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

BACHELOR THESIS

Bachelor's Degree in Industrial Electronics and Automatic Control Engineering

DESIGN OF AN APPLICATION WITH VIRTUAL REALITY FOR NEUROREHABILITATION PURPOSES

Project Report and Annexes

Author: Jordi Ginés i Ametllé
Director: Jordi Torner Ribé
Co-Director: Francesc Alpiste Penalba

Examination: October 2017

Resum

La finalitat d'aquest projecte és realitzar un estudi de la realitat virtual i de la seva aplicació en entorns de recuperació hospitalària com neurorehabilitació o fisioteràpia, ja que és una tecnologia que s'està començant a utilitzar de forma regular en aquests moments. També es crearà una aplicació 3D en realitat virtual de demostració bàsica d'un joc que sigui útil per a realitzar aquests entrenaments. Per aconseguir-ho, treballarem amb Unity, un motor gràfic utilitzat per a crear aplicacions i videojocs en 3D. El hardware per a desenvolupar el sistema d'immersió en realitat virtual serà el sistema HTC Vive, que addicionalment permet la detecció de posició i moviments a escala d'una habitació mitjançant la tecnologia de senyals de llum Lighthouse.

Aquesta aplicació es basarà en la creació d'un videojoc immersiu que pugui captar els moviments de les extremitats superiors de l'usuari i aquest pugui realitzar la seva recuperació de manera més amena i motivadora, en forma d'esquivar obstacles en el videojoc utilitzant aquests moviments. Addicionalment, es realitzarà el suport informàtic necessari per a poder utilitzar l'aplicació en unes ulleres de realitat virtual per a millorar la immersió visual. També s'obtindrà algun paràmetre que indiqui com ha sigut el transcurs de la partida.

Resumen

La finalidad de este proyecto es realizar un estudio de la realidad virtual y de su aplicación en entornos de recuperación hospitalaria como neurorehabilitación o fisioterapia, ya que es una tecnología que se está empezando a utilizar de forma regular en estos momentos. También se creará una aplicación 3D en realidad virtual de demostración básica de un juego que sea útil para realizar estos entrenamientos. Para conseguirlo, trabajaremos con Unity, un motor gráfico para la creación de aplicaciones y videojuegos en 3D. El hardware para desarrollar el sistema de inmersión en realidad virtual será el sistema HTC Vive, que adicionalmente permite la detección de posición y movimientos a escala de una habitación mediante la tecnología de señales de luz Lighthouse.

Esta aplicación se basará en la creación de un videojuego inmersivo que pueda captar los movimientos de las extremidades superiores del usuario y éste pueda realizar su recuperación de manera más amena y motivadora, en forma de esquivar obstáculos en el videojuego utilizando dichos movimientos. Adicionalmente, se realizará el soporte informático necesario para poder utilizar la aplicación en unas gafas de realidad virtual para mejorar la inmersión visual. También se va a obtener algún parámetro que indique como ha transcurrido la partida.

Abstract

The purpose of this project is to carry out a study of virtual reality and its applications in hospital environments such as neurorehabilitation or physiotherapy, since it is a technology which is starting to be used nowadays. A 3D virtual reality application for a basic demonstration of a game which could be useful for these trainings will be also done. To achieve this, we will work with Unity, a graphic motor for 3D application and game creation. The hardware for the development of the virtual reality immersive system will be the HTC Vive system, which additionally allows position and movement detection in a room scale using the light signalling technology Lighthouse.

This application will focus on the creation of an immersive videogame to track upper extremity movement from the user so he can do his rehabilitation training in a more enjoyable and motivating way, in the form of avoiding obstacles in the videogame using those movements. In addition, the necessary informatic support will be done so the application can be used in a virtual reality headset to improve visual immersion. It will also be obtained some parameter indicating how the game play was.

Acknowledgements

The achievement of this thesis would not have been possible without the generous support of the following people:

- All teachers I had during my university and earlier school stages.
- My director and co-director of this thesis, Jordi Torner and Francesc Alpiste.
- All my family for always supporting me

I am very much thankful to all of them for letting my knowledge go further, helping me determine my dedications in life and developing me as a better person.

Index

1. Introduction

1.1. Origins and motivations

Modern virtual reality is a recent technology applied to the videogames, but since it has still some problems to function completely and smooth for users, there will be some years until it completely develops. However, what is available now is still an immersive and surprising experience for all users who try it, and it generates new applications and approaches for many professionals every day.

Apart from personal entertainment devices, many other applications are starting to develop from it, in fields such as marketing, interactive spaces, communications, design, army operations or healthcare.

In the healthcare treatments, there are many patients who have had brain or coordination problems and they lack a great mobility range of motion in their upper extremities. Creating an application that motivates them to do their training will generate a positive feedback circle making their progresses better with time and improving the quality of their lives.

It is challenging and motivating to take part in this modern technology and in a field like healthcare, where it can also have a solid social function.

1.2. Objectives and Scope

The main objective of this project is the design and development of a demo game which can be helpful for patients in neurorehabilitation therapies. More concretely, the game will be aimed to patients who need any kind of motion training in their upper extremities, covering articulations such as the elbow and the shoulder.

To achieve this, first of all it will be needed to do a research about the virtual reality technologies available. After that, hardware and software to implement the application has to be accurately selected.

The demo game has to have a minimum set of characteristics to operate correctly. One is to have some kind of accurate movement detection using the hardware selected. Once the patient has done the move, it is also important that the extremity moved returns to the initial position to ensure a better treatment. This calibration and synchronization has to be considered during the game design.

More essential features are that the game has to have motivational elements, for example increasing score, advancing levels, beat enemies or capability to include various levels of difficulty. The patient has to feel some kind of advance and be motivated to do their training the better possible.

Finally, intuitive interfaces have to be designed to allow a correct interplay before and after the game is played, and to display useful data to the professional therapist that will accompany the patient while the exercises are being done and the game is played.

This project will not include future improvements needed or recommended once the project has been reviewed and used by professional therapists and patients over time. Present functions will need to be reviewed and new functions that the professionals find useful could be implemented. Even though, some recommendations will be included about aspects that may need improvement.

2. State of the Field

2.1. Neurorehabilitation

Neurorehabilitation is a medical speciality that focuses on training or recovering lost functions once brain or neurological damage has happened, and to achieve new functions in the case of children recovery.

After brain injury or brain disease there are potential widespread biochemical and functional brain changes. This adapted brain may react to environmental stimuli in a significantly altered manner. While an acutely injured brain may be neurochemically unstable and place the person at risk for seizures, irritability, and other reactions, in the early stage of recovery, the brain has increased potential for injury-induced plasticity.

Neurorehabilitation goal is maximizing the potential for recovery of function within these circumstances. In addition to the early neurochemical facilitation to brain recovery, the same fundamental neural and behavioural signals driving plasticity during learning in the intact brain are engaged during relearning in the damaged or diseased brain. Neuroplasticity research suggests that people may continue to recover functions for many years. Although researchers have also pointed out that the rate and extent of recovery occurs is not continuous: recovery is easier and faster earlier and becomes increasingly more difficult as time progresses. Some studies have shown that most people continue to make significant physical, cognitive, and behavioural recovery gains as many as five years or more post Traumatic Brain Injury. **[1]**

Neurorehabilitation implies treatment of diseases and injures **[2]** such as:

- Spinal Cord Injury
- Traumatic Brain Injury
- Cerebrovascular Injury or Stroke
- Cerebral Palsy
- Multiple Sclerosis

- Parkinson's Disease
- Amyotrophic Lateral Sclerosis
- Post-Polio Syndrome
- Dysmelia
- Guillain-Barre Syndrome

The main goal of the treatment of people with injuries to the central nervous system will be the improvement of autonomy in the activities of daily living. Therapies used may include psychological therapy, occupation therapy, teaching or re-training patients on mobility skills, communication processes, and other aspects of that person's daily routine. An important part of these therapies is physiotherapy.

2.2. Physiotherapy

The neurological physiotherapy puts at the disposal of the patient with brain damage, a series of knowledge aimed at improving, among other aspects, the quality and the effectiveness of the movements, the postural control, improve steps, stability, reduce spasticity, tremors, fatigue and the quest for autonomy, independence in daily life and thus improve the quality of life for both the patient and the family member or caregiver.

Figure 2.1. Physiotherapy in neurorehabilitation. Source **[3]**

Considering that the damage is in the nervous system, clinical practice must be based on current knowledge of neuroplasticity, motor control and motor learning.

To achieve better autonomy for patients, therapy focuses on recovering the functional movement through specific exercises of muscle tone and the sensitivity, improved balance, work of re-education, stimulation of the functional activity of the upper extremities, and modulation of muscular hypertonia. Once the patient has standardized the muscle tone, he can do stimulation with active movements, aided by the therapist. This is taken to achieve improvements in stand-up sit-down balance and basic functions of the oral and facial mobility, balance of foot, walking, movements of the arms, etc.

2.3. Available Applications

Some platforms to achieve a more interactive training or rehabilitation have been used over the past years. Virtual reality systems are recently being studied to be included, converging to new advances and more companies developing solutions. This allows the creation of complex and real life immersive systems. These platforms are developed in an almost personalized way by various companies, and although they can vary between countries, they have some common traits. The systems usually include sensors, cameras and allow the collection of medical data from professionals or therapist. Some recent technology examples will be described below.

2.3.1. NBETTER

NBETTER system developed by Neurostyle is aimed to help for stroke rehabilitation.

Figure 2.2. NBETTER Stroke Rehabilitation System. Source **[4]**

The technology consists in a Brain Computer Interface which has been proved that makes an improvement on motor function recovery. It allows a direct communication between the brain and an external device like a computer or a robot. This is achieved using an Electroencephalography (EEG) machine connected to the brain. The system also comes with a physical feedback mechanism and a visual feedback screen with interactive games.

The device is used for neurorehabilitation of stroke patients and synchronizes brain rehabilitation with physical rehabilitation. Brain Computer Interface makes the patient visualize or imagine moving the stroke affected upper limb extremity. **[4]** The computer software detects the changes on the brain and activates a robot for sensor motor feedback and also a movement in the game screen for visual feedback. This provides direct brain feedback and reinforces brain plasticity and by repeating the exercises, the patient will strength and create new neural connection to allow to do tasks that were not possible due to the stroke brain damage.

2.3.2. OmniVR

OmniVR Virtual Rehabilitation system is one of the first rehabilitation systems using virtual reality. It is a general system that offers a great amount of exercises and programs for the patient's individual needs.

The system is very easy to use for patients and professionals. It consists of a 3D camera and specialized computer software in order to precisely capture the movements done by the patient. No controllers or platforms are used.

Figure 2.3. OmniVR Virtual Rehabilitation. Source **[5]**

Once the system is started, the therapist can select the proper exercise program and parameters for each patient and treatment goal. It is important that a therapist is always in the rehabilitation room so the patient can receive the correct medical knowledge and feedback. It includes exercises aimed to stroke rehabilitation, general debility, aging people, sitting balance, walking exercises, wheelchair control, cognitive impairment treatments or upper extremities movements. It is a quite complete platform.

2.3.3. Timocco

Timocco is therapeutic gaming environment designed for children with special needs. The product is recommended for children with Attention Deficit Hyperactivity Disorder, Autism, Cerebral Palsy, Developmental Coordination Disorder or Learning Disabilities.

Timocco includes a range of development, educational and non-competitive games for the improvement of motor and cognitive capacities of the users. The activities are carried out with body movements that are detected by a tracking camera which follows the movements if the player is holding specific objects.

Figure 2.4. Timocco system. Source **[6]**

Each activity focuses on a different cognitive or motor skills, such as postural correction, bilateral coordination, eye-hand coordination, visual discrimination, etc. The activities are fully configurable, to adapt to the capabilities of each user.

Timocco offered game have been developed case by case by Occupational Therapists to provide optimal interaction and motivation for developing children abilities. Results from each game are recorded to allow an easy tracking of the progress.

2.4. Serious Game

Serious games are games whose main purpose is far from entertainment, because they look for using this entertainment created on the player in learning, training, education, or the creation of awareness on a topic.

These games take place in a three-dimensional virtual environment. They also allow a safe practice on situations that in real life could lead to danger in a virtual three-dimensional environment. The main advantage of serious games is that allow users to enjoy an interactive and educational process in which they can make mistakes without suffering the direct consequences on the reality. **[7]**

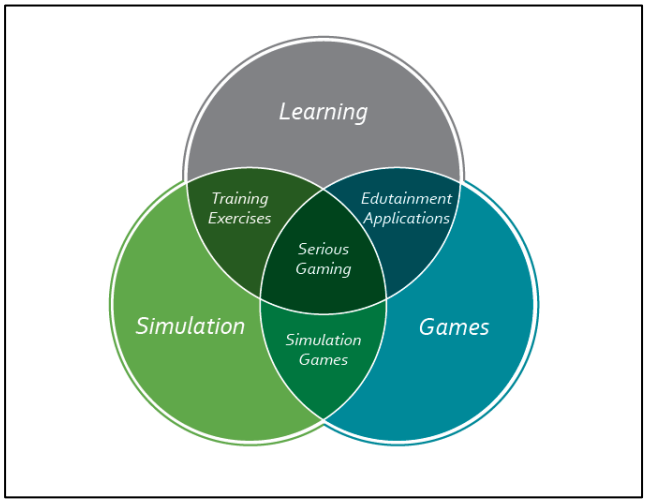

Figure 2.5. Serious – Gaming. Source **[7]**

A serious game should enforce the player specific rules, make successive tasks increasingly difficult, offer an attractive prize, limit the number of objectives the player should try to reach in the game, allow trial and error and give frequent feedback to the player.

These games can be used for education, healthcare, politics, art, communication, marketing, etc. In the health sector, in addition to entertain, they also allow to educate and promote better access for users to the world of health care.

These video games are also an opportunity to motivate and change people's behaviour with respect to their diseases and injuries. However, the application developed in this project has to be considered more as a videogame or interactive experience. Although it will incorporate some game mechanics to keep the users engaged, it focuses on a specific experience and immersive training and lacks some elements which are present in serious games, such as an effective and progressive learning on some subject.

2.5. Virtual Reality Role

Virtual Reality can be used in neurorehabilitation in occupational therapy. It can create immersive environments to help patients to train and do better tasks in a virtual world as they do not have their greatest level of motor capacity and functionality in the real world.

To help the processes of neuroplasticity (the establishment of new connections in the brain) it is essential the use of functional activities that have a sense for the person that are motivating and that will arouse their interest and increased attention on the task.

Another way of investigation is researching in therapies that have shown which helps patients with brain damage after a stroke in their rehabilitation. **[8]**

In the virtual reality system, the patient is asked to do things. It can be to stop a ball that comes from the front, go around obstacles while walking, having to reach for objects. Nothing is real, but for the patient can be. Because, according to the researchers, in the case of hemiplegia, the brain tends to use a part of the body and have to "convince" to use it.

Thus, faced with a stimulus immersive, the patient can reach what in real life one resists because he does not realize. The aim is to manipulate the brain in a less damaging environment so it can build trust and become better also in real life.

Finally, such in videogames and serious games, virtual reality can lead to immersive and new environments that increase the productivity, learning or creativity of the users, and users will have and increased motivation and willingness to work with these systems.

In this project, there will be used some virtual reality hardware to track the head and hands position of the users. This can help the patients doing some kind of physiotherapy to make their exercises with more motivation, in a process called gamification. An interactive environment or game is created, with some kind of game mechanics like increasing the level of difficulty or gaining points. The user then will be more willing to do repetitive or boring exercises considered as obligations, because new challenges are created.

2.6. Supporting Studies

The considerable number of recent technologies used for neurorehabilitation find support and feedback on clinic and academic studies. It is needed to determine if virtual reality has a positive impact on brain or limb motor recovery. For example, it is believed that patients exercise harder and longer if they are in an immersive virtual rehabilitation environment providing positive reinforcement feedback. Some studies conclusions have been considered when choosing virtual reality for the development of this project.

The first evidence found is a case report made by George Fulk in 2005. It is the first case to include a virtual reality based balance training for a patient with Multiple Sclerosis. It analysed the performance of a 48-year-old woman with long time history on Multiple Sclerosis. **[9]** The outcomes were that after the training, the patient demonstrated improvements in gait speed, gait endurance, and balance; and maintained the improvements after two months of the end of the experiment. Although the plan was formulated based on the patient's goals, it shows a task oriented with repetitive and intensive exercise plan that can be used in future studies.

Sergei V. Adamovich, Gerard G. Fluet, Eugene Tunik, Alma S. Merians reviewed in 2009 over a hundred references on experimental scientific evidences and studies regarding the benefits of training in virtual environments in their review *"Sensorimotor training in virtual*

reality: A review". They analysed the effects and engaging that virtual reality and virtual feedback to the patients can provide for sensorimotor stimulation and for inducing brain reorganization. They also compared this effects to the classic exercises in real world used for neurorehabilitation training. **[10]**

They concluded that virtual reality may be an optimal tool for designing therapies that target neuroplasticity mechanism in the nervous systems and provide training in complex situations that are impossible to create in the real world. Virtual reality provides a rehabilitation tool that can be used to harness the nervous system's capacity for sensorimotor adaptation. Virtual environments can manipulate the specificity and frequency of visual and auditory feed-back, and can provide adaptive learning algorithms and graded rehabilitation activities. As a counterpart, they also admit that although positive results are shown, virtual reality technology used in rehabilitation is starting to be used and larger controlled studies to determine how virtual reality affects neural processes are needed. More studies have been performed during the past years.

The article *"Effects of virtual reality-based bilateral upper-extremity training on brain activity in post-stroke patients"* shows the research made by Su-Hyun Lee, Yu-Mi Kim and Byoung-Hee Lee from Sahmyook University in the Republic of Korea. It shows the beneficial effects of virtual reality for regaining motor mobility on upper extremities for neurorehabilitation purposes.

In the research **[11]** , twenty patients who suffered stroke injuries were evaluated and divided into two groups. Both performed a bilateral upper extremities exercise plan, but one group used a traditional training and the other used a training based on virtual reality. All training was conducted 30 minutes per day, three times per week for six weeks, followed by brain activity evaluation with electroencephalography. In addition, the patients were offered conventional physical therapy for 30 minutes per session, five times per week for six weeks.

Upper-extremity function, muscle strength, and brain waves were evaluated before the intervention. Ten patients were randomly included into the virtual reality-based bilateral upper-extremity training and the other ten patients served as a control group and trained while watching an irrelevant video. Virtual reality immersion involved a visual expression

technique using animations and provided cognitive information for feedback. This was achieved by the use of cameras, a computer, and a monitor for information displaying.

The animation consisted of symmetric and asymmetric upper-extremity training. A monitor displayed the virtual reality program and simultaneously recorded the patient's bilateral upper-extremity movements. Patients received real-time feedback through the monitor while training.

The study concluded that the virtual reality environment with repeated bilateral movement learning increased the efficiency of training by inciting the patient's interest, providing real-time feedback to modify the patient's movement, and improving behavioural understanding.

The brain data collected also evidenced that the virtual reality group exhibited improved concentration and brain activity compared to the other group. This difference might be due to an external stimulus such as virtual reality that involves a direct multi-sensory system, inducing concentration and enhancing brain activity.

2.7. Similar Projects

Another important base of the project are similar initiatives which were done during the past one or two years in Spanish universities. These are important because they provide an environment for information exchange and continuous development.

The program SPECS, from Universitat Pompeu Fabra (UPF), developed a system to help patients with cerebral damage after a stroke in their neurorehabilitation. It consists of sensors and virtual reality glasses, and makes the patients do everyday activities like walking or holding objects in a virtual environment. It allows a faster recovery by showing a faster recovering in the virtual environment and reinforcing brain neural paths. The system has achieved remarkable results and has been proven to work until 3 years after the stroke happened, when the brain is still adapting to the new situation. It has been tested it with over 600 stroke patients, but not other types of brain damage.

Another research project was made by Universidad Carlos III de Madrid (UC3M). It consisted on a virtual reality system for shoulder motor rehabilitation. The objective was to

rehabilitate the shoulder based on two movements, the adduction and abduction of the shoulder (to raise or low the arms forming a "T" with the body). The virtual environment was developed as a football field, where the patient would be shown as a goal keeper and would have to stop the balls coming to him using the upper extremities movements.

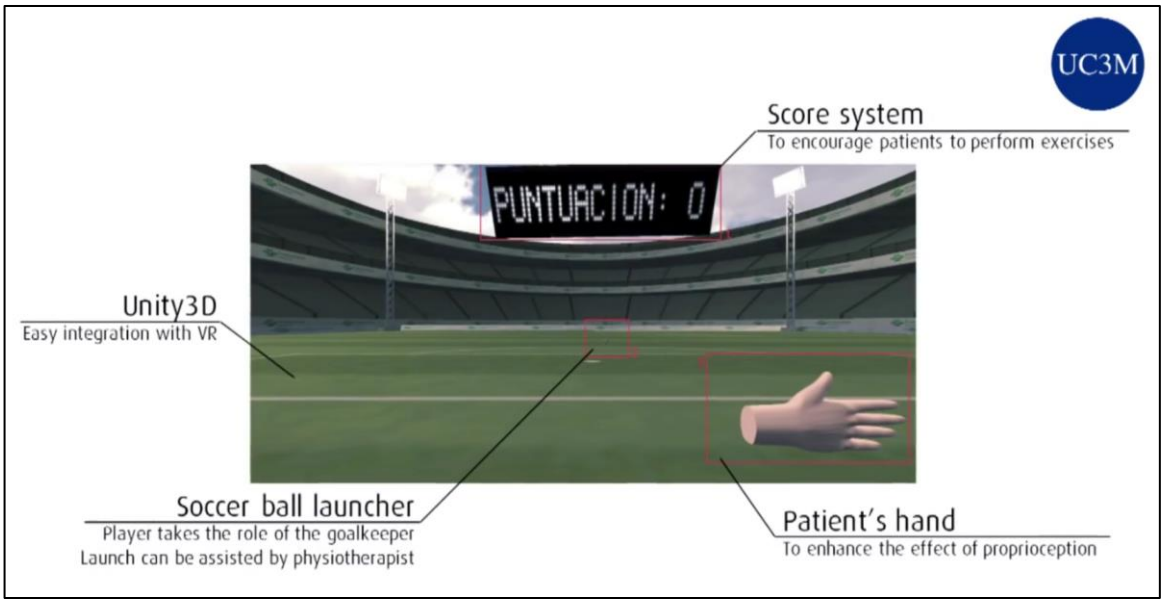

Figure 2.6. UC3M shoulder rehabilitation system. Source **[12]**

An evaluation was done working with physical therapists and the feedback was positive. Therapists felt that the patients improved the mobility of the shoulder joint, increased their muscle mass and recovered strength. However, the idea was still a prototype and more work is required before clinical analysis or commercial options are available.

3. Virtual Reality

Technically, **[13]** virtual reality is the term used to describe a three-dimensional, computer generated environment which can be explored and interacted with by a person. That person becomes part of this virtual world or is immersed within this environment and whilst there, is able to manipulate objects or perform a series of actions.

Virtual reality aims to immerse the user in an alternative world, wrapping it in the story, and allowing him to interact with the environment. In summary, make the user feel to 100 per cent in another place, being able to move or make decisions in real time.

The world of video games in the late 80 and early 90 tried to apply this technology with releases like Sega VR or Virtual Boy Nintendo but their helmets were something rudimentary. Virtual reality was still a technical concept without reasonable prospects for the industry. In 2010, Palmer Luckey began to work in a virtual reality helmet called Oculus Rift. This project started the current development boom of virtual reality devices.

3.1. Types

3.1.1. Non-immersive

The display of the virtual elements is done through a screen. The devices are classical computers and screens everyone is used to work with. The interaction is done through accessories such as keyboard, mouse or the microphone, which become the medium of entertainment. It takes advantage of the possibilities offered by Internet to interact in real time with different people or environments which, in reality, do not exist. This type is most suitable for scientific visualizations, although it is more widespread is as a means of entertainment. **[14]**

3.1.2. Semi-immersive

It arranges four displays in the form of a cube, three for the walls and another for the floor, which surround the user. This needs for a pair of glasses and a device for monitoring head

movement. They are used mainly where the user is required to keep in touch with real world elements.

Figure 3.1. CAVE Semi-immersive system. Source **[15]**

3.1.3. Fully Immersive

It is based on the simulation of a three-dimensional environment in which the user perceives through sensorial stimulation and feels the virtual world that is exploring. It achieves this through devices such as gloves, headsets, helmets or special suits, as they all capture the position and rotation of distinct parts of the body. This type of system is ideal for applications of training. In this project, fully immersive reality will be used, using as peripherical a Head Mounted Display (HMD) and two controllers with sensors to detect the player's position.

3.2. Products

3.2.1. Head Mounted Display (HMD)

It is the main way to achieve fully visual immersion. A head-mounted display, is a display device similar to a helmet, which allows to reproduce images created by computer on a screen which is very close to the eyes, or even to project the image directly on the retina of the eye.

Due to its proximity to the eyes the virtual reality helmet you get that the images displayed are much greater than those perceived by normal screens, and it can encompassl the entire field of vision of the user. Thanks to the fact that the headset is subject to the head, this can follow the movements of the user, making this feel integrated into the environments created by computer.

Figure 3.2. LG virtual reality HMD. Source **[16]**

3.2.2. Haptic Devices

Our tactile senses are a crucial component to complete the illusion of virtual reality world environments. Haptic or kinaesthetic technologies recreate the sense of touch by applying forces, vibrations, or motions to the user. Haptic devices may incorporate tactile sensors that measure forces exerted by the user on the interface.

It is a technology currently being developed for virtual reality experiences. There has been little use until today, as the price of the input and output devices necessary for its functioning is high, and a high percentage of the computer computation power is used in other necessary products like the HMDs.

There are plenty of devices being developed, such as walking platforms for movement, sensor gloves and footwear, or full body suits. Two of the available solutions will be explained next:

AxonVR builds haptic technology that enables realistic feeling in virtual reality for the first time. Its textile that simulates lifelike touch, allowing you to feel the texture, shape, motion,

vibration, and temperature of virtual objects. Its skeleton is a lightweight exoskeleton that extends the realism by applying physical forces to your body. These advanced haptic technologies bring virtual experiences to life with unprecedented levels of realism. **[17]**

Figure 3.3. AxonVR suit. Source **[17]**

Virtuix Omni is an active virtual reality platform that enables natural movement. Active VR, where your movement in the virtual world is controlled by your physical actions like walking or running, creates an unprecedented sense of immersion that cannot be experienced sitting down. It is a way to run in virtual reality safely and comfortably, without risk of falling or colliding with people, walls and other objects that hurt. **[18]**

Figure 3.4. Virtuix Omni platform. Source **[18]**

3.3. Technical Considerations

HMD or headsets used for virtual reality have some technical characteristics that need to be considered and affect the user experience.

3.3.1. Display Technology

The most common is having one liquid crystal display (LCD) or OLED display per eye. Resolution is an important parameter because it is directly related to the image quality perceived by the user. Nowadays full HD (1080*1200 pixels) resolution is typical.

3.3.2. Refresh Rate

It is the number of images shown by second. 60 Hz or frames per second is considered a good ratio which may not cause any problems, but some headsets already have 90 Hz or 120 Hz. For comparison, cinema movies are filmed at a ratio of 24 Hz.

3.3.3. Latency

It is the time between the user moves his head and that the displayed image is reset to that movement. The manufacturers try to reduce it to a minimum since excessive latency can cause dizziness and less realism in users. A successfully experience usually has a latency of 20ms or less.

3.3.4. Head and Position Tracking

The HMD uses internal sensors (gyroscope, accelerometer, magnetometer) to detect where the user head is. Position tracking is achieved through a sensor, usually external to the glasses, which detects exactly where is the head of the user in the room and any changes that may occur in that position.

3.3.5. Senses Dissonance

One of the problems of virtual reality is that there exists a dissonance between what our eyes see on the screen and the type of movement that the body really feels. If we rode a roller coaster in a virtual environment, our eyes capture the vertical tilt, but the vestibular

system of our body (the sensory system that provides the leading contribution to the sense of balance and spatial orientation for the purpose of coordinating movement and balance) does not register this change. This can provoke dizziness, eye discomfort and fatigue. However, modern technologies are being investigated to reduce these effects, like changing in a dynamic way the shadow of the user's Field of View **[19]** .

4. Hardware

4.1. Virtual Reality Systems

Although smartphones can provide reasonable virtual reality experiences, their hardware is still far from what is possible with powerful computers and major video games consoles. This means that, in terms of pure visual fidelity and frame rate, dedicated external computers are still the best choice for virtual reality. Some of the commercial options for immersive virtual reality will be detailed next.

4.1.1. Google Cardboard

The idea behind Google Cardboard is to transform any smartphone with Android in a virtual reality platform in a cheap way, because the user will be using his own smartphone once he made sure the model fits correctly. The original model and some variations use simple cardboard and allow the user to assemble their own glasses.

Figure 4.1. Google Cardboard VR. Source **[20]**

There are a great amount applications and games available on Google Play, but also with any YouTube turning on the mode of virtual reality that provides the official application of YouTube. YouTube is also producing videos in 360º for this device.

The main advantage is its cheap price and availability for all society. It reduces the cost of virtual reality glasses into only 5 euros. There is also an immense amount of content to be able to enjoy virtual reality experiences. However, the quality of experience is just decent. Google Cardboard is so basic that it depends on our smartphone for everything, including the follow-up to our head or motion detection. The field of view is also very limited. The glasses provide visual surrounding but not a real sense of immersion. They also do not have a good rate of latency, and after few minutes there may appear sensations of dizziness or nausea.

4.1.2. PlayStation VR

This Sony platform is a part of their videogame console ecosystem of entertainment: a PlayStation 4 is needed to play the games. However, the whole price is much lower than when a PC with high processing and graphic capacity is needed. Being a closed ecosystem and working only with one type of device (PS4), does not generates connection or configuration problems, and there will not appear conflicts that could arise before on the PC configurations compatible with Oculus and HTC.

The system is very comfortable to use and consists of the headset or HMD, a camera for movement detection and the PlayStation own controllers. It is connected to the console by a long cable which allows some freedom of movement. A good feature is that no television is needed to use the glasses. The glasses have two OLED 5.7-inch screens and a 1.920x1.080 pixels resolution in 16:9 format.

Figure 4.2. PlayStationVR. Source **[21]**

There are plenty of games and application available, made by Sony or other development companies, since the success of the games console attracts potential players. Some disadvantages are that screen resolution is lower than Oculus and HTC devices (both with 2.160 x1.200), although its refresh rate is greater, 120Hz. The field of vision is also smaller. Since the price of the system is also lower, these glasses are a very good option quality price.

4.1.3. Oculus Rift

This is the reference project for virtual reality, since its initial project made other manufacturers on the technology field to start investing in virtual reality platforms as well. It is said that these are the glasses everyone wanting to have the highest quality experiences in this area should seek, although the investment needed is higher than the previous options.

The Oculus Rift has an approximated cost of 600 euros and may need more investment in computer parts to meet the hardware requirements to have an appropriate working experience. It connects to the computer also by cable, no wireless option is produced.

Figure 4.3. Oculus Rift. Source **[22]**

The supply of content for the Oculus Rift has a specific store called Oculus Share and a Windows application that will also serve as a store in which find games or applications.

It is the hardware that started modern virtual reality systems and developed most technologies involving virtual reality, and they have one of the greatest experiences handling

with user immersion and satisfaction. For example, the screen resolution is the highest on the market together with HTC Vive resolution.

One very significant negative aspect is that the player is expected to be static or to sit down in order to use the glasses, and this is not the most appropriate for immersion purposes. The hardware requirements make the economic investment likely to be high.

4.1.4. HTC Vive

This system was developed with the support and collaboration of Steam to start their VR section, SteamVR. Steam is the biggest videogame digital distributor worldwide. The section has virtual reality games and experiences but also adapts traditional content so users can enjoy them with the glasses of virtual reality. Lately, Oculus support was added too.

HTC Vive are direct competitors of the Oculus Rift, however HTC Vive experience is more complete. In addition to the virtual reality glasses, it includes two wireless controllers for the hands and two base stations that are especially interesting because they expand the virtual experience and allow to enjoy it not just sitting, but standing and in room environment.

Figure 4.4. HTC Vive. Source **[23]**

The use of these base stations and of the two wireless controllers make the HTC a more complete solution and achieves the maximum immersion possible, but, in Spain the Vive system has an approximate cost of 700 euros, a little more than the Oculus. Like Oculus, the price does not include extra hardware elements for the PC.

A powerful PC is needed in order to get all that experience of virtual reality. The minimum requirements are almost identical to the Oculus technology, the most important thing is to have a right graphics card (NVIDIA GeForce GTX 970, AMD Radeon R9 480 or one with higher power) in addition to 4 GB of RAM, an HDMI port 1.4. If we comply with all the requirements we can have a more complete solution on hardware. Since Steam is a heavily used platform, the potential user base and options for applications or games is giant. Some important titles for this system are 'Tilt Brush', a 3D painting and designing environment, and 'Job Simulator', which allows to simulate different jobs, but there exist a lot of more options.

4.2. Chosen Hardware

Dedicated external hardware is the best choice for virtual reality. The hardware also has to be able to support different software for the monitoring of the patient. PlayStationVR system has notable features, but it is a closed environment which would made the final user of the application to also buy the videogame hardware apart from the glasses, and make it difficult to program the application and additional features. It also needs the approval of a private videogame company to make the project openly available in their store channels.

The following table compares the two remaining systems, Oculus Rift and HTC Vive.

Feature	Oculus Rift	HTC Vive
Display	OLED 2160 x 1200	OLED 2160 x 1200
Field of View	110 degrees	110 degrees
Refresh Rate	90 Hz	90 Hz
Tracking Area	5x5 feet or 8x8 feet depending on sensors	15x15 feet
Sensors	Accelerometer, gyroscope, magnetometer, Constellation tracking camera.	Accelerometer, gyroscope, Lighthouse laser tracking system, front-facing camera
Platform	Oculus Home	SteamVR, VivePort

The minimum PC specifications for both systems are almost identical: graphics card NVIDIA GeForce GTX 970 or AMD Radeon RX 480 for HTC and NVIDIA GeForce GTX 960 or AMD Radeon RX 470, 4 GB of RAM for HTC and 8 GB for Oculus Rift, HDMI and USB ports. These are the two powerful and both need to run in a computer.

The price and technical consideration for the screen in both systems are comparable. The system chosen is the HTC Vive due to it has a greater tracking area and a more precise tracking system using the Lighthouse technology. This will allow to capture in a more precise way the extremities movements of the patients when they are holding the controllers. Also, having a platform such as SteamVR, is another positive aspect to take into account, since the games developed with HTC Vive technology will have a more direct access to the major online game store platform in the world.

4.3. Motion Capture in HTC

The base stations make use of a technology called Lighthouse that generates a digital map of the room in which the virtual scene will take place, and the user can move through it within that integrated virtual reality. The result is prodigious, but it needs an empty space or room that might be difficult to get in conventional house or apartments, so no the user does not collide with any remaining obstacles when playing.

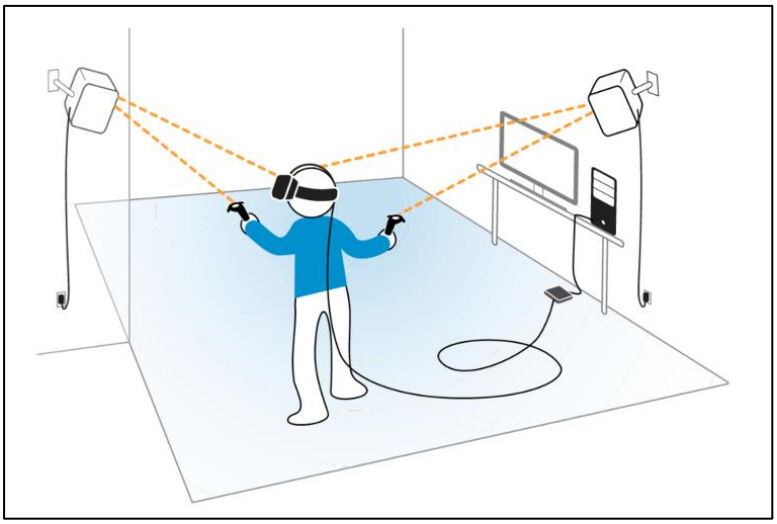

Figure 4.5. HTC Vive Lighthouse tracking technology. Source **[24]**

The accuracy of this system is very high. It provides full interaction at 360 degrees of movement and real immersion. Two stations placed in the room create a network of 360º motion detection to achieve higher virtual reality experiences. In addition, the configuration of these stations is quite simple, they are synchronized automatically and wirelessly and only need a power supply cable to run. As a practical result, it is possible to use a Lighthouse controller with an expected accuracy of under 2mm in the controller position.

The main idea behind the tracking technology is pretty simple—by flooding a room with non-visible light, Lighthouse (the base stations in HTC) functions as a reference point for any positional tracking device (like a VR headset or a game controller) to figure out where it is in real 3D space. **[25]** The system is cheap to produce and does not need any cameras or optical tracking.

The base stations contain bunch of stationary InfraRed (IR) LEDs, and a pair of InfraRed active laser emitters that spin. These create the light that fills the room. Sixty times every second, the LEDs flash, then one of the two spinning lasers sweep a beam of light across the room.

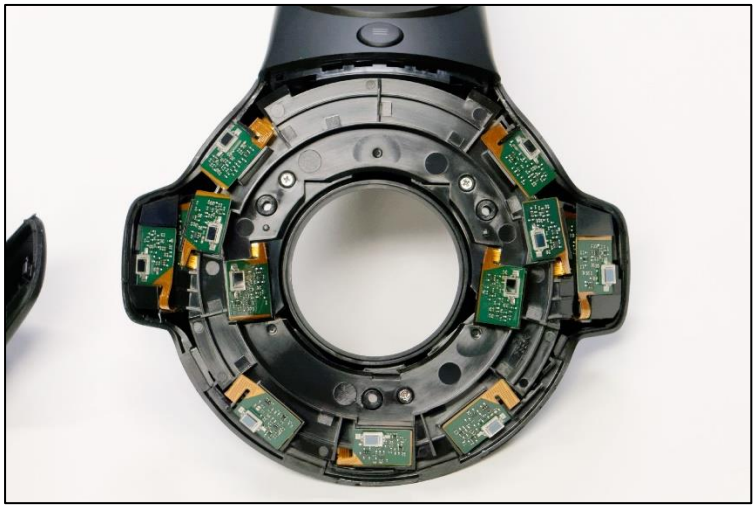

Figure 4.6. Detail of the sensors on a Vive Controller. Source **[26]**

The headset or controller is covered with little photo sensors that detect the flashes and the laser beams. When a flash occurs, the headset simply starts counting until any of its photo sensors gets hit by a laser beam, and it calculates the position very accurately by using the relation between where the photo sensor is on the headset or controller (which of the

multiple sensors has been hit by the stream of light) and when it has been hit (the amount of time that has been counted).

Since a lot of the photo sensors are hit by a laser at the same time, the 3D shape calculating the position and rotation of the object can be created.

The device can also use data from an internal gyroscope and accelerometer to track its own location in real-time, so the location can be known during the periods of time between the base station sweep or even if all sensors have been temporarily blocked from view.

5. Game Engines

5.1. Available Game Engines

A game engine or graphic engine is a software framework designed for the creation and development of video games. The main functionalities of a game engine can include a renderer for 2D or 3D graphics, collision detection, sound, scripting, animations, streaming, memory management, threading, scene graph, or video support for cinematics.

For this project, due its time limits, it has been used Unity3D because it allows to do complete quite projects with a high learning speed, and has multiplatform capacities. Unity Technologies also disposes of a substantial number of tutorials and helping methods. The industry of videogames uses a wide variety of game engines.

5.1.1. GameMaker: Studio

GameMaker has been a classic game engine for many years. Using a single development workflow GameMaker Studio 2 allows to export your game directly to Windows desktop, Mac OS X, Android, iOS, HTML5, videogame consoles, and a lot more.

It is a platform that makes it very easy to start into game development, and has the required tools to make less experienced designers in development feel comfortable and advance in their prototypes. Uses a 'Drag and Drop' development environment and the developer can have its game with not having to write a lot of code. GameMaker's built-in language (GML) is simple and helps to learn programming.

Figure 5.1. Game Maker Studio logo. Source **[27]**

Especially designed to create games in two dimensions, builds in the popular physics library Box 2D and direct connection to the main monetization services, ads and analytics market.

A free version can be downloaded from their website. Users of this game engine have the possibility to create games with 2D physics, create a thriving player base with cross-platform network play, and create stunning visual effects using shaders, only by learning a minimum of programming, design and art to develop it.

5.1.2. Stencyl

It is a very curious engine and easy to use. Stencyl is a program or graphics engine that will allow to create your own games without the need of programming knowledge.

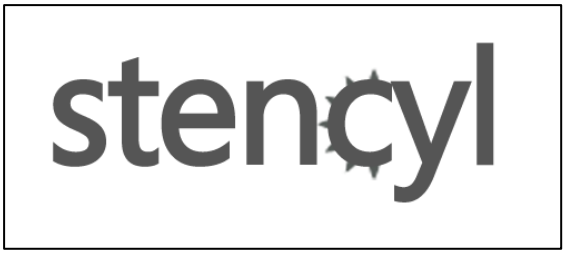

Figure 5.2. Stencyl logo. Source **[28]**

Its operation is simple: consists of applying logic and joining pieces as if it were a puzzle, although we can also download other platforms and modules for free. This drag-and-drop gameplay designer comes from another project, MIT Scratch. However, Scratch system is extended with new functionality and more functional ready to use blocks.

This makes an engine based on essentials and accelerates the workflow. Advanced users can also create and share their own blocks, extend the engine through code, import libraries and write their own custom classes that interact with the engine.

The developer will be able to commercialize the game for Windows browsers, and other PC platforms for free. For revenue, games can put embed ads or use in-game purchases to unlock additional content or power-ups while offering part, or all of the game for free, creating a bigger player base. Like others, it also has an extended platform support, with supports for: iOS (iPhone/iPad), Android, Windows, Mac, Linux, Flash and HTML5.

5.1.3. Unreal Engine

Unreal Engine is one of the most important graphic engines in the gaming industry. The last version, Unreal Engine 4 is a complete suite of creation tools flexible enough to ensure success for teams of all sizes.

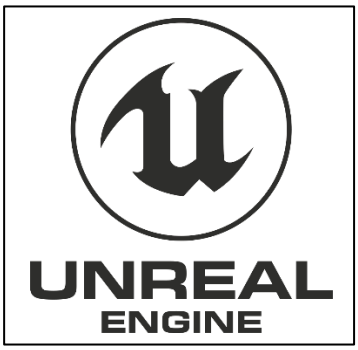

Figure 5.3. Unreal Engine logo. Source **[29]**

The most interesting part of the Unreal Engine 4 is the ability to create the games for free without paying any monthly instalments. The company 2 for a 5% of the profits generated from the game if it generates more than 3000 $ per trimester term.

Unreal 4 is one of the most powerful graphics engines and also has a heavy amount of versatility. It is possible to design a high quality realistic environment or a lot more colourful, visually simpler and cartoon style.

It also adds multiplatform and support for virtual reality. VR requires complex scenes rendered at very high framerates. Unreal Engine is designed for heavy demanding applications and it can meet these requirements. Unreal uses a pure C++ engine designed for high performance.

The only downside of the Unreal Engine 4 would be that it is one of the game engines more difficult to use. Many hours have to be dedicated to get something decent, both in its map editor and its code development. However, the results also have a really high quality in all aspects.

5.1.4. Unity3D

Unity3D is one of the most widely used graphic engines for all those who start in the world of video game. Unity 5 engine allows users to use a free version to develop their games, in the case they are programming for Windows. The free version is as complete as the paid version, with no restrictions.

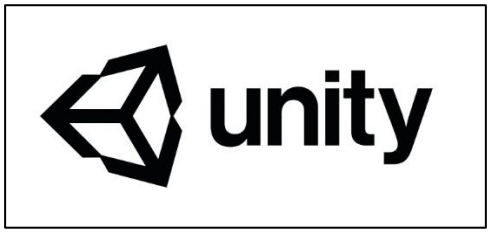

Figure 5.4. Unity logo. Source **[30]**

The Unity Editor is a creative focus for artists, designers, developers and other members of the team. Available on Windows and Mac, includes tools for the design of 2D scenes and 3D, instant play mode for quick editing and iteration and a powerful animation system.

Unity 5 allows users to create genuine high quality realistic graphic environments in a very brief time of use. The ease with which the user can create scenarios and learn to use Unity it is pretty high. Unity Technologies also puts to public disposal a substantial number of tutorials and methods put at our disposal to learn how to use it.

It has support for more than platforms between mobile, desk, console, TV, VR, AR and the web. Unity makes it easy to take developed contents to the last technologies and platforms, such as virtual, augmented or mixed reality.

A great characteristic for programmers is its real-time error handling. It finds and addresses issues of high priority that the potential customers might be experiencing in the future, and it collects and reacts to the application errors, in real time, between platforms and devices.

Unity has an online catalog where users can upload particular solutions. This is called Asset Store. The Asset Store provides free or paid ready to use content which elements that can provide solutions to developing and finishing the game faster. Start your development and reaches the finish line faster.

5.2. Chosen Engine

The two first options shown, GameMaker and Stencyl, are simpler programming game engines. They are good for beginners wanting to start to develop a simple game or developers wanting to create 2D games or applications for mobile platforms such as Android. Since it is needed the implementation and design of a 3D immersive environment, and Virtual Reality support for HTC Vive glasses, the two main options considered were Unreal Engine 4 and Unity3D. Both allow to create environments and program object interactions, and import or export elements from other projects in an uncomplicated way.

The chosen engine to develop the application is Unity3D. First of all, a free version containing all features is available for students or educational purposes. The programming language is C#, which is a common language and the developer of the project already has some knowledge in it. Unreal Engine 4 is a very powerful engine used in complex projects, but it requires more learning time in programming and map design to achieve the same results wanted for the demo in this project. Finally, Unity engine has a bigger user base, which mean more resources for learning or to include on the projects are available. The engine community is important to solve any doubts or issues which could happen during the game development. It also can help to share and spread the possible final outcomes of the project if the result is successful.

5.3. Developing with Unity3D

In this section, it will be explained the key features and considerations of Unity3D development environment. Also, the basic components of a game and how they can interact using this technology. Documentation used is partially taken from Unity3D help **[31]** and tutorials.

5.3.1. Creating and Opening projects

When Unity Editor is launched, the Home Screen appears. The Project tab appears once you have some edited projects on your computer. Get Started gets the user more information and tutorials about Unity.

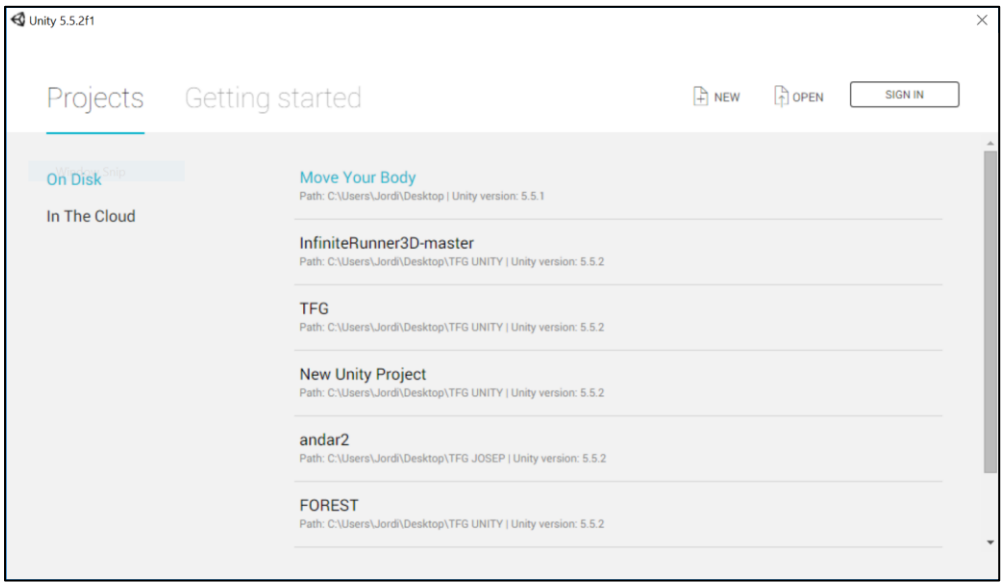

Figure 5.5. Unity Home Screen

To open a new Project, click New. To open an existing Unity Project stored on the computer, click the Project name in the Projects tab, or click Open to browse the computer for the Project folder.

In the top right corner of the Home Screen, click 'New' to see the create a project window. It can also be accessed later on the Editor. When creating a project, the following options must be considered:

- Project name: Sets the name of the Project. This names the folder which stores the Assets, Scenes and other files related to the Project.
- Location: Defines where in the computer's file system will be stored the Project.
- 3D / 2D: Choose whether the project is in 2D or 3D. This sets up various presets in Unity to make getting started easier. This is set to 3D by default.
- Add Asset Package: Adds pre-made content to the Project. The Asset Packages provided with Unity include pre-made models, particle effects and example scripts, along with other useful tools and content.

5.3.2. Editor Environment

The main editor window is made up of tabbed windows which can be rearranged, grouped, detached and docked. The main windows used in this project are:

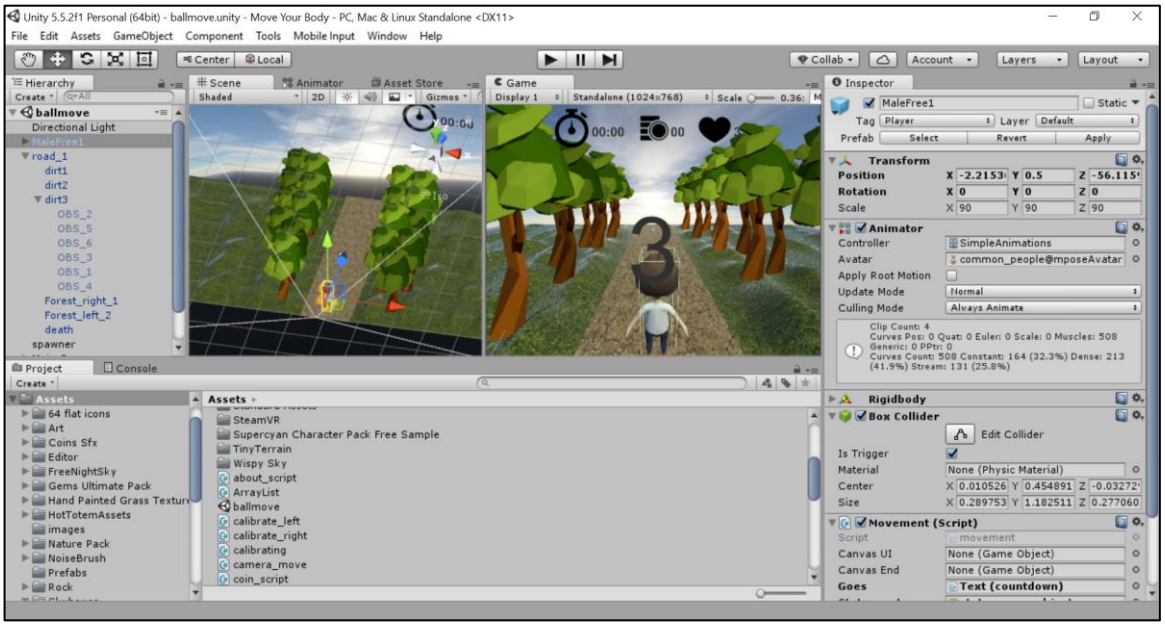

Figure 5.6. Unity Editor

- **Project Window:** displays the library of assets that are available to use in the project. When you import assets into the Project, they appear here.
- **Scene View:** allows to visually navigate and edit the scene. The scene view can show a 3D or 2D perspective, depending on the type of the project.
- **Game View:** It is the representation of the final, published game. It is needed to use one or more Cameras to control what the players actually sees when they are playing the game.
- **Hierarchy Window:** is a hierarchical text representation of every object in the scene. Each item in the scene has an entry in the hierarchy, so the two windows are inherently linked. The hierarchy reveals the structure of how objects are attached to one another.
- **Inspector Window** allows to view and edit all the properties of the currently selected object. Because distinct types of objects have different sets of properties, the layout and contents of the inspector window will vary.

- **Console Window**: shows errors, warnings and other messages generated by Unity.
- **Animator Window:** allows to create, view and modify Animator Controller assets. An Animator Controller asset is created within Unity and allows you to maintain a set of animations for a character or object.

There is also the Unity Asset Store. It is home to a growing library of free and commercial assets created both by Unity Technologies and also members of the community. A wide variety of assets is available, covering everything from textures, models and animations to whole project examples, tutorials and Editor extensions. The assets are accessed from a simple interface built into the Unity Editor and are downloaded and imported directly into your project.

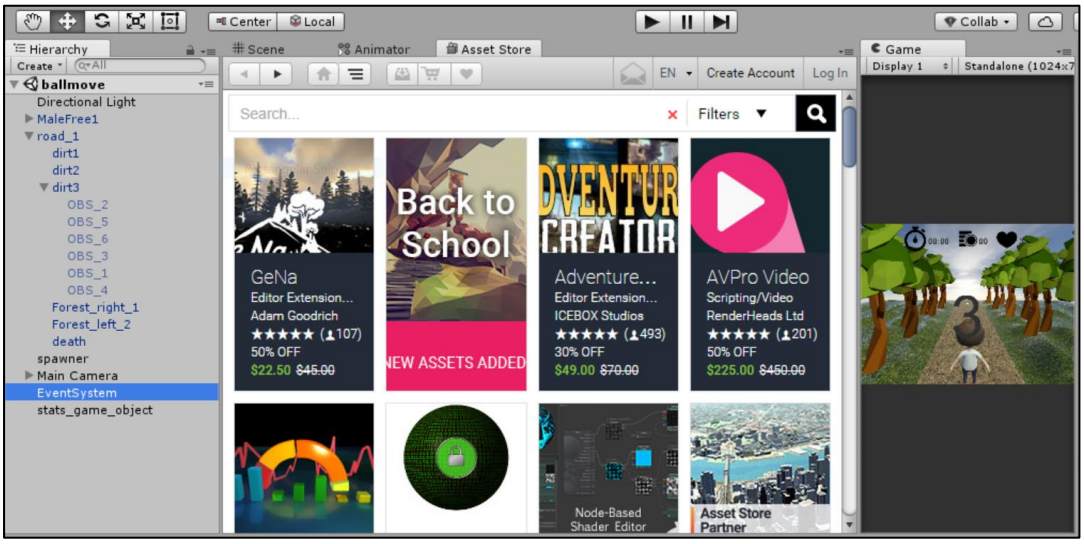

Figure 5.7. Asset Store view

5.3.3. Unity elements

There are some significant elements that need to be reviewed when editing and developing an application with Unity3D.

- **Asset:** An asset is representation of any item that can be used in your game or project. An asset may come from a file created outside of Unity, such as a 3D model, an audio file, an image, or any of the other types of file that Unity supports. There are also some asset types that can be created within Unity, such as an Animator Controller, an Audio Mixer or a Render Texture.

- **Scene:** Scenes contain the environments and menus of the game. Think of each unique Scene file as a unique level. In each Scene, you place your environments, obstacles, and decorations, designing and building your game in pieces.
- **GameObject:** Every object in the game is a GameObject, characters, environment items, camera, etc. It is needed to give a GameObject properties to perform its function. For that purpose, a variety of components are added to the GameObject.
- **Prefab:** It is an asset type that allows to store a GameObject object complete with components and properties. The prefab acts as a template from which you can create new object instances in the scene. Any edits made to a prefab asset are immediately reflected in all instances produced from it.
- **Scripts:** The behaviour of a GameObject is controlled by the Components that are attached to them. Scripts allow to implement personalized gameplay features. Unity allows to create Components using scripts. These can trigger game events, modify Component properties over time and respond to user input in any way you like. In this project, scripts are programmed in C# (C Sharp).
- **Transform:** The Transform is used to store a GameObject position, rotation, scale and parenting state. A GameObject will always have a Transform component attached.
- **Tags:** A Tag is a reference word which you can assign to one or more GameObjects. Tags help you identify GameObjects for scripting purposes. They ensure you do not need to manually add GameObjects to a script.
- **Primitive objects:** Unity can work with 3D models of any shape that can be created with modelling software. However, there are also a number of primitive object types that can be created directly within Unity, namely the **Cube**, **Sphere**, **Capsule**, **Cylinder**, **Plane** (XZ plane) and **Quad** (XY plane).

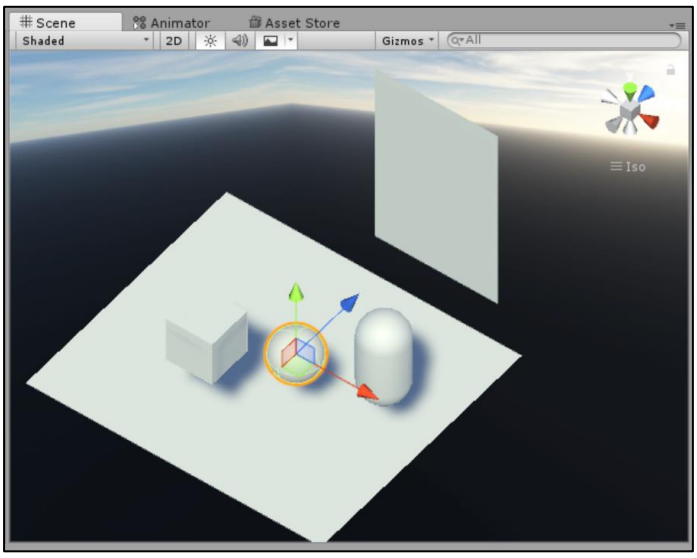

Figure 5.8. Unity primitive objects

- **Light:** Lights are an essential part of every scene. They define the colour and mood of your 3D environment. It is common to work with more than one light in each scene.
- **Camera:** Cameras in Unity are used to display the game world to the player. There will always be at least one camera in a scene, but more than one is possible. Multiple cameras can make a two-player split screen or create advanced custom effects. They can be animated or controlled with physics.

6. Game Design

6.1. Game Planning

Before starting the development of the game, it is a good practice for the development person or team to generate a Game Design Document (GDD). The game design document is a highly descriptive document that summarizes and synthetizes how the is going to be (concept, history, gender, number of platforms, production equipment, etc.).

Since the developed game is an interactive experience demo, a shortened version of this game document will be presented:

Introduction:

Move Your Body is an endless runner 3D serious game for PC or VR glasses that helps people with different degrees of arm mobility problems, motivate themselves to recover faster and in a more entertaining way.

Description:

Once the game is started, the player starts running on a road while finding obstacles that the character has to avoid, either jumping over them or moving the character left or right.

Visual Style:

Cartoon low polygonal style, cell-shading.

Audio Style:

There will be no music or dialogues.

Starting Out:

The game screen will be accessed by a previous menu. The game starts with a start menu that allows to configure options, such as setting the player's final movement position. In the main start menu, there will be 4 different options shown: Start, Calibrate, Options, About.

Start will get the game started. Options will set different options of difficulty regrading obstacles generation and game speed. Calibrate will take the position of the VR controllers. About will display the creator of the game and other references.

Locations:

The main game will be settled as a forest looking environment. The obstacles to dodge will be trees, logs and rocks.

Controls:

It will be used the position sensors of the HTC Vive VR controllers.

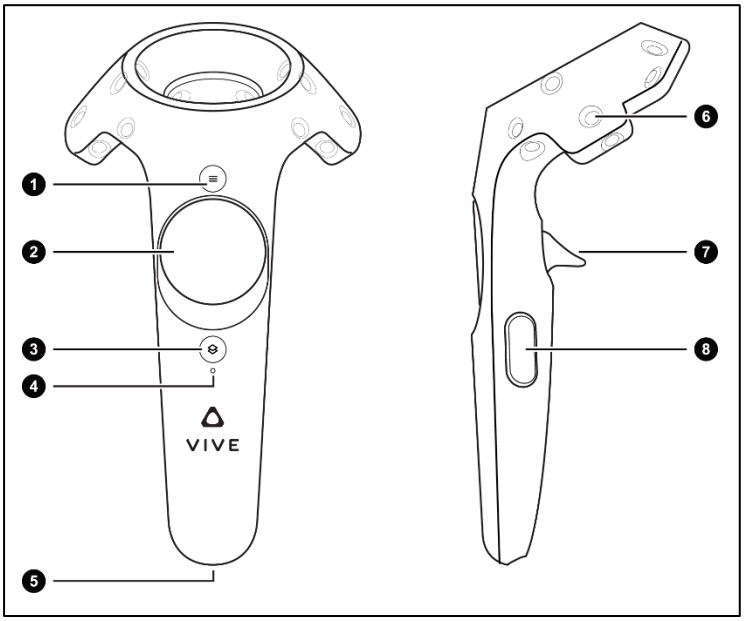

Figure 6.1. HTC Vive controller. Source **[31]**

PC controllers, keyboard and mouse, will also be used to get access through the menus, and to test and develop the option for the game:

Left click: Accept (Menu)

Left arrow key: Move Left (In-game)

Right arrow key: Move Right (In-game)

6.2. Endless Runner

The game designed is an endless runner. It is a genre of game where the player must progress always in the same direction, usually escaping from an enemy or danger, and whose aim is to advance as much as possible before dying. Usually the main actions are jump and dodge obstacles, but many games of this type also include some type of attack or power ups dynamics.

The game has been chosen to be developed as an Endless Runner for several reasons. First of all, an endless runner provides a continuous stimulus in form of having to dodge the obstacles. Since the application will be for mobility training, this ensures that after a more or less fixed period of time the player will have to do a move, resulting in a continuous exercise. The game also has always the same dynamics so, although diverse levels of difficulty can be included, the progress of the player can be easily tracked. There are no save points available, every time the gameplay starts it runs until the player loses all lifes and the game ends. Once the player has started the game, he will keep motivated to do the better punctuation possible each time to not lose all the progress.

Infinite running games have two main characteristics: The character must always move in a forward direction without stopping, and the character is a 3D model who runs and has feet. The genre does not include games where the player is able to stop in some moments.

Figure 6.2. Subway Surfers in-game screenshot. Source **[32]**

The game developed will be designed in 3D and in a third-person camera characterization. This means that the player will be able to see the character model moving and dodging obstacles. These games usually present a way of obtaining points and saving some kind of record as score. In this project, the collectable items will be coins.

A lot of endless runners are available for platforms like Android and iPhone, due to the tactile environment generates a big deal of facility of use and control in this game situations. One well known example is Subway Surfers.

6.3. Art Design

It is intended to keep the game graphics simple and functional, while having some sense of a retro or cartoon style. This is achieved by using 3D models with have the indication of low poly. It is one of the trends of design being popular during last years and really good and innovative creations are created.

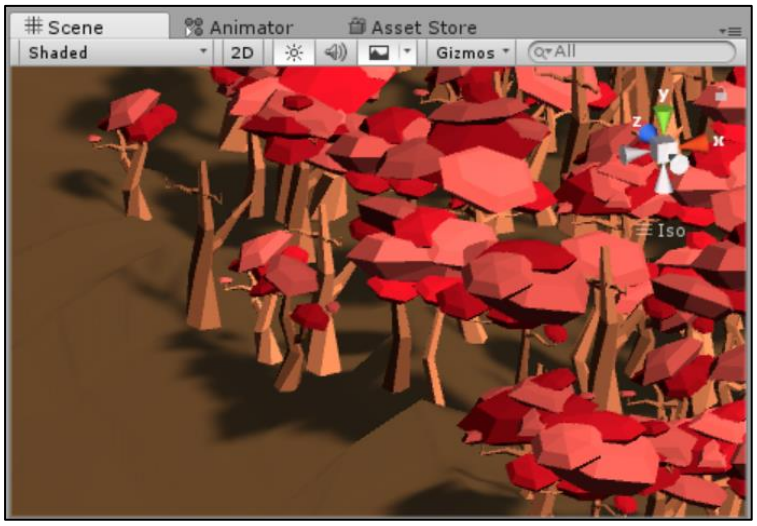

Figure 6.3. Low poly trees example scene.

Low poly is a form of 3D modelling based on using the fewest number of polygons as possible. It sacrifices the details to create simpler images. Objects may appear lacking in detail and objects supposed to be circular will look in a more polygonal way. The objective is to represent complex compositions, with minimum resources. There can be created landscapes, animals, portraits, characters, animations, etc.

A strong point of using this style for the game developed is that it provides optimization while still being visually attractive to the users. In some circumstances, a not very powerful PC, such as a laptop, may be used, so it is important to make sure that

6.3.1. Asset credit

One of the objectives of the project was to only use free assets to keep an affordable budget on this part. Since this is an education work, the assets can be used giving recognition to the creators.

During the design and programming of the game, there have been used downloaded assets from Unity Asset Store. A wide variety of assets is available, covering everything from textures, models and animations to whole project examples, tutorials and Editor extensions. The following asset where used in this project:

Asset	Category	Creator
64 Flat Game Icons	Textures & Materials/Icons & UI	Office57
Simple Gems Ultimate Animated Customizable Pack	3D Models	AurynSky
LowPolyForest	3D Models/Environments	Hot Totem Assets
Cartoon Nature Pack	3D Models/Environments	Mooshoo Labs
Noise Brush	Editor Extensions/Terrain	Denis Pahunov
Rock	3D Models/Environments/Landscapes	Vsify
Character Pack: Free Sample	3D Models/Characters/Humanoids	Supercyan
Tiny Terrain	Editor Extensions/Terrain	Jim Rogers
Wispy Skybox	Textures & Materials/Skies	Mundus Limited

6.4. Project Design

The project has been divided into two scenes, one for the Start Menu and one for the Game Screen. The interfaces needed in the game screen have been also created in that scene.

All the GameObjects needed for the correct functioning of both scenes will be added and configurated. Jumps between scenes will be on the game start and on the game end. Once the user or player has input the correct parameters in the Start Menu, the game will start. When the player loses and the game is finished, the game scene will be closed to return to the Start Menu. Scripts and Assets needed will be included in the project folders. Whereas a GameObject needs to be generated multiple times, a prefab of that object in question will be made.

The SteamVR Asset will be included. This allows in a uncomplicated way to incorporate into the project the camera for the virtual reality headset and the controllers for position and rotation detection in the room, since the plugin comes with the major part of the scripting built-in in order for these items to work.

Application scripts are included at the second annex of this project report.

6.5. Path Generation

6.5.1. Road prefabs

The first step in the designing was creating the road 3D model prefab. For that purpose, two terrains have been designed. One will be on the left of the road where the player character is and one will be on the right. The terrains are for decoration and simulate a forest with few trees. For the central part of the road it has been chosen a plain XZ. Grass and dirt textures have been applied to both models' surface.

Each road prefab is divided into 3 equal parts. The third part will be used for obstacle and collectible generation that the player character will have to dodge or pick up. The first two parts will be used as rest and preparation so the player can be prepared to use the movements to dodge the obstacles.

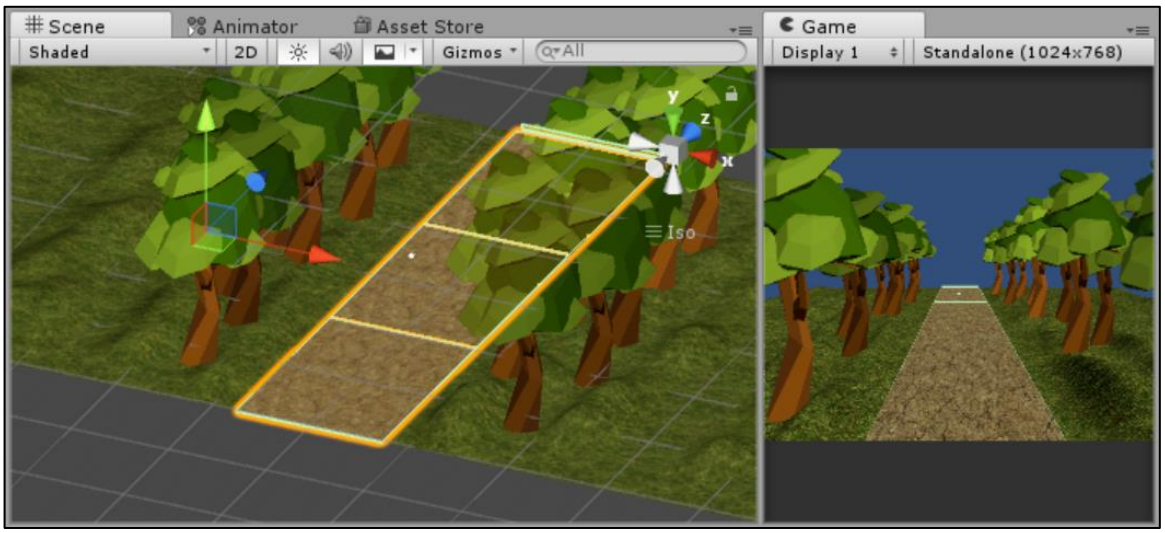

Figure 6.4. Road model.

The player character lateral movement will be limited to not exceed the limits of both terrains. The central road will have 3 lanes, that is to say, the player can do at most two movements in a row before hitting the limits.

6.5.2. Road generation and destruction

To achieve the infinite game sensation and approach characteristic of an endless runner, roads must be continuously generated.

At the initial scene, there will be only one road prefab generated, but upon starting the game there will be five instantiated and created one after another in the Z axis. As seen in the following figure, some road prefabs have been already added after the start point (the white rectangle that will contain the user interface) and some have been deleted.

There will be only 5 instances of road prefabs at a time through all the game. This allows the computer to use less graphic power and ensures that the game will not be wasting computational power in elements that are not visible from the player point of view even if the user turns his head to the opposite direction the player character is running.

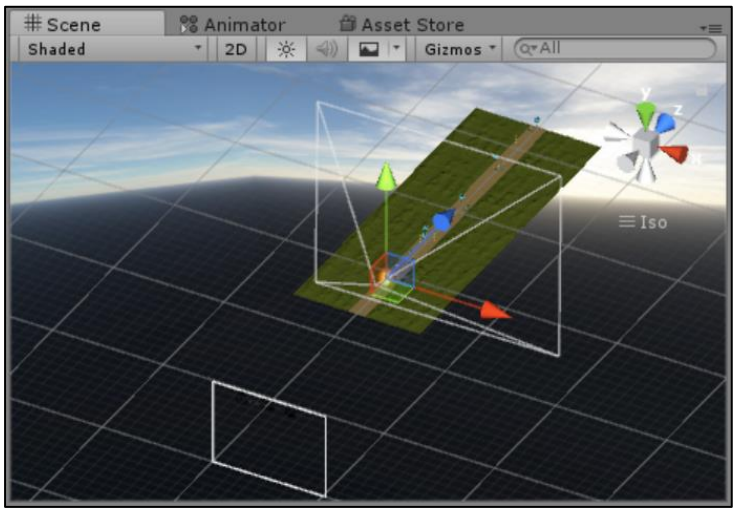

Figure 6.5. Road generation.

There will be an empty Game Object at the end of each road prefab, so when a collision with the player character with it is detected, the previous road that lies behind the player will be destroyed and the next road will be generated at the end of the screen. The effect and the irregular surface at the terrains around the central road allow these transitions to be certainly smooth.

6.5.3. Obstacles and collectibles

The player character will have to move around the obstacles to avoid losing life and get the collectibles to gain score. For the obstacles, there has been chosen a rock 3D model and a tree 3D model, and they were added to the project as a prefab to generate them into the central road when it is needed. The coins were made changing the scale transform of a basic cylinder and adding a texture from an image. The coins are packed in groups of three.

Figure 6.6. Game obstacles and coins.

6.5.4. Random generation

At the final part of each road prefab, there has been added 6 Transform GameObject. This Game Objects are empty and only contain information about their position, rotation and scale. This will be used as the fixed positions to generate obstacles and collectibles.

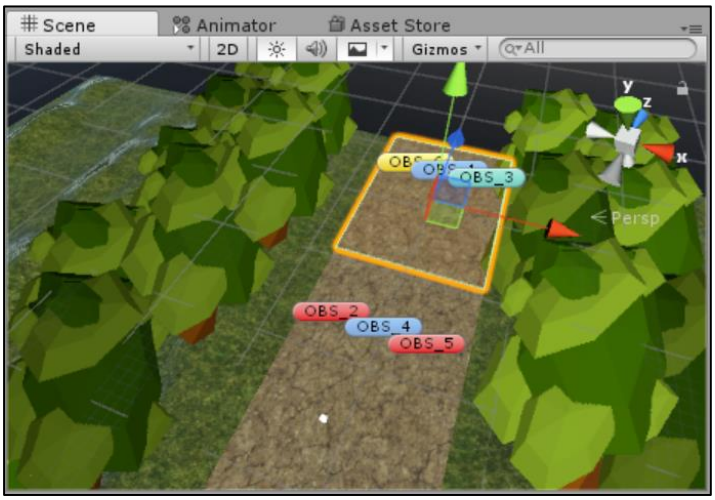

Figure 6.7. Item generating locations.

Using scripting random generation, these obstacles will be added randomly to any of the three lanes, so the game keeps unpredictable. The options for generation are adding an obstacle to the road (a rock or a tree), a group of coins, or adding nothing. The game has been designed so extremely difficult or impossible situations, like having obstacles blocking all the lanes, are rarely to happen.

6.6. GameObject Interactions

6.6.1. Collision Detection

A Rigidbody is the main component that enables physical behaviour for a GameObject. With a Rigidbody attached, the object will immediately respond to gravity. If one or more Collider components are also added, the GameObject is moved by incoming collisions. There are some cases where you might want a GameObject to have a Rigidbody without having its motion controlled by the physics engine.

In the case of the character player for this game, it will be moved directly from script code but still it is intended to be detected by collider triggers. For this reason, the player character Rigidbody will have the kinematic motion option selected in it.

Collider components define the shape of an object for the purposes of physical collisions. Colliders are invisible and do not need to be the exact size and form of the object they are associated to. One can use simpler colliders for computer optimization purposes. With careful positioning and sizing, colliders can often approximate the shape of an object quite well while keeping a low processor overhead.

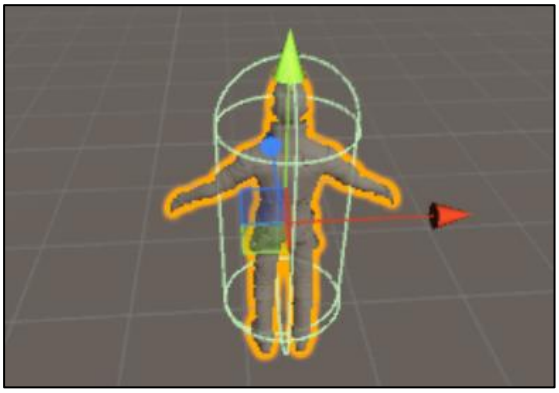

Figure 6.8. Capsule collider for third person character.

The simplest colliders are the primitive collider types. In 3D, these are the Box Collider, Sphere Collider and Capsule Collider. In this project, Box Colliders will be mostly used.

Colliders can be added to an object without a Rigidbody component to create non-motion elements on the scene. These are referred to as static colliders. Static colliders can interact with dynamic colliders (colliders attached to GameObjects with a Rigidbody on them) but since they do not have a Rigidbody, they will not move in response to collisions.

The scripting system can detect when collisions occur. A collider can also be configured as a trigger (using the "Is trigger" property) so it does not behave as a solid object and will simply allow other colliders to pass through. When a collider enters its space, it can be also detected on the trigger's Object scripts.

Colliders for obstacles and collectables in the game will be static, since once those are generated they will hold a fixed position on the road. They will also be configurated as

Triggers, and they will not generate any physics movements or collision if the player character runs through them. An invisible object containing a collider will be placed at the end of each road prefab, so it can be accomplished the road generation and destruction explained below when the player character passes through it.

Figure 6.9. In-game colliders view.

6.6.2. Obstacles

Once upon every game option is configurated, the player starts the game with 3 lifes. If the player gets hit by an obstacle and is not able to move its way out of it, he will lose a life. Once the player has lost all lifes, the player character will stop and the game will end and show a final screen.

6.6.3. Collectibles

The player can also collect coins to gain more points at the end of the game. This generates another reason for the player to do the movements. The coins will rotate and have a shiny aspect to generate more attention on the game screen.

6.6.4. Stats and Spawner Game Object

The majority of GameObjects will be infinitely generated and destroyed over the game. It was necessary to create two empty GameObjects at the start of the game. This serve as script holders for scripts needed to execute through all the game.

The stats manager generates the statistics for the game. It counts the time played, the coins collected, the player's life remaining, and the movements done to the left and to the right. It is used or accessed for some other GameObjects or Interfaces while the game is active.

The Spawner holds the script which allow the spawning and destruction for the game road. The position has been calculated so one terrain and road does not overlap with the previous or next one.

6.6.5. Player Character

The player character is a 3D animated model of a child. Upon the game starts, the player character will move followed by the camera, in a forward constant direction, at a speed that can be previously selected at the game menu. When the player has hit the necessary objects, the forward direction movement will stop.

The player left and right movements will be made by changing its position to the left or to the right in just one movement detection or keyboard hit. The amount of space depends on the central road, and has been adjusted to be a succession of slow position changes to allow a fast and almost instant transition. The player cannot go out of the bounds of the central road when it is at any of both left or right limits.

The player moves to the right or left when detects the arrow key hit on the keyboard or the movement of the right or left extremities from any other position to the desired final calibrated position set at the Start Menu. For that, it has been used the Transform (Rotation and Position) of Controller (Left) and Controller (Right), available with the SteamVR plugin. This can detect the position in real time at a room scale.

Figure 6.10. HTC Vive controllers model.

The 3D models of the controllers will be hidden since all the information needed is their Transform (Position, Rotation and Scale). There will be set a region around the final position of the movement to ensure that it is detected correctly when a minimum move is done. This region has been set to ± 22 cm.

Referring to interaction with other objects, when the player hits an obstacle, it will lose a life and continue running trespassing through the obstacles. When the player loses all three lifes, it will stop at the final obstacle hit. When the player collects coins, the coins will disappear and the number of coins collected will be added to the points statistics. The aim of the game is last all the time or score possible, so the player keeps motivated to train more or without paying attention to it with these little challenges.

The player has some built-in animations that have been configurated and used to add more flow to its movements.

6.6.6. Animations

Unity supports importing animation from external sources, and offers the ability to create animation clips from scratch within the editor using the Animation window. The player character will use three humanoid animation: running, idle, and waving hand.

For this purpose, an Animator Controller have to be attached to the Game Object. It allows to arrange and maintain a set of animations for a character or other animated Object.

The controller has references to the animation clips used within it, and manages the various animation states and the transitions between them using a State Machine, which could be thought of as a kind of flow-chart. The Animator allows to include any animations created or download to that specific character and to program the transitions and conditions between these animations.

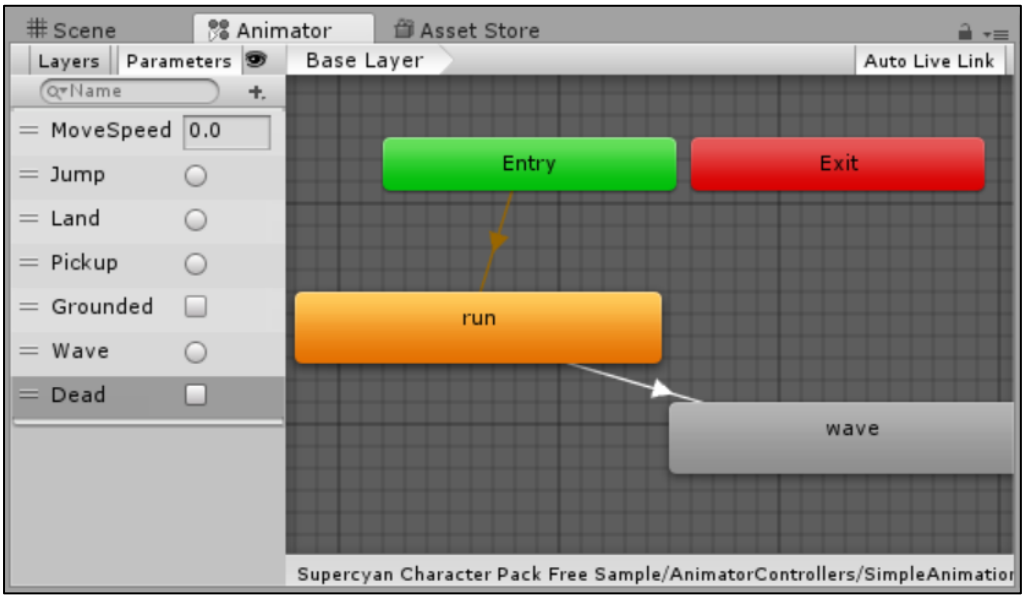

Figure 6.11. Animator view.

The player character, when the game starts will use the run animation in a loop as it is running forward. When the player loses, and has to remain motionless, instead of a dead animation that may not be friendly for all players of the game, it has been added a wave animation, that will show the user that the game is over. After that, the character will remain still until a new game is started.

6.7. User Interfaces

Unity UI (User Interface) system allows to create user interfaces fast and intuitively. The Canvas is the area that contains all UI elements. The Canvas is a GameObject with a Canvas component on it, and all UI elements must be children of such a Canvas.

The Canvas area is shown as a rectangle in the Scene View. This makes it easy to position UI elements without needing to have the Game View visible at all times. Canvas uses the Event System object to help the Messaging System.

The representation system chose for this project is to associate the Canvas as a child of the Main Camera. The UI elements are rendered by this camera, which means that the Camera settings affect the appearance of the UI.

Canvas can include components such as Text, Image, Button, Toggle, Slider, Input Field, Scroll, etc.

6.7.1. Start Menu

The Start Menu UI Canvas is all that will be shown in the Menu scene. It has an image of a forest in the background.

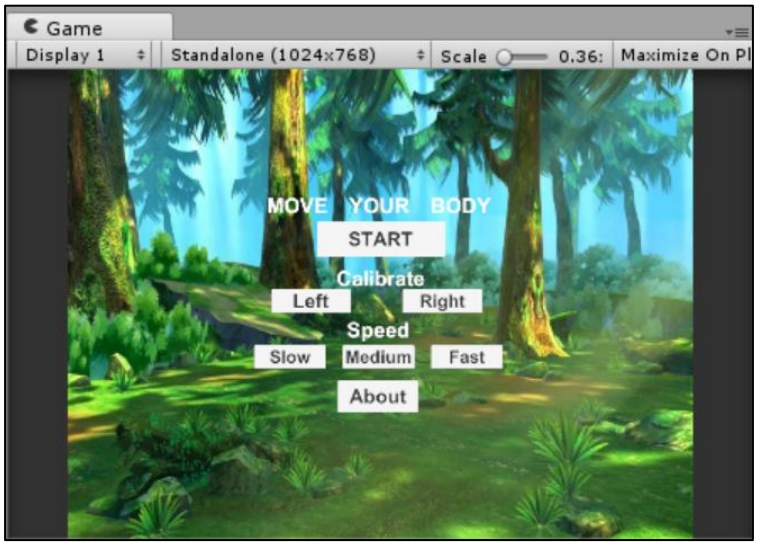

Figure 6.12. Start Menu view.

The different buttons allow to select different game functions, such as calibrating the final position from the virtual reality controllers in the left and right upper extremities, selecting the speed of the game between three distinct levels, and clicking on an about tab to display more information about the game. When the start button is hit, the game jumps to the game scene.

6.7.2. Countdown UI

When the Start button has been hit, and before the game starts, there will be a UI displaying a countdown from 3 to 0, waiting one second for each number shown in the Canvas text.

This allows the player to prepare and get himself in a correct position before the player character starts running. This timer also helps to generate the road instance after the player in a better way.

Figure 6.12. Game start countdown.

6.7.3. In-game UI

An in-game time UI has also been created so the player can have real time information on how his run is being done. This interface shows with icons and text the time played, the coins collected and the lifes remaining or hits with obstacles that the player can still take. This is so it does not interfere with the road and obstacles view.

All the UI is displayed in black colour and is shown in the following figure.

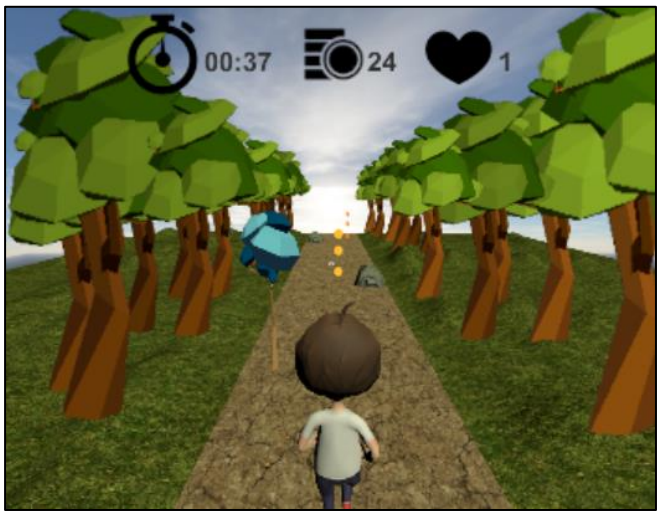

Figure 6.13. On top, view of the in-game UI.

6.8. Results UI

When the game is finished, the player character will stop and all the statistics that have been kept through the game will show up in a final interface. This final interface is made of an object called panel, which allows the interface to cover in a full or semi translucid way all the scene it has below. The statistics displayed are the time lasted in the game and the coins collected for player information and improvement purposes, and the correct movements that have been done to the left and to the right for analysis purposes.

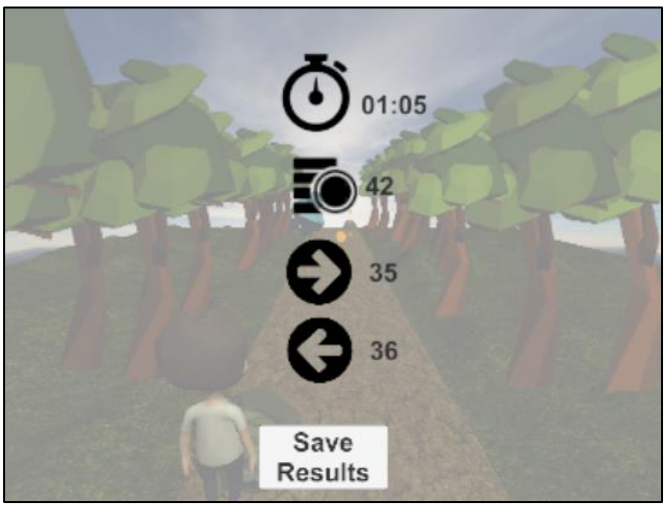

Figure 6.14. End results UI.

6.9. Data Handling and Acquisition

Data handling, acquisition and processing or analysing is a powerful tool for medical purposes, because it allows to search for patterns or analyse the results of any type of treatment or medication using real data which has been collected.

As an example of exporting data with Unity, for the standalone demo game developed, when the Save Results button is hit, the game will save the current hour and day and the statistics data collected by the stats manager into a .csv file. It is a file which can be opened which Excel to handle more easily these statistics in case they need to be copied or treated. The file path generated is always the same and will be: C:\MoveYourBody\Saved Data\Game Stats.csv. After that, the game will return to the Start Menu so the user can leave or start another game play if the player wishes it.

In the case of this game, useful data which could be information on the position and rotation of each extremity over time. This could get data like the number of movements the player could do in a game play, the average speed of each movement or the progression of the rotation angle between the extremities and the floor in the case of some movements. This is left to rehabilitation therapist criteria but could be implemented to the application analysis database in a near future.

6.10. Analysis Database

The game created in this project has been developed as a rehabilitation exercise intended to be included into a bigger application that includes an analysis database. To access it, the first menu will be a login screen to ensure only specific users can use the application and their progress will be registered. Once the user has entered the correct username and password, it will access the application. It will have 4 main characteristics:

- **New Gameplay:** This is the main characteristic used when the patients are present to perform the rehabilitation exercise. It allows to start a new gameplay or immersive experience and select the patient who is doing it from the patient database. Activities included refer to upper extremities exercises, seating exercises, walking and balance

improvements, cognitive exercises, memory training, occupational therapy applications, etc. These are aimed to physiotherapy, gaining mobility or neurorehabilitation treatment processes.

Once the activity and patient have been chosen, in the case of the game developed in this project, it will appear a menu which will allow to first configure and after start playing the game.

- **Gameplay Data Manager:** Allows to see the gameplay and studies previously done within the application. Once a new gameplay is started, the first data which is saved is the date and time and therapist name, to achieve greater tracking capabilities for each study. When the user enters in this menu, the data from the last gameplays is shown. The date, patient, therapist and gameplay results, such as punctuation, gameplay time, or other useful data for therapy, are displayed. A particular gameplay data can be deleted if it is considered not interesting, but after that it will not be recoverable.

- **Patients Manager:** It shows useful patients information, and allows to delete or add patients to the patient database according to the needs of the hospital or professional therapists. Data like name, surname, birth date, genre, treatment centre, and therapist name is saved for each patient.

- **Historical Records:** Each patient in the database has a historical record of all gameplays he has done, and its associated data and punctuations. This is so the results of the rehabilitation can be consulted over time, sometimes displaying it in a graphic way to achieve a better visualization.

7. Advantages and Disadvantages, Social Impact

Virtual reality is a technological revolution. As such it offers multiple advantages and some inconvenience for society. Every group of people or society has to decide how to regulate it. Although a lot of applications are starting to develop good impressions, its use can also have some risks.

In the health sector and as it has been shown in this project, it helps providing improvements to brain and mental health recovery thanks to immersive environments, improving the quality of life of the patients and their families. In physical recovery or medical therapies, it has also had an impact on making the therapies more appealing or creating new tools for the medical professionals.

Areas like science and technology education, tourism, leisure, simulation, communication and social networks have a brilliant potential to evolve due to the virtual reality. Virtual reality allows to simulate real or fictitious environments giving us the feeling of being there. This feature has an immense potential in different areas, such as for example:

- Psychological or educational treatments, virtual reality has been used for treating fear control, anorexia and bulimia, child education, etc.
- Video games environments that will allow us to move with total freedom over the world. This also applies to any type of interactive narrative, as the same principle applies to a film in virtual reality.
- Learn a job from home, as well as any type of practical training, without the need to move to the place in question (whether by being away or being dangerous, for example).
- Physical meeting places (simulated) with friends and family who live in another part of the world, thanks to the feeling of presence that develops within the virtual reality.
- Telework physical, through the use of haptic device and automated systems.
- Virtual art installations, exploiting the lack of certain physical limitations that allows an environment created by computer.

To sum up, virtual reality is not only for videogames but more of a multipurpose tool, so as our knowledge advances, new applications will be developed with this technology.

However, the long-term effects of virtual reality are still unknown. There have been cases of when the experience of VR can make someone feel bad, either with a headache, dizziness, blurred vision, or a combination of all these.

It is believed that many side effects are only temporary, but studies of long-term research are still scarce, so it does not know for sure. VR manufacturers are aware of this and they put warnings in the documentation of their products. **[33]**

For example, the documentation of the health and safety of the Oculus Rift lists a possibility of the following symptoms:

- Seizures
- Loss of consciousness
- visual fatigue
- Cramps on eye or muscle
- Involuntary movements
- Blurred or double vision or other vision abnormalities
- Dizziness
- Disorientation
- Balance Disorders
- Hand-eye coordination deteriorated
- Excessive sweating
- increased salivation
- Nausea
- Discomfort or pain in the head or eyes
- Drowsiness
- Fatigue

There is also evidence that the VR can change the thinking and daily behaviour, due in part to the fact that can be extremely realistic. There is a problem with desensitization, the person is no longer affected by extreme acts of violence, crime or military emergencies in real life and fails to show empathy or compassion as a result. In some situations, they actively seek out this type of scenario for the adrenaline rush and sense of power. **[34]**

Many people, when adopting a virtual avatar, can get to love more their fictional life than his own life, giving a tendency towards the replacement of their own life, in exchange for what he would have achieved in real life but never achieved. It can also cause health problems, such as obesity, the oxidation of the bones, etc. because if people gets too hooked to virtual reality, can tend to be more isolated and rely even more on the virtual world. This can become a vicious circle problem.

The user needs to use virtual reality devices in an isolated and safe space, because it separates your senses of sight and hearing to create more immersive experience. Some people express their view as if people spends a lot of hours per day with these devices, social autism could increase.

In conclusion, virtual reality and its associated technologies have a lot of positive benefits for people entertainment and training, but nowadays technology needs to improve to solve some of the negative aspects for health that it causes. Also, people have to take care of not to use it excessively.

Conclusions

During this project, it has been shown that virtual reality technology, even though the actual one will have more important developments near future, is helpful in a great quantity of fields and for helping to improve society's quality of life, specifically using concrete applications in the health care sector, such as neurorehabilitation.

The view of using virtual reality just for videogames or entertainment has been dismissed. Its fields of application can be very different: military, healthcare, fashion, education, engineering, telecommunications, films, sports or scientific visual. These fields can work together having synergies thanks to this technology, like in the game developed where entertainment and health therapies are combined.

It can be said that the objectives proposed for this project have been accomplished. The analysis of different alternatives and the development of a functional demo game application has been correctly completed. The gameplay results and interfaces achieved are considered satisfactory.

It has to be noted that a considerable number of hours has been dedicated to learning the application development environment. Unity is a powerful tool to develop videogames and applications, and this project can serve as an example or base for other users to learn the workflow of this software and to create their own projects or add functionalities, options or extras to this one.

Although the application is functional and useful, some drawbacks are present. For accomplishing a more complete product, the game should have been tested in neurorehabilitation patients to consider real test circumstances. In the future, the application could be improved with more options, like more personalised exercises for each articulation, analysis of the historical user performance data or real-time data handling, like rotation or acceleration parameters. These may be determined once the application has been used in real life training situations for some months.

It is an accurate guess to say that in the future, virtual reality applications will be used for many purposes in society, in which they can be beneficiation.

Financial Analysis

This chapter covers the budget needed to successfully complete the current project from the economical perspective. Expenses are divided in three different sections: material and software, human and indirect costs. Depreciations of the elements shown are computed considering that the project lasted 8 months.

Material and Software Costs

The material and software costs of this project are IT oriented. In order to develop the project, a computer with peripherical and HTC Vive system was needed. No other machinery was needed. The computer was used for all tasks. Some software licenses were also included.

Product	Price	Estimated Lifetime	Time used	Cost
Laptop	1200 €	4 years	8 months	200 €
HTC Vive	700 €	3 years	8 months	156 €
Unity3D (Student)	Free	1 year	8 months	Free
Microsoft Visual Studio IDE	Free	1 year	8 months	Free
Microsoft Office	100 €	1 year	8 months	67 €
Microsoft Windows 10	135 €	4 years	8 months	23 €

Human Resources Costs

This project was executed by one Junior Developer or Software Engineer. It has been calculated an approximated cost of 30 € per hour prior taxes. A lot of time was dedicated to formation and documentation about virtual reality, the computer software and

programming languages used. For future projects, the human resources costs could be decreased.

Task	Hours	Cost per Hour	Cost
General Planning	15	30 € / hour	450 €
Brainstorming	10	30 € / hour	300 €
Information research	35	30 € / hour	1050 €
Implementation tool study	25	30 € / hour	750 €
Programming language and tutorials learning	110	30 € / hour	3300 €
Application Design	55	30 € / hour	1650 €
Application Implementation	230	30 € / hour	6900 €
Project Report	100	30 € / hour	3000 €

Indirect Costs

We also need to consider some indirect cost which come from the use of the previous resources.

Component	Use	Cost	Cost
Electricity	0,22 Kw * 570 hours	0,11517 € / kWh	15 €
Internet connection	8 months	30 € / month	240 €
Lab and spaces rent	8 months	300 € / month	2400 €

Total Costs

Using data from the previous sections we can obtain all costs within the project. An additional of around a 3% of the budget has been added for emergency or not planned expenses.

Component	Cost
Material and Software	446 €
Human Resources	17400 €
Indirect	2655 €
Unexpected expenses	611 €
PROJECT TOTAL	**21112 €**

The total cost of the project is 21112 (twenty-one thousand, one hundred and twelve) euros.

Bibliography and References

References

[1] Perna, Robert B. 2014. "Neurorehabilitation: Evolving Practice". *International Journal Of Neurorehabilitation* 01 (01).

[2] "Activity By Aetiology | Institut Guttmann". 2017. *Institut Guttman - Neurorehabilitation Hospital. http://www.guttmann.com/en/activity-aetiology.*

[3] "IC Clínica De Fisioterapia | Fisioterapia Neurológica". 2017. *Icfisioterapia.Com.* http://www.icfisioterapia.com/fisioterapia_neurologica.html.

[4] "Stroke Treatment | Stroke Rehab | Brain Exercise | Neuro Style". 2017. Neuro Style. http://neuro-style.com/nbetter-stroke-rehabilitation-system/.

[5] Accelerated Care Plus Corporation. 2017. "Virtual Rehabilitation". Acplus.Com. http://acplus.com/programs/Pages/Virtual-Rehabilitation.aspx.

[6] Timocco, What. 2017. "Fun Therapy Games For Kids With Special Needs | Timocco". Timocco.Com. https://www.timocco.com/.

[7] Frederick Joseph F. 2017. "Defining Serious Games – Leadership & Flow". *Flowleadership.Org.* http://flowleadership.org/serious-games/.

[8] Jané, Carmen. 2017. "Realidad Virtual Para La Rehabilitación Neurológica". *Elperiodico.* http://www.elperiodico.com/es/sociedad/20160925/realidad-virtual-para-la-rehabilitacion-neurologica-5408914.

[9] Fulk, George D. 2005. "Locomotor Training And Virtual Reality-Based Balance Training For An Individual With Multiple Sclerosis". Journal Of Neurologic Physical Therapy 29 (1): 34-42. doi:10.1097/01.npt.0000282260.59078.e4.

[10] Adamovich, Sergei V., Gerard G. Fluet, Eugene Tunik, and Alma S. Merians. 2009. "Sensorimotor Training In Virtual Reality: A Review". Neurorehabilitation 25 (29–44). doi:10.3233/NRE-2009-0497.

[11] Lee, Suhyun, Yumi Kim, and Byoung-Hee Lee. 2016. "Effect Of Virtual Reality-Based Bilateral Upper Extremity Training On Upper Extremity Function After Stroke: A Randomized Controlled Clinical Trial". Occupational Therapy International 23 (4): 357-368. doi:10.1002/oti.1437.

[12] "Virtual Reality For Motor Rehabilitation Of The Shoulder | UC3M". 2016. Uc3m.Es. https://www.uc3m.es/ss/Satellite/UC3MInstitucional/en/Detalle/Comunicacion_C/1438738168390/1371216052710/Virtual_Reality_for_Motor_Rehabilitation_of_the_Shoulder.

[13] "What Is Virtual Reality? - Virtual Reality Society". 2017. *Virtual Reality Society*. https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html.

[14] "Types Of VR System". 2017. *Agocg.Ac.Uk*. http://www.agocg.ac.uk/reports/virtual/37/chapter2.htm.

[15] "Cave Automatic Virtual Environment". 2017. *En.Wikipedia.Org*. https://en.wikipedia.org/wiki/Cave_automatic_virtual_environment.

[16] Reality, Virtual. 2017. "LG's VR HMD Shows High-End Promise, May Offer Mixed Reality". *Tom's Hardware.* http://www.tomshardware.com/news/lg-vr-hmd-prototype-gdc,33792.html.

[17] "Virtual Reality You Can Feel". 2017. *Axonvr*. https://axonvr.com/#haptx-haptics-evolved.

[18] "Virtuix Omni First Of Its Kind Active Virtual Reality Motion Platform". 2017. *Virtuix.Com*. http://www.virtuix.com/.

[19] Perry, York, and York Perry. 2017. "El Mareo Al Usar Dispositivos De Realidad Virtual Ya Tiene Cura - Fayerwayer". *Fayerwayer*. https://www.fayerwayer.com/2016/06/el-mareo-al-usar-dispositivos-de-realidad-virtual-ya-tiene-solucion/.

[20] "Google Cardboard – Google VR". 2017. *Vr.Google.Com*. https://vr.google.com/cardboard/.

[21] "Playstation®VR". 2017. *Playstation*. https://www.playstation.com/es-es/explore/playstation-vr/.

[22] "Oculus". 2017. *Step into Rift.* https://www.oculus.com/.

[23] "VIVE™ | Buy Vive Hardware". 2017. *Vive VR Product*. https://www.vive.com/eu/product/.

[24] "The Source For Tech Buying Advice". 2017. *Techradar.* http://www.techradar.com/.

[25] Buckley, Sean. 2017. "This Is How Valve'S Amazing Lighthouse Tracking Technology Works". *Gizmodo.Com*. http://gizmodo.com/this-is-how-valve-s-amazing-lighthouse-tracking-technol-1705356768.

[26] Malventano, Allyn. 2016. "Steamvr HTC Vive In-Depth - Lighthouse Tracking System Dissected And Explored | The Steamvr Lighthouse Tracking System". *Pcper.Com*. https://www.pcper.com/reviews/General-Tech/SteamVR-HTC-Vive-depth-Lighthouse-Tracking-System-Dissected-and-Explored/SteamV.

[27] "Gamemaker | Yoyo Games". 2017. *Yoyo Games*. https://www.yoyogames.com/gamemaker.

[28] "Stencyl: Make Iphone, Ipad, Android & Flash Games Without Code". 2017. *Stencyl.Com*. http://www.stencyl.com/.

[29] "Game Engine Technology By Unreal". 2017. *Unrealengine.Com*. https://www.unrealengine.com/en-US/what-is-unreal-engine-4.

[30] "Unity - Game Engine". 2017. *Unity*. https://unity3d.com/es.

[31] Technologies, Unity. 2017. "Unity - Manual: Manual De Unity". *Docs.Unity3d.Com*. https://docs.unity3d.com/es/current/Manual/index.html.

[32] "Screenshot Subway Surfers". 2017. *Gametop.Com*. http://www.gametop.com/download-free/subway-surfers/screenshot0.html.

[33] Davis, Nicola. 2017. "Long-Term Effects Of Virtual Reality Use Need More Research, Say *Scientists". The Guardian. https://www.theguardian.com/technology/2016/mar/19/long-erm-effects-of-virtual-reality-use-need-more-research-say-scientists.*

[34] "Virtual Reality And Ethical Issues - Virtual Reality Society". 2017. *Virtual Reality Society*. https://www.vrs.org.uk/virtual-reality/ethical-issues.html.

Complementary Bibliography

Krucoff, Max O., Shervin Rahimpour, Marc W. Slutzky, V. Reggie Edgerton, and Dennis A. Turner. 2016. "Enhancing Nervous System Recovery Through Neurobiologics, Neural Interface Training, And Neurorehabilitation". *Frontiers In Neuroscience* 10.

Arnab, Sylvester, Ian Dunwell, and Kurt Debattista. 2013. *Serious Games For Healthcare*. Hershey, PA: Medical Information Science Reference.

"Toda La Actualidad Sobre La Realidad Virtual Y Aumentada". 2017. *Realovirtual.Com*. https://www.realovirtual.com/.

Technologies, Unity. 2017. "Unity - Manual: Unity User Manual (2017.1)". *Docs.Unity3d.Com*. https://docs.unity3d.com/Manual/index.html.

M. LaValle, Steven. 2016. *Virtual Reality*. EBook. 1st ed. Cambridge University Press. http://vr.cs.uiuc.edu/vrbook.pdf.

Annexes

A1. User Manual

A guide for users using the application will be detailed next.

Once the user opens the Move Your Body application (.exe in the PC), the Start Menu shows up. It has the following buttons to configure the options:

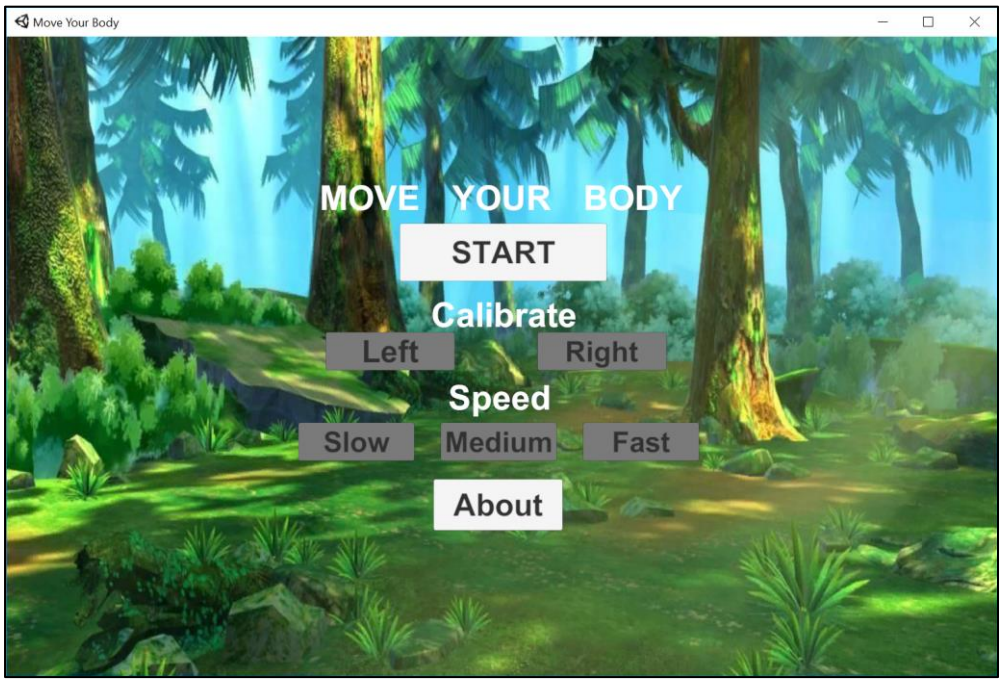

Figure A1.1. Start Menu view

- **Calibrate:** Saves the position and rotation values of the final position of the training movement that will allow the player to move the character later in the game. There is a button for each extremity. Once the position is saved correctly, the button turns into green colour.

Figure A1.2. Calibrate buttons

- **Speed:** Allows the user to select between three speed levels to decrease or increase the difficulty of the game. Only one value can be selected at the same time.

Figure A1.3. Speed select buttons

- **About:** Shows basic information of the creation of game, useful in case the user downloaded the project from another source or wants to know more about it.
- **Start:** Starts the game for the player. The game will start after a 3 to 0 countdown to allow the player to prepare.

The following figure shows the game view. The player is expected to do movements from the left or the right to dodge all obstacles and collect the coins, while keeping the character in the road. The game also displays to the player three real time statistics, from left to right: Time, Coins and Lives.

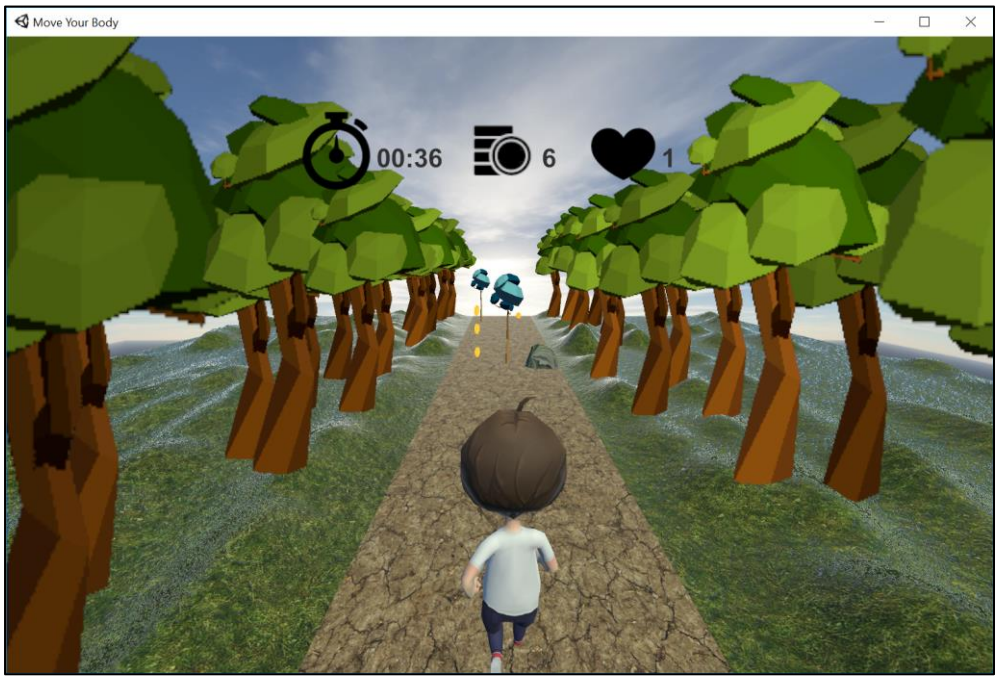

Figure A1.4. In game view.

The player is expected to get to the final position that was calibrated for each move needed to do. Once the move has been done, the player has to get near the initial position again in order to be allowed to do another move with that specific extremity.

The movements of the player character to the right will be achieved when a movement with the right upper extremity is correctly done. The movement to the left will be the same but in the left extremity.

Figure A1.5. Example of a left movement during a game test.

When the player has been hit by obstacles three times and loses all life, the game finishes, making the character stop and showing a screen with the end results. These end results are: played time, coins collected, left movements and right movements.

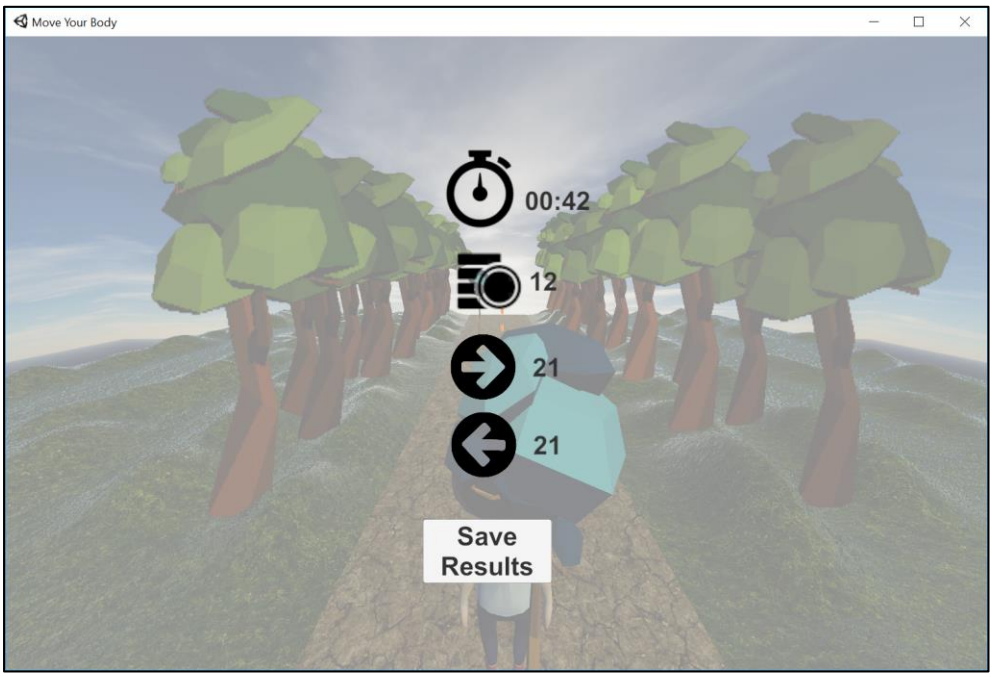

Figure A1.6. Game finished view.

Hitting the Save Results button will save these final results into a .csv file which can be opened in any spreadsheet editor like Excel, in order to manipulate this data more efficiently.

D2 | 6

	A	B	C	D	E	F	G
1	Date	Game Time	Coins	Movements Left	Movements Right		
2	9/14/2017 8:16:46 PM	0:46	6	6	6		
3							
4							
5							
6							
7							

Figure A1.7. Saved game stats.

When the Save Results button is pressed, the application will return to the Start Menu.

A2. Application Scripts

The scripts used in the application development will be copied here, with a little explanation of how they work.

about_script Shows the information image contained in the tab About in the starting Menu.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class about_script : MonoBehaviour {
    public GameObject about_image;
    public GameObject boton;
    // Use this for initialization
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(About);
    }

       // Update is called once per frame
       void Update () {

       }
    void About()
    {
        about_image.SetActive(true);
    }
}
```

calibrate_left Gets the position and rotation from the left controller of the VR system when the button on the start Menu is hit.

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

public class calibrate_left : MonoBehaviour {
    public GameObject boton;
    public GameObject izquierda;
    public static float x_pos_left;
    public static float y_pos_left;
    public static float z_pos_left;
    public static float x_rot_left;
    public static float y_rot_left;
    public static float z_rot_left;
    // Use this for initialization
```

```
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(Calibrate);
        x_pos_left = 1;
        y_pos_left = 1;
        z_pos_left = 1;
        x_rot_left = 1;
        y_rot_left = 1;
        z_rot_left = 1;
        GetComponent<Image>().color = Color.grey;
    }

    // Update is called once per frame
    void Calibrate()
    {
        x_pos_left = izquierda.transform.position.x;
        y_pos_left = izquierda.transform.position.y;
        z_pos_left = izquierda.transform.position.z;
        x_rot_left = izquierda.transform.rotation.x;
        y_rot_left = izquierda.transform.rotation.y;
        z_rot_left = izquierda.transform.rotation.z;

        GetComponent<Image>().color = Color.green;
    }

}
```

calibrate_right Gets the position and rotation from the right controller of the VR system when the button on the start Menu is hit.

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

public class calibrate_right : MonoBehaviour {
    public GameObject boton;
    public GameObject derecha;
    public static float x_pos_right;
    public static float y_pos_right;
    public static float z_pos_right;
    public static float x_rot_right;
    public static float y_rot_right;
    public static float z_rot_right;
    // Use this for initialization
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(Calibrate);
        x_pos_right = 1;
        y_pos_right = 1;
        z_pos_right = 1;
        x_rot_right = 1;
```

```
        y_rot_right = 1;
        z_rot_right = 1;
        GetComponent<Image>().color = Color.grey;
    }

       // Update is called once per frame
       void Update () {

       }
    void Calibrate()
    {
        x_pos_right = derecha.transform.position.x;
        y_pos_right = derecha.transform.position.y;
        z_pos_right = derecha.transform.position.z;
        x_rot_right = derecha.transform.rotation.x;
        y_rot_right = derecha.transform.rotation.y;
        z_rot_right = derecha.transform.rotation.z;

        GetComponent<Image>().color = Color.green;
        //Debug.Log(x_pos_right+"   "+ y_pos_right+"   "+ z_pos_right);
    }


}
```

camera_move Moves the camera to view the player character model through all the gameplay.

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

public class camera_move : MonoBehaviour {
    public stats_manager stats;
    public countdown goes;
       // Use this for initialization
       void Start () {

       }

       // Update is called once per frame
       void Update () {
        if(stats.life>0 && goes.go==true)transform.position += new Vector3(0, 0,
5*speed_select.speed); //forward movement

    }

}
```

coins_script Rotates the coin collectables and does actions when the player character model collides with them.

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

public class coin_script : MonoBehaviour {
    private GameObject stats;

    public stats_manager score_ball;
       // Use this for initialization
       void Start () {
        stats = GameObject.Find("stats_game_object");
        score_ball = stats.GetComponent<stats_manager>();
    }

       // Update is called once per frame
       void Update () {
        transform.Rotate(Vector3.back * Time.deltaTime*50);
    }
    void OnTriggerEnter(Collider other)
    {
        //Debug.Log(Time.time);
        score_ball.coins += 1;
        Debug.Log("hola");
        if (other.tag == "Player") Destroy(gameObject);

    }


}
```

countdown Generates a countdown in an interface from 3 to 0 before starting the game.

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

public class countdown : MonoBehaviour {
    public GameObject UI;
    public bool go = false;
    private int countr = 3;
    public float counting;
    public GameObject text;
    private Text counter;
       // Use this for initialization
       void Start () {
        counter = text.GetComponent<Text>();
       }
```

```
        // Update is called once per frame
        void Update () {
         counting += Time.deltaTime;
         if(counting>=1)
         {
             counting = 0;
             --countr;
             counter.text = countr.ToString();
         }
         if(countr<=0)
         {
             counter.text = "";
             UI.SetActive(true);
             go = true;

         }
        }

}
```

death Destroys game road when the player has gone past it.

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

public class death : MonoBehaviour {

    // Use this for initialization
    void Start () {

       }

       // Update is called once per frame
       void Update () {

       }

    void OnTriggerEnter(Collider other)
    {
        if(other.tag=="Player")
        {
            Destroy(gameObject.transform.parent.gameObject);
        }



    }


}
```

disappear_about Hides the information image contained in the tab About in the starting Menu.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class disappear_about : MonoBehaviour {
    public GameObject boton;
    // Use this for initialization
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(Click);
    }

       // Update is called once per frame
       void Click() {
        gameObject.SetActive(false);
       }

}
```

movement Moves the player character in a forward fixed speed selected at the start menu. Considers road bounds and input data from keyboard or controllers to move. Movements to the let or to the right of the player character will coincide with movement detection of each extremity.

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

public class movement : MonoBehaviour {
    public GameObject canvasUI;
    public GameObject canvasEnd;
    public countdown goes;
    private Animator animation;
    public GameObject stat_count;
    private stats_manager stats;
    public GameObject izquierdo;
    public GameObject derecho;

    private float contador=50;
    private float bounds = 50;
    private bool moveright = true;
    private bool moveleft = true;

    private float positionlimit = 0.22f;
    private float rotationlimit = 5f;
    private bool movedetectleft1 = false;
```

```
    private bool movedetectleft2 = false;

    private bool movedetectright1 = false;
    private bool movedetectright2 = false;

    // Use this for initialization
    void Start () {

    animation = gameObject.GetComponent<Animator>();
    stats = stat_count.GetComponent<stats_manager>();

    }

       // Update is called once per frame
       void Update () {

        //Debug.Log(calibrate_right.x_pos_right + "   " +
calibrate_right.y_pos_right + "   " + calibrate_right.z_pos_right);
        //Debug.Log(derecho.transform.position.x + "   " +
derecho.transform.position.y + "   " + derecho.transform.position.z);
        //Debug.Log(movedetectright1 + "   " + movedetectright2);

        if (
        ((izquierdo.transform.position.x > (calibrate_left.x_pos_left +
positionlimit)) || (izquierdo.transform.position.x < (calibrate_left.x_pos_left
- positionlimit))) ||
        ((izquierdo.transform.position.y > (calibrate_left.y_pos_left +
positionlimit)) || (izquierdo.transform.position.y < (calibrate_left.y_pos_left
- positionlimit))) ||
        ((izquierdo.transform.position.z > (calibrate_left.z_pos_left +
positionlimit)) || (izquierdo.transform.position.z < (calibrate_left.z_pos_left
- positionlimit))))

        {
            movedetectleft1 = true;
        }

        if ((movedetectleft1) &&
        (((izquierdo.transform.position.x < (calibrate_left.x_pos_left +
positionlimit)) && (izquierdo.transform.position.x > (calibrate_left.x_pos_left
- positionlimit))) &&
        ((izquierdo.transform.position.y < (calibrate_left.y_pos_left +
positionlimit)) && (izquierdo.transform.position.y > (calibrate_left.y_pos_left
- positionlimit))) &&
        ((izquierdo.transform.position.z < (calibrate_left.z_pos_left +
positionlimit)) && (izquierdo.transform.position.z > (calibrate_left.z_pos_left
- positionlimit)))))
        {
            movedetectleft1 = false;
            movedetectleft2 = true;
        }

        if (
```

```
        ((derecho.transform.position.x > (calibrate_right.x_pos_right +
positionlimit)) || (derecho.transform.position.x < (calibrate_right.x_pos_right
- positionlimit))) ||
        ((derecho.transform.position.y > (calibrate_right.y_pos_right +
positionlimit)) || (derecho.transform.position.y < (calibrate_right.y_pos_right
- positionlimit))) ||
        ((derecho.transform.position.z > (calibrate_right.z_pos_right +
positionlimit)) || (derecho.transform.position.z < (calibrate_right.z_pos_right
- positionlimit))))
        {
            movedetectright1 = true;
        }

        if ((movedetectright1) &&
        (((derecho.transform.position.x < (calibrate_right.x_pos_right +
positionlimit)) && (derecho.transform.position.x > (calibrate_right.x_pos_right
- positionlimit))) &&
        ((derecho.transform.position.y < (calibrate_right.y_pos_right +
positionlimit)) && (derecho.transform.position.y > (calibrate_right.y_pos_right
- positionlimit))) &&
        ((derecho.transform.position.z < (calibrate_right.z_pos_right +
positionlimit)) && (derecho.transform.position.z > (calibrate_right.z_pos_right
- positionlimit)))))
        {
            movedetectright1 = false;
            movedetectright2 = true;
        }


        if (goes.go == true)
        {
            if (stats.life > 0) transform.position += new Vector3(0, 0,
5*speed_select.speed); //forward movement
            else
            {
                animation.SetBool("Dead", true);
                canvasUI.SetActive(false);
                canvasEnd.SetActive(true);
            }
            if (moveright)
            {
                if (Input.GetKeyDown(KeyCode.RightArrow) || movedetectright2)
                {
                    ++stats.right;
                    movedetectright2 = false;
                    MoveRight();
                }
            }
            if (moveleft)
            {
                if (Input.GetKeyDown(KeyCode.LeftArrow) || movedetectleft2)

                {
                    ++stats.left;
```

```
                movedetectleft2 = false;
                MoveLeft();
            }

        }

        if (movedetectright2)
        {
            movedetectright2 = false;
        }

        if (movedetectleft2)
        {
            movedetectleft2 = false;
        }

        //Debug.Log("aixo es la meva posicio"+transform.position);
    }

}

void MoveRight()
{

    while(contador>0)
    {
        contador -= Time.deltaTime;
        transform.position += Vector3.right * Time.deltaTime;
    }
    contador = 50;

if (Mathf.Abs(transform.position.x + 5f) > bounds)
     {
        moveright = false;
        moveleft = true;
    }
    else
    {
        moveright = true;
        moveleft = true;
    }
}
void MoveLeft()
{

    while (contador > 0)
    {
        contador -= Time.deltaTime;
        transform.position -= Vector3.right * Time.deltaTime;
    }
    contador = 50;
      if (Mathf.Abs(transform.position.x ) + 5f > bounds)
     {
        moveleft = false;
```

```
            moveright = true;
        }
        else
        {
            moveright = true;
            moveleft = true;
        }
    }
    void OnTriggerEnter(Collider other)
    {
        if(other.tag=="Rock" || other.tag =="Tree")
        {
            --stats.life;
        }

    }



}
```

play Loads the game scene from the menu.

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

public class play : MonoBehaviour {
    public GameObject boton;
    // Use this for initialization
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(Click);
    }

       // Update is called once per frame
       void Click () {

        SceneManager.LoadScene("ballmove");
    }

}
```

save_results Saves registered game stats and returns to the start menu.

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

```
using UnityEngine.SceneManagement;
using UnityEngine.UI;
public class save_results : MonoBehaviour {
    public stats_manager result;
    public show_time time;
    public GameObject boton;
    public
    // Use this for initialization
    void Start () {
        Button btn = boton.GetComponent<Button>();
        btn.onClick.AddListener(results);
    }

       // Update is called once per frame
        void results()
    {
        System.IO.File.WriteAllText("C:/MoveYourBody/Saved Data/Game Stats.csv",
"Date" + ";" + "Game Time" + ";" + "Coins" + ";" + "Movements Left" + ";" +
"Movements Right" + "\n" +
                                    System.DateTime.Now.ToString() + ";" +
time.final_time.text + ";" + result.coins_final.text + ";"
+result.total_left.text + ";" +result.total_right.text);
        SceneManager.LoadScene("menu");
    }

}
```

show_coins Converts the coins statistic into text for player interfaces.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class show_coins : MonoBehaviour {
    public stats_manager stats;
    public Text time;
    private float aaaa;
    // Use this for initialization
    void Start () {

}

       // Update is called once per frame
       void Update () {
        aaaa = Mathf.Round(stats.coins);
        time.text =aaaa.ToString();
    }

}
```

show_heart Converts the lifes statistic into text for player interfaces.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class show_heart : MonoBehaviour {
    public stats_manager stats;
    public Text time;
    private float aaaa;
    // Use this for initialization
    void Start () {

       }

       // Update is called once per frame
       void Update () {
        aaaa = Mathf.Round(stats.life);
        time.text = aaaa.ToString();
    }

}
```

show_time Converts the time statistic into text for player interfaces.

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

public class show_time : MonoBehaviour {
    public Text final_time;
    public stats_manager stats;
    public Text time;
    private float aaaa;
    // Use this for initialization
    void Start () {

       }

       // Update is called once per frame
       void Update () {

            aaaa = Mathf.Round(stats.seconds);
            time.text = stats.minutes + ":" + aaaa.ToString();

            if (stats.minutes < 10 && stats.seconds < 10)
            {
                time.text = "0" + stats.minutes + ":" + "0" + aaaa.ToString();
            }
            if (stats.minutes < 10 && stats.seconds >= 10)
            {
```

```
                time.text = "0" + stats.minutes + ":" + aaaa.ToString();
            }
            if (stats.minutes >= 10 && stats.seconds < 10)
            {
                time.text = "0" + stats.minutes + ":" + aaaa.ToString();
            }
        final_time.text = time.text;

    }

}
```

spawn_blocks Generates random prefab obstacles and coins in the game in predetermined locations on the game road.

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

public class spawn_blocks : MonoBehaviour {
    public stats_manager stats;
    public List<GameObject> tag_targets = new List<GameObject>();
    public GameObject rock;
    public GameObject tree;
    public GameObject coins_hori;
    public GameObject coins_verti;
    private int random_value_1;
    private int random_value_2;
    private int random_val;
    // Use this for initialization
    void Start () {
        foreach (Transform child in transform)
        {

                tag_targets.Add(child.gameObject);

        }

        spawn();

    }

       // Update is called once per frame

    void spawn()
    {
        random_value_1 = (int)Random.Range(0, tag_targets.Count-1);
        GameObject rock_spawn = tag_targets[random_value_1];

        GameObject rocka = Instantiate(rock, new
Vector3(rock_spawn.transform.position.x, rock_spawn.transform.position.y,
rock_spawn.transform.position.z), Quaternion.identity) ;
```

```
        rocka.transform.parent = gameObject.transform;
        tag_targets.RemoveAt(random_value_1);
        random_value_1= (int)Random.Range(0, tag_targets.Count - 1);
        GameObject tree_spawn= tag_targets[random_value_1];

        GameObject tree_inst= Instantiate(tree, new
Vector3(tree_spawn.transform.position.x, tree_spawn.transform.position.y,
tree_spawn.transform.position.z), Quaternion.identity);
        tree_inst.transform.parent = gameObject.transform;
        tag_targets.RemoveAt(random_value_1);
        random_value_1 = (int)Random.Range(0, tag_targets.Count - 1);
        GameObject coin_hori= tag_targets[random_value_1];

        GameObject coins_hor_inst =Instantiate(coins_hori, new
Vector3(coin_hori.transform.position.x, coin_hori.transform.position.y,
coin_hori.transform.position.z), Quaternion.identity);
        coins_hor_inst.transform.parent = gameObject.transform;
        tag_targets.RemoveAt(random_value_1);
        random_value_1 = (int)Random.Range(0, tag_targets.Count - 1);

    }



}
```

spawner Generates road in front of the player character and destroys the road behind him to ensure low memory and PC resources usage.

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

public class spawner : MonoBehaviour {
    public GameObject forest_chunk;
    public GameObject player;
    public float count=0.5f;
    private float chunk_posi;
    private int generate = 0;
    private float next = 1;
    private float chunk_next = 3772f;
    // Use this for initialization
    void Start () {
        chunk_posi = forest_chunk.transform.position.z;

    }

       // Update is called once per frame
       void Update () {
        if (count <= 0)
        {
            count = 0.5f;
```

```
            spawning();
        }
        else
        {
            count-=Time.deltaTime;
        }


    }
    void spawning()
    {
        if (generate < 5)
        {
            chunk_posi += +613f;
            Instantiate(forest_chunk, new
Vector3(forest_chunk.transform.position.x, forest_chunk.transform.position.y,
chunk_posi), Quaternion.identity);
            generate++;
        }
        if ((player.transform.position.z > 613f*next) && (generate>=5))
        {
            Instantiate(forest_chunk, new
Vector3(forest_chunk.transform.position.x, forest_chunk.transform.position.y,
chunk_next), Quaternion.identity);
            chunk_next += +613f;
            next++;
        }


    }

}
```

speed_select Selects the game speed using three buttons on the start menu.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class speed_select : MonoBehaviour {
    public GameObject slow;
    public GameObject normal;
    public GameObject fast;
    private Image slow_speed_spr;
    private Image normal_speed_spr;
    private Image fast_speed_spr;
    private Button slow_speed_but;
    private Button normal_speed_but;
    private Button fast_speed_but;
    public static float speed;
    // Use this for initialization
    void Start () {
```

```
        speed = 0.7f;
        slow_speed_spr = slow.GetComponent<Image>();
        normal_speed_spr = normal.GetComponent<Image>();
        fast_speed_spr = fast.GetComponent<Image>();
        slow_speed_but = slow.GetComponent<Button>();
        normal_speed_but = normal.GetComponent<Button>();
        fast_speed_but = fast.GetComponent<Button>();
        slow_speed_spr.color = Color.grey;
        normal_speed_spr.color = Color.grey;
        fast_speed_spr.color = Color.grey;
        slow_speed_but.onClick.AddListener(slow_green);
        normal_speed_but.onClick.AddListener(normal_green);
        fast_speed_but.onClick.AddListener(fast_green);
    }

       // Update is called once per frame
       void slow_green ()
    {
        slow_speed_spr.color = Color.green;
        normal_speed_spr.color = Color.grey;
        fast_speed_spr.color = Color.grey;
        speed = 0.45f;

    }
    void normal_green()
    {
        slow_speed_spr.color = Color.grey;
        normal_speed_spr.color = Color.green;
        fast_speed_spr.color = Color.grey;
        speed = 0.7f;
    }
    void fast_green()
    {
        slow_speed_spr.color = Color.grey;
        normal_speed_spr.color = Color.grey;
        fast_speed_spr.color = Color.green;
        speed = 0.95f;
    }

}
```

stats_manager Handles statistics through the game and shows them in the final screen when the game is finished.

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

public class stats_manager : MonoBehaviour {
    public GameObject canvasUI;
    public GameObject canvasfinal;
```

```
    public Text total_right;
    public Text total_left;
    public Text coins_final;
    public countdown contador;
    public int right;
    public int left;
    public int coins;
    public float secandos;
    public int seconds;
    public int minutes;
    public int life;
       // Use this for initialization
       void Start () {
        left= 0;
        right = 0;
        coins = 0;
        seconds = 0;
        minutes = 0;
        life = 3;
    }

       // Update is called once per frame
       void Update () {
        total_left.text = left.ToString() ;
        total_right.text = right.ToString();

        if (contador.go && life>=0)
        {

            secandos += Time.deltaTime;
            if (secandos >= 1)
            {
                ++seconds;
                secandos = 0;
            }

            if (seconds >= 60)
            {
                seconds = 0;
                minutes += 1;
            }
            //Debug.Log(life);
        }
        if(life<=0)
        {
            total_left.text = left.ToString();
            total_right.text = right.ToString();
            coins_final.text = coins.ToString();
            canvasfinal.SetActive(true);
            GameObject.Destroy(canvasUI);
        }

    }
}
```