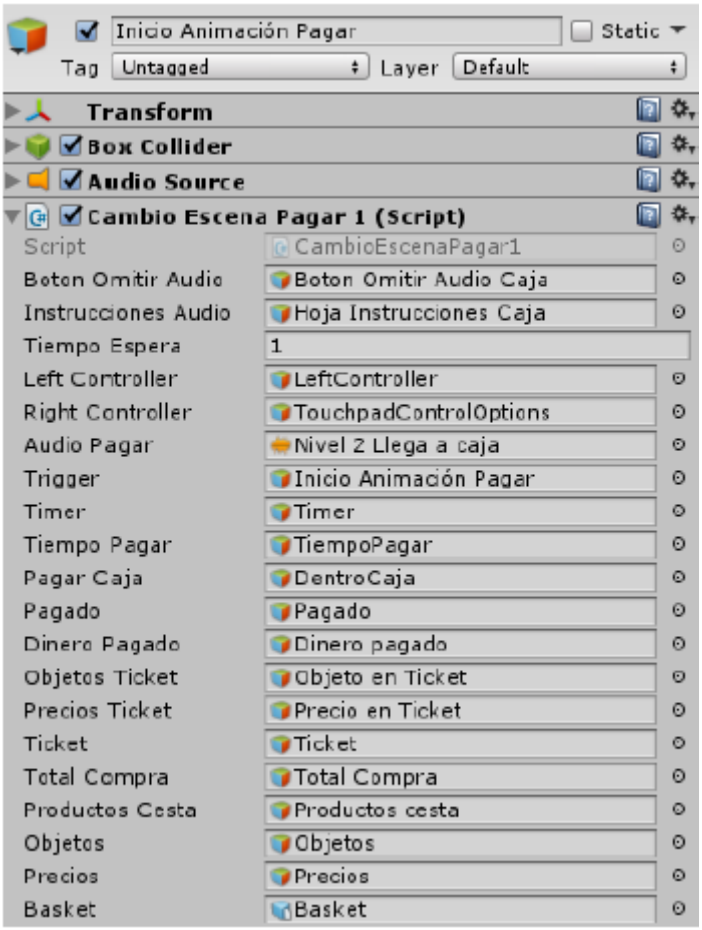


Figure 8.118. “Inicio Animación Pagar” game object (Source: Prepared by the authors)

“TiempoPagar” is where the script that has to control the time the patient takes to pay is located.

The last game object of this game is “Final Juego”, which is an invisible wall, located at the exit door. It has an script that closes the scene when the activity is completed and the patient has collided with it. After that, the final menu is shown.

Figure 8.119. "Final Juego" game object (Source: Prepared by the authors)

8.7. Game controls

One of the most important device for the VR application is the game controller. In this case, there are two of them, one for each hand. Taking into account that most of the patients who use this application have not used this technology, game controls are as simple as possible. Only four buttons are used: 3 of them are in the right controller and the other one is on the left controller.

Game controls are explained to the patient with an audio in the tutorial. Also, while the patient is listening to this audio, while explaining each button/control, an image referenced to this button will be shown.

Right and left controllers are the same. There is an image below with the number of each button and its referenced explanation:

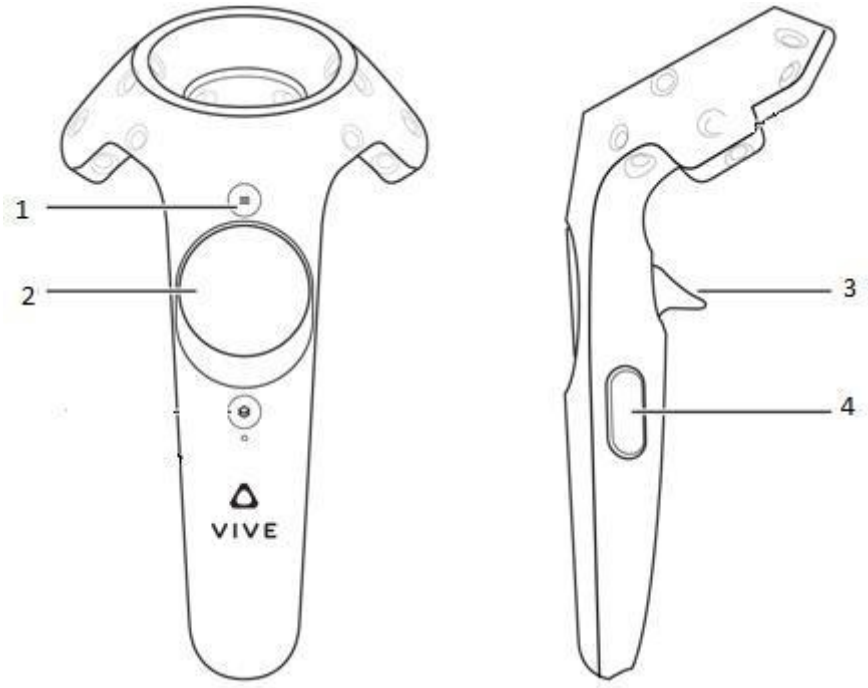


Figure 8.120. Controller (Source: Prepared by the authors)

- **1:** This is known as the "Menu button" and it is the only button the patient has to press in the left controller, in levels 2 and 3 of the "Shopping List" game. With this button the patient can have a look the list of products whenever they want. Once this button is pressed, the list appears on the screen for 15 seconds. The number of times that this button is pressed is collected in the database.
- **2:** This is the "Trackpad" button. Pressed in the right controller, allows the patient to move inside the supermarket environment in the whole application. Depending on the part of the trackpad that is pressed, the patient will go in one or another direction. So, if the user presses the upper part of the button, the avatar will move forward. If the user presses the bottom of the "Trackpad", it will move backwards. If the patient presses the right or left part of this button, the avatar will move to the right or to left direction.
- **3:** This is the "Trigger" button. This is the button used to select and drag the different products in the supermarket to the basket. The user has to approach the right controller to the product they want to take and, once it is highlighted, hold this button to pick it up. If the user releases the button, the product falls because of the gravity. Moreover, this button is also used to press the different buttons that appear in the screen during the "Go shopping" game mode. For example, to press the skip button.
- **4:** This is a particular button because it is only used in the tutorial. When this button is pressed in the right command the last instruction of the scene is repeated. This button is useful at the beginning since the patient can be nervous, and they can miss some important information of the instructions.

9. Database

The database is one of the important parts of the application, as it is the tool that allows doctors to follow patient's progress. It is where the results of each of the patients is collected and stored. In order not to use patients' names, they are all identified by an identification number (ID).

For this data collection, it "MySQL" program has been used. This software is developed by "Oracle" and "Microsoft SQL Server" and it is one of the most used programs in the world for the creation of databases, as it is easy to use and understand.

First of all, the program has to be installed in "Unity". Once it is installed, it is necessary to create a list of the variables included which will be in the database, indicating for each of them the type of variable it is. This is possible to make it with "DB Browser" program, as thanks to it the doctor can see and sort the data obtained and stored through "MySQL". That is why the list of variables has to be created in "DB Browser".

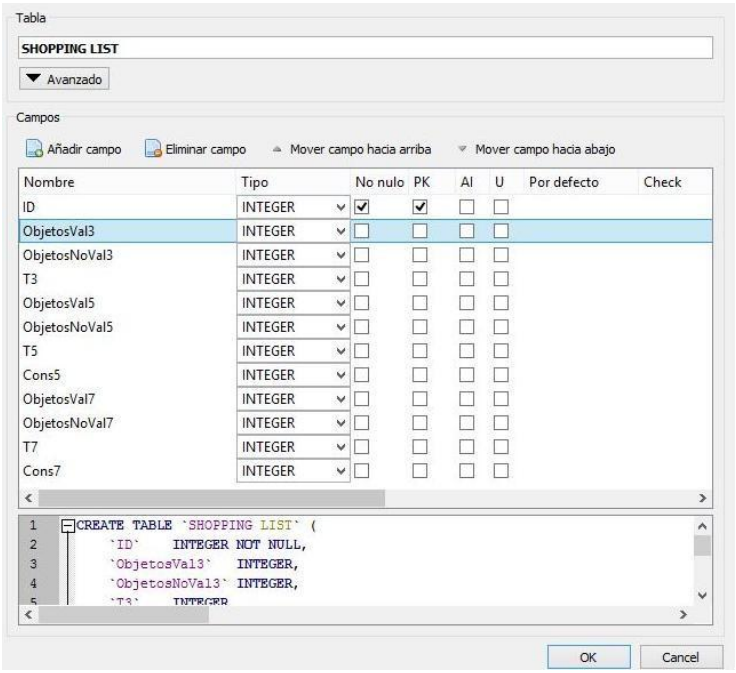


Figure 9.1. Some of the variables used in the app in "DB Browser" (Source: Prepared by the authors)

After this, it is necessary to connect it with the application. To do that, it is needed a script in which the collection of the different variables for each level is programmed in C# language. This script is included in the game object "GameControl", and appears in every scene of the application, as it has to collect all the data.

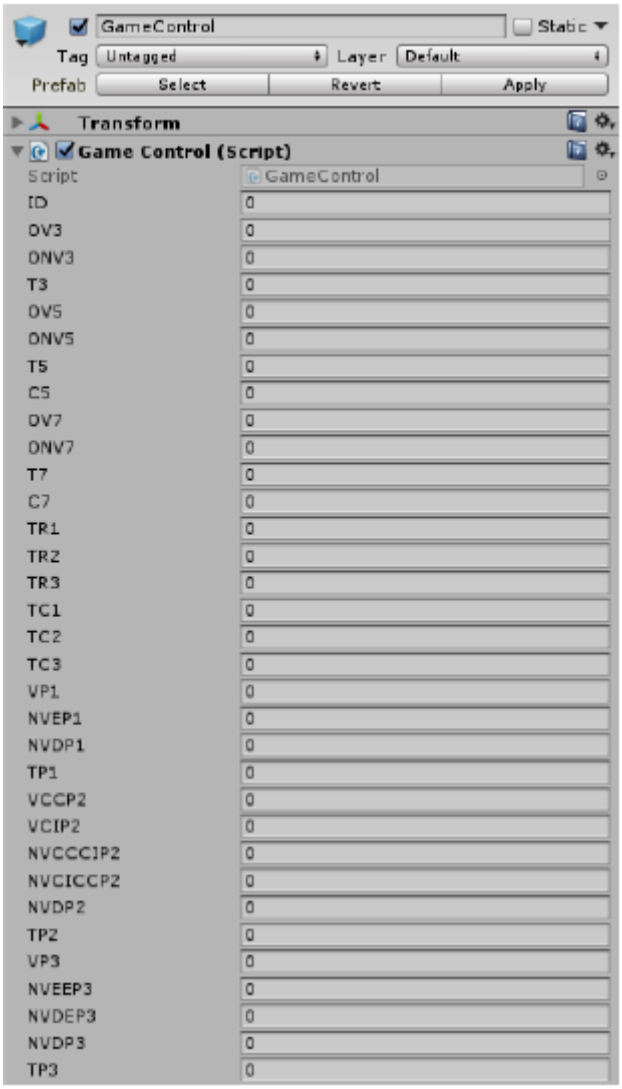
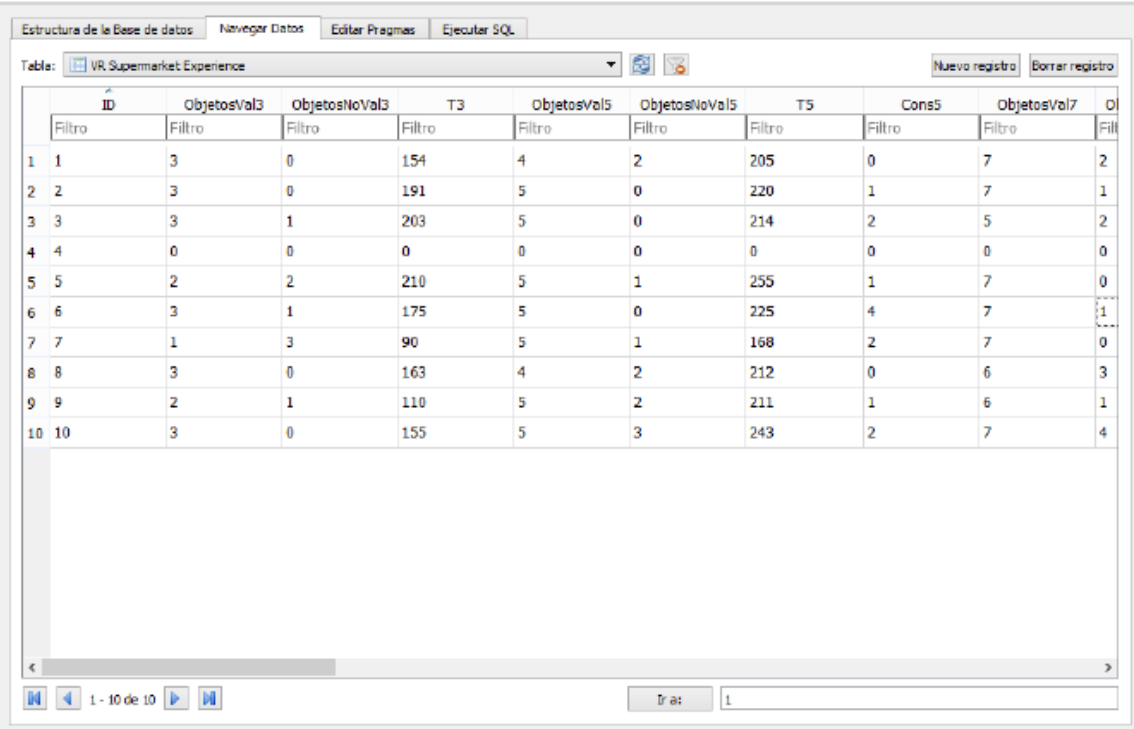


Figure 9.2. "GameControl" game object (Source: Prepared by the authors)

As it can be seen, all the variables of the application are included in this game object. All these variables will be explained in the next section. All data is collected in "GameControl" object while the patient is performing the exercise. Once the patient finishes the game and the doctor presses the "Salir" button in the "Final Menu" scene, the collected data is saved and stored in the database.

The database is read by the doctor with "DB Browser" program. In addition to that, results can also be exported to a ".csv" format, which allows the doctor to see and sort the results with, for example, "Microsoft Excel".



	ID	ObjetosVal3	ObjetosNoVal3	T3	ObjetosVal5	ObjetosNoVal5	T5	Cons5	ObjetosVal7	O
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro	Fil
1	1	3	0	154	4	2	205	0	7	2
2	2	3	0	191	5	0	220	1	7	1
3	3	3	1	203	5	0	214	2	5	2
4	4	0	0	0	0	0	0	0	0	0
5	5	2	2	210	5	1	255	1	7	0
6	6	3	1	175	5	0	225	4	7	1
7	7	1	3	90	5	1	168	2	7	0
8	8	3	0	163	4	2	212	0	6	3
9	9	2	1	110	5	2	211	1	6	1
10	10	3	0	155	5	3	243	2	7	4

Figure 9.3. "DB Browser" program with data collected from "Shopping list" game (Source: Prepared by the authors)

The previous figure is an example of the data collected in a game. As it can be seen, there are 10 ID that represent 10 patients which have done all three levels. All data can be sorted according to any of the variables.

At last, there is also a "filter" in which the doctor has the possibility to look for specific information by searching a key word or number. This can be useful for some situations in which there are a lot of patients and all of them do not appear on the screen.

9.1. Data collected

All data collected in the application is considered to be important for the subsequent study. Doctors of "Hospital Clínic" indicated to previous students of the project what data has to be collected in several games, such as "Shopping list", "Shop assistant" and "Autonomous communities". For the game "Go shopping", the data collected is the one considered interesting for the doctors, as it was not possible to arrange a meeting with them.

The different variables that appear in "Game control", and that can be seen at figure 9.3, will be explained below, sorted according to the game mode in which they appear.

In the "ID menu" only a variable is collected:

- **ID:** The identification number of the patient that will perform the activities. It is the most important one.

As tutorial's goal is that the patient can learn about how controllers work and get used to the supermarkets' environment, no data is collected.

In the "Shopping List" game mode, the data collected is:

- **OV3: "**ObjetosVal3" in "MySQL". This variable indicates the number of products correctly introduced in the basket in the first level. The maximum are 3 products.
- **ONV3:** "ObjetosNoVal3" in "MySQL". This variable indicates the number of products incorrectly introduced in the basket in the first level.
- **T3:** Also "T3" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the first level until it leaves the door to finish it.
- **OV5: "**ObjetosVal5" in "MySQL". This variable indicates the number of products correctly introduced in the basket in the second level. The maximum are 5 products.
- **ONV5:** "ObjetosNoVal5" in "MySQL". This variable indicates the number of products incorrectly introduced in the basket in the second level.
- **T5:** Also "T5" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the second level until it leaves the door to finish it.
- **C5:** "Cons5" in "MySQL". This variable posts the number of times that the user consults the shopping list in the second level.
- **OV7: "**ObjetosVal7" in "MySQL". This variable indicates the number of products correctly introduced in the basket in the third level. The maximum are 7 products.
- **ONV7:** "ObjetosNoVal7" in "MySQL". This variable indicates the number of products incorrectly introduced in the basket in the third level.
- **T7:** Also "T7" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the third level until it leaves the door to finish it.
- **C7:** "Cons7" in "MySQL". This variable posts the number of times that the user consults the shopping list in the third level.

In the "Shop Assistant" game mode, the data collected is:

- **TR1:** Also "TR1" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the first level until it puts the three products in their correct shelves.
- **TR2:** Also "TR2" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the second level until it puts the four products in their correct shelves.
- **TR3:** Also "TR3" in "MySQL". This variable indicates the elapsed time in seconds since the patient begins the third level until it puts the six products in their correct shelves.

In the “Autonomous Communities” game mode, the data collected are:

- **TC1:** Also “TC1” in “MySQL”. This variable indicates the elapsed time in seconds since the patient begins the first level until it puts the three products in their correct stands.
- **TC2:** Also “TC2” in “MySQL”. This variable indicates the elapsed time in seconds since the patient begins the second level until it puts the four products in their correct stands.
- **TC3:** Also “TC3” in “MySQL”. This variable indicates the elapsed time in seconds since the patient begins the third level until it puts the six products in their correct stands.

Finally, in the “Go shopping” game mode, the data collected are:

- **VP1: “**ValP1” in “MySQL”. This variable indicates “1” if the patient paid the price of the purchase exactly in the first level.
- **NVEP1:** “NoValExcesoP1” in “MySQL”. This variable indicates “1” if the patient paid more than the price of the purchase in the first level.
- **NVDP1:** “NoValDefectoP1” in “MySQL”. This variable indicates “1” if the patient paid less than the price of the purchase in the first level.
- **TP1:** Also “TP1” in “MySQL”. This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the first level until it leaves the door to finish it.
- **VCCP2: “**ValCambioCorrectoP2” in “MySQL”. This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was correct, so the patient pressed the green button for correct change in the second level.
- **VCIP2:** “ValCambioIncorrectoP2” in “MySQL”. This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was incorrect, so the patient pressed the red button for incorrect change in the second level.
- **NVCCCIP2: “**NoValCambioCorrectoP2” in “MySQL”. This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was correct, but the patient pressed the red button for incorrect change in the second level.
- **NVCICCP2:** “NoValCambioIncorrectoP2” in “MySQL”. This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was incorrect, but the patient pressed the green button for correct change in the second level.
- **NVDP2:** “NoValDefectoP2” in “MySQL”. This variable indicates “1” if the patient paid less than the price of the purchase, so they paid incorrectly, in the second level.
- **TP2:** Also “TP2” in “MySQL”. This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the second level until it leaves the door to finish it.
- **VP3:** “ValP3” in “MySQL”. This variable indicates "1" if the patient paid correctly and, after the random change appeared, they guessed which coins had to be eliminated in the third level.

- **NVEEP3: “**NoValExcesoEliminarP3” in “MySQL”. This variable indicates "1" if the patient paid correctly and, after the random change appeared, they eliminated more coins than necessary in the third level.
- **NVDEP3:** “NoValDefectoEliminarP3” in “MySQL”. This variable indicates "1" if the patient paid correctly and, after the random change appeared, if they eliminated fewer coins than necessary in the third level.
- **NVDP3:** “NoValDefectoP3” in “MySQL”. This variable indicates “1” if the patient paid less than the price of the purchase, so they paid incorrectly, in the third level.
- **TP3:** Also “TP3” in “MySQL”. This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the third level until it leaves the door to finish it.

10. Results

10.1. Improvements and changes in the application

One of the main problems of the application was the excessive weight it had. This fact makes the app to start slowly and unable the possibility to start it in a mobile phone or in a computer not too sophisticated as the one we have on the laboratory.

Unity3D have different ways to create objects. One of this ways is creating different polygons in the scene and then adding a texture, but this method is very heavy for an application like ours. If we need to duplicate the same object we can do it in two different ways, creating another group of polygons and textures or creating a prefab from the first one, which means a lower use of memory and a lighter app.

Not only objects make the weight of the application, it is very important to take care about the textures used for each one and the resolution. Very heavy textures and a very high resolution mean a slower app and a more weight.

At last, we have to take care of something very obvious, that the app is a continuation of other projects and, maybe, not everything we have inside the app is used nowadays. This means more weight and more processes to run.

So the main changes in the application are explained below:

10.1.1. Prefabs

Is a system used by Unity3D to create, store and configure different objects without the necessity of redoing them in every scene, and avoiding the fact of duplicating the weight.

Prefabs also allow us to save the configuration of a GameObject. This helps the user to do not have to reconfigure every object every time we need to use it.

In our project this is a very important tool because inside the supermarket we have our products duplicated several times. This means that a big amount of the weight of our application is created by all the products created lots of times.

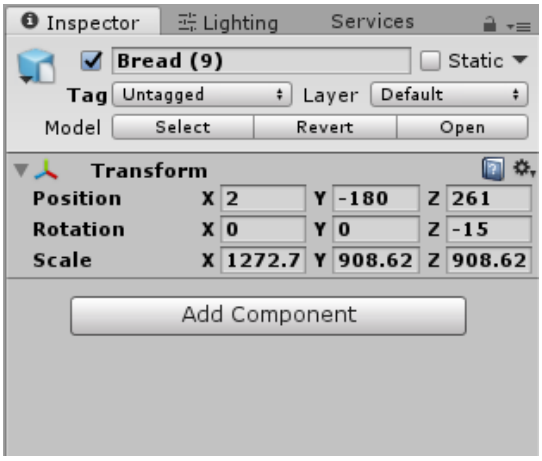


Figure 10.1. Model of "Bread" product that was in previous projects (Source: prepared by the authors)

To reduce weight we have created every product again and we made it as a prefab. After creating all the products we had to put them all again on the shelves and make all the supermarket for every scene. As a result, we made all the products of supermarkets for all twelve levels again, with every object as a prefab. We also created again all walls, terrain, lights… and all the different stuff of the supermarket to have the entire scene made with prefabs, making every scene the lightest possible.

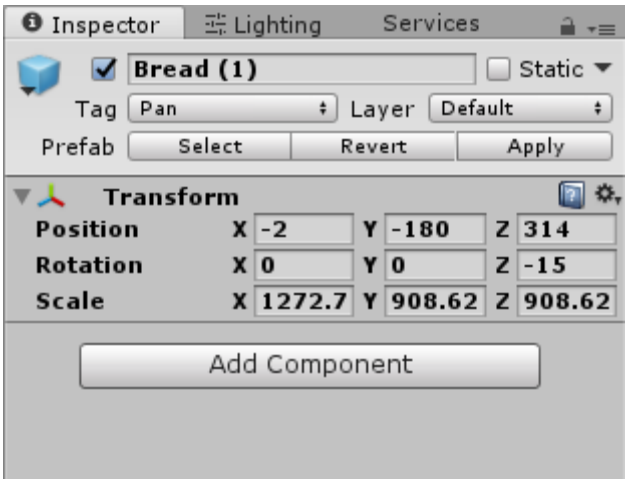


Figure 10.2. Actual prefab "Bread" product (Source: Prepared by the authors)

10.1.2. Grabbable products and other characteristics

As we said before, creating a prefab also means saving the configuration of every product, that is why we created the prefab based on the first product of every shelve. By doing this we have saved inside the prefab the grabbable condition, the colliders and the gravity.

Thanks to this we made every product in the supermarket interactable with each other. And now all products are affected by the gravity.

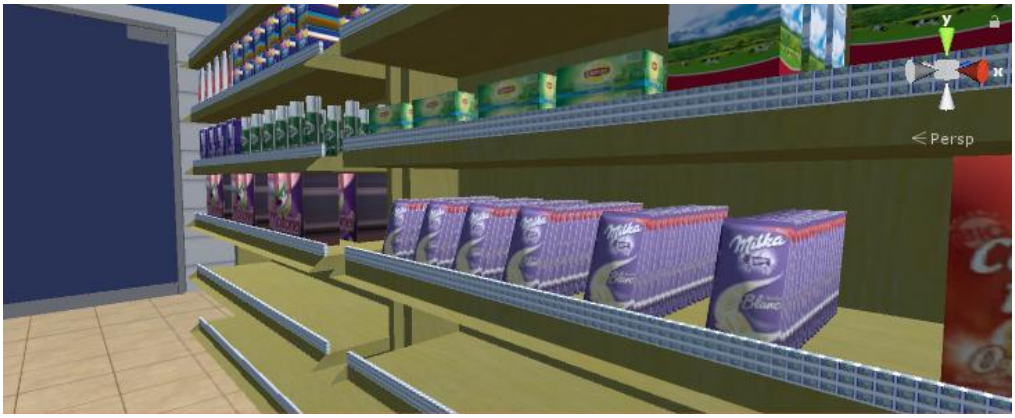


Figure 10.3. "Chocolate" product without gravity (Source: Prepared by the authors)

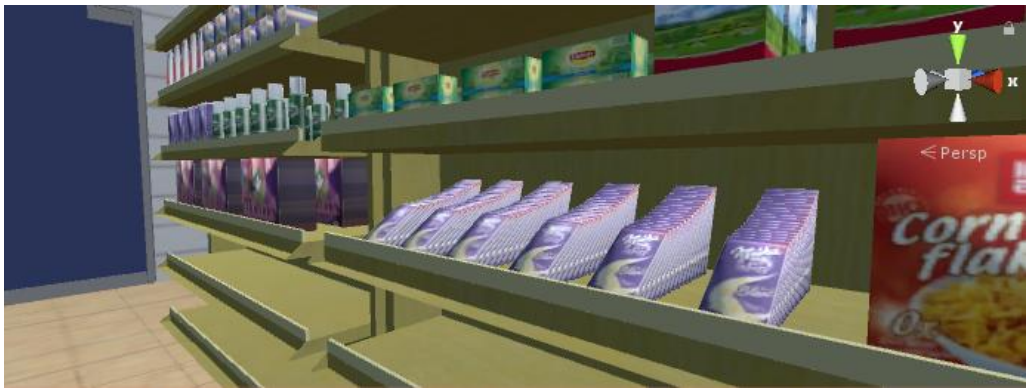


Figure 10.4. "Chocolate" product affected by the gravity (Source: Prepared by the authors)

We have to notice that, in the previous project, only the first row of every product was grabbable and the rest of them were only images. Thanks to this we improved the immersion on the app because the user can grab something from the back and if it collides with another, this will move like in real life.

10.1.3. Correction in colliders

When creating all the prefabs, we noticed that some objects with not uniform forms had colliders bigger than they needed. This means that, in the moment the game starts and the gravity has effect, products look like they are floating.

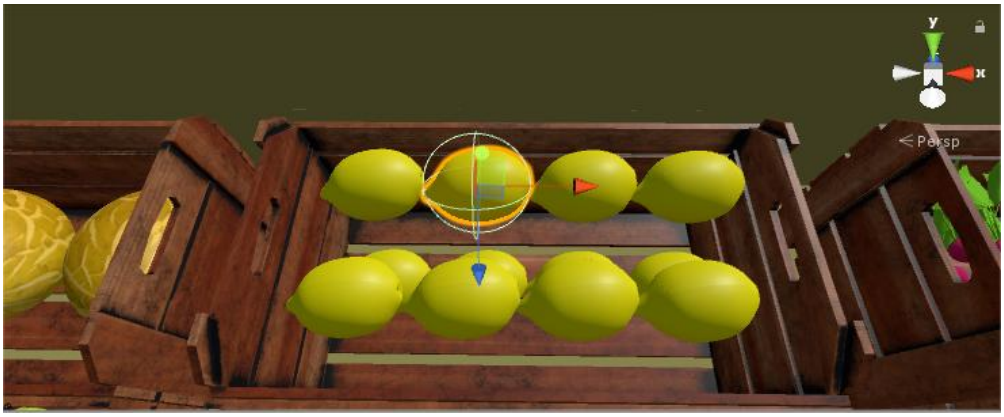


Figure 10.5. Sphere collider used in previous projects (Source: Prepared by the authors)

Figure 10.6. "Lemon" product when is affected by gravity (Source: Prepared by the authors)

To correct this situation and help to improve immersion, we created new colliders more adjusted to the real form of the objects. After creating these new colliders we can notice that after starting the game, objects look like they are touching the other.

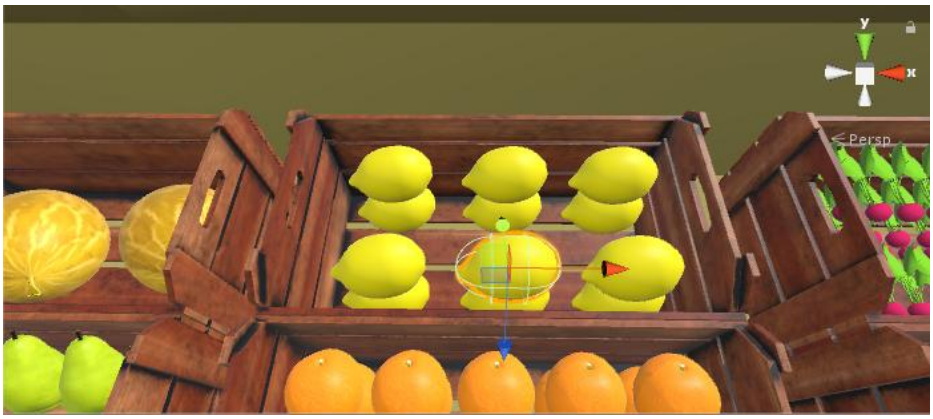


Figure 10.7. New collider in "lemon" product (Source: Prepared by the authors)

10.1.4. Uniformity on the supermarket's environment

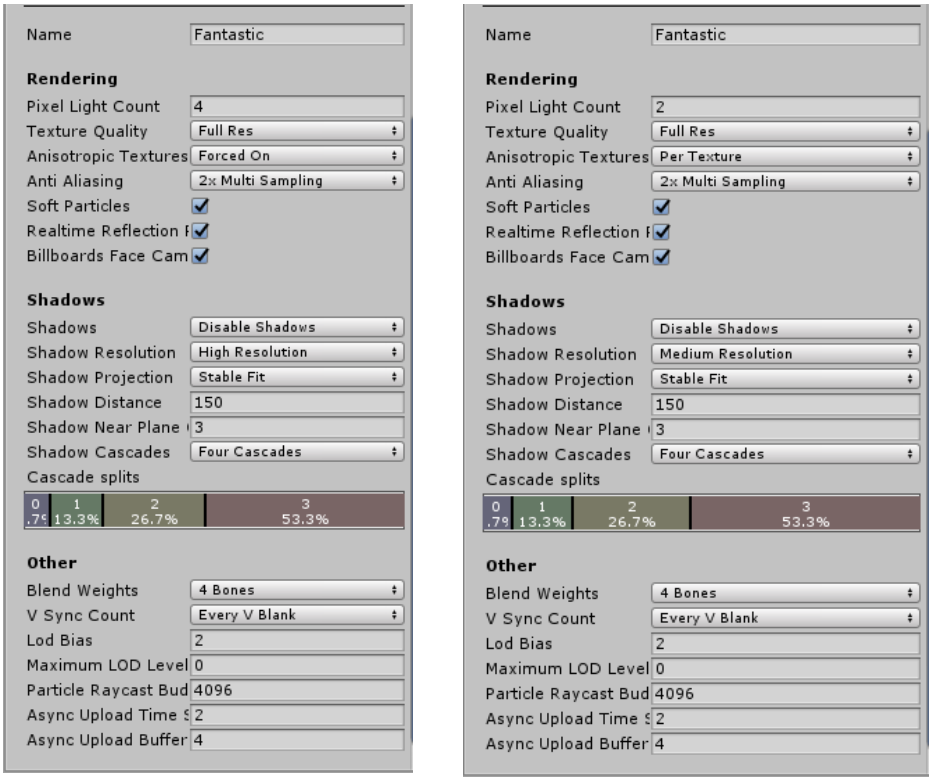
Creating everything with prefabs also helped us to make every scene equal to the others, so this will help the users to recognize every scene easily. Things like the position of all products, the height of the fruit and vegetable section or the textures are now exactly equal for all twelve levels and for all twelve final scenes.

Figure 10.8. Old vegetables and fruits section (Source: Prepared by the authors)

Figure 10.9. New vegetables and fruits section (Source: Prepared by the authors)

10.1.5. Resolution and shadows

To reduce even more weight on the application we looked at the resolution we were working with. We noticed that we were working with full resolution on the objects and the shadows. This full resolution configuration made the application to work slowly because the app was running all the objects at the same time.



Figures 10.10. and 10.11. On the left side, the old resolutions and shadows. On the right side, the new ones. (Source: Prepared by the authors)

To correct this situation, we changed some values on the resolution configuration. First of all we reduced the pixel light count from four to two. We changed the anisotropic textures from forced on to per texture. This changes made our application to have a perfect resolution in all the products the costumer had in front but the system do not run the objects we have far away from us, reducing the start time and the use of memory.

Talking about the shadows we disabled them and we changed their resolution from high to medium. We consider that shadows are important for the immersion but it is not necessary to run them in full resolution.

10.1.6. Reduction of weight, CPU usage and RAM memory

After making all this changes on our application we found a very good result on our weight reduction work. They were made in a total of twenty eight scenes (12 levels, 12 final scenes of the games, first and second part of the tutorial, ID and Menu) and made a change in the weight of the application, the use of the CPU and the use of memory.

We started with a weight of 5.55 Gigabytes (5.965.191.364 bytes), a use of the CPU of 53% and a use of memory of 53% too. We also can look for a memory use inside the Unity3D program, at the beginning was 2.00 Gigabyte.

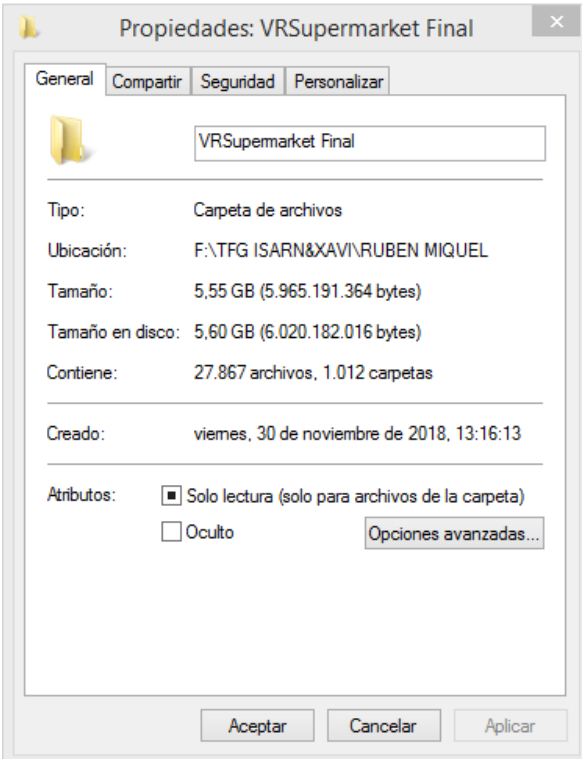


Figure 10.12. Previous weight of the application (Source: Prepared by the authors)

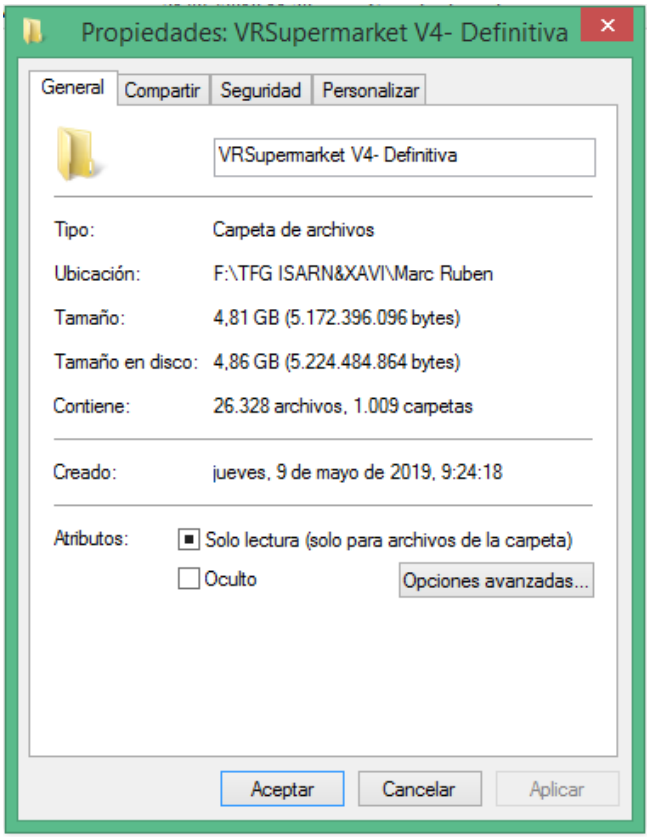


Figure 10.13. Actual weight of the application (Source: Prepared by the authors)

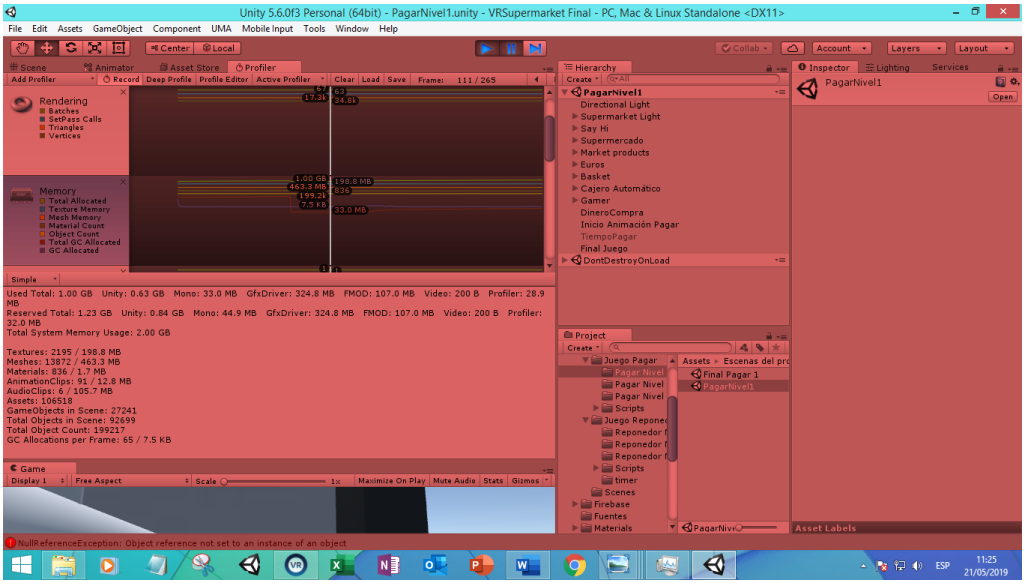


Figure 10.14. Memory used inside Unity3D before all changes (Source: Prepared by the authors)

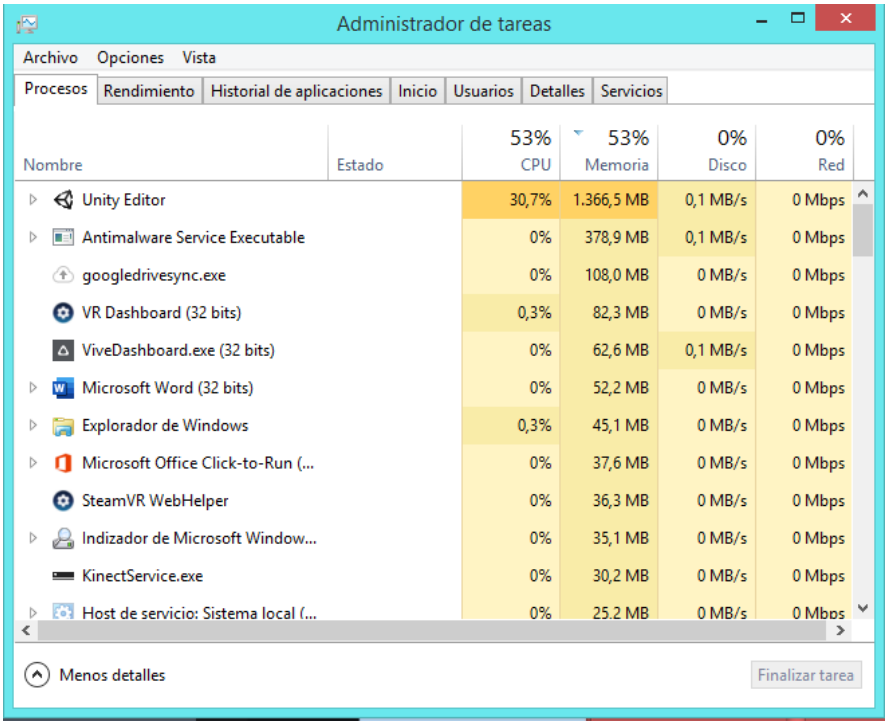


Figure 10.15. Task Manager of Windows before the changes (Source: Prepared by the authors)

After all the changes made, we finished with a weight of 4.81 Gigabytes (5.172.396.096 bytes), a use of the CPU of 59% and a use of memory of 48%. Again we can look the memory use of the program, 1.90 Gigabytes after the weight reduction.

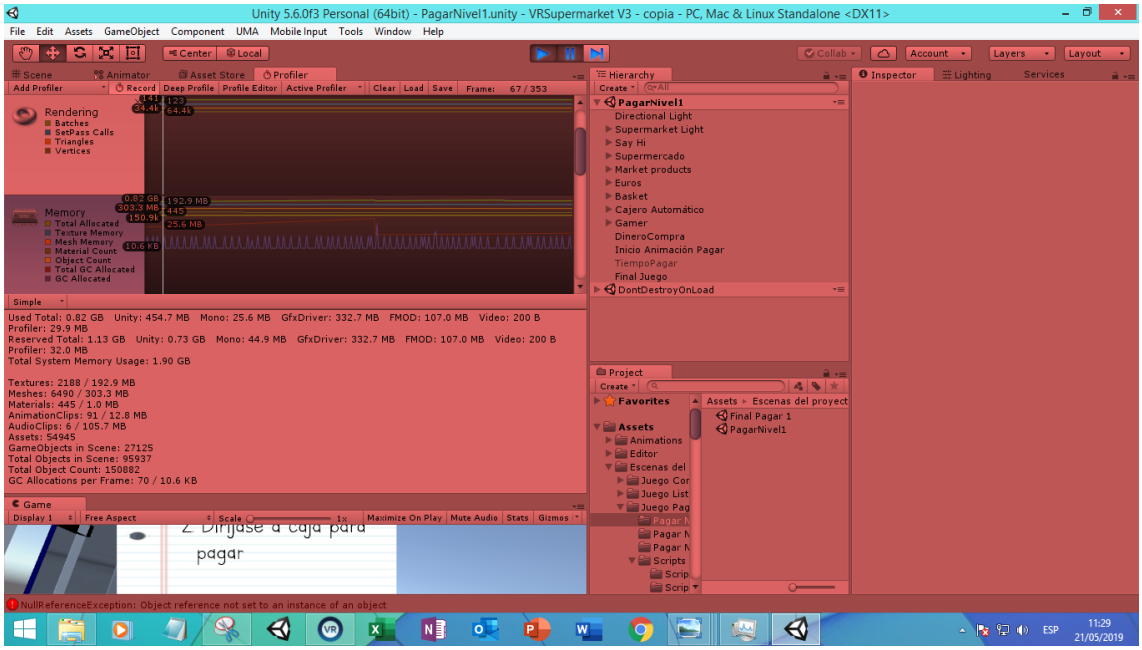


Figure 10.16. Memory used inside Unity3D after all changes (Source: Prepared by the authors)

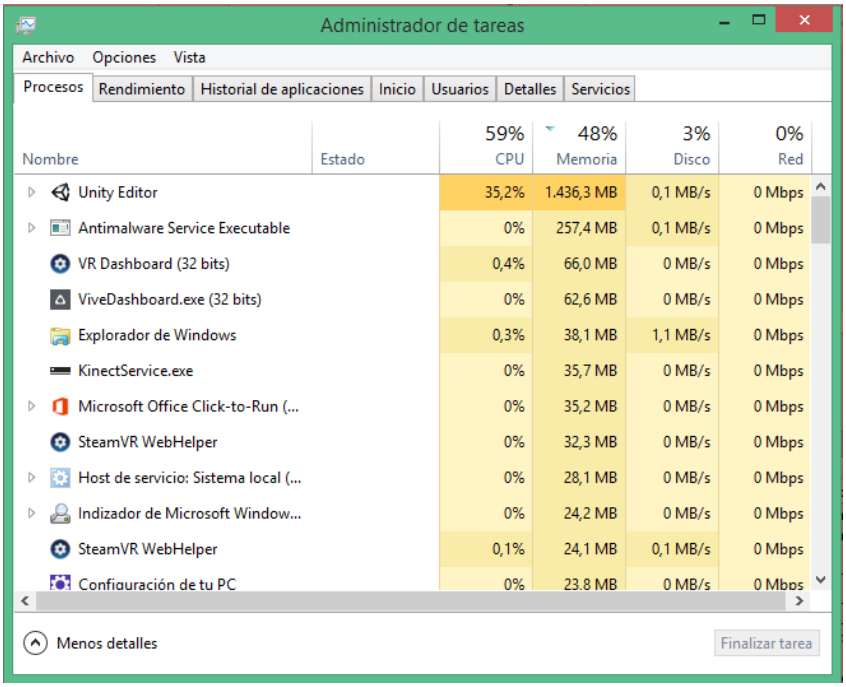


Figure 10.17. Task Manager of Windows after the changes (Source: Prepared by the authors)

As we can see we have got a total reduction of 0.74 Gigabytes, about a 13.3% of the initial weight. We also noticed a reduction on the use of the memory by 5%, but this is difficult to calculate because the use of CPU and memory also counts other programs running at that moment.

10.2 ADFO test

Something very important for us is to know if our application is really usable for real patients with different types of cognitive diseases, that's why, after improving the application and test it inside our laboratory, we tested it with real patients in ADFO (Associació de Disminuïts Físics d'Osona).

We moved to Vic on 30th May of 2019 to test the application on eight real patients. All of them had suffered from stroke disease but each one presented different illnesses depending on the part of the brain the stroke had affected more.

We met patients with difficulties to understand our language, others with non-functional limbs or vision problems. All these different situations were something we haven't noticed in the laboratory and this helped us to think about the future lines of our project.

During the morning we tested the application with patients helped by a specialized physiotherapist on these types of illnesses, who helped us a lot to know how to treat with the patients and how to use the glasses depending on the illness of each patient.

10.2.1 Recognition of diseases

Now we are going to recognize every disease we saw in each patient, to help us understand their feelings among the application and the results of the game after test it.

Patient 1	**Disease:**
	Stroke Disease
Problems:	This patient had several difficulties to understand us because he didn't understand Spanish and English properly. She had a little affectation on her left hand, making a little bit difficult to grab the control.
Results:	This patient started with difficulties to understand the application but with a little help from us and the physiotherapist was able to grab objects and move for the supermarket. After the game she told us that she felt tired.

Table 2. Patient 1 difficulties and considerations (Source: Prepared by the authors)

Patient 2	**Disease:**
	Stroke Disease
Problems:	This patient had several affectations all the left part of his body which made difficult for them to grab one controller. By the other hand, he was wearing big glasses and he could not put the 3D glasses and his own ones. This made difficult for him to read the list. At last, this patient had mobility problems and he had to play sat down.
Results:	The patient was very motivated and this helped him to understand the application. Being sat down created a new problem because this game is created to play on foot, and he felt it was very short. At the end, the patient was able to move and recognize some objects.

Table 3. Patient 2 difficulties and considerations (Source: Prepared by the authors)

Patient 3	**Disease:**
	Stroke Disease
Problems:	This patient showed some comprehension difficulties that made it hard to teach him how to use the controllers. He also have a little affectation to his left side, that made difficult for him to grab the controller.
Results:	This patient was highly motivated and that helped us a lot to make him play with the application. He had several difficulties to use the controller but with the help of the physiotherapist he was able to use the finish the level.

Table 4. Patient 3 difficulties and considerations (Source: Prepared by the authors)

Patient 4	**Disease:**
	Stroke Disease
Problems:	The patient had an affectation on his left side that made difficult for him to grab the controller. He also had an affectation in speech that made difficult for us to communicate with him.
Results:	The patient showed some difficulties with the glasses because he did not see properly. We could solve it adjusting the glasses. He was able to walk through the supermarket and grab objects with a bit of help.

Table 5. Patient 4 difficulties and considerations (Source: Prepared by the authors)

Patient 5	**Disease:**
	Stroke Disease
Problems:	This patient had an affectation to her left side, this made difficult for her to grab the controller. She needed help from the physiotherapist to grab the controller.
Results:	The patient was the first one to solve the level by her own, thanks to her high motivation. Despite her difficulties to grab the controller she was able to find the basket properly.

Table 6. Patient 5 difficulties and considerations (Source: Prepared by the authors)

Patient 6	**Disease:**
	Stroke Disease
Problems:	This patient had a mobility affectation and he needed to play sat down. He also had an affectation on his left side causing problems to grab the controller
Results:	Despite being sat down the patient could move around the supermarket and grab objects from the lower shelves. The patient had difficulties to find the basket because of the low mobility of her left hand.

Table 7. Patient 6 difficulties and considerations (Source: Prepared by the authors)

Patient 7	**Disease:**
	Stroke Disease
Problems:	This patient also had a mobility disease and had to play sat down. He had her left hand with low mobility but he could lift her arm.
Results:	The patient was able to move for the supermarket and grab objects with not many difficulties despite being sat down. The patient could find the basket easily and was able to complete the level. The patient showed great motivation for the game.

Table 8. Patient 7 difficulties and considerations (Source: Prepared by the authors)

Patient 8	**Disease:**
	Stroke Disease
Problems :	We did not notice any great physical disease, only the need to wear glasses.
Results:	The patient solved the level with facility, she did not have any difficulties to use the controllers or move through the supermarket.

Table 9. Patient 8 difficulties and considerations (Source: Prepared by the authors)

As we can see not only cognitive diseases were found, many patients had aftermaths due to the stroke disease and that's something we did not think about in the laboratory.

10.2.2 Surveys

Not only know that the application runs well it is important, we also wanted to know if patients we comfortable with the glasses or if they noticed any dizziness during the game. We consider that this application will not be functional if patients are not comfortable with the platform and that is why we made a Survey to each patient.

As we can see in the survey made to the patients almost everyone felt comfortable with the glasses and anyone feel any kind of dizziness, they were very motivated and this helped us to work better.

We also can see that everyone would play with this game more often and they only said that would like to practice more. We think that these eight patients with the correct training and more days playing would be able to play with it properly because almost everyone found quite easy to use the controllers.

In general lines everyone recognized the products and thinks that there is nothing to change in the application.

After this session with real patients we noticed new necessities for our application and a few changes to do. We were unable to do this changes because the proximity to the end of our dissertation but we will explain it on future lines.

	Patient 1	**Patient 2**	**Patient 3**	**Patient 4**	**Patient 5**	**Patient 6**	**Patient 7**	**Patient 8**
Did you feel comfortable?	YES	YES	YES	MORE OR LESS	YES	YES	YES	YES
Did you feel any sensation of dizziness?	NO	NO	NO	A BIT	NO	NO	NO	NO
Would you like to do this game more often?	YES	YES	YES	YES, BUT PRACTICING MORE	YES	YES	YES	YES
Did you find difficult to know how to use the controllers?	YES	NO	A BIT	NO WITH HELP	NO	NO	NO	NO
Did you feel difficult to recognize any product?	NO	THE LIST BECAUSE OF NOT WEARING GLASSES	SOME OF THEM	NO	NO	NO, ONLY READ LETTERS	NO	YES, BECAUSE NOT WEARING GLASSES
Would you change something of the game?	NO	NO	NO	NO	NO	NO	NO	YES, FEW PRODUCTS

Table 10. Results of the surveys made to the patients (Source: Prepared by the authors)

11. Environmental impact analysis and regulations

This project environmental impact is insignificant, but there are some points to take into account as well as the necessary regulations that must be followed by any computer program for medical purposes.

To analyze this insignificant environmental impact generated by this project, it is necessary to talk about the concept of Life Cycle Assessment ("LCA"). The "LCA", is a technique which evaluates the environmental impacts associated to all the stages of the useful life of any product, from the extraction of raw material to the processing, manufacture, distribution, use, repair, maintenance and recycling.

Extrapolating the "LCA" technique to our project, the most important environmental impact that can be found is related to the energy consumption that has been necessary to carry out the project. Without energy, none of the devices necessary for the development of the project is capable of functioning. Nowadays, the electricity production process is based on the consumption of fossil fuels (oil, gas and coal) or nuclear energy. As a result, significant amounts of gases are emitted into the atmosphere causing the greenhouse effect and, consequently, climate change.

Regarding the devices used, it is necessary to be aware of the abrasive methods used for the extraction of the materials necessary for its manufacture, in addition to the energy consumption associated with this. It is also necessary to emphasize that, the instructions must be followed to, if necessary, proceed with the correct recycling and reuse of any of the devices used.

It is for all this, that companies which are responsible for the production of the elements used in this project, must follow some laws to try to reduce the maximum environmental impact. The most important laws about it are:

- Restriction of the Use of Certain Hazardous Substances in Electrical and Electronic Equipment (RoHS). It restricts the use of hazardous materials in the manufacture of electronic and electrical equipment.
- ISO 9000 and 14000 certifications. These are a series of certifications that companies must follow to manage and reduce their environmental impact. They include guidelines to reduce energy consumption and waste management among many others.

In addition, in reference to our project, as it is a project in which an application is developed for medical purposes, and, in one way or another, it is planned to be used to monitor or treat patients with a mild cognitive degenerative disease, below are some of the most important laws that must be followed, some of them at European level and others at national level:

- Law 14/1986, Spanish law of general health
- European Directive 90/385/CEE responsible for regulating active implantable medical products.

- European Directive 93/42/CEE responsible for regulating health products.
- European Decision 2010/227/UE related with Database on Health Products.
- European Regulation number 207/2012, about electronic instructions on Health Products.

In conclusion, anything what is done and regulations say is enough to back away energy consumption. It is true that energy consumption is essential for the planet development, but we must be aware that the current pace endangers the depletion of our natural resources. Therefore, we must all take care to make a responsible use and try to consume renewable energy sources.

Conclusions

This project has consisted on the creation of a Virtual reality application applied to cognitive training to improve the results obtained by actual techniques in mild cognitive impairment treatments.

With the use of this application, patients with MCI can train and slow down the growths of the effects of their diseases while having fun. This is why this kind of new techniques are more effective than traditional ones.

Once the project is finished, we can say that the project goals and all the objectives set throughout the project have been achieved.

First of all, we reduced the weight of the application in more than a twelve per cent. With the knowledge acquired during these past months we have been able to not only reduce weight, also improving the use of memory and approaching the app to all kind of users, not only the ones with powerful computers.

Secondly, we made all the application exactly equal in every level to improve the immersion and make it easier for users to meet the supermarket and recognize every product. This also helped to improve all the products and make them grabbable and affected by the gravity, this is very useful for patients because its immersion is better and they fell like being in a real supermarket.

On the other hand, we tested the application with several patients and we received an incredible feedback. Thank to this test we found some new problems that had to be solved and we corroborate that the application is useful and comfortable for patients with other diseases a part from stroke.

Thanks to our previous knowledge of “Unity” and the programming language “C#”, we were able to solve little problems with the scripts.

Nowadays the application is ready to keep testing with different types of patients and it only needs to solve little problems to make it fully accessible for everyone.

This project helped us to understand the amount of people suffering from this kind of diseases and how needed are this kind of therapies to make their recovery easier.

Finally, we can say that this project helped us to be more organized, distribute our time, have a working method and to improve our English.

Future lines

Nowadays our application is almost full developed. We have four different games and 3 levels of difficulty for each one and in this game we have introduced lots of types of cognitive trainings like using your memory, calculate or link. Despite of that, some minor changes can be made to make it even more functional

First of all, even having the application tested with patients, it is needed to still keep testing the application with real patients and including all types of cognitive diseases. This will help to find more errors and make it more accessible to everyone, not depending on their physical problems.

After testing the application with several patients and talking with the physiotherapist we noticed that some minor changes are needed to make the application more accessible. It is important for people with problems with balance to keep the basket well and make easier the introduction of the products. According to these lines, we think that having any kind of button will help people with side affectations to exercise their hands. If we link these two ideas, we think it is needed to develop a method to make appear the basket by pushing a button.

Another future project we have deduced from our visit to ADFO was the creation of a kind of control via joystick to help people with reduced mobility. As with three patients we have to use the game with them sat down and, as we said before, this game it is not created to play sat, including a Joystick will help people to move around the supermarket and turn 360 degrees without the need of turn themselves.

There is only one point in all this project that needs a bit more of development and this is the database. Nowadays this game uses MYSQL to register all the data from every patient but it is not fully developed. It is needed to make an easier database for the doctor or physiotherapist to help them register all the progresses of the patients from all four games and every level.

As we said before it is very difficult to set the future lines for this project because it is almost full developed. The most important improvement will be make it accessible for everyone and test it during several days to show the evolution of the patients and see if this application helps anyone.

Budget

In this section the financial analysis of the project is presented. For the realization of this study, it has been considered that the project began on 02/07/2018 and ended on 01/09/2019, which, taking into account holidays, corresponds to 101 days of work. Below are the different costs associated with the project. These costs have been divided into two large groups, direct costs and indirect costs.

Direct cost

Direct costs are divided into salary, social security costs and the value of the equipment and material needed to develop the project.

Salary

For the salary section it has been considered that this project has been executed by an engineer with a gross salary of 30€ per hour. In addition, taking into account that the working hours have varied depending on the day, a total average of 6 hours worked per day have been considered.

	Quantity	Time (h)	Cost/hour (€/h)	Cost (€)
Engineers	2	600	30	36.000
TOTAL				**36.000 €**

Table 11. Salary cost

Social security

For the calculation of this section, it has been taken into account that in Spain, for this type of project, the cost associated with social security is the 23.6% of the contribution base.

	Number of workers	Salary (€)	Contribution base (%)	Cost (€)
Social security	2	18.000	23,60	8.496
TOTAL				**8.496 €**

Table 12. Social security cost

Equipment

This section includes all the material and software that has been necessary for the development of this project.

	Quantity	Cost/unit (€)	Cost (€)
PC	1	1.200	1.200
Monitor	1	500	500
Laptop	1	1.000	1.000

HTC Vive	1	600	600
“Unity 3D”	2	0	0
“Visual Studio”	2	0	0
“Paint”	2	0	0
“MySQL”	2	0	0
“DB Browser”	2	0	0
TOTAL			**3.300 €**

Table 13. Equipment cost

Indirect cost

In this project, the indirect costs that appear are only those shown below.

Installation and associated expenses

	Months	Cost/month (€)	Cost (€)
Rent	5	600	3.000
Internet	5	60	300
Light	5	60	300
Water	5	30	150
Gas	5	30	150
TOTAL			**3.900 €**

Table 14. Installation and associated expenses cost

Total cost

Finally, the following table presents the total cost of the project.

Concept	Cost (€)
Salary	36.000
Social Security	8.496
Equipment	3.300
TOTAL DIRECT COST	**47.796**
Installation and associated expenses	3.900
TOTAL INDIRECT COST	**3.900**
TOTAL	**51.696 €**

Table 15. Total cost of the project

Bibliography

Agencia Española de Medicamentos y Productos Sanitarios. 2017. "Publicados los nuevos reglamentos europeos de productos sanitarios". Last modified May 5, 2017. https://www.aemps.gob.es/informa/notasInformativas/productosSanitarios/2017/NI-PS_09-2017-reglamentos-europeos-PS.htm

Baldelli, M.V., M. Motta, E. Abati, A. Pirani, V. Manzi and E. Mariani. 1993. "Effects of Reality Orientation Therapy on patients in the community". *Archives of Gerontology and Geriatrics* 17, no.3 (November):211-8. https://www.sciencedirect.com/science/article/abs/pii/016749439390052J?via=ihub

Burdea, G. and P. Coiffet. 1996. Tecnologías de la Realidad Virtual. Barcelona: Editorial Paidós Ibérica S. A.

Cao S. 2016. "Virtual Reality Applications in Rehabilitation". *18th International Conference HCI International 2016 Toronto*, ON, Canada, July 17–22, 2016. Proceedings, Part I. Editor Masaaki Kurosu The Open University of Japan Chiba-Shi.

Clare, L. and B. Woods. 2003. "Cognitive rehabilitation and cognitive training for early-stage Alzheimer's disease and vascular dementia". Cochrane Database of Systematic Reviews 2003, Issue 4. https://www.cochranelibrary.com/cdsr/doi/10.1002/14651858.CD003260/full

Difede, J., and H. Gunter Hoffman. 2002. "Virtual reality exposure therapy for World Trade Center Post-traumatic Stress Disorder: A Case Report". *CyberPsychology & Behavior: Multimedia and Virtual Reality on Behavior and Society* 5, no.6 (December): 529–35. http://doi.org/10.1089/109493102321018169

European Commission. 2018. "Medical devices". Accesed December 16, 2018. http://ec.europa.eu/growth/single-market/european-standards/harmonised-standards/medical-devices_en

Fernández-Calvo et al. 2011; Man, Chung, and Lee 2012; Hill et al. 2016; Optale et al. 2010; Kizony et al. 2012; «escalaalzheimer.pdf», s.d.; Pourmand et al. 2017.

Fernández-Calvo, Bernardino, Roberto Rodríguez-Pérez, Israel Contador, Alicia Rubio-Santorum, and Francisco Ramos. 2011. "Eficacia del entrenamiento cognitivo basado en nuevas tecnologías en pacientes con demencia tipo Alzheimer" 23: 44-50.

Greenwald, S.W., A. Kulik, S. Beck, S. Cobb, S. Parsons, N. Newbutt, ... and P. Maes. 2017. "Technology and Applications for Collaborative Learning in Virtual Reality". *Repository of the International Society of Learning Sciences*: 18–22. http://doi.org/10.22318/cscl2017.115

Groom, V., J.N. Bailenson, and C. Nass. 2009. "The influence of racial embodiment on racial bias in immersive virtual environments". *Social Influence* 4, no.3 (September): 231–248.

http://doi.org/10.1080/15534510802643750

Hill, Nicole T.M., M. BMSc., Loren Mowszowski, D. Psych., Sharon L. Naismith, Verity L. Chadwick, and B.Sc. (Hons.). 2016. "Computerized Cognitive Training in Older Adults With Mild Cognitive Impairment or Dementia: A Systematic Review and Meta-Analysis". *The American Journal of Psychiatry* 174, no. 4 (November): 329-340. https://ajp.psychiatryonline.org/doi/10.1176/appi.ajp.2016.16030360

Jensen, L. and F. Konradsen. 2018. "A review of the use of virtual reality head-mounted displays in education and training". *Education and Information Technologies* 23, no.4 (July): 1515–1529 https://doi.org/10.1007/s10639-017-9676-0

Kizony, R., M. Korman, G. Sinoff, E. Klinger and N. Josman. 2012. "Using a virtual supermarket as a tool for training executive functions in people with mild cognitive impairment": 10-12.

Lampit A, H. Hallock, M. Valenzuela. 2014. "Computerized cognitive training in cognitively healthy older adults: a systematic review and meta-analysis of effect modifiers". *PLOS Medicine*. https://doi.org/10.1371/journal.pmed.1001756

LaValle, S.M. 2016. "Virtual reality". *IEEE Computer Graphics and Applications* 14, no.1 (Jan.): 15-16. http://doi.org/10.1109/38.250913

Man, David W.K., Jenny C.C. Chung and Grace Y.Y. Lee. 2012. "Evaluation of a virtual reality - based memory training programme for Hong Kong Chinese older adults with questionable dementia: a pilot study". *International Journal of Geriatric Psychiatry* 27, no.5 (June): 513-20. https://doi.org/10.1002/gps.2746

Motter, J.N., M.A. Pimontel, D. Rindskopf, 2016. "Computerized cognitive training and functional recovery in major depressive disorder: A meta-analysis". *Journal of Affective Disorders* 189, no.1 (January): 184–191 https://www.sciencedirect.com/science/article/pii/S0165032715308454?via%3Dihub

Mundo Virtual. n.d. "¿Qué es la realidad virtual?". Accesed October 10, 2018. http://mundo-virtual.com/que-es-la-realidad-virtual/

Munro, A., R. Breaux and J. Patrey. 2002. *Cognitive aspects of virtual environments design. Handbook of virtual environments: design, implementation, and applications*, 415-434. CRC Press.

Optale, Gabriele, Cosimo Urgesi, Valentina Busato, Silvia Marin, Lamberto Piron, Konstantinos Priftis, Luciano Gamberini, Salvatore Capodieci, and Adalberto Bordin. 2010. "Controlling Memory Impairment in Elderly Adults Using Virtual Reality Memory Training: A Randomized Controlled Pilot Study". *SAGE journals.* https://doi.org/10.1177/1545968309353328

Petersen, R.C., R. Doody, A. Kurz, R.C. Mohs, J.C. Morris and P.V. Rabins. 2001. "Current concepts

in mild cognitive impairment". *Archives of Neurology* 58, no.12.

Pourmand, Ali, Steven Davis, Danny Lee, Scott Barber and Neal Sikka. 2017. "Emerging Utility of Virtual Reality as a Multidisciplinary Tool in Clinical Medicine" 6, no.5 (October): 263-70. https://doi.org/10.1089/g4h.2017.0046

Prince M., R. Bryce, E. Albanese, A. Wimo, W. Ribeiro and C.P. Ferri. 2013. "The global prevalence of dementia: a systematic review and metaanalysis." Alzheimers Dement 9, no.1 (January):63–75. https://doi.org/10.1016/j.jalz.2012.11.007

Priore, C., G. Castelnuovo, and D. Liccione. 2003. "Experience with V-STORE: considerations on presence in virtual environments for effective neuropsychological rehabilitation of executive functions". *CyberPsychology & Behavior* 6, no.3 (July): 281-287. http://online.liebertpub.com/doi/abs/10.1089/109493103322011579

Requena, C., F. Maestú, P. Campo, A. Fernández and T. Ortiz. 2006. "Effects of cholinergic drugs and cognitive training on dementia: 2-year follow-up.". *Dementia and Geriatric Cognitive Disorders* 22, no.4 (September): 339-45. https://www.karger.com/Article/Abstract/95600

Rizzo, A. "Skip" and S. Thomas Koenig. 2017. "Is Clinical Virtual Reality Ready for Primetime?". *Neuropsychology* 31, no. 8: 877–899. http://dx.doi.org/10.1037/neu0000405

Rosenthal, R., W.A. Gantert, C. Hamel, J. Metzger, T. Kocher, P. Vogelbach, ... and D. Hahnloser. 2008a. "The future of patient safety: Surgical trainees accept virtual reality as a new training tool". *Patient Safety in Surgery* 2, no.1 (April): 16. http://doi.org/10.1186/1754-9493-2-16

Rosenthal, R., W.A. Gantert, C. Hamel, J. Metzger, T. Kocher, P. Vogelbach, ... and D. Hahnloser. 2008b. "The future of patient safety: Surgical trainees accept virtual reality as a new training tool". *Patient Safety in Surgery* 2, no.1 (April): 16. http://doi.org/10.1186/1754-9493-2-16

Sitzer, D.I., E.W. Twamley and D.V. Jeste. 2006. "Cognitive training in Alzheimer's disease: a meta-analysis of the literature". *Acta Psychiatrica Scandinavica* 114, no.2 (June): 75-90. https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1600-0447.2006.00789.x

Snowden, J. S. 1989. "Semantic dementia: A form of circumscribe cerebral atrophy". *Behavioural Neurology* 2, no.3: 167-182. Netherlands: IOS Press. http://psycnet.apa.org/index.cfm?fa=search.displayRecord&uid=1992-02247-001

Annex A. Pilot test for the "Hospital Clínic" of Barcelona

IMMERSIVE VIRTUAL REALITY COGNITIVE TRAINING FOR OLDER ADULTS WITH MIDDLE COGNITIVE IMPAIRMENT (MCI) OR MIDDLE DEMENTIA (MD)

June 2019

Table of contents

A1. VR Application

"VR Supermarket Experience"

Its purpose is to be used in the treatment of mild cognitive impairments through the virtual simulation of an everyday act such as going to a supermarket to make the purchase.

Figure A1.1. Sections of the supermarket (Source: Prepared by the author)

The shop scenario was a simulated convenience shop consisting of five goods trays, one fruit tray, three refrigerators, and one cashier. Participants were asked to search around the shop and buy the requested items.

A1.1. Menus

A1.1.1. ID menu

Figure A1.2. ID Menu (Source: Prepared by the author)

A1.1.2. Main menu

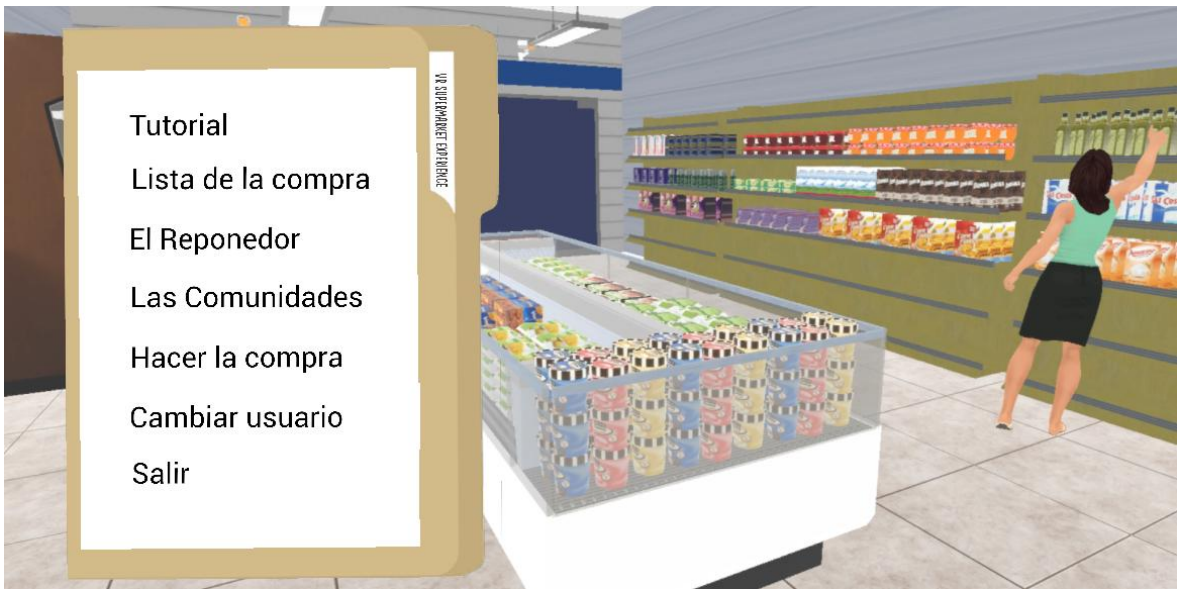


Figure A1.3. Main menu (Source: Prepared by the author)

A1.1.3. Levels menu

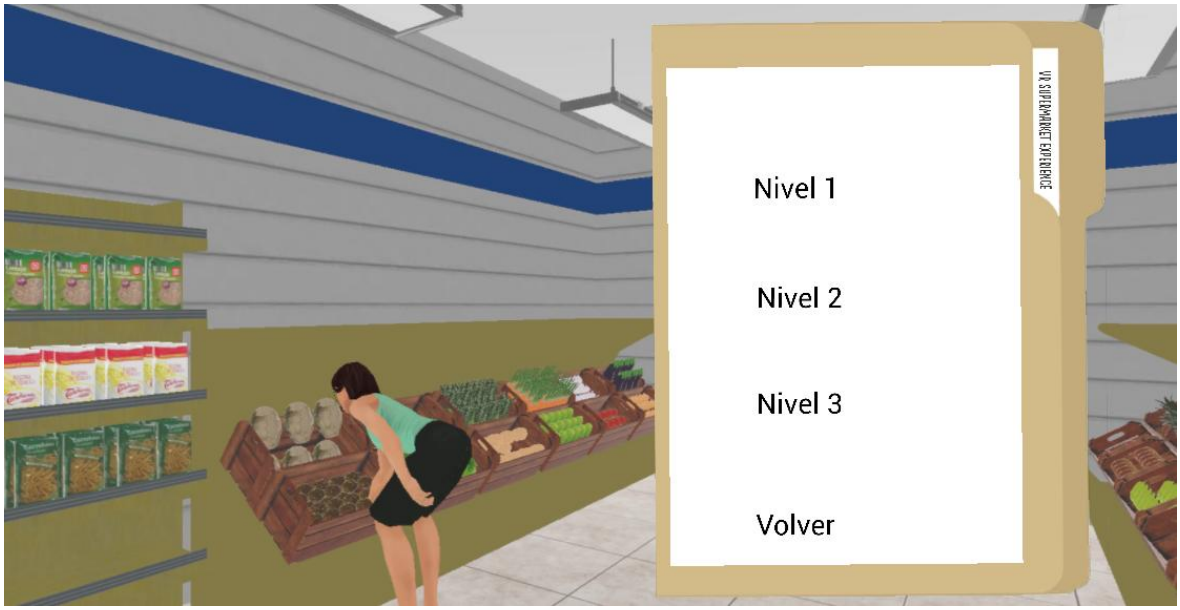


Figure A1.4. Levels menu (Source: Prepared by the author)

A1.1.4. Final menu

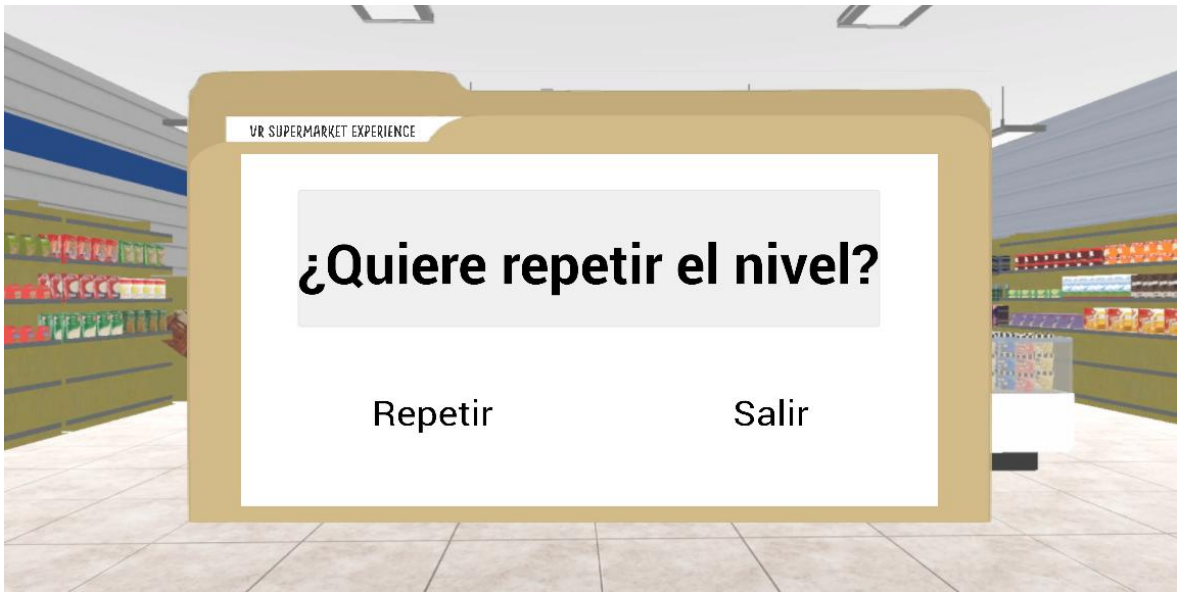


Figure A1.5. Final menu (Source: Prepared by the author)

A1.2. Tutorial

The first part of the application VR Supermarket Experience is a tutorial to help the patient. It will be useful because the patient can begin to get used the application and know how to relate to the environment. He will use the commands to pass different proofs that will allow the patient to familiarize himself with the mechanics of the application.

First of all, the user will learn to move the head to look around. Virtual reality allows a 360 degrees vision by turning the head and in this first part we want to help the patient realize that.

The user will be in a small room with four walls. In three of them there will be a red sphere that will change to a green colour when it will be viewed from the front. This exercise will make the person turn on itself.

A1.3. Shopping list

In this part of the application there are 3 levels and 2 videos. The first video explains what you have to do in the level 1 and the second video explains what you will find in levels 2 and 3. The videos provide tips for a better understanding of the application.

In this part of the application the patient is in the supermarket and a shopping list appears on the screen. The patient has to take these products from the supermarket and put them in the basket. The user can take the products he wants but the objective of the application is to take the products that are only on the list.

Levels have a growing difficulty as the user progresses. In the first level the shopping list will always be visible. In the levels 2 and 3 the shopping list will be visible for a stipulated time. The differences between level 2 and level 3 are that in the level 2 there are five products in the list and in the level 3 there are seven products. The display time of the shopping list is also higher in level 2. During both levels you can consult the shopping list by pressing a button.

To increase the difficulty, some distractions for the user have been included in levels 2 and 3. These distractions consist in that the supermarket replenishment will go to the user and will tell him something. In level 2 there is only one distraction. On the other hand, in level 3 there are two distractions.

Figure A1.6. Shopping list scene (Source: Prepared by the author)

Cognitive functions worked in this section:

- **Language**
  - Associate the product with its noun.
  - The activity is proposed to improve the recall capacity of the language and vocabulary by the patient. The noun of the object is shown on screen during the test so the patient may look for it in the supermarket at stroll speed.
- **Episodic memory**
  - To be able to assess the state of the patient's episodic memory there were designed activities in which short-term memory capacity was involved.
  - Description of the task: A list of the products is given to the patient to be memorized.

A1.4. Shop assistant

In this activity the patient can play another different role. He becomes a shop assistant and his objective is returning a different market product at their correct place.

The products are on a table that it is in the centre of the market. The patient must identify the objects and find in which section must be the product.

The environment is the same that the "Shopping list" activity and the products of the shelves are in the same place too. Also, the controllers and game objects are very similar than the other activity to reuse resources and save working time.

This activity has three levels:

- Level 1: the patient must return 3 products.
- Level 2: the patient must return 4 products.
- Level 3: the patient must return 6 products.

All the results of the levels will be stored in the same database used in other activities.

Figure A1.7. Level 1 products of "Shop assistant" (Source: Prepared by the author)

The products entered in the basket will appear on the screen. When the user thought that he has finished the purchase, he must leave through the exit door.

Cognitive functions worked in this section:

- **Episodic memory**
  - To be able to assess the state of the patient's episodic memory it was designed an activity in which short-term memory capacity was evaluated.
  - Description of the task: Items are shown to the patient with images so that they can be memorized and ordered in their proper place
- **Semantic memory**
  - Activities to decrease the neurodegenerative disorder characterized by the loss of semantic memory, that is to say, the ability to remember meaning and family of words by the patient.
  - Description of the task: Ordering products in their proper place
  - Specific products and trademarks present a higher level of difficulty than the generic ones.

- **Executive function**
    - Deficits in EF refer to a collection of impairment in attention, planning, problem-solving, multitasking, monitoring and behavioral control. This activity was developed to assess the patient's EF capacity.
    - Description of the task: Order products on shelves (product from picture). It is asked to replace some products to the corresponding place.

A1.5. Autonomous communities

These activities are based on the recognition of some products and relate them to their corresponding region.

The patient must choose the product and replace it in the corresponding region stand trying to do the fewer mistakes as possible.

The structure and objects of this activity are very similar that the "Shop Assistant".

In this activity and levels, we can find similar objects that are like the "Shop assistant" activity.

Figure A1.8. Level 2 products of "Autonomous communities" (Source: Prepared by the author)

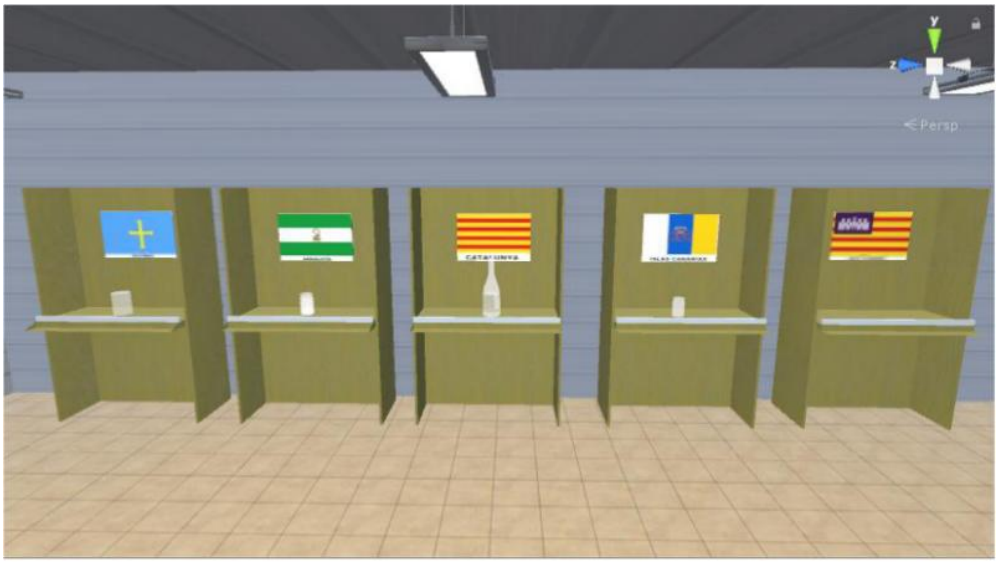


Figure A1.9. Level 2 flags of "Autonomous communities" (Source: Prepared by the author)

Cognitive functions worked in this section:

- **Episodic memory**
  - Description of the task: Items are shown to the patient with images so that they can be memorized.
- **Semantic memory**
  - Cultural and geographic characteristics of products present a high level of difficulty to be remembered.
- **Executive function**
  - Description of the task: Order products on shelves (product from picture). It is asked to replace some products to the corresponding place.

A1.6. Go shopping

In this game the patient must take a certain number (depending on the level) of any of the supermarket products and insert them in the basket in the same way as in the "Shopping list" game.

Once the client has taken the determined number of products, they should go to the cashier to proceed with the payment of these. From that moment, the basket and the objects that appear in the screen disappear. Then, the purchase ticket, with the products, their price and the total price of the purchase, appears on the screen. This is the moment in which the patient must proceed with the payment. The patient must enter in the indicated area of the automatic cashier, the bills or the coins necessary to make the payment of the purchase.

Depending on the level, this payment will have to be exact or not, so the change does not appear

at all levels. As has been done previously with the basket, the money that is entered in the cashier also appears on the screen, so that the patient knows if the money has been entered correctly.

Differences between levels:

- Level 1. The patient has to take 3 products and pay the price of the purchase exactly.
- Level 2. The patient has to take 5 products. They can only pay with banknotes so a random change will appear on the screen. The patient must click if the change is correct or not in the correspondent button.
- Level 3. The patient has to take 5 products. They can only pay with banknotes so a random change will appear on the screen. The patient must take and drag to the red button the extra coins that appear in the screen in order to correct the change.
-

Figure A1.10. "Go shopping" level 2 scene (Source: Prepared by the author)

Cognitive functions worked in this section:

- **Semantic memory**
  - Activities to decrease the neurodegenerative disorder characterized by the loss of semantic memory, that is to say, the ability to remember meaning and family of words by the patient.
  - Description of the task: Paying the purchase.
  - The patient has to relate the image of a coin or a banknote with its corresponding value.

- **Executive function**
    - Deficits in EF refer to a collection of impairment in attention, planning, problem-solving, multitasking, monitoring and behavioral control. This activity was developed to assess the patient's EF capacity.
    - Description of the task: Calculating the cost in the cashier zone, and depending on the level do different mental calculations such as add or subtract.

A1.7. HTC Commands

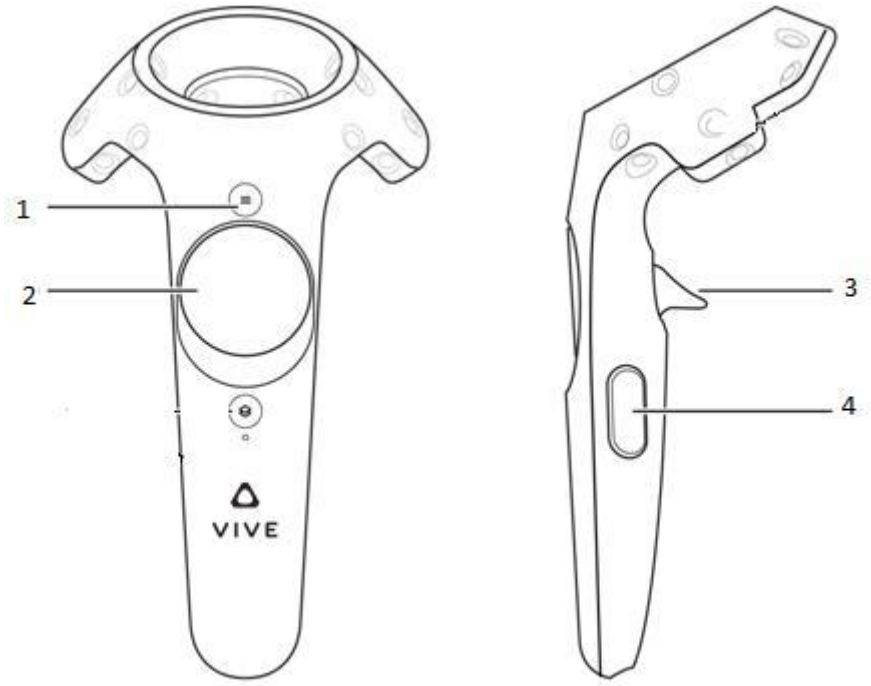


Figure A1.11. HTC Commands (Source: Prepared by the author)

Number 1: this button is only used in the levels 2 and 3 of the Shopping List part. It is used to consult the list. When the button number 1 of the left command is pressed the list appears 15 seconds. The number of times that the button number 1 is pressed in the levels 2 and 3 will appear in the database.

Number 2: It is the Trackpad button. It is always in the right command and it is used to modify the direction of the user. When the user presses the left part of the Trackpad he moves to the left. When he presses the right part he moves to the right. When the patient presses the up part of the Trackpad he goes forward and when he presses the down part of the Trackpad he goes backwards. It is used in all the parts of the application.

Number 3: It is the trigger button. It is used in the four parts of the application. It is used to catch the different products. When the trigger button of the right command is pressed, and a product is

selected the user catches the product. Moreover, this button is also used to press the different buttons that appear in the screen during the “Go shopping” game mode.

Number 4: This button is only used in the tutorial. When the button number 4 of the right command is pressed the last instruction is repeated. It is useful at the beginning because the patient can be nervous, and he can lose some important information of the instruction.

A1.8. Database

One of the most important things about the application is the collection of data. It is very important for the doctor to have a follow-up of each patient’s progress.

The database is done with the program “MySQL”. It is a very popular program to do a database and it is developed by “Oracle” and “Microsoft SQL Server”. The database can be read with the program “DB Browser”.

A2. Pilot study script

Hypothesis	The two study objectives are (1) to develop and implement an immersive VR-based memory training program for older adults with middle cognitive impairment (MCI) or middle dementia (MD), and (2) to examine the efficacy of a VR memory training program on episodic memory and self-appraisal of cognitive functions as compared with a conventional therapist-led memory training program. The study presents two hypotheses, similar to Man *Et al.,* 2012: (1) Positive changes would be exhibited in the VR-based and therapist-led memory training. (2) The VR-based memory group show better memory functions and greater functional independence as compared with the therapist-led group.
Participants	20. 10 EG (Experimental Group, 10 CG (Control Group) Crossover. Finally, all patients will use VR configuration.
Control Group	Non-VR based memory programme
Experimental Group	An immersive VR- based memory training programme
Training content	An immersive VR- based memory training program Each scenario was designed according to a gradation structure: tasks ranged from simple to more complex in terms of the number and similarity of objects to be remembered as well as the duration of distraction. (a) duration of recall according to the spaced retrieval technique (Thivierge et al., 2008); (b) types of memory stimuli delivered: from visual-auditory and visual only to auditory only; and (c) migration from semantic to episodic memory, by requiring the participants to gradually remember what to do and the sequence of actions.
Sessions number	24 individual sessions, 30 minutes/session 2 sessions/week 12 weeks/patient 12h/patient Technical and therapeutic support will be needed for the experiment in: "Centre de dia c/Reina Amalia, 37, al lado de la parada de metro Paralelo." 12h * 20 = 240h (2 mornings a week). Probable experiment duration: if 10h/week then 24 weeks (6 months)
Methodology	Each session of the experimental and control training lasted approximately 30 minutes and was conducted on separate days. The clinical and neuropsychological evaluation was performed before the onset of the training (pre-training) and at the end of the training phase (post-training). Firstly, a guided tutorial is done to familiarize the users in an initial session. The application content consists in play a serious game with 5activities:

	Tutorial Shopping list Shop assistant Autonomous communities Go shopping He or she would be told about a 30-min task involving moving around, reading, and memorizing the items They then took out those items from the shop trays and refrigerators. The time taken to finish the task was recorded. During the 3-month training phase, 2 VR sessions will be administered every week. Each VR session lasted approximately 15 minutes and was followed, after a pause of 1 minute, by 15 minutes in which the participant was invited to make an oral summary of the experience. The therapist-led training adopted a psychoeducational approach and it will be very similar to the VR, but with traditional activities.
Baseline Neuropsychological Evaluation	General cognitive abilities Verbal Memory Executive functions Visuospatial processing Daily living activities Depression (Optale et al., 2010) Multifactorial Memory Questionnaire and Fuld Object Memory Evaluation. CDR scale. Mini Mental State Examination (MMSE) The MMSE was developed by Folstein's group (Folstein et al., 1975) Geriatric Depression Scale (Yesavage et al., 1983). Multifactorial Memory Questionnaire (MMQ; Troyer and Rich, 2002). Fuld Object Memory Evaluation (FOME; Fuld, 1977) Lawton Instrumental Activities of Daily Living (HKLawton IADL) scale (Lawton and Brody, 1969; Tong and Man, 2002). (Man *Et al.*, 2012) Other tests could be applied at the end of the experience: Technology Acceptance Model (TAM) to measure the test-takers' acceptance could be answered when the study finish. (Shih-ChingYeh, Et al. 2012) Simulator Sickness Questionnaire (SSQ) (Kennedy,R.S. Etal. 1993) cited by Shi Cao 2016
Levels of difficulty	Levels have a growing difficulty as the user progresses. In the first level the shopping list will always be visible. In the levels 2 and 3 the shopping list will be visible for a stipulated time. The differences between level 2 and level 3 are that in the level 2 there are five products in the list and in the level 3 there are seven products. The display time of the shopping list is also higher in level 2. During both levels you can consult the shopping list by pressing a button. To increase the difficulty, some distractions for the user have been included in levels 2 and 3. These distractions consist in that the

	supermarket replenishment will go to the user and will tell him something. In level 2 there is only one distraction. On the other hand, in level 3 there are two distractions. The products entered in the basket will appear on the screen. When the user thought that he has finished the purchase, he must leave through the exit door. In the "Shopping list", "Shop assistant" and "Autonomous communities" activities the levels depend on the number of objects: Level 1: the patient must return 3 products. Level 2: the patient must return 4 products. Level 3: the patient must return 6 products. In the "Go shopping" activity the differences between levels are: Level 1. The patient has to take 3 products and pay the price of the purchase exactly. Level 2. The patient has to take 5 products. They can only pay with banknotes so a random change will appear on the screen. The patient must click if the change is correct or not in the correspondent button. Level 3. The patient has to take 5 products. They can only pay with banknotes so a random change will appear on the screen. The patient must take and drag to the red button the extra coins that appear in the screen in order to correct the change.
Collected Data	Gender, n (%) Male Female Level of education, n (%) <1 year 1–2 years >2 years Mean age (SD), years *Background of Subjects* Find out the subjects' spending habits, including (1) whether they have the habit of shopping or not; (2) the place(s) they frequent when go shopping (3) whether they prefer shopping alone or with someone else; and (4) their shopping frequency. (Yeh *Et al.* 2012) In the "Shop Assistant" game mode, the data collected is: "ObjetosVal3". This variable indicates the number of products correctly introduced in the basket in the first level. The maximum are 3 products. "ObjetosNoVal3". This variable indicates the number of products incorrectly introduced in the basket in the first level. "T3". This variable indicates the elapsed time in seconds since the patient begins the first level until it leaves the door to finish it. "ObjetosVal5". This variable indicates the number of products correctly introduced in the basket in the second level. The maximum are 5 products. "ObjetosNoVal5". This variable indicates the number of products incorrectly introduced in the basket in the second level. "T5". This variable indicates the elapsed time in seconds since the patient begins the second level until it leaves the door to finish it.

	"Cons5". This variable posts the number of times that the user consults the shopping list in the second level."ObjetosVal7" in "MySQL". This variable indicates the number of products correctly introduced in the basket in the third level. The maximum are 7 products. "ObjetosNoVal7". This variable indicates the number of products incorrectly introduced in the basket in the third level. "T7". This variable indicates the elapsed time in seconds since the patient begins the third level until it leaves the door to finish it. "Cons7". This variable posts the number of times that the user consults the shopping list in the third level. In the "Shop Assistant" game mode, the data collected are: "TR1". This variable indicates the elapsed time in seconds since the patient begins the first level until it puts the three products in their correct shelves. "TR2". This variable indicates the elapsed time in seconds since the patient begins the second level until it puts the four products in their correct shelves. "TR3". This variable indicates the elapsed time in seconds since the patient begins the third level until it puts the six products in their correct shelves. In the "Autonomous Communities" game mode, the data collected are: "TC1". This variable indicates the elapsed time in seconds since the patient begins the first level until it puts the three products in their correct stands. "TC2". This variable indicates the elapsed time in seconds since the patient begins the second level until it puts the four products in their correct stands. "TC3". This variable indicates the elapsed time in seconds since the patient begins the third level until it puts the six products in their correct stands. Finally, in the "Go shopping" game mode, the data collected are: "ValP1". This variable indicates "1" if the patient paid the price of the purchase exactly in the first level. "NoValExcesoP1". This variable indicates "1" if the patient paid more than the price of the purchase in the first level. "NoValDefectoP1". This variable indicates "1" if the patient paid less than the price of the purchase in the first level. "TP1". This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the first level until it leaves the door to finish it. "ValCambioCorrectoP2". This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was correct, so the patient pressed the green button for correct change in the second level. "ValCambioIncorrectoP2". This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was incorrect, so the patient pressed the red button for incorrect change in the second level.

	"NoValCambioCorrectoP2". This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was correct, but the patient pressed the red button for incorrect change in the second level. "NoValCambioIncorrectoP2". This variable indicates "1" if the patient paid correctly, the random change that appeared on the screen was incorrect, but the patient pressed the green button for correct change in the second level. "NoValDefectoP2". This variable indicates "1" if the patient paid less than the price of the purchase, so they paid incorrectly, in the second level. "TP2". This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the second level until it leaves the door to finish it. "ValP3". This variable indicates "1" if the patient paid correctly and, after the random change appeared, they guessed which coins had to be eliminated in the third level. "NoValExcesoEliminarP3". This variable indicates "1" if the patient paid correctly and, after the random change appeared, they eliminated more coins than necessary in the third level. "NoValDefectoEliminarP3". This variable indicates "1" if the patient paid correctly and, after the random change appeared, they eliminated less coins than necessary in the third level. "NoValDefectoP3". This variable indicates "1" if the patient paid less than the price of the purchase, so they paid incorrectly, in the third level. "TP3". This variable indicates the elapsed time in seconds since the patient enters in the payment zone in the third level until it leaves the door to finish it.
Expected results	In mild cognitive impairment, VR is efficacious on global cognition, memory, working memory, and attention and helps improve psychosocial functioning, including depressive symptoms. Effects on other domains such as executive function and processing speed are negligible.

A3. References

AUTHOR	Fernández-Calvo, B. *Et al*. 2011
PARTICIPANTS	EA light 45 patients 3 groups: stimulation program with BBA (EABB) or a traditional stimulation program (EAPI), based on paper-and-pencil tasks, for twelve weeks. A third group, the control group (EANT), did not receive any treatment during this period.
SESSIONS, INTERVENTION	36h Wii Nintendo, Program of games of reactivation and stimulation of mental capacities applied in an order: perception, memory of work, calculation, analysis (semantic memory) and acuity (recognition of objects and estimation of quantities). Compared with Integrated Psychostimulation Program (IPP). Each game with 3 levels of difficulty EAPI (12h)
OUTCOME MEASURES	Pre-post design, considering neuropsychological, behavioral, and functional standard measures as outcome variables. ADAS-Cog: Escala de Evaluación de la Enfermedad de Alzheimer; EDC: Escala Depresión Cornell. NPI-Q: Inventario Neuropsiquiátrico-Questionario; RDRS-2; Escala de Evaluación Rápida de Discapacidad – 2.
RESULTS	The BBA program was more effective than IPP to reduce cognitive decline and depressive symptoms in patients with AD.

AUTHOR	Man, D.W.K. *Et al.* 2012
PARTICIPANTS	24 patients EG, 20 GC. Older adults with questionable dementia
SESSIONS, INTERVENTION	Non-immersive form of VR 10 individual sessions of 30 min 15 min VR, 15 min the patient describes the experience.
OUTCOME MEASURES	Multifactorial Memory Questionnaire and Fuld Object Memory Evaluation. **Assessment tools** CDR scale. Mini Mental State Examination (MMSE) The MMSE was developed by Folstein's group (Folstein et al., 1975) Geriatric Depression Scale (Yesavage et al., 1983). **Outcome measures** Multifactorial Memory Questionnaire (MMQ; Troyer and Rich, 2002). Fuld Object Memory Evaluation (FOME; Fuld, 1977) Lawton Instrumental Activities of Daily Living (HKLawton IADL) scale (Lawton and Brody, 1969; Tong and Man, 2002).
RESULTS	The authors suggest that VRMT may improve memory function in elderly adults by enhancing focused attention.

AUTHOR	Optale, G. *Et al.* 2010
PARTICIPANTS	EG: 10 women and 5 men; CG: 11 women and 5 men
SESSIONS, INTERVENTION	EG: 10 women and 5 men; CG: 11 women and 5 men 36 sessions initial training 24 sessions booster training phase. 30 min/ session EG: Strolling with joystick path to be remembered. GC: music therapy HMD V6
OUTCOME MEASURES	General cognitive abilities: A, Mini Mental State Examination (MMSE); B, Mental Status in Neurology (MS). Neuropsychologicalabilities: A,DigitSpan(DS);B,Verbal StoryRecall(VSR);C,PhonemicVerbalFluency(PVF);D,DualTask Performance (DTP); E, Cognitive Estimation Test (CET); F, Clock Drawing Test (CDT). Daily living activities and depression:A,Activities of Daily Living–Functions (ADL-F); B. Activities of Daily Living– Mobility (ADL-M); C. Instrumental Activities of Daily Living (IADL); D. Geriatric Depression Scale (GDS).
RESULTS	The EG showed significant improvements in memory tests, especially in long-term recall The authors suggest that VRMT may improve memory function in elderly adults by enhancing focused attention.

AUTHOR	Hill, N. *Et al.* (Review) 2016. Meta-analysis
PARTICIPANTS	14,961 articles 660 full-text versions. N=25 studies EXAMPLES: MCI 16 (Finn et al. 2015) 71 (Tarnanas et al. 2014) 106 (Barban et al. 2016) DEMENTIA 11 (Galante et al. 2007), 13 (Lee et al. 2013), 30 (Fernandez-Calvo et al. 2011), 44 (Man et al. 2012)
SESSIONS, INTERVENTION	Virtual reality simulating household tasks, 12 sessions, 30 min each Virtual reality museum task, 40 sessions, 90 min Virtual reality Virtools platform, 36 sessions, 30 min In-house virtual reality enhanced recumbent stationary bike coin 18 and dragon collection , 24 sessions, 20-45 min COGPACK Verbal memory, nonverbal memory, executive functions, attention, speed, 52 sessions, 90 min Commercial games, 24 sessions, 45 min. In-house computerized errorless learning program, 12 sessions, 60 min Virtual reality home and shop simulation, 10 sessions, 30 min
OUTCOME MEASURES	Mild cognitive impairment. Working memory Without outlier (53) Verbal learning Trim & Fill Verbal memory Non–verbal learning Non–verbal memory Attention Executive function Processing speed Language Visuospatial skills Psychosocial Without outlier (46) IADL
RESULTS	Small to moderate effects were found for global cognition, attention, working memory, learning, and memory, with the exception of nonverbal memory, and for psychosocial functioning, including depressive symptoms. In dementia, statistically significant effects were found on overall cognition (k=11, g=0.26, 95% CI=0.01–0.52) and visuospatial skills, but these were driven by three trials of virtual reality or Nintendo Wii. Effects on other domains such as executive function and processing speed are negligible. There is insufficient data to determine whether training gains can be maintained over the long-term. we had insufficient data to evaluate the durability of CCT effects and whether these may reduce conversion to dementia.

AUTHOR	Ekman, U. *Et al*. 2018
PARTICIPANTS	12 with chronic neglect
SESSIONS, INTERVENTION	Performed the interventions 3 times/week during 5 weeks, in total 15 hours. Virtual-reality based scanning training that combines visual, audio and sensori-motor stimulation called RehAtt
OUTCOME MEASURES	fMRI
RESULTS	Patients improved their performance in the Posner fMRI task

AUTHOR	Lampit, A. . *Et al.* 2014
PARTICIPANTS	51 studies age > 60, Healthy not MCI
SESSIONS, INTERVENTION	> 4 h of practice on standardized computerized tasks or video games DOSE 24 h or less (36) more tan 20 hours (15) SESSION LENGTH 30 min or less (13) 31-60 min (29) >60 min (8) CCT Types: Multi-domain 23 Attention 6 Speed of processing 9 Video Game 4 Working Memory 9
OUTCOME MEASURES	Cognitive tests Verbal memory, (B) nonverbal memory, (C) WM, (D) processing speed, (E) executive functions, (F) attention, and (G) visuospatial skills.
RESULTS	Small to moderate effect sizes were found for nonverbal memory, g = 0.24 (95% CI 0.09 to 0.38); verbal memory, g = 0.08 (95% CI 0.01 to 0.15); working memory (WM), g = 0.22 (95% CI 0.09 to 0.35); processing speed, g = 0.31 (95% CI 0.11 to 0.50); and visuospatial skills, g = 0.30 (95% CI 0.07 to 0.54). No significant effects were found for executive functions and attention. Passive control has bad results, and purely home-based training does not result cognitive benefits in unimpaired older adults.

AUTHOR	Vemuri, P. *Et al*. (review)2 016
PARTICIPANTS	MCI. Not described
SESSIONS, INTERVENTION	Not described. cognitive stimulation, cognitive training, and cognitive rehabilitation.
OUTCOME MEASURES	fMRI
RESULTS	Cognitive interventions can be divided into cognitive stimulation, cognitive

	training, and cognitive rehabilitation. Cognitive stimulation engages participants in a range of general activities and discussions and is commonly conducted in groups. It aims at general enhancement of cognitive and social functioning. Cognitive training focuses on guided practice on a set of tasks that reflect particular cognitive functions, such as memory, attention or problem solving or offers instruction, and *practice* of mnemonic approaches such as the method of loci or visual imagery (i.e., strategy training). Cognitive rehabilitation intends to identify and address the individual's needs and goals, which may require strategies for taking in new information or compensatory methods such as using memory aids [13]. Cognitive stimulation consistently improves global cognition, primarily in individuals with mild-to- moderate dementia However, cognitive training did not result in any statistically significant effects in any domain for early stages of Alzheimer's disease. In MCI, cognitive training resulted in small benefits for episodic memory and other cognitive functions, although there is debate about these effects

AUTHOR	Shi Cao 2 016
PARTICIPANTS	MCI. Not described
SESSIONS, INTERVENTION	Not described. Cognitive stimulation, cognitive training, and cognitive rehabilitation.
OUTCOME MEASURES	fMRI
RESULTS	Cognitive interventions can be divided into cognitive stimulation, cognitive training, and cognitive rehabilitation. Cognitive stimulation engages participants in a range of general activities and discussions and is commonly conducted in groups. It aims at general enhancement of cognitive and social functioning. Cognitive training focuses on guided practice on a set of tasks that reflect particular cognitive functions, such as memory, attention or problem solving or offers instruction, and *practice* of mnemonic approaches such as the method of loci or visual imagery (i.e., strategy training). Cognitive rehabilitation intends to identify and address the individual's needs and goals, which may require strategies for taking in new information or compensatory methods such as using memory aids. Cognitive stimulation consistently improves global cognition, primarily in individuals with mild-to- moderate dementia However, cognitive training did not result in any statistically significant effects in any domain for early stages of Alzheimer's disease. In MCI, cognitive training resulted in small benefits for episodic memory and other cognitive functions, although there is debate about these effects.

AUTHOR	Yeh S-C ET al. 2012
PARTICIPANTS	60 Dementia in EG 30 healthy people in the control group
SESSIONS, INTERVENTION	HMD The guide may specify the required goods by means of texts (showing the goods' name(s) only) or picture(s) (of the goods only). The tests are available

	in two modes, one with the shopping list displayed and the other without, and the test-taker must memorize the required goods if the shopping list is not displayed. 3-stage scenario: at first, the system will display a shopping list with the name(s) of 1-10 item(s) of goods (depending on the therapist's design), then, the test-taker will enter the store to look for and click to select the item(s) on the shopping list; the shopping list and a shopping cart list may/may not be displayed on the upper left corner of the screen. Cashier desk to calculate and select money, randomly wrong change can be provided. Path is plotted over store's view on floor and its length is calculated.
OUTCOME MEASURES	Only descriptive data are published. Not cognitive improvement is related. Record the test-taker's location-time history data, the goods at which the test-taker has gazed, and whether the test-taker has selected a certain goods or not Technology Acceptance Model(TAM)
RESULTS	A variety of tasks of multi-layered difficulty-level hierarchy, such as memorizing a shopping list, looking for certain goods, and checking out, has been designed for customized and adaptive assessment, training, and treatment of MD. In the meantime, the study also recorded the test-taker's performance data and history date of task execution (including path and central-vision movement) in full and measured the subjects' technology acceptance.

Annex B. Explanatory audios

Table of contents

B1. Tutorial audios

- Bienvenido a VR Supermarket Experience. Con el fin de mejorar la adaptación a la realidad virtual a continuación empezaremos el tutorial. A partir de ahora todas las instrucciones del tutorial podrán volver a reproducirse si usted lo necesita pulsando el botón del lateral del mando, en la parte inferior.
- En esta primera parte vamos a aprender a girar sobre nosotros mismos para observar nuestro alrededor. El primer paso será mirar la esfera roja que tiene enfrente hasta que cambie a color verde.
- Genial. Así es cómo deberá actuar para desplazarse hacia delante. Si quiere desplazarse hacia la izquierda deberá girar sobre si mismo hacia esa dirección. Inténtelo.
- ¡Muy bien! Ahora dese la vuelta para mirar a la esfera restante
- ¡Enhorabuena! Ya estamos finalizando la primera parte del tutorial. Ahora procederemos a explicarle como desplazarse usando los mandos. Mire a la puerta que se encuentra a su derecha. Apriete el panel central del mando derecho en la dirección en la que desea moverse. Por ejemplo si se quiere mover hacia la derecha deberá presionar la parte derecha del panel central. Si se quiere mover hacia la izquierda deberá presionar la parte izquierda. Si quiere avanzar hacia delante deberá presionar la parte superior del botón central. Si por lo contrario quiere retroceder deberá presionar la parte inferior. Siguiendo estos pasos ya se puede acercar a la puerta para seguir con el tutorial.
- ¡Ahora pondremos en práctica lo aprendido! Siga las flechas para llegar al final del pasillo
- ¡Perfecto! Ya nos queda poco. Ahora vamos a familiarizarnos con el ambiente del programa y vamos a interactuar con los objetos que nos encontremos. Deberá coger el paquete de castañas que se encuentra en la parte izquierda de la mesa central y colocarlo en la silueta que aparece en el estante de enfrente. Para coger los objetos debe presionar el gatillo trasero del mando derecho.
- Ahora fíjese que ha aparecido un objeto nuevo en la mesa. Deberá colocarlo en la silueta que aparece en la estantería de enfrente.
- Por último, como ha hecho con los elementos anteriores, deberá colocar el nuevo objeto que ha aparecido en su correspondiente lugar.
- ¡Felicidades! Ha finalizado con éxito el tutorial
- Ahora se podrá familiarizar con el entorno en el que trabajará más adelante. Se puede mover por todo el supermercado tal y como ha aprendido en el tutorial. También puede coger los objetos que desee. Para poner los objetos en la cesta que tiene situada en el mando izquierdo deberá acercar el objeto a la cesta y automáticamente

se colocará dentro de la cesta. Cuando desee salir del tutorial tendrá que ir a la puerta de salida del supermercado.

- ¡Adelante!

B2. Shopping list audios

- Hoy hay que hacer la compra. En la lista aparecen los productos que deberá poner en la cesta. La lista desaparecerá al cabo de unos segundos. Cuando no recuerde los productos que hay en la lista puede pulsar el botón del menú del mando izquierdo para que le vuelva a aparecer. ¡Comencemos!
- ¡Buenos días! Esperamos que tenga una buena compra.
- ¡Hola buenas! ¿Se ha fijado usted en las ofertas de la sección de congelados?
- ¡Enhorabuena! Ha completado con éxito el ejercicio. Ahora ya se puede dirigir a la puerta de salida del supermercado para terminar el nivel.

B3. Shop assistant audios

- En el centro del supermercado hay una mesa con diferentes productos. Deberá colocar los productos de la mesa en el estante que le corresponda del supermercado. ¡Adelante!
- ¡Enhorabuena! Ha completado con éxito el ejercicio.

B4. Autonomous communities audios

- En la habitación hay estantes con diferentes banderas de comunidades autónomas. En la otra parte de la habitación se encuentra una mesa con diferentes productos. Cada producto es típico de una comunidad autónoma distinta. Tendrá que colocar cada producto en el estante correspondiente. ¡Comencemos!
- ¡Enhorabuena! Ha completado con éxito el ejercicio.

B5. Go shopping audios

B5.1. Level 1

- Bienvenido. Hoy, hay que hacer la compra. Tendrá que añadir a la cesta los 3 productos que prefiera de todo el supermercado, y, posteriormente dirigirse a caja para terminar la compra. ¡Adelante!
- ¡Muy bien! Ya puede dirigirse a caja para proceder con el pago.
- Ahora, tendrá que pagar la compra. Para ello, dispone de billetes y monedas encima

del mostrador, a la izquierda de la cajera. Deberá introducir el importe exacto en el lugar especificado del cajero automático. Cuando considere que ya ha introducido el importe exacto, diríjase a la puerta de salida para terminar el nivel. ¡Comencemos!

- Bienvenido. Hoy, hay que hacer la compra. Tendrá que añadir a la cesta los 5 productos que prefiera de todo el supermercado, y, posteriormente dirigirse a caja para terminar la compra. ¡Comencemos!

B5.2. Level 2

- Ahora tendrá que pagar la compra. Para ello, dispone de billetes encima del mostrador, a la izquierda de la cajera. Deberá introducir el importe que considere oportuno en el lugar especificado del cajero automático. Si es necesario, la cajera le devolverá el cambio. Tenga cuidado, esta cajera a veces se equivoca. ¡Adelante!
- Aquí tiene el cambio. ¿Es correcto? Pulse el botón verde si está conforme con el cambio, o el rojo en caso contrario.
- ¡Muchas gracias por su compra! Ya puede dirigirse a la puerta de salida para terminar el nivel. ¡Hasta pronto!
- ¡Uy perdone! Ahora sí se le entrega el cambio correcto. Ha estado usted muy atento.

B5.3. Level 3

- Aquí tiene el cambio. ¿Es correcto? Coja y arrastre las monedas o billetes que considere que sobran hasta el botón rojo, si es que sobran. Una vez considere que el cambio es correcto, pulse el botón verde. ¡Adelante!

Annex C. Program Codes

Table of contents

C1. “Tutorial” programmed codes

AudioInicioTuto

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AudioInicioTuto : MonoBehaviour
{
    public GameObject LeftController;
    public GameObject RightController;
    public GameObject ProductosCesta;
    public GameObject Objetos;
    public GameObject trigger;
    AudioSource audiosourceinicio;
    public AudioClip AudioInicio;


    // Use this for initialization
    void Start()
    {
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = AudioInicio;
        audiosourceinicio.Play();
        LeftController.SetActive(false);
        RightController.SetActive(false);
        StartCoroutine(tiempoaudioinicial());
        StartCoroutine(inivideo());

    }

    private IEnumerator inivideo()
    {
        yield return new WaitForSeconds(11.0f);
        trigger.SetActive(true);
        StartCoroutine(finvideo());
    }

    private IEnumerator finvideo()
    {
        yield return new WaitForSeconds(12.0f);
        trigger.SetActive(false);
    }

    IEnumerator tiempoaudioinicial()
    {
        yield return new WaitForSeconds(AudioInicio.length);
        LeftController.SetActive(true);
        RightController.SetActive(true);
        ProductosCesta.SetActive(true);
        Objetos.SetActive(true);

    }
}
```

CloseFinalDoors

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CloseFinalDoors : MonoBehaviour {
       public GameObject door1;
       public GameObject door2;
       public GameObject snap1;
       public GameObject controllerLeft;
       public GameObject controllerRight;
    public GameObject PlanoTrigger;
    public int Y = 0;
    Animator animation1;
       Animator animation2;
       AudioSource audiosourcepuerta;
       AudioSource audiosourceexplicacion;
       //public AudioClip puerta;
       public AudioClip explicacion;
       public int contador2;

       // Use this for initialization
       void Start () {
              animation1 = door1.GetComponent<Animator>();
              animation2 = door2.GetComponent<Animator>();
              audiosourcepuerta = door1.GetComponent<AudioSource> ();
              //audiosourcepuerta.clip = puerta;
              audiosourceexplicacion = snap1.GetComponent<AudioSource> ();
              audiosourceexplicacion.clip = explicacion;
       }

       void OnTriggerEnter (Collider collider)
       {
        Y = 1;
        animation1.enabled= true;
              animation2.enabled= true;
              contador2 = contador2 + 1;
              if (contador2 == 1) {

                     //audiosourcepuerta.Play();
                     audiosourceexplicacion.Play ();
                     controllerLeft.SetActive (false);
                     controllerRight.SetActive (false);
                     StartCoroutine (tiempo ());
            StartCoroutine (planoTrigger());
        }

       }
       IEnumerator tiempo()
       {
              yield return new WaitForSeconds (explicacion.length);
              controllerLeft.SetActive (true);
              controllerRight.SetActive (true);
        PlanoTrigger.SetActive(false);
    }
    IEnumerator planoTrigger()
    {
        yield return new WaitForSeconds(18.0f);
        PlanoTrigger.SetActive(true);
    }}
```

CloseInitialDoors

```
using System;
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CloseInitialDoors : MonoBehaviour {
       public GameObject door1;
       public GameObject door2;
       public GameObject arrow1;
       Animator animation1;
       Animator animation2;
       public int contador1 = 0;
    public int contador2 = 0;
    public int X = 0;
    AudioSource audiosourcedireccion;
       //public AudioClip puerta;
       public AudioClip direccion;
       // Use this for initialization
       void Start () {
              animation1 = door1.GetComponent<Animator>();
              animation2 = door2.GetComponent<Animator>();
              audiosourcedireccion = arrow1.GetComponent<AudioSource> ();
              audiosourcedireccion.clip = direccion;
       }


       void OnTriggerEnter (Collider collider)
       {
        X = 1;
        animation1.SetBool ("close",true);
              animation2.SetBool ("close",true);
              contador1 = 1;
              if ((contador1 == 1) & (contador2 == 0)) {
                     audiosourcedireccion.Play ();
            contador2 = 1;
              }

       }
}
```

GoToFinalTuto

```
using System.Collections;
```

```
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class GoToFinalTuto : MonoBehaviour {

	public AudioClip finalsound;

	// Use this for initialization
	void Start () {

		StartCoroutine (FinalTuto ());

	}
	IEnumerator FinalTuto()
	{
		yield return new WaitForSeconds (finalsound.length);
		SceneManager.LoadScene ("Test scene");
	}

}
```

IrMenuRepeticion

```
namespace VRTK
{
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class IrMenuRepeticion : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Tutorial");
        }
    }
}
```

OpenInitialDoors

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class OpenInitialDoors : MonoBehaviour {
    public GameObject flag;
    public GameObject door1;
       public GameObject door2;
       Animator animation1;
       Animator animation2;
       // Use this for initialization
       void Start () {
              animation1 = door1.GetComponent<Animator>();
              animation2 = door2.GetComponent<Animator>();
       }


       void OnTriggerEnter (Collider collider)
       {
        if (flag.GetComponent<Renderer>().material.color == Color.green)
        {
            animation1.SetBool("open", true);
            animation2.SetBool("open", true);
        }

       }
}
```

RaycastTuto

```
using System;
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;

public class RaycastTuto : MonoBehaviour
{
    //public GameObject door1;
    //public GameObject door2;
    public GameObject totem1;
    public GameObject totem2;
    public GameObject totem3;
    //public GameObject LeftController;
    public GameObject RightController;
    private float distanceToSee = 0.0f;
    public GameObject PlanoVideoIzquierda;
    public GameObject PlanovideoDerecha;
    public GameObject PlanoTrackPad;

    AudioSource audiosourcestart;
    AudioSource audiosourceinicio;
    AudioSource audiosourceizquierda;
    AudioSource audiosourcederecha;
    AudioSource audiosourcegiro;
    public AudioClip audiostart;
    public AudioClip audioinicio;
    public AudioClip audioizquierda;
    public AudioClip audioderecha;
    public AudioClip audiogiro;
    public GameObject buttongrip;


    private int counter;
    private int counter2;
    private int counter3;
    private bool tot1 = false;
    private bool tot2 = false;
    private bool tot3 = false;
    //Animator  animation1;
    //Animator animation2;

    // Use this for initialization
    void Start()
    {
        //animation1 = door1.GetComponent<Animator>();
        //animation2 = door2.GetComponent<Animator>();
       // LeftController.SetActive(false);
        RightController.SetActive(false);
        audiosourcestart = GetComponent<AudioSource>();
        audiosourcestart.clip = audiostart;
        audiosourcestart.Play();
        StartCoroutine(Repeticion());
        StartCoroutine(iniciar());
        audiosourceizquierda = totem1.GetComponent<AudioSource>();
        audiosourceizquierda.clip = audioizquierda;
        audiosourcederecha = totem2.GetComponent<AudioSource>();
        audiosourcederecha.clip = audioderecha;
        audiosourcegiro = totem3.GetComponent<AudioSource>();
        audiosourcegiro.clip = audiogiro;

    }

    IEnumerator Repeticion()
    {
        yield return new WaitForSeconds(15.0f);
```

```
        buttongrip.SetActive(true);
    }

    IEnumerator iniciar()
    {
        yield return new WaitForSeconds(audiostart.length);
        StartCoroutine(imagen());
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = audioinicio;
        audiosourceinicio.Play();
        StartCoroutine(tiempo());
    }

    private IEnumerator imagen()
    {
        yield return new WaitForSeconds(3.0f);
        buttongrip.SetActive(false);
    }

    IEnumerator tiempo()
    {
        yield return new WaitForSeconds(audioinicio.length);
        distanceToSee = 200.0f;
    }

    IEnumerator PlanoIzquierda()
    {
        yield return new WaitForSeconds(5.0f);
        PlanoVideoIzquierda.SetActive(true);
    }


    IEnumerator tiempototem1()
    {
        yield return new WaitForSeconds(audioizquierda.length);
        totem2.SetActive(true);
        PlanoVideoIzquierda.SetActive(false);

    }

    IEnumerator tiempototem2()
    {
        yield return new WaitForSeconds(audioderecha.length);
        totem3.SetActive(true);
        PlanovideoDerecha.SetActive(false);
    }

    IEnumerator tiempototem3()
    {
        yield return new WaitForSeconds(audiogiro.length);
        //LeftController.SetActive(true);
        RightController.SetActive(true);
        PlanoTrackPad.SetActive(false);
    }
    // Update is called once per frame
    void Update()
    {
        RaycastHit hit;


        if (Physics.Raycast(this.transform.position, this.transform.forward, out
hit, distanceToSee))
```

```
        {
            if (hit.collider.gameObject == totem1)
            {
                totem1.GetComponent<Renderer>().material.color = Color.green;
                tot1 = true;
                counter = counter + 1;
                if (counter == 1)
                {
                    StartCoroutine(tiempototem1());
                    audiosourceizquierda.Play();
                    StartCoroutine(PlanoIzquierda());
                }
            }
            if (hit.collider.gameObject == totem2)
            {

                totem2.GetComponent<Renderer>().material.color = Color.green;
                tot2 = true;
                counter2 = counter2 + 1;
                if (counter2 == 1)
                {
                    StartCoroutine(tiempototem2());
                    audiosourcederecha.Play();
                    PlanovideoDerecha.SetActive(true);
                }
            }
            if (hit.collider.gameObject == totem3)
            {

                totem3.GetComponent<Renderer>().material.color = Color.green;
                tot3 = true;
                counter3 = counter3 + 1;
                if (counter3 == 1)
                {
                    PlanoTrackPad.SetActive(true);
                    StartCoroutine(tiempototem3());
                    audiosourcegiro.Play();
                }
            }


            //if (tot1 == true && tot2 == true && tot3 == true) {
            //audiosourcegiro.enabled = false;
            //totem1.SetActive(false);
            //totem2.SetActive(false);
            //totem3.SetActive(false);

            //animation1.enabled= true;
            //animation2.enabled= true;
            //LeftController.SetActive (true);
            //RightController.SetActive (true);
            //}

        }
    }
}
```

C2. “Shopping list” programmed codes

Nivel 1

AudioInicioMainScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AudioInicioMainScene : MonoBehaviour {
    public GameObject LeftController;
    public GameObject RightController;
    public GameObject Nota;
    public GameObject Timer;
    public GameObject ProductosCesta;
    public GameObject Objetos;


    AudioSource audiosourceinicio;

    public AudioClip AudioInicio;


    // Use this for initialization
    void Start()
    {
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = AudioInicio;
        audiosourceinicio.Play();
        LeftController.SetActive(false);
        RightController.SetActive(false);
        StartCoroutine(tiempoaudioinicial());

    }
    IEnumerator tiempoaudioinicial()
    {
        yield return new WaitForSeconds(AudioInicio.length);
        Nota.SetActive(true);
        LeftController.SetActive(true);
        RightController.SetActive(true);
        ProductosCesta.SetActive(true);
        Objetos.SetActive(true);

    }
}
```

IrMenuRepeticion1

```
namespace VRTK
{
    using System.Collections;
```

```
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class IrMenuRepeticion1 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Lista 1");
        }
    }
}
```

Listatres

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ListaTres : MonoBehaviour {

       public Text Lista;
       private string[] listas = {
           "Garbanzos\nAceite\nSal",
              "Leche\nPan\nPlátano",
              "Pollo\nGambas\nManzana",
        "Bistec\nPizza\nSandía",
        "Lentejas\nMiel\nTomate",
        "Pepino\nVinagre\nMelocotón",
        "Azúcar\nMantequilla\nHelado",
        "Arroz\nPan bimbo\nGuisantes"
    };
       public int pet;

       // Use this for initialization
       void Start () {

              pet =Random.Range(0,listas.Length);
              print (listas[pet]);

              Lista.text = listas [pet];

       }

       // Update is called once per frame
       void Update () {

       }
}
```

SobreEscribirDatos

```
using System.Collections;
using System.Collections.Generic;
```

```
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatos : MonoBehaviour {

	private GameObject GameControlDatos;
	private int ID;
	private int OV3 ;
	private int ONV3 ;
	private int T3 ;

	// Use this for initialization
	void Start () {

		GameControlDatos = GameObject.Find ("GameControl");
		ID= GameControlDatos.GetComponent<GameControl>().ID;
		OV3 = GameControlDatos.GetComponent<GameControl> ().OV3;
		ONV3 = GameControlDatos.GetComponent<GameControl> ().ONV3;
		T3 = GameControlDatos.GetComponent<GameControl> ().T3;
	}

	public void Escena1(){

		string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
		IDbConnection dbconn;
		dbconn = (IDbConnection) new SqliteConnection(conn);
		dbconn.Open(); //Open connection to the database.

		IDbCommand dbcmd = dbconn.CreateCommand();
		string sqlQuery = "UPDATE Datos SET
ObjetosVal3='"+OV3+"',ObjetosNoVal3='"+ONV3+"',T3='"+T3+"' WHERE ID="+ID;
		dbcmd.CommandText = sqlQuery;
		IDataReader reader = dbcmd.ExecuteReader();


		reader.Close();
		reader = null;

		dbcmd.Dispose();
		dbcmd = null;

		dbconn.Close();
		dbconn = null;


	}
}
```

StopGravity

```
using System;
using System.Collections;
	using System.Collections.Generic;
```

```
using UnityEngine;
using UnityEngine.UI;

public class StopGravity : MonoBehaviour {

       public Text ObjetosCesta;
       private string ObjetosPrevios;
       private string ObjetosPrevios2;
       public GameObject referencia;
       private int V1=0;
       private int V2=0;
       private int V3=0;
       public int Valid=0;
       public int NoVal=0;
       private int pet2;
       public GameObject time;
       public int timeint;
    public GameObject señal;
    private Collider capa;
    public GameObject quitar2;
    private Collider cesta2;
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    private int x = 0;

    // Use this for initialization
    void Start () {

        pet2 = referencia.GetComponent<ListaTres>().pet;
    }

    // Update is called once per frame
    void OnTriggerEnter (Collider other) {

        if (other.tag != "Untagged") {

                    other.GetComponent<Rigidbody> ();
                    //other.attachedRigidbody.isKinematic = true;
                    other.gameObject.transform.parent = transform;

                    if (pet2 == 0) {

                          if (other.tag == "Garbanzos") {
                                V1 = 1;
                                print (V1 + ("V1"));
                                print (Valid+("Valid"));
                          }
                          if (other.tag == "Aceite") {
                                V2 = 1;
                                print (V2 + ("V2"));
                                print (Valid+("Valid"));
                          }
                          if (other.tag == "Sal") {
                                V3 = 1;
                                print (V3 + ("V3"));
                                print (Valid+("Valid"));
                          }
                          if (other.tag !="Garbanzos" && other.tag !="Aceite" &&
other.tag !="Sal")
                          {
```

```
                        NoVal = NoVal + 1;
                        print (NoVal+("NoVal"));
                    }

                }
                if (pet2 == 1) {

                    if (other.tag == "Leche") {
                        V1 = 1;
                        print (V1 + ("V1"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag == "Pan") {
                        V2 = 1;
                        print (V2 + ("V2"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag == "Plátanos") {
                        V3 = 1;
                        print (V3 + ("V3"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag !="Leche" && other.tag !="Pan" &&
other.tag !="Plátanos")
                    {
                        NoVal = NoVal + 1;
                        print (NoVal+("NoVal"));
                    }

                }
                if (pet2 == 2) {

                    if (other.tag == "Pollo") {
                        V1 = 1;
                        print (V1 + ("V1"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag == "Gambas") {
                        V2 = 1;
                        print (V2 + ("V2"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag == "Manzana") {
                        V3 = 1;
                        print (V3 + ("V3"));
                        print (Valid+("Valid"));
                    }
                    if (other.tag !="Pollo" && other.tag != "Gambas" &&
other.tag !="Manzana")
                    {
                        NoVal = NoVal + 1;
                        print (NoVal+("NoVal"));
                    }

                }

                if (pet2 == 3) {

                    if (other.tag == "Bistec") {
                        V1 = 1;
                        print (V1 + ("V1"));
                        print (Valid+("Valid"));
```

```
                        }
                        if (other.tag == "Pizza") {
                              V2 = 1;
                              print (V2 + ("V2"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag == "Sandía") {
                              V3 = 1;
                              print (V3 + ("V3"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag !="Bistec" && other.tag !="Pizza" &&
other.tag != "Sandía")
                        {
                              NoVal = NoVal + 1;
                              print (NoVal+("NoVal"));
                        }

                  }
                  if (pet2 == 4) {

                        if (other.tag == "Lentejas") {
                              V1 = 1;
                              print (V1 + ("V1"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag == "Miel") {
                              V2 = 1;
                              print (V2 + ("V2"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag == "Tomate") {
                              V3 = 1;
                              print (V3 + ("V3"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag != "Lentejas" && other.tag != "Miel" &&
other.tag !="Tomate")
                        {
                              NoVal = NoVal + 1;
                              print (NoVal+("NoVal"));
                        }

                  }
                  if (pet2 == 5) {

                        if (other.tag == "Pepino") {
                              V1 = 1;
                              print (V1 + ("V1"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag == "Vinagre") {
                              V2 = 1;
                              print (V2 + ("V2"));
                              print (Valid+("Valid"));
                        }
                        if (other.tag == "Melocotón") {
                              V3 = 1;
                              print (V3 + ("V3"));
                              print (Valid+("Valid"));
                        }
```

```
                        if (other.tag != "Pepino" && other.tag !="Vinagre" &&
other.tag != "Melocotón")
                        {
                                NoVal = NoVal + 1;
                                print (NoVal+("NoVal"));
                        }

                }
                if (pet2 == 6) {

                        if (other.tag == "Azúcar") {
                                V1 = 1;
                                print (V1 + ("V1"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag == "Mantequilla") {
                                V2 = 1;
                                print (V2 + ("V2"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag == "Helado") {
                                V3 = 1;
                                print (V3 + ("V3"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag != "Azúcar" && other.tag !=
"Mantequilla" && other.tag != "Helado")
                        {
                                NoVal = NoVal + 1;
                                print (NoVal+("NoVal"));
                        }

                }
                if (pet2 == 7) {

                        if (other.tag == "Arroz") {
                                V1 = 1;
                                print (V1 + ("V1"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag == "Pan bimbo") {
                                V2 = 1;
                                print (V2 + ("V2"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag == "Guisantes") {
                                V3 = 1;
                                print (V3 + ("V3"));
                                print (Valid+("Valid"));
                        }
                        if (other.tag !="Arroz" && other.tag != "Pan bimbo" &&
other.tag !="Guisantes")
                        {
                                NoVal = NoVal + 1;
                                print (NoVal+("NoVal"));
                        }
                }


                other.gameObject.SetActive (false);

        ObjetosCesta.text= string.Format("\n"+other.tag+ObjetosPrevios2);
```

```
                    ObjetosPrevios= ObjetosCesta.text;
                    ObjetosPrevios2= ObjetosPrevios;
        }

       }

       void Update (){

        capa = señal.GetComponent<Collider>();

        if (capa.enabled == true)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = true;
        }
        if (capa.enabled == false)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = false;
        }
        if ((Valid == 3) & (x == 0))
        {
            x = 1;
            audiosourcemusic = GetComponent<AudioSource>();
            audiosourcemusic.clip = Audiomusic;
            audiosourcemusic.Play();
            StartCoroutine(exito());
        }
        Valid = V1 + V2 + V3;
              timeint= (int)time.GetComponent<TimeCounter>().time;

              }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
    }
}
```

Nivel 2

AudioInicioMainScene23

```
using System;
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AudioInicioMainScene23 : MonoBehaviour {
    public GameObject LeftController;
    public GameObject RightController;
    AudioSource audiosourceinicio;
    public AudioClip AudioInicio;
    public GameObject Nota;
    public GameObject Imagen;
    public GameObject Timer;
    public GameObject ProductosCesta;
    public GameObject Objetos;

    // Use this for initialization
    void Start () {
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = AudioInicio;
        audiosourceinicio.Play();
        LeftController.SetActive(false);
        RightController.SetActive(false);
        StartCoroutine(tiempoaudioinicial());
        StartCoroutine(imagenmando());

    }

    private IEnumerator imagenmando()
    {
        yield return new WaitForSeconds(10.0f);
        Imagen.SetActive(true);
    }

    IEnumerator tiempoaudioinicial()
    {
        yield return new WaitForSeconds(AudioInicio.length);
        Imagen.SetActive(false);
        Nota.SetActive(true);
        StartCoroutine(iniciolista());


    }

    private IEnumerator iniciolista()
    {
        yield return new WaitForSeconds(10.0f);
        ProductosCesta.SetActive(true);
        Objetos.SetActive(true);
        Nota.SetActive(false);
        LeftController.SetActive(true);
        RightController.SetActive(true);
    }
}
```

IrMenuRepeticion2

```
namespace VRTK
{
    using System.Collections;
```

```
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class IrMenuRepeticion2 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Lista 2");
        }
    }
}
```

ListaCinco

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ListaCinco : MonoBehaviour {

       public Text Lista;
       private string[] listas = {
              "Ajo\nVinagre\nSal\nNaranja\nGambas",
              "Arroz\nPollo\nGuisantes\nAceite\nCebolla",
              "Lasaña\nBistec\nMantequilla\nAlmendras\nGarbanzos",
        "Manzana\nPizza\nLeche\nAzúcar\nMacarrones",
        "Sal\nPan\nPera\nTomate\nHarina",
        "Pan bimbo\nMermelada\nHelado\nLentejas\nPepino"
    };

       public int pet;

       // Use this for initialization
       void Start () {

              pet =Random.Range(0,listas.Length);
              print (listas[pet]);

              Lista.text = listas [pet];

       }

       // Update is called once per frame
       void Update () {

       }
}
```

SobreEscribirDatos5

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatos5 : MonoBehaviour {

	private GameObject GameControlDatos;
	private int ID;
	private int OV5 ;
	private int ONV5 ;
	private int T5 ;
	private int C5;

	// Use this for initialization
	void Start () {

		GameControlDatos = GameObject.Find ("GameControl");
		ID= GameControlDatos.GetComponent<GameControl>().ID;
		OV5 = GameControlDatos.GetComponent<GameControl> ().OV5;
		ONV5 = GameControlDatos.GetComponent<GameControl> ().ONV5;
		T5 = GameControlDatos.GetComponent<GameControl> ().T5;
		C5 = GameControlDatos.GetComponent<GameControl> ().C5;
	}

	public void Escena2(){

		string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
		IDbConnection dbconn;
		dbconn = (IDbConnection) new SqliteConnection(conn);
		dbconn.Open(); //Open connection to the database.

		IDbCommand dbcmd = dbconn.CreateCommand();
		string sqlQuery = "UPDATE Datos SET
ObjetosVal5='"+OV5+"',ObjetosNoVal5='"+ONV5+"',T5='"+T5+"',Cons5='"+C5+"' WHERE
ID="+ID;
		dbcmd.CommandText = sqlQuery;
		IDataReader reader = dbcmd.ExecuteReader();


		reader.Close();
		reader = null;

		dbcmd.Dispose();
		dbcmd = null;

		dbconn.Close();
		dbconn = null;


	}
}
```

StopGravity5

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using UnityEngine.UI;

public class StopGravity5 : MonoBehaviour {

	public Text ObjetosCesta;
	private string ObjetosPrevios;
	private string ObjetosPrevios2;
	public GameObject referencia;
	private int V1=0;
	private int V2= 0;
	private int V3= 0;
	private int V4= 0;
	private int V5= 0;
	private int pet2;
	public int Valid=0;
	public int NoVal=0;
    public GameObject señal;
    private Collider capa;
    public GameObject quitar2;
    private Collider cesta2;
    public GameObject time;
    public int timeint;
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    private int x = 0;

    // Use this for initialization
    void Start () {

		pet2 = referencia.GetComponent<ListaCinco> ().pet;

	}

	void OnTriggerEnter (Collider other) {

		if (other.tag != "Untagged") {

			other.GetComponent<Rigidbody> ();
			//other.attachedRigidbody.isKinematic = true;
			other.gameObject.transform.parent = transform;

			if (pet2 == 0) {

				if (other.tag == "Ajo") {
					V1 = 1;
					print (V1);
				}
				if (other.tag == "Vinagre") {
					V2 = 1;
					print (V2);
				}
				if (other.tag == "Sal") {
					V3 = 1;
					print (V3);
				}
				if (other.tag == "Naranja") {
					V4 = 1;
					print (V4);
				}
				if (other.tag == "Gambas") {
```

```
                                V5 = 1;
                                print (V5);
                        }
                        if (other.tag != "Ajo" && other.tag !="Vinagre" &&
other.tag !="Sal" && other.tag != "Naranja" && other.tag != "Gambas")
                        {
                                NoVal = NoVal + 1;
                                print (NoVal+("NoVal"));
                        }
                }

                if (pet2 == 1) {

                        if (other.tag == "Arroz") {
                                V1 = 1;
                                print (V1);
                        }
                        if (other.tag == "Pollo") {
                                V2 = 1;
                                print (V2);
                        }
                        if (other.tag == "Guisantes") {
                                V3 = 1;
                                print (V3);
                        }
                        if (other.tag == "Aceite") {
                                V4 = 1;
                                print (V4);
                        }
                        if (other.tag == "Cebolla") {
                                V5 = 1;
                                print (V5);
                        }
                        if (other.tag !="Arroz" && other.tag !="Pollo" &&
other.tag !="Guisantes" && other.tag != "Aceite" && other.tag !="Cebolla")
                        {
                                NoVal = NoVal + 1;
                                print (NoVal+("NoVal"));
                        }
                }

                if (pet2 == 2) {

                        if (other.tag == "Lasaña") {
                                V1 = 1;
                                print (V1);
                        }
                        if (other.tag == "Bistec") {
                                V2 = 1;
                                print (V2);
                        }
                        if (other.tag == "Mantequilla") {
                                V3 = 1;
                                print (V3);
                        }
                        if (other.tag == "Almendras") {
                                V4 = 1;
                                print (V4);
                        }
                        if (other.tag == "Garbanzos") {
                                V5 = 1;
                                print (V5);
```

```
                        }
                        if (other.tag != "Lasaña" && other.tag !="Bistec" &&
other.tag != "Mantequilla" && other.tag != "Almendras" && other.tag
!="Garbanzos")
                        {
                            NoVal = NoVal + 1;
                            print (NoVal+("NoVal"));
                        }
                    }

            if (pet2 == 3)
            {

                if (other.tag == "Manzana")
                {
                    V1 = 1;
                    print(V1);
                }
                if (other.tag == "Pizza")
                {
                    V2 = 1;
                    print(V2);
                }
                if (other.tag == "Leche")
                {
                    V3 = 1;
                    print(V3);
                }
                if (other.tag == "Azúcar")
                {
                    V4 = 1;
                    print(V4);
                }
                if (other.tag == "Macarrones")
                {
                    V5 = 1;
                    print(V5);
                }
                if (other.tag != "Manzana" && other.tag != "Pizza" && other.tag
!= "Leche" && other.tag != "Azúcar" && other.tag != "Macarrones")
                {
                    NoVal = NoVal + 1;
                    print(NoVal + ("NoVal"));
                }
            }

            if (pet2 == 4)
            {

                if (other.tag == "Sal")
                {
                    V1 = 1;
                    print(V1);
                }
                if (other.tag == "Pan")
                {
                    V2 = 1;
                    print(V2);
                }
                if (other.tag == "Pera")
                {
                    V3 = 1;
```

```
                    print(V3);
                }
                if (other.tag == "Tomate")
                {
                    V4 = 1;
                    print(V4);
                }
                if (other.tag == "Harina")
                {
                    V5 = 1;
                    print(V5);
                }
                if (other.tag != "Sal" && other.tag != "Pan" && other.tag !=
"Pera" && other.tag != "Tomate" && other.tag != "Harina")
                {
                    NoVal = NoVal + 1;
                    print(NoVal + ("NoVal"));
                }
            }

            if (pet2 == 5)
            {

                if (other.tag == "Pan bimbo")
                {
                    V1 = 1;
                    print(V1);
                }
                if (other.tag == "Mermelada")
                {
                    V2 = 1;
                    print(V2);
                }
                if (other.tag == "Helado")
                {
                    V3 = 1;
                    print(V3);
                }
                if (other.tag == "Lentejas")
                {
                    V4 = 1;
                    print(V4);
                }
                if (other.tag == "Pepino")
                {
                    V5 = 1;
                    print(V5);
                }
                if (other.tag != "Pan bimbo" && other.tag != "Mermelada" &&
other.tag != "Helado" && other.tag != "Lentejas" && other.tag != "Pepino")
                {
                    NoVal = NoVal + 1;
                    print(NoVal + ("NoVal"));
                }
            }

            other.gameObject.SetActive (false);

                    ObjetosCesta.text=
string.Format("\n"+other.tag+ObjetosPrevios2);
                    ObjetosPrevios= ObjetosCesta.text;
                    ObjetosPrevios2= ObjetosPrevios;
```

```
            }
        }

        // Update is called once per frame
        void Update () {
          capa = señal.GetComponent<Collider>();

          if (capa.enabled == true)
          {
              cesta2 = quitar2.GetComponent<Collider>();
              cesta2.enabled = true;
          }
          if (capa.enabled == false)
          {
              cesta2 = quitar2.GetComponent<Collider>();
              cesta2.enabled = false;
          }
          if ((Valid == 5) & (x == 0))
          {
              x = 1;
              audiosourcemusic = GetComponent<AudioSource>();
              audiosourcemusic.clip = Audiomusic;
              audiosourcemusic.Play();
              StartCoroutine(exito());
          }
          Valid = V1 + V2 + V3 + V4 + V5;
                timeint= (int)time.GetComponent<TimeCounter>().time;
        }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
    }
}
```

Nivel 3

IrMenuRepeticion3

```
namespace VRTK
{
```

```
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class IrMenuRepeticion3 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Lista 3");
        }
    }
}
```

ListaSiete

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ListaSiete : MonoBehaviour {

       public Text Lista;
       private string[] listas = {
              "Pan\nMantequilla\nLentejas\nTomate\nZanahoria\nSal\nAceite",
              "Helado\nBistec\nGarbanzos\nAzúcar\nLeche\nPatata\nPasta de
dientes",
              "Pan bimbo\nTampax\nMermelada\nMelocotón\nPollo\nPizza\nPimienta",
        "Gambas\nVinagre\nHarina\nArroz\nCerezas\nMiel\nPiña",
        "Limón\nAlubias\nSandía\nCafé\nDesodorante\nFresas\nAlmendras"
    };
       public int pet;

       // Use this for initialization
       void Start () {

              pet =Random.Range(0,listas.Length);
              print (listas[pet]);

              Lista.text = listas [pet];

       }

       // Update is called once per frame
       void Update () {

       }
}
```

SobreEscribirDatos7

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatos7 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
       private int OV7 ;
       private int ONV7 ;
       private int T7 ;
    private int C7;
       // Use this for initialization
       void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              OV7 = GameControlDatos.GetComponent<GameControl> ().OV7;
              ONV7 = GameControlDatos.GetComponent<GameControl> ().ONV7;
              T7 = GameControlDatos.GetComponent<GameControl> ().T7;
         C7 = GameControlDatos.GetComponent<GameControl>().C7;
    }

       public void Escena3(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET
ObjetosVal7='"+OV7+"',ObjetosNoVal7='"+ONV7+"',T7='"+T7+"',Cons7='"+C7+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

StopGravity7

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

```
using UnityEngine.UI;

public class StopGravity7 : MonoBehaviour {

	public Text ObjetosCesta;
	private string ObjetosPrevios;
	private string ObjetosPrevios2;
	public GameObject referencia;
	private int V1=0;
	private int V2= 0;
	private int V3= 0;
	private int V4= 0;
	private int V5= 0;
	private int V6= 0;
	private int V7= 0;
	private int pet2;
	public int Valid=0;
	public int NoVal=0;
	public GameObject time;
	public int timeint;
    public GameObject señal;
    private Collider capa;
    public GameObject quitar2;
    private Collider cesta2;
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    private int x = 0;

    // Use this for initialization
    void Start () {

		pet2 = referencia.GetComponent<ListaSiete> ().pet;

	}

	void OnTriggerEnter (Collider other) {

		if (other.tag != "Untagged") {

			other.GetComponent<Rigidbody> ();
			//other.attachedRigidbody.isKinematic = true;
			other.gameObject.transform.parent = transform;

			if (pet2 == 0) {

				if (other.tag == "Pan") {
					V1 = 1;
					print (V1);
				}
				if (other.tag == "Mantequilla") {
					V2 = 1;
					print (V2);
				}
				if (other.tag == "Lentejas") {
					V3 = 1;
					print (V3);
				}
				if (other.tag == "Tomate") {
					V4 = 1;
					print (V4);
```

```
                        }
                        if (other.tag == "Zanahoria") {
                              V5 = 1;
                              print (V5);
                        }
                        if (other.tag == "Sal") {
                              V6 = 1;
                              print (V5);
                        }
                        if (other.tag == "Aceite") {
                              V7 = 1;
                              print (V5);
                        }
                        if (other.tag != "Pan" && other.tag != "Mantequilla"
&& other.tag != "Lentejas" && other.tag !="Tomate" && other.tag !="Zanahoria" &&
other.tag !="Sal" && other.tag !="Aceite")
                        {
                              NoVal = NoVal + 1;
                              print (NoVal+("NoVal"));
                        }
                  }

                  if (pet2 == 1) {

                        if (other.tag == "Helado") {
                              V1 = 1;
                              print (V1);
                        }
                        if (other.tag == "Bistec") {
                              V2 = 1;
                              print (V2);
                        }
                        if (other.tag == "Garbanzos") {
                              V3 = 1;
                              print (V3);
                        }
                        if (other.tag == "Azúcar") {
                              V4 = 1;
                              print (V4);
                        }
                        if (other.tag == "Leche") {
                              V5 = 1;
                              print (V5);
                        }
                        if (other.tag == "Patata") {
                              V6 = 1;
                              print (V5);
                        }
                        if (other.tag == "Pasta de dientes") {
                              V7 = 1;
                              print (V5);
                        }
                        if (other.tag !="Helado" && other.tag !="Bistec" &&
other.tag !="Garbanzos" && other.tag != "Azúcar" && other.tag != "Leche" &&
other.tag !="Patata" && other.tag != "Pasta de dientes")
                        {
                              NoVal = NoVal + 1;
                              print (NoVal+("NoVal"));
                        }
                  }
                  if (pet2 == 2) {
```

```
                        if (other.tag == "Pan bimbo") {
                               V1 = 1;
                               print (V1);
                        }
                        if (other.tag == "Tampax") {
                               V2 = 1;
                               print (V2);
                        }
                        if (other.tag == "Mermelada") {
                               V3 = 1;
                               print (V3);
                        }
                        if (other.tag == "Melocotón") {
                               V4 = 1;
                               print (V4);
                        }
                        if (other.tag == "Pollo") {
                               V5 = 1;
                               print (V5);
                        }
                        if (other.tag == "Pizza") {
                               V6 = 1;
                               print (V5);
                        }
                        if (other.tag == "Pimienta") {
                               V7 = 1;
                               print (V5);
                        }
                        if (other.tag !="Pan bimbo" && other.tag != "Tampax"
&& other.tag !="Mermelada" && other.tag != "Melocotón" && other.tag != "Pollo" &&
other.tag != "Pizza" && other.tag != "Pimienta")
                        {
                               NoVal = NoVal + 1;
                               print (NoVal+("NoVal"));
                        }
                  }
            if (pet2 == 3)
            {

                if (other.tag == "Gambas")
                {
                    V1 = 1;
                    print(V1);
                }
                if (other.tag == "Vinagre")
                {
                    V2 = 1;
                    print(V2);
                }
                if (other.tag == "Harina")
                {
                    V3 = 1;
                    print(V3);
                }
                if (other.tag == "Arroz")
                {
                    V4 = 1;
                    print(V4);
                }
                if (other.tag == "Cerezas")
                {
                    V5 = 1;
```

```
                print(V5);
            }
            if (other.tag == "Miel")
            {
                V6 = 1;
                print(V5);
            }
            if (other.tag == "Piña")
            {
                V7 = 1;
                print(V5);
            }
            if (other.tag != "Gambas" && other.tag != "Vinagre" && other.tag
!= "Harina" && other.tag != "Arroz" && other.tag != "Cerezas" && other.tag !=
"Miel" && other.tag != "Piña")
            {
                NoVal = NoVal + 1;
                print(NoVal + ("NoVal"));
            }
        }
        if (pet2 == 4)
        {

            if (other.tag == "Limón")
            {
                V1 = 1;
                print(V1);
            }
            if (other.tag == "Alubias")
            {
                V2 = 1;
                print(V2);
            }
            if (other.tag == "Sandía")
            {
                V3 = 1;
                print(V3);
            }
            if (other.tag == "Café")
            {
                V4 = 1;
                print(V4);
            }
            if (other.tag == "Desodorante")
            {
                V5 = 1;
                print(V5);
            }
            if (other.tag == "Fresas")
            {
                V6 = 1;
                print(V5);
            }
            if (other.tag == "Almendras")
            {
                V7 = 1;
                print(V5);
            }
            if (other.tag != "Limón" && other.tag != "Alubias" && other.tag
!= "Sandía" && other.tag != "Café" && other.tag != "Desodorante" && other.tag !=
"Fresas" && other.tag != "Almendras")
            {
```

```
                    NoVal = NoVal + 1;
                    print(NoVal + ("NoVal"));
                }
            }
            other.gameObject.SetActive (false);

                    ObjetosCesta.text=
string.Format("\n"+other.tag+ObjetosPrevios2);
                    ObjetosPrevios= ObjetosCesta.text;
                    ObjetosPrevios2= ObjetosPrevios;
             }
       }

       // Update is called once per frame
       void Update () {
         capa = señal.GetComponent<Collider>();

         if (capa.enabled == true)
         {
             cesta2 = quitar2.GetComponent<Collider>();
             cesta2.enabled = true;
         }
         if (capa.enabled == false)
         {
             cesta2 = quitar2.GetComponent<Collider>();
             cesta2.enabled = false;
         }
         if ((Valid == 7) & (x == 0))
         {
             x = 1;
             audiosourcemusic = GetComponent<AudioSource>();
             audiosourcemusic.clip = Audiomusic;
             audiosourcemusic.Play();
             StartCoroutine(exito());
         }
         Valid = V1 + V2 + V3 + V4 + V5 + V6 + V7;
               timeint= (int)time.GetComponent<TimeCounter>().time;
       }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
    }
}
```

C3. “Shop assistant” programmed codes

Nivel 1

ChangeLevel1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeLevel1 : MonoBehaviour
{


    public int id;
    private int _leche;
    private int _pizza;
    private int _bimbo;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {
```

```
        if (e.snappedObject.name == "leche")
        {
            productosCestaController.SetObjectTrue(id);


        }

        if (e.snappedObject.name == "bimbo")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "pizza")
        {
            productosCestaController.SetObjectTrue(id);

        }
    }
}
```

SalirNivel4

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel4 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Reponedor 1");
    }
}
```

SobreEscribirDatosReponedor1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosReponedor1 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TR1;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TR1 = GameControlDatos.GetComponent<GameControl> ().TR1;
       }

       public void EscenaReponedor1(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo R1='"+TR1+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Nivel 2

5.2.1. ChangeLevel2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeLevel2 : MonoBehaviour
{


    public int id;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {



        if (e.snappedObject.name == "harina")
        {
```

```
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "berenjena")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "piña")
        {
            productosCestaController.SetObjectTrue(id);
        }

        if (e.snappedObject.name == "ternera")
        {
            productosCestaController.SetObjectTrue(id);
        }



    }


}
```

Salirnivel5

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel5 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Reponedor 2");
```

```
    }
}
```

SobreEscribirDatosReponedor2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosReponedor2 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TR2;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TR2 = GameControlDatos.GetComponent<GameControl> ().TR2;
       }

       public void EscenaReponedor2(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo R2='"+TR2+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Nivel 3

ChangeLevel3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeLevel3 : MonoBehaviour
{


    public int id;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {



        if (e.snappedObject.name == "Chips")
                                    {
```

```
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "Endivia")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "Barra Pan")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "Guisantes")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "pasta")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "Yogurt")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "sandia")
        {
            productosCestaController.SetObjectTrue(id);

        }
    }
}
```

Salirnivel6

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel6 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Reponedor 3");
    }
}
```

5.3.3. SobreEscribirDatosReponedor3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosReponedor3 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TR3;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TR3 = GameControlDatos.GetComponent<GameControl> ().TR3;
       }

       public void EscenaReponedor3(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo R3='"+TR3+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Others

Audio_inicio_reponedor

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class audio_inicio_reponedor : MonoBehaviour
{
    public GameObject LeftController;
    public GameObject RightController;
    AudioSource audiosourceinicio;
    public AudioClip AudioInicio;
    // Use this for initialization
    void Start()
    {
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = AudioInicio;
        audiosourceinicio.Play();
        LeftController.SetActive(false);
        RightController.SetActive(false);
        StartCoroutine(tiempoaudioinicial());
    }

    IEnumerator tiempoaudioinicial()
    {
        yield return new WaitForSeconds(AudioInicio.length);
        LeftController.SetActive(true);
        RightController.SetActive(true);
    }
}
```

Contador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
 using MySql.Data.MySqlClient;

public class Contador : MonoBehaviour {


    private float contador = 5.0f;
    int aciertos = 0;
    string escena;
    string levelname;


    public void Aciertos(int valor)
    {
        aciertos = valor;
    }

}
```

Pause

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class pause : MonoBehaviour {

    public GameObject ActivarPause;

       public void pausar()
    {
        ActivarPause.SetActive(true);
    }
}
```

Pausebutton

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class pausebutton : MonoBehaviour {

       // Use this for initialization
       void Start () {

       }

       // Update is called once per frame
       void Update () {

       }
}
```

ProductosCestasController

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using MySql.Data.MySqlClient;


public class ProductosCestaController : MonoBehaviour
{

    public int nObjects = 3;
    public bool[] objectState;
    public GameObject ObjectToDisable;
    public GameObject Contador;
    private bool all;

    private int naciertos = 0;

    public GameObject contador;
    public int timeint;
    public GameObject time;

    private void Start()
    {

        Contador _contador = contador.GetComponent<Contador>();

        Contador.SetActive(false);
        ObjectToDisable.SetActive(false);

        objectState = new bool[nObjects];
        for (int i = 0; i < nObjects; i++) {
            objectState[i] = false;
        }

    }

    public void SetObjectTrue(int id) {
        objectState[id] = true;
        Debug.Log("Set Active" + id);
        naciertos++;
        Contador _contador = contador.GetComponent<Contador>();
        _contador.Aciertos(naciertos);
        CheckAllState();




    }

    public void CheckAllState()
    {
        all = true;
        for (int i = 0; i < nObjects; i++)
        {


            if (!objectState[i])
            {
                all = false;
            }
```

```
        }

        if (all)
        {
            ObjectToDisable.SetActive(true);
            Contador.SetActive(true);
            time.SetActive(false);

        }
        else
        {
            ObjectToDisable.SetActive(false);
            Contador.SetActive(false);
        }
    }

    void Update()
    {
        timeint = (int)time.GetComponent<TimeCounter>().time;
    }


}
```

C4. “Autonomous Communities” programmed codes

Nivel 1

ChangeComunidades1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeComunidades1 : MonoBehaviour
{


    public int id;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {


        if (e.snappedObject.name == "fuet")
```

```
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "platano")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "gazpacho")
        {
            productosCestaController.SetObjectTrue(id);

        }
    }
}
```

Salirnivel

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start () {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Comunidades 1");
    }

}
```

SobreEscribirDatosComunidades1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosComunidades1 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TC1;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TC1 = GameControlDatos.GetComponent<GameControl> ().TC1;
       }

       public void EscenaComunidades1(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo C1='"+TC1+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Nivel 2

ChangeComunidades2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeComunidades2 : MonoBehaviour
{


    public int id;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {


        if (e.snappedObject.name == "mojopicon")
        {
            productosCestaController.SetObjectTrue(id);
```

```
        }

        if (e.snappedObject.name == "aceitunas")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "sobrasada")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "fabada")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "cava")
        {
            productosCestaController.SetObjectTrue(id);

        }
    }
}
```

Salirnivel2

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel2 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Comunidades 2");
    }

}
```

SobreEscribirDatosComunidades2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosComunidades2 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TC2;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TC2 = GameControlDatos.GetComponent<GameControl> ().TC2;
       }

       public void EscenaComunidades2(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo C2='"+TC2+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Nivel 3

ChangeComunidades3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;
using UnityEngine.SceneManagement;


public class ChangeComunidades3 : MonoBehaviour
{


    public int id;



    ProductosCestaController productosCestaController;

    void Awake()
    {
        VRTK_SDKManager.instance.AddBehaviourToToggleOnLoadedSetupChange(this);
    }

    private void Start()
    {
        productosCestaController = FindObjectOfType<ProductosCestaController>();
    }

    void OnDestroy()
    {

VRTK_SDKManager.instance.RemoveBehaviourToToggleOnLoadedSetupChange(this);
    }
    void OnEnable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone +=
ChangeLevel_ObjectSnappedToDropZone;

        object test1 = null;
        SnapDropZoneEventArgs test2 = new SnapDropZoneEventArgs();
        ChangeLevel_ObjectSnappedToDropZone(test1, test2);

    }
    void OnDisable()
    {

        GetComponent<VRTK_SnapDropZone>().ObjectSnappedToDropZone -=
ChangeLevel_ObjectSnappedToDropZone;
    }


    private void ChangeLevel_ObjectSnappedToDropZone(object sender,
SnapDropZoneEventArgs e)
    {


        if (e.snappedObject.name == "txakoli")
        {
            productosCestaController.SetObjectTrue(id);
```

```
        }

        if (e.snappedObject.name == "vinotinto")
        {
            productosCestaController.SetObjectTrue(id);

        }

        if (e.snappedObject.name == "mejillones")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "naranjas")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "cerezas")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "ensaimada")
        {
            productosCestaController.SetObjectTrue(id);

        }
        if (e.snappedObject.name == "morcilla")
        {
            productosCestaController.SetObjectTrue(id);

        }
    }
}
```

Salirnivel3

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class salirnivel3 : MonoBehaviour {
    AudioSource audiosourceexito;
    public AudioClip Audioexito;
    AudioSource audiosourcemusic;
    public AudioClip Audiomusic;
    // Use this for initialization
    void Start()
    {
        audiosourcemusic = GetComponent<AudioSource>();
        audiosourcemusic.clip = Audiomusic;
        audiosourcemusic.Play();
        StartCoroutine(exito());
    }

    private IEnumerator exito()
    {
        yield return new WaitForSeconds(Audiomusic.length);
        audiosourceexito = GetComponent<AudioSource>();
        audiosourceexito.clip = Audioexito;
        audiosourceexito.Play();
        StartCoroutine(salida());
    }

    private IEnumerator salida()
    {
        yield return new WaitForSeconds(Audioexito.length);
        SceneManager.LoadScene("Final Comunidades 3");
    }
}
```

SobreEscribirDatosComunidades3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosComunidades3 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int TC3;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
              TC3 = GameControlDatos.GetComponent<GameControl> ().TC3;
       }

       public void EscenaComunidades3(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Tiempo C3='"+TC3+"' WHERE
ID="+ID;
              dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

C5. “Go Shopping” programmed codes

Nivel 1

AudioInicioJuegoPrecios

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using VRTK;

public class AudioInicioJuegoPrecios : MonoBehaviour {

    public GameObject BotonOmitirAudio;
    public GameObject InstruccionesAudio;
    private int BotonOmitirUsed;
    public GameObject ProductosCesta;
    public GameObject Objetos;
    public GameObject Precios;
    public GameObject Timer;
    public GameObject TiempoPagar;
    public GameObject PagarCaja;
    public GameObject Pagado;
    public GameObject DineroPagado;
    public GameObject ObjetosTicket;
    public GameObject PreciosTicket;
    public GameObject Ticket;
    public GameObject TotalCompra;
    AudioSource audiosourceinicio;
    public AudioClip AudioInicio;


    // Use this for initialization
    void Start()
    {
        audiosourceinicio = GetComponent<AudioSource>();
        audiosourceinicio.clip = AudioInicio;
        audiosourceinicio.Play();
        BotonOmitirAudio.SetActive(true);
        InstruccionesAudio.SetActive(true);
        Timer.SetActive(false);
        TiempoPagar.SetActive(false);
        PagarCaja.SetActive(false);
        Pagado.SetActive(false);
        DineroPagado.SetActive(false);
        Ticket.SetActive(false);
        ObjetosTicket.SetActive(false);
        PreciosTicket.SetActive(false);
        TotalCompra.SetActive(false);
        StartCoroutine(tiempoaudioinicial());
    }
    void Update()
    {
        if (BotonOmitirAudio.activeSelf == true)
        {
            BotonOmitirUsed =
BotonOmitirAudio.GetComponent<VRTK_InteractableObject>().Grab;
            if (BotonOmitirUsed == 1)
            {
                BotonOmitirAudio.SetActive(false);
                InstruccionesAudio.SetActive(false);
                audiosourceinicio.Stop();
                ProductosCesta.SetActive(true);
```

```
                Objetos.SetActive(true);
                Precios.SetActive(true);
                StopCoroutine(tiempoaudioinicial());
            }
        }
    }

    IEnumerator tiempoaudioinicial()
    {
        yield return new WaitForSeconds(AudioInicio.length);
        BotonOmitirAudio.SetActive(false);
        InstruccionesAudio.SetActive(false);
        ProductosCesta.SetActive(true);
        Objetos.SetActive(true);
        Precios.SetActive(true);

    }
}
```

CambioEscenaPagar1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using VRTK;

public class CambioEscenaPagar1 : MonoBehaviour
{
    public GameObject BotonOmitirAudio;
    public GameObject InstruccionesAudio;
    private int BotonOmitirUsed;
    private int contador;
    public int TiempoEspera;
    public GameObject LeftController;
    public GameObject RightController;
    AudioSource audiosourceinicio;
    public AudioClip AudioPagar;
    public GameObject Trigger;
    public GameObject Timer;
    public GameObject TiempoPagar;
    public GameObject PagarCaja;
    public GameObject Pagado;
    public GameObject DineroPagado;
    public GameObject ObjetosTicket;
    public GameObject PreciosTicket;
    public GameObject Ticket;
    public GameObject TotalCompra;
    public GameObject ProductosCesta;
    public GameObject Objetos;
    public GameObject Precios;
    public GameObject Basket;

    // Use this for initialization
    void Start()
    {
        Timer.SetActive(false);
        TiempoPagar.SetActive(false);
        Pagado.SetActive(false);
        DineroPagado.SetActive(false);
        Ticket.SetActive(false);
        ObjetosTicket.SetActive(false);
```

```
        PreciosTicket.SetActive(false);
        TotalCompra.SetActive(false);
        LeftController.SetActive(true);
        RightController.SetActive(true);
    }
    void OnTriggerEnter(Collider collider)
    {
        contador = contador + 1;
        if (contador == 1)
        {
            BotonOmitirAudio.SetActive(true);
            InstruccionesAudio.SetActive(true);
            ProductosCesta.SetActive(false);
            Objetos.SetActive(false);
            Precios.SetActive(false);
            Destroy(Basket);
            audiosourceinicio = GetComponent<AudioSource>();
            audiosourceinicio.clip = AudioPagar;
            audiosourceinicio.Play();
            RightController.SetActive(false);
            LeftController.SetActive(false);
            StartCoroutine(tiempoaudio());
        }
    }
    void Update()
    {
        if (BotonOmitirAudio.activeSelf == true)
        {
            BotonOmitirUsed =
BotonOmitirAudio.GetComponent<VRTK_InteractableObject>().Grab;
            if (BotonOmitirUsed == 1)
            {
                BotonOmitirAudio.SetActive(false);
                InstruccionesAudio.SetActive(false);
                audiosourceinicio.Stop();
                LeftController.SetActive(true);
                RightController.SetActive(true);
                Timer.SetActive(true);
                TiempoPagar.SetActive(true);
                PagarCaja.SetActive(true);
                Pagado.SetActive(true);
                DineroPagado.SetActive(true);
                ObjetosTicket.SetActive(true);
                PreciosTicket.SetActive(true);
                TotalCompra.SetActive(true);
                Ticket.SetActive(true);
                StopCoroutine(tiempoaudio());
                StartCoroutine(tiempodesaparece());
            }
        }
    }
    IEnumerator tiempoaudio()
    {
        yield return new WaitForSeconds(AudioPagar.length);
        BotonOmitirAudio.SetActive(false);
        InstruccionesAudio.SetActive(false);
        LeftController.SetActive(true);
        RightController.SetActive(true);
        Timer.SetActive(true);
        TiempoPagar.SetActive(true);
        PagarCaja.SetActive(true);
        Pagado.SetActive(true);
```

```
        DineroPagado.SetActive(true);
        ObjetosTicket.SetActive(true);
        PreciosTicket.SetActive(true);
        TotalCompra.SetActive(true);
        Ticket.SetActive(true);
        StartCoroutine(tiempodesaparece());
    }
    IEnumerator tiempodesaparece()
    {
        yield return new WaitForSeconds(TiempoEspera);
        Trigger.SetActive(false);
    }
}
```

CestaJuegoPreciosN1

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CestaJuegoPreciosN1 : MonoBehaviour {

      public Text ObjetosCesta;
    public Text PrecioCesta;
    public Text TotalCompra;
    public Text ObjetosCestaTicket;
    public Text PrecioCestaTicket;
    private string ObjetosPrevios;
      private string ObjetosPrevios2;
    private decimal Precio;
    public decimal TotalCompra1;
    private string Precio2;
    private string TotalCompra2;
    private string PreciosPrevios;
    private string PreciosPrevios2;
    public GameObject señal;
    private Collider capa;
    public GameObject quitar2;
    private Collider cesta2;
    private int Contador = 0;
    public int NumObjetos = 3;
    AudioSource audiosourceirapagar;
    public AudioClip AudioIrAPagar;

    // Use this for initialization
    void Start ()
        {
        TotalCompra1 = 0.00m;
        }

        // Update is called once per frame
        void OnTriggerEnter (Collider other) {

        if (other.tag != "Untagged")
        {
            Contador = Contador + 1;
            if (Contador == NumObjetos)
            {
                audiosourceirapagar = GetComponent<AudioSource>();
                audiosourceirapagar.clip = AudioIrAPagar;
                audiosourceirapagar.Play();
```

```
}

other.GetComponent<Rigidbody>();
//other.attachedRigidbody.isKinematic = true;
other.gameObject.transform.parent = transform;
{
    {
        if (other.tag == "Bistec")
        {
            Precio = 4.50m;
        }
        if (other.tag == "Pizza")
        {
            Precio = 2.59m;
        }
        if (other.tag == "Sandía")
        {
            Precio = 3.55m;
        }
        if (other.tag == "Garbanzos")
        {
            Precio = 1.15m;
        }
        if (other.tag == "Aceite")
        {
            Precio = 3.59m;
        }
        if (other.tag == "Sal")
        {
            Precio = 0.35m;
        }
        if (other.tag == "Leche")
        {
            Precio = 1.12m;
        }
        if (other.tag == "Pan")
        {
            Precio = 0.55m;
        }
        if (other.tag == "Helado")
        {
            Precio = 2.70m;
        }
        if (other.tag == "Gambas")
        {
            Precio = 15.99m;
        }
        if (other.tag == "Lentejas")
        {
            Precio = 3.50m;
        }
        if (other.tag == "Alubias")
        {
            Precio = 1.10m;
        }
        if (other.tag == "Arroz")
        {
            Precio = 0.69m;
        }
        if (other.tag == "Harina")
        {
            Precio = 0.59m;
```

```
}
if (other.tag == "Macarrones")
{
    Precio = 0.75m;
}
if (other.tag == "Manzana")
{
    Precio = 0.53m;
}
if (other.tag == "Plátanos")
{
    Precio = 0.44m;
}
if (other.tag == "Melocotón")
{
    Precio = 0.66m;
}
if (other.tag == "Pera")
{
    Precio = 0.45m;
}
if (other.tag == "Naranja")
{
    Precio = 0.45m;
}
if (other.tag == "Pimiento rojo")
{
    Precio = 0.50m;
}
if (other.tag == "Almendras")
{
    Precio = 1.50m;
}
if (other.tag == "Cacahuetes")
{
    Precio = 1.35m;
}
if (other.tag == "Avellanas")
{
    Precio = 1.99m;
}
if (other.tag == "Pistachos")
{
    Precio = 4.50m;
}
if (other.tag == "Castañas")
{
    Precio = 4.35m;
}
if (other.tag == "Mezcla")
{
    Precio = 1.80m;
}
if (other.tag == "Chips")
{
    Precio = 0.80m;
}
if (other.tag == "Patata")
{
    Precio = 0.30m;
}
if (other.tag == "Espárragos")
```

```
{
    Precio = 1.20m;
}
if (other.tag == "Pepino")
{
    Precio = 0.50m;
}
if (other.tag == "Zanahoria")
{
    Precio = 0.40m;
}
if (other.tag == "Tomate")
{
    Precio = 0.60m;
}
if (other.tag == "Ajo")
{
    Precio = 0.75m;
}
if (other.tag == "Cebolla")
{
    Precio = 0.50m;
}
if (other.tag == "Berenjena")
{
    Precio = 0.70m;
}
if (other.tag == "Kiwi")
{
    Precio = 0.89m;
}
if (other.tag == "Coliflor")
{
    Precio = 1.20m;
}
if (other.tag == "Endivia")
{
    Precio = 1.00m;
}
if (other.tag == "Aguacate")
{
    Precio = 2.30m;
}
if (other.tag == "Té")
{
    Precio = 3.55m;
}
if (other.tag == "Café")
{
    Precio = 2.20m;
}
if (other.tag == "Miel")
{
    Precio = 4.99m;
}
if (other.tag == "Chocolate")
{
    Precio = 1.10m;
}
if (other.tag == "Mermelada")
{
    Precio = 1.30m;
```

```
}
if (other.tag == "Ensalada")
{
    Precio = 1.10m;
}
if (other.tag == "Moras")
{
    Precio = 2.55m;
}
if (other.tag == "Uvas")
{
    Precio = 1.60m;
}
if (other.tag == "Setas")
{
    Precio = 2.25m;
}
if (other.tag == "Cerezas")
{
    Precio = 2.60m;
}
if (other.tag == "Fresas")
{
    Precio = 3.54m;
}
if (other.tag == "Pollo")
{
    Precio = 3.42m;
}
if (other.tag == "Pescado")
{
    Precio = 4.68m;
}
if (other.tag == "Mantequilla")
{
    Precio = 0.80m;
}
if (other.tag == "Queso")
{
    Precio = 2.40m;
}
if (other.tag == "Yogur")
{
    Precio = 0.69m;
}
if (other.tag == "Pan bimbo")
{
    Precio = 1.20m;
}
if (other.tag == "Guisantes")
{
    Precio = 0.70m;
}
if (other.tag == "Lasaña")
{
    Precio = 3.30m;
}
if (other.tag == "Patatas congeladas")
{
    Precio = 1.40m;
}
if (other.tag == "Maíz")
```

```
{
    Precio = 1.20m;
}
if (other.tag == "Palitos de pescado")
{
    Precio = 3.50m;
}
if (other.tag == "Piña")
{
    Precio = 2.70m;
}
if (other.tag == "Melón")
{
    Precio = 3.05m;
}
if (other.tag == "Limón")
{
    Precio = 0.65m;
}
if (other.tag == "Rabano")
{
    Precio = 1.15m;
}
if (other.tag == "Coco")
{
    Precio = 2.60m;
}
if (other.tag == "Cereales")
{
    Precio = 1.80m;
}
if (other.tag == "Azúcar")
{
    Precio = 0.60m;
}
if (other.tag == "Huevos")
{
    Precio = 2.30m;
}
if (other.tag == "Pimienta")
{
    Precio = 0.80m;
}
if (other.tag == "Vinagre")
{
    Precio = 0.70m;
}
if (other.tag == "Tampax")
{
    Precio = 3.70m;
}
if (other.tag == "Pasta de dientes")
{
    Precio = 1.20m;
}
if (other.tag == "Desodorante")
{
    Precio = 2.30m;
}
if (other.tag == "Papel higiénico")
{
    Precio = 3.40m;
```

```
                    }
                    if (other.tag == "Gel")
                    {
                        Precio = 1.40m;
                    }

                }
            }
            TotalCompra1 = TotalCompra1 + Precio;
            other.gameObject.SetActive(false);

            ObjetosCesta.text = string.Format("\n" + other.tag +
ObjetosPrevios2);
            ObjetosPrevios = ObjetosCesta.text;
            ObjetosPrevios2 = ObjetosPrevios;

            Precio2 = Precio.ToString() + "€";
            PrecioCesta.text = string.Format("\n" + Precio2 + PreciosPrevios2);
            PreciosPrevios = PrecioCesta.text;
            PreciosPrevios2 = PreciosPrevios;

            ObjetosCestaTicket.text = ObjetosCesta.text;
            PrecioCestaTicket.text = PrecioCesta.text;
        }

    }

    void Update()
    {
        TotalCompra2 = TotalCompra1.ToString() + " €";
        TotalCompra.text = string.Format("\n" + "TOTAL " + TotalCompra2);

        capa = señal.GetComponent<Collider>();

        if (capa.enabled == true)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = true;
        }
        if (capa.enabled == false)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = false;
        }
    }
}
```

FinalPagar1

```
namespace VRTK
{
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class FinalPagar1 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Pagar 1");
        }
    }
}
```

ObtenerPrecioCompraN1

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using VRTK;

public class ObtenerPrecioCompraN1 : MonoBehaviour {

    private GameObject TotalCompra;
    private decimal APagar;
    private string PrecioCompra1;
    public decimal ValorAObtener;
    public GameObject Trigger;

    void Start ()
    {}

    void OnTriggerEnter(Collider collider)
    {
            TotalCompra = GameObject.Find("DentroCesta");
            APagar =
TotalCompra.GetComponent<CestaJuegoPreciosN1>().TotalCompra1;
            ValorAObtener = APagar;
            Trigger.GetComponent<BoxCollider>().enabled = false;
    }
}
```

PagarCajaN1

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PagarCajaN1 : MonoBehaviour {

    public Text DineroPagado;
    private string DineroPrevio;
       private string DineroPrevio2;
    private decimal Valor;
    public GameObject Señal;
    private Collider Capa;
    public GameObject Quitar;
    private Collider Caja;
    public decimal TotalPagado=0.00m;
    private GameObject ValorCompra;
    public decimal PrecioCompra;
    public int Valid = 0;
    public int NoValExc = 0;
    public int NoValDef = 0;


    // Use this for initialization
    void Start ()
    {
    }

    void OnTriggerEnter (Collider other)
    {
        if (other.tag != "Untagged") {

                    other.GetComponent<Rigidbody> ();
                    //other.attachedRigidbody.isKinematic = true;
                    other.gameObject.transform.parent = transform;
            {
                {
                    if (other.tag == "20.00€")
                    {
                        Valor = 20.00m;
                        print(Valor);
                    }
                    if (other.tag == "10.00€")
                    {
                        Valor = 10.00m;
                        print(Valor);
                    }
                    if (other.tag == "5.00€")
                    {
                        Valor = 5.00m;
                        print(Valor);
                    }
                    if (other.tag == "2.00€")
                    {
                        Valor = 2.00m;
                        print(Valor);
                    }
                    if (other.tag == "1.00€")
                    {
```

```
                    Valor = 1.00m;
                    print(Valor);
                }
                if (other.tag == "0.50€")
                {
                    Valor = 0.50m;
                    print(Valor);
                }
                if (other.tag == "0.20€")
                {
                    Valor = 0.20m;
                    print(Valor);
                }
                if (other.tag == "0.10€")
                {
                    Valor = 0.10m;
                    print(Valor);
                }
                if (other.tag == "0.05€")
                {
                    Valor = 0.05m;
                    print(Valor);
                }
                if (other.tag == "0.02€")
                {
                    Valor = 0.02m;
                    print(Valor);
                }
                if (other.tag == "0.01€")
                {
                    Valor = 0.01m;
                    print(Valor);
                }
            }
        }

        other.gameObject.SetActive (false);

        DineroPagado.text= string.Format("\n"+other.tag+ DineroPrevio2);
        DineroPrevio = DineroPagado.text;
        DineroPrevio2 = DineroPrevio;

        TotalPagado = TotalPagado + Valor;
        ValorCompra = GameObject.Find("DineroCompra");
        PrecioCompra =
ValorCompra.GetComponent<ObtenerPrecioCompraN1>().ValorAObtener;

        if (TotalPagado < PrecioCompra)
        {
            NoValDef = 1;
            Valid = 0;
            NoValExc = 0;
            print(NoValDef + "NoValDef");
        }
        if (TotalPagado == PrecioCompra)
        {
            NoValDef = 0;
            Valid = 1;
            NoValExc = 0;
            print(Valid + "Valid");
        }
        if (TotalPagado > PrecioCompra)
```

```
            {
                NoValDef = 0;
                Valid = 0;
                NoValExc = 1;
                print(NoValDef + "NoValExc");
            }
        }
      }

      void Update ()
    {
        Capa = Señal.GetComponent<Collider>();

        if (Capa.enabled == true)
        {
            Caja = Caja.GetComponent<Collider>();
            Caja.enabled = true;
        }
        if (Capa.enabled == false)
        {
            Caja = Quitar.GetComponent<Collider>();
            Caja.enabled = false;
        }
      }
}
```

SobreEscribirDatosP1

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosP1 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int VP1;
    private int NVEP1;
    private int NVDP1;
    private int TP1;

    // Use this for initialization
    void Start () {

              GameControlDatos = GameObject.Find ("GameControl");
              ID= GameControlDatos.GetComponent<GameControl>().ID;
        VP1 = GameControlDatos.GetComponent<GameControl> ().VP1;
        NVEP1 = GameControlDatos.GetComponent<GameControl> ().NVEP1;
        NVDP1 = GameControlDatos.GetComponent<GameControl> ().NVDP1;
        TP1 = GameControlDatos.GetComponent<GameControl>().TP1;
    }

       public void EscenaPagar1(){

              string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
              IDbConnection dbconn;
              dbconn = (IDbConnection) new SqliteConnection(conn);
              dbconn.Open(); //Open connection to the database.

              IDbCommand dbcmd = dbconn.CreateCommand();
              string sqlQuery = "UPDATE Datos SET Pagado válido P1='" + VP1 +
"',Pagado no válido por Exceso P1='" + NVEP1 +
            "',Pagado no válido por Defecto P1='" + NVDP1 + "',Tiempo P1='" + TP1
+ "' WHERE ID="+ID;

        dbcmd.CommandText = sqlQuery;
              IDataReader reader = dbcmd.ExecuteReader();


              reader.Close();
              reader = null;

              dbcmd.Dispose();
              dbcmd = null;

              dbconn.Close();
              dbconn = null;


       }
}
```

Nivel 2

CestaJuegoPreciosN2

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CestaJuegoPreciosN2 : MonoBehaviour {

       public Text ObjetosCesta;
    public Text PrecioCesta;
    public Text TotalCompra;
    public Text ObjetosCestaTicket;
    public Text PrecioCestaTicket;
    private string ObjetosPrevios;
       private string ObjetosPrevios2;
    private decimal Precio;
    public decimal TotalCompra1;
    private string Precio2;
    private string TotalCompra2;
    private string PreciosPrevios;
    private string PreciosPrevios2;
    public GameObject señal;
    private Collider capa;
    public GameObject quitar2;
    private Collider cesta2;
    private int Contador = 0;
    public int NumObjetos = 5;
    AudioSource audiosourceirapagar;
    public AudioClip AudioIrAPagar;

    // Use this for initialization
    void Start ()
        {
        TotalCompra1 = 0.00m;
        }

        // Update is called once per frame
        void OnTriggerEnter (Collider other) {

        if (other.tag != "Untagged")
        {
            Contador = Contador + 1;
            if (Contador == NumObjetos)
            {
                audiosourceirapagar = GetComponent<AudioSource>();
                audiosourceirapagar.clip = AudioIrAPagar;
                audiosourceirapagar.Play();
            }

            other.GetComponent<Rigidbody>();
            //other.attachedRigidbody.isKinematic = true;
            other.gameObject.transform.parent = transform;
            {
                {
                    if (other.tag == "Bistec")
                    {
                        Precio = 4.50m;
                    }
                    if (other.tag == "Pizza")
                                                {
```

```
    Precio = 2.59m;
}
if (other.tag == "Sandía")
{
    Precio = 3.55m;
}
if (other.tag == "Garbanzos")
{
    Precio = 1.15m;
}
if (other.tag == "Aceite")
{
    Precio = 3.59m;
}
if (other.tag == "Sal")
{
    Precio = 0.35m;
}
if (other.tag == "Leche")
{
    Precio = 1.12m;
}
if (other.tag == "Pan")
{
    Precio = 0.55m;
}
if (other.tag == "Helado")
{
    Precio = 2.70m;
}
if (other.tag == "Gambas")
{
    Precio = 15.99m;
}
if (other.tag == "Lentejas")
{
    Precio = 3.50m;
}
if (other.tag == "Alubias")
{
    Precio = 1.10m;
}
if (other.tag == "Arroz")
{
    Precio = 0.69m;
}
if (other.tag == "Harina")
{
    Precio = 0.59m;
}
if (other.tag == "Macarrones")
{
    Precio = 0.75m;
}
if (other.tag == "Manzana")
{
    Precio = 0.53m;
}
if (other.tag == "Plátanos")
{
    Precio = 0.44m;
}
```

```
if (other.tag == "Melocotón")
{
    Precio = 0.66m;
}
if (other.tag == "Pera")
{
    Precio = 0.45m;
}
if (other.tag == "Naranja")
{
    Precio = 0.45m;
}
if (other.tag == "Pimiento rojo")
{
    Precio = 0.50m;
}
if (other.tag == "Almendras")
{
    Precio = 1.50m;
}
if (other.tag == "Cacahuetes")
{
    Precio = 1.35m;
}
if (other.tag == "Avellanas")
{
    Precio = 1.99m;
}
if (other.tag == "Pistachos")
{
    Precio = 4.50m;
}
if (other.tag == "Castañas")
{
    Precio = 4.35m;
}
if (other.tag == "Mezcla")
{
    Precio = 1.80m;
}
if (other.tag == "Chips")
{
    Precio = 0.80m;
}
if (other.tag == "Patata")
{
    Precio = 0.30m;
}
if (other.tag == "Espárragos")
{
    Precio = 1.20m;
}
if (other.tag == "Pepino")
{
    Precio = 0.50m;
}
if (other.tag == "Zanahoria")
{
    Precio = 0.40m;
}
if (other.tag == "Tomate")
{
```

```
    Precio = 0.60m;
}
if (other.tag == "Ajo")
{
    Precio = 0.75m;
}
if (other.tag == "Cebolla")
{
    Precio = 0.50m;
}
if (other.tag == "Berenjena")
{
    Precio = 0.70m;
}
if (other.tag == "Kiwi")
{
    Precio = 0.89m;
}
if (other.tag == "Coliflor")
{
    Precio = 1.20m;
}
if (other.tag == "Endivia")
{
    Precio = 1.00m;
}
if (other.tag == "Aguacate")
{
    Precio = 2.30m;
}
if (other.tag == "Té")
{
    Precio = 3.55m;
}
if (other.tag == "Café")
{
    Precio = 2.20m;
}
if (other.tag == "Miel")
{
    Precio = 4.99m;
}
if (other.tag == "Chocolate")
{
    Precio = 1.10m;
}
if (other.tag == "Mermelada")
{
    Precio = 1.30m;
}
if (other.tag == "Ensalada")
{
    Precio = 1.10m;
}
if (other.tag == "Moras")
{
    Precio = 2.55m;
}
if (other.tag == "Uvas")
{
    Precio = 1.60m;
}
```

```
if (other.tag == "Setas")
{
    Precio = 2.25m;
}
if (other.tag == "Cerezas")
{
    Precio = 2.60m;
}
if (other.tag == "Fresas")
{
    Precio = 3.54m;
}
if (other.tag == "Pollo")
{
    Precio = 3.42m;
}
if (other.tag == "Pescado")
{
    Precio = 4.68m;
}
if (other.tag == "Mantequilla")
{
    Precio = 0.80m;
}
if (other.tag == "Queso")
{
    Precio = 2.40m;
}
if (other.tag == "Yogur")
{
    Precio = 0.69m;
}
if (other.tag == "Pan bimbo")
{
    Precio = 1.20m;
}
if (other.tag == "Guisantes")
{
    Precio = 0.70m;
}
if (other.tag == "Lasaña")
{
    Precio = 3.30m;
}
if (other.tag == "Patatas congeladas")
{
    Precio = 1.40m;
}
if (other.tag == "Maíz")
{
    Precio = 1.20m;
}
if (other.tag == "Palitos de pescado")
{
    Precio = 3.50m;
}
if (other.tag == "Piña")
{
    Precio = 2.70m;
}
if (other.tag == "Melón")
{
```

```
                    Precio = 3.05m;
                }
                if (other.tag == "Limón")
                {
                    Precio = 0.65m;
                }
                if (other.tag == "Rabano")
                {
                    Precio = 1.15m;
                }
                if (other.tag == "Coco")
                {
                    Precio = 2.60m;
                }
                if (other.tag == "Cereales")
                {
                    Precio = 1.80m;
                }
                if (other.tag == "Azúcar")
                {
                    Precio = 0.60m;
                }
                if (other.tag == "Huevos")
                {
                    Precio = 2.30m;
                }
                if (other.tag == "Pimienta")
                {
                    Precio = 0.80m;
                }
                if (other.tag == "Vinagre")
                {
                    Precio = 0.70m;
                }
                if (other.tag == "Tampax")
                {
                    Precio = 3.70m;
                }
                if (other.tag == "Pasta de dientes")
                {
                    Precio = 1.20m;
                }
                if (other.tag == "Desodorante")
                {
                    Precio = 2.30m;
                }
                if (other.tag == "Papel higiénico")
                {
                    Precio = 3.40m;
                }
                if (other.tag == "Gel")
                {
                    Precio = 1.40m;
                }

            }
        }
        TotalCompra1 = TotalCompra1 + Precio;
        other.gameObject.SetActive(false);

        ObjetosCesta.text = string.Format("\n" + other.tag +
ObjetosPrevios2);
```

```
            ObjetosPrevios = ObjetosCesta.text;
            ObjetosPrevios2 = ObjetosPrevios;

            Precio2 = Precio.ToString() + "€";
            PrecioCesta.text = string.Format("\n" + Precio2 + PreciosPrevios2);
            PreciosPrevios = PrecioCesta.text;
            PreciosPrevios2 = PreciosPrevios;

            ObjetosCestaTicket.text = ObjetosCesta.text;
            PrecioCestaTicket.text = PrecioCesta.text;
        }

    }

    void Update()
    {
        TotalCompra2 = TotalCompra1.ToString() + " €";
        TotalCompra.text = string.Format("\n" + "TOTAL " + TotalCompra2);

        capa = señal.GetComponent<Collider>();

        if (capa.enabled == true)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = true;
        }
        if (capa.enabled == false)
        {
            cesta2 = quitar2.GetComponent<Collider>();
            cesta2.enabled = false;
        }
    }
}
```

FinalPagar2

```
namespace VRTK
{
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class FinalPagar2 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Pagar 2");
        }
    }
}
```

ObtenerPrecioCompraN2i3

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using VRTK;

public class ObtenerPrecioCompraN2i3 : MonoBehaviour
{
    private GameObject TotalCompra;
    private decimal APagar;
    private string PrecioCompra1;
    public decimal ValorAObtener;
    public GameObject Trigger;

    void Start()
    { }

    void OnTriggerEnter(Collider collider)
    {
        TotalCompra = GameObject.Find("DentroCesta");
        APagar = TotalCompra.GetComponent<CestaJuegoPreciosN2>().TotalCompra1;
        ValorAObtener = APagar;
        Trigger.GetComponent<BoxCollider>().enabled = false;
    }
}
```

PagarCajaN2

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using VRTK;


public class PagarCajaN2 : MonoBehaviour
{
    public Text DineroPagado;
    private string DineroPrevio;
    private string DineroPrevio2;
    private decimal Valor;
    public GameObject Señal;
    private Collider Capa;
    public GameObject Quitar;
    private Collider Caja;
    private decimal TotalPagado = 0.00m;
    private GameObject ValorCompra;
    private decimal PrecioCompra;

    private decimal Cambio;
    public decimal Cambio1;
    private int ValorRandom;
    private decimal CambioRandom;
    private decimal CambioRandom1;
    public int TiempoAparecerCambio = 2;
    AudioSource aparececambio;
    public AudioClip ApareceCambio;
    private List<decimal> Cambios = new List<decimal>();

    public GameObject Menu;
```

```
public GameObject TextoCambio;
public GameObject Euros;
public GameObject VeinteEuros;
public GameObject DiezEuros;
public GameObject CincoEuros;
public GameObject DosEuros;
public GameObject DosEuros1;
public GameObject UnEuro;
public GameObject CincuentaCentimos;
public GameObject VeinteCentimos;
public GameObject VeinteCentimos1;
public GameObject DiezCentimos;
public GameObject CincoCentimos;
public GameObject DosCentimos;
public GameObject DosCentimos1;
public GameObject UnCentimo;
public GameObject CeroEuros;

public GameObject BotonCorrecto;
private int BotonCorrectoUsed;
public GameObject BotonIncorrecto;
private int BotonIncorrectoUsed;
AudioSource botoncorrectopulsado;
public AudioClip AudioBotonCorrecto;
AudioSource botonincorrectopulsado;
public AudioClip AudioBotonIncorrecto;
public int TiempoCambioCorrecto = 6;
public int TiempoCambioIncorrecto = 1;


public int AciertoCC = 0;
public int AciertoCI = 0;
public int ErrorCCCI = 0;
public int ErrorCICC = 0;
public int NoValDef = 0;

// Use this for initialization
void Start()
{
}

void OnTriggerEnter(Collider other)
{
    if (other.tag != "Untagged")
    {
        other.GetComponent<Rigidbody>();
        //other.attachedRigidbody.isKinematic = true;
        other.gameObject.transform.parent = transform;
        {
            {
                if (other.tag == "20.00€")
                {
                    Valor = 20.00m;
                }
                if (other.tag == "10.00€")
                {
                    Valor = 10.00m;
                }
                if (other.tag == "5.00€")
                {
                        Valor = 5.00m;
                }
```

```
                }
            }

            other.gameObject.SetActive(false);

            DineroPagado.text = string.Format(other.tag + "  " + DineroPrevio2);
            DineroPrevio = DineroPagado.text;
            DineroPrevio2 = DineroPrevio;

            TotalPagado = TotalPagado + Valor;
            ValorCompra = GameObject.Find("DineroCompra");
            PrecioCompra =
ValorCompra.GetComponent<ObtenerPrecioCompraN2i3>().ValorAObtener;

            Cambio = TotalPagado - PrecioCompra;
            Cambio1 = Cambio;

            if (TotalPagado < PrecioCompra)
            {
                NoValDef = 1;
            }
            if (TotalPagado >= PrecioCompra)
            {
                NoValDef = 0;
                StartCoroutine(tiempocambio());
                Cambios.Add(Cambio);
                Cambios.Add(Cambio + 1.00m);
                Cambios.Add(Cambio + 1.20m);
                Cambios.Add(Cambio + 1.50m);
                Cambios.Add(Cambio + 2.00m);
                ValorRandom = UnityEngine.Random.Range(0, Cambios.Count);
                CambioRandom = Cambios[ValorRandom];
                CambioRandom1 = CambioRandom;
            }
        }
    }

    IEnumerator tiempocambio()
    {
        yield return new WaitForSeconds(TiempoAparecerCambio);
        Euros.SetActive(true);
        TextoCambio.SetActive(true);
        BotonCorrecto.SetActive(true);
        BotonIncorrecto.SetActive(true);
        aparececambio = GetComponent<AudioSource>();
        aparececambio.clip = ApareceCambio;
        aparececambio.Play();

        if (CambioRandom1 >= 20.00m)
        {
            VeinteEuros.SetActive(true);
            CambioRandom1 = CambioRandom1 - 20.00m;
        }
        if (CambioRandom1 >= 10.00m)
        {
            DiezEuros.SetActive(true);
            CambioRandom1 = CambioRandom1 - 10.00m;
        }
        if (CambioRandom1 >= 5.00m)
        {
            CincoEuros.SetActive(true);
            CambioRandom1 = CambioRandom1 - 5.00m;
```

```
        }
        if (CambioRandom1 >= 2.00m)
        {
            DosEuros.SetActive(true);
            CambioRandom1 = CambioRandom1 - 2.00m;
        }
        if (CambioRandom1 >= 2.00m && DosEuros.activeSelf == true)
        {
            DosEuros1.SetActive(true);
            CambioRandom1 = CambioRandom1 - 2.00m;
        }
        if (CambioRandom1 >= 1.00m)
        {
            UnEuro.SetActive(true);
            CambioRandom1 = CambioRandom1 - 1.00m;
        }
        if (CambioRandom1 >= 0.50m)
        {
            CincuentaCentimos.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.50m;
        }
        if (CambioRandom1 >= 0.20m)
        {
            VeinteCentimos.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.20m;
        }
        if (CambioRandom1 >= 0.20m && VeinteCentimos.activeSelf == true)
        {
            VeinteCentimos1.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.20m;
        }
        if (CambioRandom1 >= 0.10m)
        {
            DiezCentimos.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.10m;
        }
        if (CambioRandom1 >= 0.05m)
        {
            CincoCentimos.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.05m;
        }
        if (CambioRandom1 >= 0.02m)
        {
            DosCentimos.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.02m;
        }
        if (CambioRandom1 >= 0.02m && DosCentimos.activeSelf == true)
        {
            DosCentimos1.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.02m;
        }
        if (CambioRandom1 >= 0.01m)
        {
            UnCentimo.SetActive(true);
            CambioRandom1 = CambioRandom1 - 0.01m;
        }
        if (CambioRandom == 0.00m)
        {
            CeroEuros.SetActive(true);
        }
    }
```

```
    void Update()
    {
        Capa = Señal.GetComponent<Collider>();
        if (Capa.enabled == true)
        {
            Caja = Quitar.GetComponent<Collider>();
            Caja.enabled = true;
        }
        if (Capa.enabled == false)
        {
            Caja = Quitar.GetComponent<Collider>();
            Caja.enabled = false;
        }

        if (BotonCorrecto.activeSelf == true)
        {
            BotonCorrectoUsed =
BotonCorrecto.GetComponent<VRTK_InteractableObject>().Grab;
            if (BotonCorrectoUsed == 1)
            {
                BotonCorrecto.SetActive(false);
                BotonIncorrecto.SetActive(false);
                Menu.SetActive(false);
                botoncorrectopulsado = GetComponent<AudioSource>();
                botoncorrectopulsado.clip = AudioBotonCorrecto;
                botoncorrectopulsado.Play();
                if (CambioRandom == Cambio)
                {
                    AciertoCC = 1;
                }
                else
                {
                    ErrorCICC = 1;
                }
            }
        }
        if (BotonIncorrecto.activeSelf == true)
        {
            BotonIncorrectoUsed =
BotonIncorrecto.GetComponent<VRTK_InteractableObject>().Grab;
            if (BotonIncorrectoUsed == 1)
            {
                BotonCorrecto.SetActive(false);
                BotonIncorrecto.SetActive(false);
                StartCoroutine(tiempoaudiobotonincorrecto());
                botonincorrectopulsado = GetComponent<AudioSource>();
                botonincorrectopulsado.clip = AudioBotonIncorrecto;
                botonincorrectopulsado.Play();
                if (CambioRandom != Cambio)
                {
                    AciertoCI = 1;
                }
                else
                {
                    ErrorCCCI = 1;
                }
            }
        }
    }
    IEnumerator tiempoaudiobotonincorrecto()
    {
        yield return new WaitForSeconds(AudioBotonIncorrecto.length);
```

```
        StartCoroutine(tiempobotonincorrecto2());
        VeinteEuros.SetActive(false);
        DiezEuros.SetActive(false);
        CincoEuros.SetActive(false);
        DosEuros.SetActive(false);
        DosEuros1.SetActive(false);
        UnEuro.SetActive(false);
        CincuentaCentimos.SetActive(false);
        VeinteCentimos.SetActive(false);
        VeinteCentimos1.SetActive(false);
        DiezCentimos.SetActive(false);
        CincoCentimos.SetActive(false);
        DosCentimos.SetActive(false);
        DosCentimos1.SetActive(false);
        UnCentimo.SetActive(false);
        CeroEuros.SetActive(false);
    }
    IEnumerator tiempobotonincorrecto2()
    {
        yield return new WaitForSeconds(TiempoCambioIncorrecto);
        StartCoroutine(tiempobotonincorrecto3());
        if (Cambio1 >= 20.00m)
        {
            VeinteEuros.SetActive(true);
            Cambio1 = Cambio1 - 20.00m;
        }
        if (Cambio1 >= 10.00m)
        {
            DiezEuros.SetActive(true);
            Cambio1 = Cambio1 - 10.00m;
        }
        if (Cambio1 >= 5.00m)
        {
            CincoEuros.SetActive(true);
            Cambio1 = Cambio1 - 5.00m;
        }
        if (Cambio1 >= 2.00m)
        {
            DosEuros.SetActive(true);
            Cambio1 = Cambio1 - 2.00m;
        }
        if (Cambio1 >= 2.00m && DosEuros.activeSelf == true)
        {
            DosEuros1.SetActive(true);
            Cambio1 = Cambio1 - 2.00m;
        }
        if (Cambio1 >= 1.00m)
        {
            UnEuro.SetActive(true);
            Cambio1 = Cambio1 - 1.00m;
        }
        if (Cambio1 >= 0.50m)
        {
            CincuentaCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.50m;
        }
        if (Cambio1 >= 0.20m)
        {
            VeinteCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.20m;
        }
        if (Cambio1 >= 0.20m && VeinteCentimos.activeSelf == true)
```

```
        {
            VeinteCentimos1.SetActive(true);
            Cambio1 = Cambio1 - 0.20m;
        }
        if (Cambio1 >= 0.10m)
        {
            DiezCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.10m;
        }
        if (Cambio1 >= 0.05m)
        {
            CincoCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.05m;
        }
        if (Cambio1 >= 0.02m)
        {
            DosCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.02m;
        }
        if (Cambio1 >= 0.02m && DosCentimos.activeSelf == true)
        {
            DosCentimos1.SetActive(true);
            Cambio1 = Cambio1 - 0.02m;
        }
        if (Cambio1 >= 0.01m)
        {
            UnCentimo.SetActive(true);
            Cambio1 = Cambio1 - 0.01m;
        }
        if (Cambio == 0.00m)
        {
            CeroEuros.SetActive(true);
        }
    }
    IEnumerator tiempobotonincorrecto3()
    {
        yield return new WaitForSeconds(TiempoCambioCorrecto);
        Menu.SetActive(false);
        botoncorrectopulsado = GetComponent<AudioSource>();
        botoncorrectopulsado.clip = AudioBotonCorrecto;
        botoncorrectopulsado.Play();
    }
}
```

SobreEscribirDatosP2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosP2 : MonoBehaviour {

       private GameObject GameControlDatos;
       private int ID;
    private int VCCP2;
    private int VCIP2;
    private int NVCCCIP2;
    private int NVCICCP2;
    private int NVDP2;
    private int TP2;

    // Use this for initialization
    void Start () {

            GameControlDatos = GameObject.Find ("GameControl");
        ID = GameControlDatos.GetComponent<GameControl>().ID;
        VCCP2 = GameControlDatos.GetComponent<GameControl>().VCCP2;
        VCIP2 = GameControlDatos.GetComponent<GameControl>().VCIP2;
        NVCCCIP2 = GameControlDatos.GetComponent<GameControl>().NVCCCIP2;
        NVCICCP2 = GameControlDatos.GetComponent<GameControl>().NVCICCP2;
        NVDP2 = GameControlDatos.GetComponent<GameControl>().NVDP2;
        TP2 = GameControlDatos.GetComponent<GameControl>().TP2;
    }

       public void EscenaPagar2(){

            string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
            IDbConnection dbconn;
            dbconn = (IDbConnection) new SqliteConnection(conn);
            dbconn.Open(); //Open connection to the database.

            IDbCommand dbcmd = dbconn.CreateCommand();
            string sqlQuery = "UPDATE Datos SET Acierto con el cambio correcto
P2='" + VCCP2 +
           "',Acierto con el cambio incorrecto P2='" + VCIP2 + "',Error cambio
correcto y pulsa incorrecto P2='" + NVCCCIP2 +
           "',Error cambio incorrecto y pulsa correcto  P2='" + NVCICCP2 +
"',Error Pagado no válido por Defecto P2='" + NVDP2 +
           "',Tiempo P2='" + TP2+"' WHERE ID="+ID;

        dbcmd.CommandText = sqlQuery;
            IDataReader reader = dbcmd.ExecuteReader();


            reader.Close();
            reader = null;

            dbcmd.Dispose();
            dbcmd = null;

            dbconn.Close();
            dbconn = null;
```

```
        }
}
```

Nivel 3

BotonEliminarCambio

```
    using System;
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.UI;
    using VRTK;

public class BotonEliminarCambio : MonoBehaviour
{
    private decimal Valor;
    public GameObject Señal;
    private Collider Capa;
    public GameObject Quitar;
    private Collider Boton;
    private GameObject CambioRandom;
    public decimal Diferencia;


    // Use this for initialization
    void Start()
    {
        CambioRandom = GameObject.Find("DentroCaja");
        Diferencia = CambioRandom.GetComponent<PagarCajaN3>().CambioRandom;
    }

    // Update is called once per frame
    void OnTriggerEnter(Collider other)
    {
        if (other.tag != "Untagged")
        {
            other.GetComponent<Rigidbody>();
            //other.attachedRigidbody.isKinematic = true;
            other.gameObject.transform.parent = transform;
            {
                {
                    if (other.tag == "20.00€")
                    {
                        Valor = 20.00m;
                    }
                    if (other.tag == "10.00€")
                    {
                        Valor = 10.00m;
                    }
                    if (other.tag == "5.00€")
                    {
                        Valor = 5.00m;
                    }
                    if (other.tag == "2.00€")
                    {
                        Valor = 2.00m;
                    }
                    if (other.tag == "1.00€")
                    {
                        Valor = 1.00m;
```

```
                    }
                    if (other.tag == "0.50€")
                    {
                        Valor = 0.50m;
                    }
                    if (other.tag == "0.20€")
                    {
                        Valor = 0.20m;
                    }
                    if (other.tag == "0.10€")
                    {
                        Valor = 0.10m;
                    }
                    if (other.tag == "0.05€")
                    {
                        Valor = 0.05m;
                    }
                    if (other.tag == "0.02€")
                    {
                        Valor = 0.02m;
                    }
                    if (other.tag == "0.01€")
                    {
                        Valor = 0.01m;
                    }
                }
            }
            other.gameObject.SetActive(false);

            Diferencia = Diferencia - Valor;
        }
    }

    void Update()
    {

        Capa = Señal.GetComponent<Collider>();

        if (Capa.enabled == true)
        {
            Boton = Quitar.GetComponent<Collider>();
            Boton.enabled = true;
        }
        if (Capa.enabled == false)
        {
            Boton = Quitar.GetComponent<Collider>();
            Boton.enabled = false;
        }
    }
}
```

FinalPagar3

```
namespace VRTK
{
    using System.Collections;
    using System.Collections.Generic;
    using UnityEngine;
    using UnityEngine.SceneManagement;

    public class FinalPagar3 : MonoBehaviour
    {



        // Use this for initialization
        void Start()
        {

        }
        void OnTriggerEnter(Collider collider)
        {
            SceneManager.LoadScene("Final Pagar 3");
        }
    }
}
```

PagarCajaN3

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using VRTK;

public class PagarCajaN3 : MonoBehaviour
{
    public Text DineroPagado;
    private string DineroPrevio;
    private string DineroPrevio2;
    private decimal Valor;
    public GameObject Señal;
    private Collider Capa;
    public GameObject Quitar;
    private Collider Caja;
    private decimal TotalPagado = 0.00m;
    private GameObject ValorCompra;
    private decimal PrecioCompra;

    private decimal Cambio;
    private decimal Cambio1;
    private int ValorRandom;
    public decimal CambioRandom;
    public int TiempoAparecerCambio = 2;
    AudioSource aparececambio;
    public AudioClip ApareceCambio;
    private List<decimal> Cambios = new List<decimal>();

    public GameObject Menu;
    public GameObject TextoCambio;
    public GameObject Euros;
    public GameObject VeinteEuros;
    public GameObject DiezEuros;
    public GameObject CincoEuros;
    public GameObject DosEuros;
    public GameObject DosEuros1;
    public GameObject DosEurosExtra;
    public GameObject UnEuro;
    public GameObject UnEuroExtra;
    public GameObject CincuentaCentimos;
    public GameObject CincuentaCentimosExtra;
    public GameObject VeinteCentimos;
    public GameObject VeinteCentimos1;
    public GameObject VeinteCentimosExtra;
    public GameObject DiezCentimos;
    public GameObject DiezCentimosExtra;
    public GameObject CincoCentimos;
    public GameObject DosCentimos;
    public GameObject DosCentimos1;
    public GameObject UnCentimo;
    public GameObject CeroEuros;

    public GameObject BotonCorrecto;
    private int BotonCorrectoUsed;
    public GameObject BotonEliminarDinero;
    public AudioClip AudioBotonCorrecto;
```

```
    AudioSource botoncorrectopulsado;

    private GameObject DineroEliminado;
    private decimal Diferencia;

    public int Acierto = 0;
    public int ErrorExc = 0;
    public int ErrorDef = 0;
    public int NoValDef = 0;

    void Start()
    {
    }

    void OnTriggerEnter(Collider other)
    {

        if (other.tag != "Untagged")
        {
            other.GetComponent<Rigidbody>();
            //other.attachedRigidbody.isKinematic = true;
            other.gameObject.transform.parent = transform;
            {
                {
                    if (other.tag == "20.00€")
                    {
                        Valor = 20.00m;
                    }
                    if (other.tag == "10.00€")
                    {
                        Valor = 10.00m;
                    }
                    if (other.tag == "5.00€")
                    {
                        Valor = 5.00m;
                    }
                }
            }

            other.gameObject.SetActive(false);

            DineroPagado.text = string.Format(other.tag + "   " + DineroPrevio2);
            DineroPrevio = DineroPagado.text;
            DineroPrevio2 = DineroPrevio;

            TotalPagado = TotalPagado + Valor;
            ValorCompra = GameObject.Find("DineroCompra");
            PrecioCompra =
ValorCompra.GetComponent<ObtenerPrecioCompraN2i3>().ValorAObtener;
            Debug.Log(PrecioCompra + "PrecioCompra");

            Cambio = TotalPagado - PrecioCompra;
            Cambio1 = Cambio;

            if (TotalPagado < PrecioCompra)
            {
                NoValDef = 1;
            }
            if (TotalPagado >= PrecioCompra)
            {
                NoValDef = 0;
                StartCoroutine(tiempocambio());
```

```
            Cambios.Add(0.00m);
            Cambios.Add(0.20m);
            Cambios.Add(0.50m);
            Cambios.Add(1.20m);
            Cambios.Add(2.30m);
            ValorRandom = UnityEngine.Random.Range(0, Cambios.Count);
            CambioRandom = Cambios[ValorRandom];
        }
    }
}

IEnumerator tiempocambio()
{
    yield return new WaitForSeconds(TiempoAparecerCambio);
    Euros.SetActive(true);
    TextoCambio.SetActive(true);
    BotonCorrecto.SetActive(true);
    BotonEliminarDinero.SetActive(true);
    aparececambio = GetComponent<AudioSource>();
    aparececambio.clip = ApareceCambio;
    aparececambio.Play();

    if (Cambio1 >= 20.00m)
    {
        VeinteEuros.SetActive(true);
        Cambio1 = Cambio1 - 20.00m;
    }
    if (Cambio1 >= 10.00m)
    {
        DiezEuros.SetActive(true);
        Cambio1 = Cambio1 - 10.00m;
    }
    if (Cambio1 >= 5.00m)
    {
        CincoEuros.SetActive(true);
        Cambio1 = Cambio1 - 5.00m;
    }
    if (Cambio1 >= 2.00m)
    {
        DosEuros.SetActive(true);
        Cambio1 = Cambio1 - 2.00m;
    }
    if (Cambio1 >= 2.00m && DosEuros.activeSelf == true)
    {
        DosEuros1.SetActive(true);
        Cambio1 = Cambio1 - 2.00m;
    }
    if (Cambio1 >= 1.00m)
    {
        UnEuro.SetActive(true);
        Cambio1 = Cambio1 - 1.00m;
    }
    if (Cambio1 >= 0.50m)
    {
        CincuentaCentimos.SetActive(true);
        Cambio1 = Cambio1 - 0.50m;
    }
    if (Cambio1 >= 0.20m)
    {
        VeinteCentimos.SetActive(true);
        Cambio1 = Cambio1 - 0.20m;
    }
```

```
        if (Cambio1 >= 0.20m && VeinteCentimos.activeSelf == true)
        {
            VeinteCentimos1.SetActive(true);
            Cambio1 = Cambio1 - 0.20m;
        }
        if (Cambio1 >= 0.10m)
        {
            DiezCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.10m;
        }
        if (Cambio1 >= 0.05m)
        {
            CincoCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.05m;
        }
        if (Cambio1 >= 0.02m)
        {
            DosCentimos.SetActive(true);
            Cambio1 = Cambio1 - 0.02m;
        }
        if (Cambio1 >= 0.02m && DosCentimos.activeSelf == true)
        {
            DosCentimos1.SetActive(true);
            Cambio1 = Cambio1 - 0.02m;
        }
        if (Cambio1 >= 0.01m)
        {
            UnCentimo.SetActive(true);
            Cambio1 = Cambio1 - 0.01m;
        }
        if (Cambio == 0.00m && CambioRandom == 0.00m)
        {
            CeroEuros.SetActive(true);
        }


        if (CambioRandom == 0.20m)
        {
            VeinteCentimosExtra.SetActive(true);
        }
        if (CambioRandom == 0.50m)
        {
            CincuentaCentimosExtra.SetActive(true);
        }
        if (CambioRandom == 1.20m)
        {
            UnEuroExtra.SetActive(true);
            VeinteCentimosExtra.SetActive(true);
        }
        if (CambioRandom == 2.30m)
        {
            DosEurosExtra.SetActive(true);
            VeinteCentimosExtra.SetActive(true);
            DiezCentimosExtra.SetActive(true);
        }
    }

    void Update()
    {
        if (BotonEliminarDinero.activeSelf ==true)
        {
```

```
            Diferencia =
BotonEliminarDinero.GetComponent<BotonEliminarCambio>().Diferencia;
            Debug.Log("Diferencia" + Diferencia);
        }
        if (BotonCorrecto.activeSelf == true)
        {
            BotonCorrectoUsed =
BotonCorrecto.GetComponent<VRTK_InteractableObject>().Grab;
            if (BotonCorrectoUsed == 1)
            {
                BotonCorrecto.SetActive(false);
                BotonEliminarDinero.SetActive(false);
                Menu.SetActive(false);
                botoncorrectopulsado = GetComponent<AudioSource>();
                botoncorrectopulsado.clip = AudioBotonCorrecto;
                botoncorrectopulsado.Play();
                if (Diferencia == 0.00m)
                {
                    Acierto = 1;
                }
                if (Diferencia > 0.00m)
                {
                    ErrorDef = 1;
                }
                if (Diferencia < 0.00m)
                {
                    ErrorExc = 1;
                }
            }
        }

        Capa = Señal.GetComponent<Collider>();

        if (Capa.enabled == true)
        {
            Caja = Quitar.GetComponent<Collider>();
            Caja.enabled = true;
        }
        if (Capa.enabled == false)
        {
            Caja = Quitar.GetComponent<Collider>();
            Caja.enabled = false;
        }
    }
}
```

SobreEscribirDatosP3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;

public class SobreEscribirDatosP3 : MonoBehaviour {

       private GameObject GameControlDatos;
    private int ID;
    private int VP3;
    private int NVEEP3;
    private int NVDEP3;
    private int NVDP3;
    private int TP3;

    // Use this for initialization
    void Start () {

            GameControlDatos = GameObject.Find ("GameControl");
            ID= GameControlDatos.GetComponent<GameControl>().ID;
        VP3 = GameControlDatos.GetComponent<GameControl>().VP3;
        NVEEP3 = GameControlDatos.GetComponent<GameControl>().NVEEP3;
        NVDEP3 = GameControlDatos.GetComponent<GameControl>().NVDEP3;
        NVDP3 = GameControlDatos.GetComponent<GameControl>().NVDP3;
        TP3 = GameControlDatos.GetComponent<GameControl>().TP3;
    }

       public void EscenaPagar3(){

            string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
            IDbConnection dbconn;
            dbconn = (IDbConnection) new SqliteConnection(conn);
            dbconn.Open(); //Open connection to the database.

            IDbCommand dbcmd = dbconn.CreateCommand();
            string sqlQuery = "UPDATE Datos SET Acierto P3='" + VP3 + "',Error
Exceso de dinero eliminado P3='" + NVEEP3 +
           "',Error Defecto de dinero eliminado P3='" + NVDEP3 + "',Error Pagado
no válido por Defecto P3='" + NVDP3 +
           "',Tiempo P3='" + TP3 + "' WHERE ID=" + ID;

        dbcmd.CommandText = sqlQuery;
            IDataReader reader = dbcmd.ExecuteReader();


            reader.Close();
            reader = null;

            dbcmd.Dispose();
            dbcmd = null;
```

```
            dbconn.Close();
            dbconn = null;


        }
}
```

C6. “Database”

AdminMYSQL

```
using UnityEngine;
using MySql.Data.MySqlClient;
using System.Data;

public class AdminMYSQL : MonoBehaviour {
    public string servidorBaseDatos;
    public string nombreBaseDatos;
    public string usuarioBaseDatos;
    public string contraseñaBaseDatos;

    private string datosConexion;
    private MySqlConnection conexion;


       // Use this for initialization
       void Start () {
        datosConexion = "Server=" + servidorBaseDatos
                        + ";Database=" + nombreBaseDatos
                        + ";Uid=" + usuarioBaseDatos
                        + ";Pwd=" + contraseñaBaseDatos
                        + ";";
        ConectarConServidorBaseDatos();

    }

    private void ConectarConServidorBaseDatos()
    {
        conexion = new MySqlConnection(datosConexion);
        try
        {
            conexion.Open();
            Debug.Log("Conexion con BD correcta");
        }
        catch (MySqlException error)
        {
            Debug.LogError("Imposible conectar con las Base de Datos" + error);
        }
    }
        public MySqlDataReader Select(string _select)
        {
            MySqlCommand cmd = conexion.CreateCommand();
            cmd.CommandText = "SELECT * FROM " + _select;
            MySqlDataReader Resultado = cmd.ExecuteReader();
        return Resultado;
        }

        public MySqlDataReader Insert(string _insert)
        {
        MySqlCommand cmd = conexion.CreateCommand();
        cmd.CommandText = "INSERT INTO " + _insert;
        MySqlDataReader Resultado = cmd.ExecuteReader();
        return Resultado;
        }

}
```

Login

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using MySql.Data.MySqlClient;
using UnityEngine.SceneManagement;



public class Login : MonoBehaviour
{
    public InputField usuarioTxt;
    public InputField contraseñaTxt;

    public void Logear()
    {
        string _log = "`usuarios` WHERE `usuario` LIKE '" + usuarioTxt.text +
"'AND `pass` LIKE'" + contraseñaTxt.text + "'";

        AdminMYSQL _adminMYSQL =
GameObject.Find("AdministradorBaseDeDatos").GetComponent<AdminMYSQL>();
        MySqlDataReader Resultado = _adminMYSQL.Select(_log);

        if (Resultado.HasRows)
        {
            Debug.Log("Login Correcto");
            Resultado.Close();
            SceneManager.LoadScene("MENU 1");
        }
        else
        {
            Debug.Log("Usuario o contraseña incorrectos");
            Resultado.Close();

        }
    }


}
```

Resumen

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Resumen : MonoBehaviour {

    public GameObject DontDestroy;
    public GameObject DontDestroy2;
    public InputField usuarioTxt;


    private void Awake()
    {
        DontDestroyOnLoad(DontDestroy);
        DontDestroyOnLoad(DontDestroy2);
    }
}
```

db

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;
using VRTK;

public class db : MonoBehaviour {
       public GameObject GuardarDatos;
       public GameObject Input;

       private int ID ;
       private int OV3 ;
       private int ONV3 ;
       private int T3 ;
       private int OV5 ;
       private int ONV5 ;
       private int T5 ;
       private int C5 ;
       private int OV7 ;
       private int ONV7 ;
       private int T7 ;
       private int C7 ;
    private int TR1;
    private int TR2;
    private int TR3;
    private int TC1;
    private int TC2;
    private int TC3;
    private int VP1;
    private int NVEP1;
    private int NVDP1;
    private int TP1;
    private int VCCP2;
    private int VCIP2;
    private int NVCCCIP2;
    private int NVCICCP2;
    private int NVDP2;
```

```
    private int TP2;
    private int VP3;
    private int NVEEP3;
    private int NVDEP3;
    private int NVDP3;
    private int TP3;

    void Start ()
    {
            //sqlite_BaseDatos ();
       }


      public void sqlite_BaseDatos ()
      {
            string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
            IDbConnection dbconn;
            dbconn = (IDbConnection) new SqliteConnection(conn);
            dbconn.Open(); //Open connection to the database.

            IDbCommand dbcmd = dbconn.CreateCommand();
            string sqlQuery = "SELECT * FROM Datos WHERE ID=
"+Input.GetComponent<Text>().text;
            dbcmd.CommandText = sqlQuery;
            IDataReader reader = dbcmd.ExecuteReader();

            while (reader.Read())
            {
                  int ID = reader.GetInt32(0);
                  int OV3 = reader.GetInt32(1);
                  int ONV3 = reader.GetInt32(2);
                  int T3 = reader.GetInt32(3);
                  int OV5 = reader.GetInt32(4);
                  int ONV5 = reader.GetInt32(5);
                  int T5 = reader.GetInt32(6);
                  int C5 = reader.GetInt32(7);
                  int OV7 = reader.GetInt32(8);
                  int ONV7 = reader.GetInt32(9);
                  int T7 = reader.GetInt32(10);
                  int C7 = reader.GetInt32(11);
            int TR1 = reader.GetInt32(12);
            int TR2 = reader.GetInt32(13);
            int TR3 = reader.GetInt32(14);
            int TC1 = reader.GetInt32(15);
            int TC2 = reader.GetInt32(16);
            int TC3 = reader.GetInt32(17);
            int VP1 = reader.GetInt32(18);
            int NVEP1 = reader.GetInt32(19);
            int NVDP1 = reader.GetInt32(20);
            int TP1 = reader.GetInt32(21);
            int VCCP2 = reader.GetInt32(22);
            int VCIP2 = reader.GetInt32(23);
            int NVCCCIP2 = reader.GetInt32(24);
            int NVCICCP2 = reader.GetInt32(25);
            int NVDP2 = reader.GetInt32(26);
            int TP2 = reader.GetInt32(27);
            int VP3 = reader.GetInt32(28);
            int NVEEP3 = reader.GetInt32(29);
            int NVDEP3 = reader.GetInt32(30);
            int NVDP3 = reader.GetInt32(31);
            int TP3 = reader.GetInt32(32);
```

```
    Debug.Log( "ID= "+ID+"  Objetos válidos ="+OV3+" Objetos no válidos ="+
ONV3+"Tiempo ="+
                            T3+"Objetos válidos 5="+OV5+"+ Objetos no
válidos5="+ONV5+" Tiempo="+T5+"Consutas 5="+C5+
                            "Objetos válidos 7="+OV7+"Objetos no válidos
7="+ONV7+"Tiempo 7="+T7+"Consultas 7="+C7+"Tiempo R1="+TR1+
                "Tiempo R2=" + TR2 + "Tiempo R3=" + TR3 + "Tiempo C1=" + TC1 +
"Tiempo C2=" + TC2 + "Tiempo C3=" + TC3 +
                "Pagado válido P1=" + VP1 + "Pagado no válido por Exceso P1=" +
NVEP1 +
                "Pagado no válido por Defecto P1=" + NVDP1 + "Tiempo P1=" + TP1 +
"Acierto con el cambio correcto P2" + VCCP2 +
                "Acierto con el cambio incorrecto P2" + VCIP2 + "Error con el
cambio correcto P2" + NVCCCIP2 +
                "Error con el cambio incorrecto P2" + NVCICCP2 + "Error Pagado no
válido por Defecto P2" + NVDP2 +
                "Tiempo P2=" + TP2 + "Acierto P3" + VP3 + "Error Exceso de dinero
eliminado P3" + NVEEP3 +
                "Error Defecto de dinero eliminado P3" + NVDEP3 + "Error Pagado
no válido por Defecto P3" + NVDP3 +
                "Tiempo P3=" + TP3);

                    GuardarDatos.GetComponent<GameControl> ().ID = ID;
                    GuardarDatos.GetComponent<GameControl> ().OV3 = OV3;
                    GuardarDatos.GetComponent<GameControl> ().ONV3 = ONV3;
                    GuardarDatos.GetComponent<GameControl> ().T3 = T3;
                    GuardarDatos.GetComponent<GameControl> ().OV5 = OV5;
                    GuardarDatos.GetComponent<GameControl> ().ONV5 = ONV5;
                    GuardarDatos.GetComponent<GameControl> ().T5 = T5;
                    GuardarDatos.GetComponent<GameControl> ().C5 = C5;
                    GuardarDatos.GetComponent<GameControl> ().OV7 = OV7;
                    GuardarDatos.GetComponent<GameControl> ().ONV7 = ONV7;
                    GuardarDatos.GetComponent<GameControl> ().T7 = T7;
                    GuardarDatos.GetComponent<GameControl> ().C7 = C7;
            GuardarDatos.GetComponent<GameControl>().TR1 = TR1;
            GuardarDatos.GetComponent<GameControl>().TR2 = TR2;
            GuardarDatos.GetComponent<GameControl>().TR3 = TR3;
            GuardarDatos.GetComponent<GameControl>().TC1 = TC1;
            GuardarDatos.GetComponent<GameControl>().TC2 = TC2;
            GuardarDatos.GetComponent<GameControl>().TC3 = TC3;
            GuardarDatos.GetComponent<GameControl>().VP1 = VP1;
            GuardarDatos.GetComponent<GameControl>().NVEP1 = NVEP1;
            GuardarDatos.GetComponent<GameControl>().NVDP1 = NVDP1;
            GuardarDatos.GetComponent<GameControl>().TP1 = TP1;
            GuardarDatos.GetComponent<GameControl>().VCCP2 = VCCP2;
            GuardarDatos.GetComponent<GameControl>().VCIP2 = VCIP2;
            GuardarDatos.GetComponent<GameControl>().NVCCCIP2 = NVCCCIP2;
            GuardarDatos.GetComponent<GameControl>().NVCICCP2 = NVCICCP2;
            GuardarDatos.GetComponent<GameControl>().NVDP2 = NVDP2;
            GuardarDatos.GetComponent<GameControl>().TP2 = TP2;
            GuardarDatos.GetComponent<GameControl>().VP3 = VP3;
            GuardarDatos.GetComponent<GameControl>().NVEEP3 = NVEEP3;
            GuardarDatos.GetComponent<GameControl>().NVDEP3 = NVDEP3;
            GuardarDatos.GetComponent<GameControl>().NVDP3 = NVDP3;
            GuardarDatos.GetComponent<GameControl>().TP3 = TP3;
        }
             reader.Close();
             reader = null;

             dbcmd.Dispose();
             dbcmd = null;
```

```
            dbconn.Close();
            dbconn = null;


        }
}
```

C7. “Others”

Accion

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class accion : MonoBehaviour
{
    AudioSource audiosourcerepetir1;
    public AudioClip audiorepetir1;
    AudioSource audiosourcerepetir2;
    public AudioClip audiorepetir2;
    AudioSource audiosourcerepetir3;
    public AudioClip audiorepetir3;
    AudioSource audiosourcerepetir4;
    public AudioClip audiorepetir4;
    AudioSource audiosourcerepetir5;
    public AudioClip audiorepetir5;
    AudioSource audiosourcerepetir6;
    public AudioClip audiorepetir6;
    AudioSource audiosourcerepetir7;
    public AudioClip audiorepetir7;
    AudioSource audiosourcerepetir8;
    public AudioClip audiorepetir8;
    private SteamVR_TrackedController controller;
    public GameObject obj1;
    public GameObject obj2;
    public GameObject obj3;
    public GameObject marca1;
    public GameObject marca2;
    public GameObject marca3;
    public GameObject marca4;
    public GameObject marca5;
    public GameObject marca6;
    AudioSource activar;
    RaycastTuto codigo1;
    CloseInitialDoors codigo2;
    CloseFinalDoors codigo3;
    audiobj2 codigo4;
    audiobj3 codigo5;
    AudioSource variable;
    private int cont0 = 0;
    private int cont1 = 0;
    private int cont2 = 0;
    private int cont3 = 0;
    private int cont4 = 0;
    private int cont5 = 0;
    private int cont6 = 0;
    private int cont7 = 0;
    // Use this for initialization
    void Start()
    {
        StartCoroutine(voz());
        StartCoroutine(flag1());
        StartCoroutine(flag2());
        StartCoroutine(flag3());
        StartCoroutine(flag4());
        StartCoroutine(flag5());
    }
```

```
private IEnumerator flag5()
{
    yield return new WaitForSeconds(1.0f);
    codigo5 = marca6.GetComponent<audiobj3>();
}

private IEnumerator flag4()
{
    yield return new WaitForSeconds(1.0f);
    codigo4 = marca5.GetComponent<audiobj2>();
}

private IEnumerator flag3()
{
    yield return new WaitForSeconds(1.0f);
    codigo3 = marca3.GetComponent<CloseFinalDoors>();
}

private IEnumerator flag2()
{
    yield return new WaitForSeconds(1.0f);
    codigo2 = marca2.GetComponent<CloseInitialDoors>();
}

private IEnumerator flag1()
{
    yield return new WaitForSeconds(1.0f);
    codigo1 = marca1.GetComponent<RaycastTuto>();
}

private IEnumerator voz()
{
    yield return new WaitForSeconds(33.0f);
    controller = GetComponentInParent<SteamVR_TrackedController>();
    controller.Gripped += repeticion;
}

private void repeticion(object sender, ClickedEventArgs e)
{
    variable.Play();
}

void Update()
{
    if (obj1.GetComponent<Renderer>().material.color == Color.green)
    {

        if (obj2.GetComponent<Renderer>().material.color == Color.green)
        {

            if (obj3.GetComponent<Renderer>().material.color == Color.green)
            {

                if (codigo2.X == 1)
                {

                    if (codigo3.Y == 1)
                    {

                        if(codigo4.A == 1)
                        {
                            if (codigo5.B == 1)
```

```
                    {
                        if (cont7 == 0)
                        {
                            audiosourcerepetir7.clip = null;
                            audiosourcerepetir7.volume = 0;
                            StartCoroutine(objeto3());
                        }
                    }

                    else
                    {
                        if (cont6 == 0)
                        {
                            audiosourcerepetir6.clip = null;
                            audiosourcerepetir6.volume = 0;
                            StartCoroutine(objeto2());
                        }

                    }
                }

                else
                {
                    if (cont5 == 0)
                    {
                        audiosourcerepetir5.clip = null;
                        audiosourcerepetir5.volume = 0;
                        StartCoroutine(objeto1());
                    }

                }
            }

            else
            {
                if (cont4 == 0)
                {
                    audiosourcerepetir4.clip = null;
                    audiosourcerepetir4.volume = 0;
                    StartCoroutine(pasillo());
                }

            }
        }

        else
        {
            if (cont3 == 0)
            {
                audiosourcerepetir3.clip = null;
                audiosourcerepetir3.volume = 0;
                StartCoroutine(esperargiro());
            }

        }
    }

    else
    {
        if (cont2 == 0)
        {
            audiosourcerepetir2.clip = null;
```

```
                    audiosourcerepetir2.volume = 0;
                    StartCoroutine(esperarderech());
                }

            }
        }

        else
        {
            if (cont1 == 0)
            {
                activar = marca4.GetComponent<AudioSource>();
                activar.clip = null;
                StartCoroutine(esperarizq());
            }

        }
    }

    else
    {
        if (cont0 == 0)
        {
            audiosourcerepetir1 = GetComponent<AudioSource>();
            audiosourcerepetir1.clip = audiorepetir1;
            variable = audiosourcerepetir1;
            cont0 = 1;
        }

    }

}

private IEnumerator objeto3()
{
    yield return new WaitForSeconds(codigo5.audioobj3.length);
    audiosourcerepetir8 = GetComponent<AudioSource>();
    audiosourcerepetir8.volume = 1;
    audiosourcerepetir8.clip = audiorepetir8;
    variable = audiosourcerepetir8;
    cont7 = 1;
}

private IEnumerator objeto2()
{
    yield return new WaitForSeconds(codigo4.audioobj2.length);
    audiosourcerepetir7 = GetComponent<AudioSource>();
    audiosourcerepetir7.volume = 1;
    audiosourcerepetir7.clip = audiorepetir7;
    variable = audiosourcerepetir7;
    cont6 = 1;
}

private IEnumerator objeto1()
{
    yield return new WaitForSeconds(codigo3.explicacion.length);
    audiosourcerepetir6 = GetComponent<AudioSource>();
    audiosourcerepetir6.volume = 1;
    audiosourcerepetir6.clip = audiorepetir6;
    variable = audiosourcerepetir6;
    cont5 = 1;
}
```

```
    private IEnumerator pasillo()
    {
        yield return new WaitForSeconds(codigo2.direccion.length);
        audiosourcerepetir5 = GetComponent<AudioSource>();
        audiosourcerepetir5.volume = 1;
        audiosourcerepetir5.clip = audiorepetir5;
        variable = audiosourcerepetir5;
        cont4 = 1;
    }

    private IEnumerator esperarizq()
    {
        yield return new WaitForSeconds(codigo1.audioizquierda.length);
        audiosourcerepetir2 = GetComponent<AudioSource>();
        audiosourcerepetir2.volume = 1;
        audiosourcerepetir2.clip = audiorepetir2;
        variable = audiosourcerepetir2;
        cont1 = 1;
    }

    private IEnumerator esperarderech()
    {
        yield return new WaitForSeconds(codigo1.audioderecha.length);
        audiosourcerepetir3 = GetComponent<AudioSource>();
        audiosourcerepetir3.volume = 1;
        audiosourcerepetir3.clip = audiorepetir3;
        variable = audiosourcerepetir3;
        cont2 = 1;
    }

    private IEnumerator esperargiro()
    {
        yield return new WaitForSeconds(codigo1.audiogiro.length);
        audiosourcerepetir4 = GetComponent<AudioSource>();
        audiosourcerepetir4.volume = 1;
        audiosourcerepetir4.clip = audiorepetir4;
        variable = audiosourcerepetir4;
        cont3 = 1;
    }
}
```

Activar

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class activar : MonoBehaviour {

    private SteamVR_TrackedController controller;
    public GameObject cubo;
    private Collider espacio;

    void Start()
    {
        controller = GetComponentInParent<SteamVR_TrackedController>();
    }

    void Update()
    {
        if (controller.triggerPressed == true)
        {
            espacio = cubo.GetComponent<Collider>();
            espacio.enabled = true;
        }
        else
        {
            espacio = cubo.GetComponent<Collider>();
            espacio.enabled = false;
        }
    }
}
```

Audiobj2

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class audiobj2 : MonoBehaviour {
    AudioSource audiosourceobj2;
    public AudioClip audioobj2;
    public int A = 0;
    // Use this for initialization
    void Start () {
        audiosourceobj2 = GetComponent<AudioSource>();
        audiosourceobj2.clip = audioobj2;
        audiosourceobj2.Play();
        StartCoroutine(finalobj2());
       }

    private IEnumerator finalobj2()
    {
        yield return new WaitForSeconds(audioobj2.length);
        A = 1;
    }
}
```

Audiobj3

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class audiobj3 : MonoBehaviour {
    AudioSource audiosourceobj3;
    public AudioClip audioobj3;
    public int B = 0;
    // Use this for initialization
    void Start()
    {
        audiosourceobj3 = GetComponent<AudioSource>();
        audiosourceobj3.clip = audioobj3;
        audiosourceobj3.Play();
        StartCoroutine(finalobj3());
    }

    private IEnumerator finalobj3()
    {
        yield return new WaitForSeconds(audioobj3.length);
        B = 1;
    }
}
```

ContadorListaConsulta

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class audiobj3 : MonoBehaviour
{
    AudioSource audiosourceobj3;
    public AudioClip audioobj3;
    public int B = 0;
    // Use this for initialization
    void Start()
    {
        audiosourceobj3 = GetComponent<AudioSource>();
        audiosourceobj3.clip = audioobj3;
        audiosourceobj3.Play();
        StartCoroutine(finalobj3());
    }

    private IEnumerator finalobj3()
    {
        yield return new WaitForSeconds(audioobj3.length);
        B = 1;
    }
}
```

DelVideoAlNivel

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class DelVideoAlNivel : MonoBehaviour {
    public int VideoTime;
    //public string SceneName;

       // Use this for initialization
       void Start ()
    {
            StartCoroutine(TiempoVideo());
        }

        IEnumerator TiempoVideo()
    {
            yield return new WaitForSeconds(VideoTime);
        SceneManager.LoadScene("Main scene");
    }

    }
```

DesactivarNota

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DesactivarNota : MonoBehaviour {


       // Use this for initialization
       void Start () {
              StartCoroutine (DesactivarNotas ());
       }

       IEnumerator DesactivarNotas()
       {
              yield return new WaitForSeconds (15);
              gameObject.SetActive (false);
       }
}
```

EmailPassword

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Firebase.Auth;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
using System;

public class EmailPassword : MonoBehaviour
{

    private FirebaseAuth auth;
    public InputField UserNameInput, PasswordInput;
    public Button SignupButton, LoginButton;
    public Text ErrorText;

    void Start()
    {
        auth = FirebaseAuth.DefaultInstance;
        //Just an example to save typing in the login form
        UserNameInput.text = "demofirebase@gmail.com";
        PasswordInput.text = "abcdefgh";

        SignupButton.onClick.AddListener(() => Signup(UserNameInput.text,
PasswordInput.text));
        LoginButton.onClick.AddListener(() => Login(UserNameInput.text,
PasswordInput.text));
    }


    public void Signup(string email, string password)
    {
        if (string.IsNullOrEmpty(email) || string.IsNullOrEmpty(password))
        {
            //Error handling
            return;
        }

        auth.CreateUserWithEmailAndPasswordAsync(email,
password).ContinueWith(task =>
        {
            if (task.IsCanceled)
            {
                Debug.LogError("CreateUserWithEmailAndPasswordAsync was
canceled.");
                return;
            }
            if (task.IsFaulted)
            {
                Debug.LogError("CreateUserWithEmailAndPasswordAsync error: " +
task.Exception);
                if (task.Exception.InnerExceptions.Count > 0)

UpdateErrorMessage(task.Exception.InnerExceptions[0].Message);
                return;
            }

            FirebaseUser newUser = task.Result; // Firebase user has been
created.
            Debug.LogFormat("Firebase user created successfully: {0} ({1})",
                newUser.DisplayName, newUser.UserId);
```

```
            UpdateErrorMessage("Signup Success");
        });
    }

    private void UpdateErrorMessage(string message)
    {
        ErrorText.text = message;
        Invoke("ClearErrorMessage", 3);
    }

    void ClearErrorMessage()
    {
        ErrorText.text = "";
    }
    public void Login(string email, string password)
    {
        auth.SignInWithEmailAndPasswordAsync(email, password).ContinueWith(task
=>
        {
            if (task.IsCanceled)
            {
                Debug.LogError("SignInWithEmailAndPasswordAsync canceled.");
                return;
            }
            if (task.IsFaulted)
            {
                Debug.LogError("SignInWithEmailAndPasswordAsync error: " +
task.Exception);
                if (task.Exception.InnerExceptions.Count > 0)

UpdateErrorMessage(task.Exception.InnerExceptions[0].Message);
                return;
            }

            FirebaseUser user = task.Result;
            Debug.LogFormat("User signed in successfully: {0} ({1})",
                user.DisplayName, user.UserId);

            PlayerPrefs.SetString("LoginUser", user != null ? user.Email :
"Unknown");
            SceneManager.LoadScene("LoginResults");
        });
    }
}
```

GameControl

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using Mono.Data.Sqlite;
using System.Data;
using System;
using UnityEngine.UI;
using VRTK;

public class GameControl : MonoBehaviour {

	public static GameControl control;
	//public GameObject BaseDatos;
	private GameObject dentrocesta;
    private GameObject dinerocaja;
    private GameObject LeftController;
	private GameObject tiempo;
	public int ID;
	public int OV3 ;
	public int ONV3 ;
	public int T3 ;
	public int OV5 ;
	public int ONV5 ;
	public int T5 ;
	public int C5 ;
	public int OV7 ;
	public int ONV7 ;
	public int T7 ;
	public int C7 ;
    public int TR1;
    public int TR2;
    public int TR3;
    public int TC1;
    public int TC2;
    public int TC3;
    public int VP1;
    public int NVEP1;
    public int NVDP1;
    public int TP1;
    public int VCCP2;
    public int VCIP2;
    public int NVCCCIP2;
    public int NVCICCP2;
    public int NVDP2;
    public int TP2;
    public int VP3;
    public int NVEEP3;
    public int NVDEP3;
    public int NVDP3;
    public int TP3;
    void Awake () {
		if (control == null) {
			DontDestroyOnLoad (gameObject);
			control = this;
		}
		else if(control !=this)
		{
			Destroy(gameObject);
		}
```

```
        }



        public void GuardarDatos (){

                //ID = BaseDatos.GetComponent<db>().ID;
                //OV3 = BaseDatos.GetComponent<db> ().OV3;
                //ONV3 = BaseDatos.GetComponent<db> ().ONV3;
                //T3 = BaseDatos.GetComponent<db> ().T3;
                Debug.Log ("ID="+ ID + "OV3="+ OV3 + "ONV3="+ ONV3 + "T3="+T3);

        }

    public void Escena1()
    {

        string conn = "URI=file:" + Application.dataPath +
"/plugins/BaseDatos.db"; //Path to database.
        IDbConnection dbconn;
        dbconn = (IDbConnection)new SqliteConnection(conn);
        dbconn.Open(); //Open connection to the database.

        IDbCommand dbcmd = dbconn.CreateCommand();
        string sqlQuery = "UPDATE Datos SET ObjetosVal3='" + OV3 + "' WHERE ID="
+ ID;
        dbcmd.CommandText = sqlQuery;
        IDataReader reader = dbcmd.ExecuteReader();


        reader.Close();
        reader = null;

        dbcmd.Dispose();
        dbcmd = null;

        dbconn.Close();
        dbconn = null;

        //dentrocesta = GameObject.Find ("DentroCesta");
        //OV3 = dentrocesta.GetComponent<StopGravity> ().Valid;
        //ONV3 = dentrocesta.GetComponent<StopGravity> ().NoVal;
        //T3 = dentrocesta.GetComponent<StopGravity> ().timeint;
        //Debug.Log ("ID="+ ID + "OV3="+ OV3 + "ONV3="+ ONV3 + "T3="+T3);
    }

    void Update()
    {


        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ListaNivel1"))
        {



            dentrocesta = GameObject.Find("DentroCesta");
            OV3 = dentrocesta.GetComponent<StopGravity>().Valid;
            ONV3 = dentrocesta.GetComponent<StopGravity>().NoVal;
            T3 = dentrocesta.GetComponent<StopGravity>().timeint;
```

```
        }
        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ListaNivel2"))
        {


            tiempo = GameObject.Find("Script");
            T5 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
            C5 =
LeftController.GetComponent<ContadorListaConsulta>().ContadorConsulta;
            dentrocesta = GameObject.Find("DentroCesta");
            OV5 = dentrocesta.GetComponent<StopGravity5>().Valid;
            ONV5 = dentrocesta.GetComponent<StopGravity5>().NoVal;



        }
        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ListaNivel3"))
        {

            tiempo = GameObject.Find("Script");
            T7 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
            C7 =
LeftController.GetComponent<ContadorListaConsulta>().ContadorConsulta;
            dentrocesta = GameObject.Find("DentroCesta");
            OV7 = dentrocesta.GetComponent<StopGravity7>().Valid;
            ONV7 = dentrocesta.GetComponent<StopGravity7>().NoVal;

        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ReponedorNivel1"))
        {

            tiempo = GameObject.Find("Script");
            TR1 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ReponedorNivel2"))
        {

            tiempo = GameObject.Find("Script");
            TR2 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ReponedorNivel3"))
        {

            tiempo = GameObject.Find("Script");
            TR3 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ComunidadesNivel1"))
```

```
        {

            tiempo = GameObject.Find("Script");
            TC1 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ComunidadesNivel2"))
        {

            tiempo = GameObject.Find("Script");
            TC2 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("ComunidadesNivel3"))
        {

            tiempo = GameObject.Find("Script");
            TC3 = tiempo.GetComponent<TimerRecogida>().timeint;
            LeftController = GameObject.Find("LeftController");
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("PagarNivel1"))
        {
            dinerocaja = GameObject.Find("DentroCaja");
            VP1 = dinerocaja.GetComponent<PagarCajaN1>().Valid;
            NVEP1 = dinerocaja.GetComponent<PagarCajaN1>().NoValExc;
            NVDP1 = dinerocaja.GetComponent<PagarCajaN1>().NoValDef;
            tiempo = GameObject.Find("TiempoPagar");
            TP1 = tiempo.GetComponent<TimerRecogida>().timeint;
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("PagarNivel2"))
        {
            dinerocaja = GameObject.Find("DentroCaja");
            VCCP2 = dinerocaja.GetComponent<PagarCajaN2>().AciertoCC;
            VCIP2 = dinerocaja.GetComponent<PagarCajaN2>().AciertoCI;
            NVCCCIP2 = dinerocaja.GetComponent<PagarCajaN2>().ErrorCCCI;
            NVCICCP2 = dinerocaja.GetComponent<PagarCajaN2>().ErrorCICC;
            NVDP2 = dinerocaja.GetComponent<PagarCajaN2>().NoValDef;
            tiempo = GameObject.Find("TiempoPagar");
            TP2 = tiempo.GetComponent<TimerRecogida>().timeint;
        }

        if (SceneManager.GetActiveScene() ==
            SceneManager.GetSceneByName("PagarNivel3"))
        {
            dinerocaja = GameObject.Find("DentroCaja");
            VP3 = dinerocaja.GetComponent<PagarCajaN3>().Acierto;
            NVEEP3 = dinerocaja.GetComponent<PagarCajaN3>().ErrorExc;
            NVDEP3 = dinerocaja.GetComponent<PagarCajaN3>().ErrorDef;
            NVDP3 = dinerocaja.GetComponent<PagarCajaN3>().NoValDef;
            tiempo = GameObject.Find("TiempoPagar");
            TP3 = tiempo.GetComponent<TimerRecogida>().timeint;
        }
    }
}
```

GlobalControl

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GlobalControl : MonoBehaviour {


	public float time;
	public  static GlobalControl Instance;

	void Awake ()
	{
		if (Instance == null) {
			DontDestroyOnLoad (gameObject);
			Instance = this;
		} else if (Instance != this) {
			Destroy (gameObject);
		}
	}
}
```

InicioSaludo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class InicioSaludo : MonoBehaviour {

	public GameObject Mujer;
	AudioSource AudioSourceMujer;
	public int contador = 0;
	//public AudioClip AudioSaludo;
	// Use this for initialization
	void Start () {
		AudioSourceMujer= Mujer.GetComponent<AudioSource>();
	//	AudioSourceMujer.clip = AudioSaludo;
	}

	// Update is called once per frame
	void OnTriggerEnter (Collider other) {

		contador = contador + 1;

		if (contador == 1) {
			AudioSourceMujer.Play ();
			Debug.Log ("hola man");
		}
	}
}
```

Keepposition

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Keepposition : MonoBehaviour {

	private float distancia = 0.1f;


	// Update is called once per frame
	void Update () {
		if (transform.position.y <  distancia) {
			transform.position = new Vector3(transform.position.x,0,
transform.position.z);
		}
	}
}
```

Keepposition2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Keepposition2 : MonoBehaviour {

	private float distancia = 0.1f;


	// Update is called once per frame
	void Update () {
		if (transform.position.y >  distancia) {
			transform.position = new Vector3(transform.position.x,0,
transform.position.z);
		}
	}
}
```

LoadSceneOnClick

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class LoadSceneOnClick : MonoBehaviour {

    public void LoadByIndex(int sceneIndex)
    {
        SceneManager.LoadScene(sceneIndex);
    }
}
```

PreviousScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PreviousScene : MonoBehaviour {

    private static string lastLevel;

    public static void setLastLevel(string level)
    {
        lastLevel = level;
    }

    public static string getLastLevel()
    {
        return lastLevel;
    }

    public static void changeToPreviousLvl()
    {
        Application.LoadLevel(lastLevel);
    }
}
```

QuitarCesta

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class quitarcesta : MonoBehaviour {
    public GameObject cesta;
    private Collider cestita;
       // Use this for initialization
       void Start ()
    {
        cestita = cesta.GetComponent<Collider>();
        cestita.enabled = false;
       }
}
```

QuitScene

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class QuitScene : MonoBehaviour {

    public void Quit()
    {
#if UNITY_EDITOR
        UnityEditor.EditorApplication.isPlaying = false;
#else
        Application.Quit ();
#endif
    }
}
```

ReproduceSound

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class ReproduceSound : MonoBehaviour {

	AudioSource audioSource;
	public AudioClip sonido;
	// Use this for initialization
	void Start () {

		audioSource = GetComponent<AudioSource> ();
		GetComponent<AudioSource> ().clip = sonido;
		audioSource.Play();
	}


}
```

SoundsRepeat

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class soundsrepeat : MonoBehaviour
{
    private SteamVR_TrackedController controller;
    private AudioSource audiosource;


       // Use this for initialization
       //void Start ()
    //{
        //audiosource = GetComponent<AudioSource>();

        //controller = GetComponentInParent<SteamVR_TrackedController>();
        //controller.MenuButtonClicked = audiosource;
    //}


    //private void audiosource(object sender);

    //{
        //audiosource.Play();
    //}

    // Update is called once per frame
    void Update () {

       }
}
```

StartTalking

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StartTalking : MonoBehaviour
{

    public GameObject Mujer;
    public GameObject LeftController;
    public GameObject RightController;
    AudioSource AudioSourceMujer;
    public AudioClip audiodistrac;
    int contador = 0;
    // Use this for initialization
    void Start()
    {
        AudioSourceMujer = Mujer.GetComponent<AudioSource>();
    }

    void OnTriggerEnter(Collider collider)
    {

        LeftController.SetActive(false);
        RightController.SetActive(false);
        StartCoroutine(TiempoEsperar());
        contador = contador + 1;

        if (contador == 1)
        {
            AudioSourceMujer.Play();
        }
    }
    IEnumerator TiempoEsperar()
    {
        yield return new WaitForSeconds(audiodistrac.length);
        LeftController.SetActive(true);
        RightController.SetActive(true);
        Destroy(gameObject);
    }
}
```

StartWalking

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class StartWalking : MonoBehaviour {

       public GameObject Mujer;
       //public GameObject LeftController;
       public GameObject RightController;
       public int TiempoEspera;//Normalmente el tiempo será de 10 seg pero por si
se varía la animación
       Animator animation1;
       AudioSource AudioSourceMujer;
       public int TiempoSaludar;
    public int tiempomoverse;
    private int contador;
       private int contadorsaludo;
    public GameObject Pared;
       // Use this for initialization
       void Start () {

              animation1 = Mujer.GetComponent<Animator> ();
              AudioSourceMujer= Mujer.GetComponent<AudioSource>();
       }

       // Update is called once per frame
       void OnTriggerEnter (Collider collider) {
        contador = contador + 1;
        if (contador == 1)

        {
            animation1.SetBool("walking", false);
            //animation1.enabled = true;
            //LeftController.SetActive (false);
            RightController.SetActive(false);
            StartCoroutine(TiempoEsperar());
            StartCoroutine(TiempoSaludo());
            StartCoroutine(Tiempomovers());
            Destroy(Pared);
            Debug.Log("Yaxx");
        }
       }
    IEnumerator Tiempomovers()
    {
        yield return new WaitForSeconds(tiempomoverse);

        RightController.SetActive(true);

    }
    IEnumerator TiempoSaludo()
    {
        yield return new WaitForSeconds(TiempoSaludar);
        contadorsaludo = contadorsaludo + 1;
        if (contadorsaludo == 1)
        {
            AudioSourceMujer.Play();
        }
    }
    IEnumerator TiempoEsperar()
       {
              yield return new WaitForSeconds(TiempoEspera);
```

```
        //      LeftController.SetActive (true);
                Mujer.SetActive (false);
         Destroy(gameObject);
        }



}
```

TimeCounter

```
using UnityEngine.UI;
using System.Collections;
using UnityEngine;
using System.Collections.Generic;

public class TimeCounter : MonoBehaviour {
        public Text Timer;
        public float time= 0.0f;
        // Use this for initialization

        void Start () {

        }

        // Update is called once per frame
        void Update () {
                time += Time.deltaTime;

                var minutes = (int)time / 60;
                var seconds = time % 60;
                var fraction = (time * 100) % 100;

                Timer.text = string.Format("{0:00} : {1:00} ", minutes, seconds);

                //Timer.text = string.Format("{0:00} : {1:00} : {2:000}", minutes,
seconds, fraction);
        }
}
```

TimerRecogida

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class TimerRecogida : MonoBehaviour {



                public GameObject time;
                public int timeint;


        // Update is called once per frame
        void Update () {

                timeint = (int)time.GetComponent<TimeCounter> ().time;

        }
}
```

WhenCollision2

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;


public class WhenCollision2 : MonoBehaviour {

	AudioSource audioSource;
	public AudioClip impact;
	// Use this for initialization
	void Start () {

		audioSource = GetComponent<AudioSource> ();
		GetComponent<AudioSource> ().clip = impact;

	}

	// Update is called once per frame
	void Update () {

	}

	IEnumerator OnTriggerEnter (Collider collider)
	{
		audioSource.Play ();
		yield return new WaitForSeconds (impact.length);

		gameObject.SetActive (false);
	}
}
```