Universidad
Carlos III de Madrid

Bachelor Thesis

Implementation of Open Source applications "Serious Game" for Rehabilitation

Author

Jaime Herreros Díaz

Tutor

Alberto Jardón Huete

October 2016

TABLE OF CONTENTS

INDEX OF FIGURES

INDEX OF TABLES

Abstract

Serious Games and Virtual Reality (VR) are present nowadays as an alternative to traditional rehabilitation therapies. This project describes the workflow to develop videogames for health monitoring as well as a source of entertainment for physiotherapy patients, primarily patients that suffer hemiparesis caused by a neurological disease like a stroke. We propose the last version of Microsoft Kinect sensors as low cost game controller and the software Unity to develop Open Source Rehabilitation Serious Games. These Serious Games try to imitate physiotherapy sessions performed in movement recovery therapies, reducing the waiting list of patients together with time and costs to hospitals. The premise is that the gameplay makes patients execute upper body exercises alongside equilibrium training, meanwhile they are monitored extracting useful data and results for the physicians.

Key words: Serious Games, Hemiparesis patients, Kinect, Unity, Open Source.

1 Introduction

1.1 Background

The world of technology is growing faster since the entrance of 21st century and, alongside the technology, the field of videogames.

This evolution is obvious if we compare the videogames played at 80’s, controlled using gaming machines installed at recreation halls, with the ones used today, consoles equipped with a complex hardware and complementary devices which let users play at home only switching on the console.

Is collectively known that videogames have the purpose of amuse their users. In spite of this, videogames are evolving with the objective of acquire a more relevant and responsible role in the modern society. [1]

Videogame’s world is reaching this role with the apparition of a recent game’s category less well known renowned as “Serious Games”. Serious games can be defined as digital games serving serious purposes like training, education, research or healthcare [2]. The implementation of these Serious Games in high-ranking areas, how is healthcare, could mean a quality step for training processes, with lower costs and great results obtained with enjoyable methods that let the user acquire specific useful skills only clicking the “play” button.

This project focusses on the creation of a Serious Game for healthcare purposes, creating a distracted way to carry out hemiparesis rehabilitation treatments for patients that have suffered a neurological disease, like a stroke, at the same time that a tracking of the patient evolution is performed without the constant presence of rehabilitation specialists.

1.2 Motivation

Each day videogame’s world progresses offering a broad gamut of devices including different types of cameras and sensors. The motivation of this project is to design, using these devices, a low cost and effective alternative to present commercial Serious Games, which actually are sold to the hospitals. To achieve this, low cost hardware devices and free Open Source Game Engines and scripts have been used to minimize the prize.

Once the videogame’s hardware is selected, an interface between the hardware and the videogame itself is necessary, this interface is the “Game Engine”. Game Engines

are toolkits aimed to ease the development of videogames, acting as a superstructure of several development efforts. This project set out and combine a list of hardware devices, Game Engines, and rehabilitation exercises that could fulfil our requirements for the creation of Serious Games.

1.3 Objectives

As previously mentioned, the main objective of this study is to choose and use a series of software tools and hardware devices in order to develop a hemiparesis rehabilitation Serious Game, which reduce the workload of physiotherapists and allow patients to have fun during the rehabilitation process. In order to achieve this, the program must fulfil the following aspects:

1. **Acquire patient’s position and movement**

   Track the patient is the most important issue when it’s time to do a diagnosis or evaluation. Include this recording in a Serious Game could be a useful help for physicians in order to follow closely their patient evolution.

2. **Integrate patients in a comfortable virtual environment**

   The mood of the patients is an essential factor at any rehabilitation therapy. To influence their interest, this Serious Game has to move the patient to other environment different to the hospital’s one. They have to forget that they are doing an unpleasant rehabilitation while they are playing with this Serious Game.

3. **Motivate the patients to achieve rehabilitation objectives**

   The self-improvement and motivation is the key in a rehabilitation process. In order to do that the videogame have to integrate a type of reward system which motivates the patient when he is playing with the purpose of surpass his previous performance score.

4. **Store and retrieve patient’s results**

   All the results obtained during the game session have to be stored in a suitable format, easily accessible for the physicians that must check continuously the patient’s performance.

5. **Use an Open Source software**

   The principal advantage of develop an Open Source application is the possibility of be manipulated and enhanced by any user. An Open Source app lets to modify the principal codes and structure of the software with the purpose of change and enhance it, and how it can be manipulated by any user that has access to the app, is easier to give different perspectives to this application.

1.4 Structure

The dissertation starts clarifying what are the target users that will use the Serious Games developed in this project (chapter 2). These users are mostly stroke survivors, so to go in depth about the Stroke, the thesis continue with its definition and classification, causes, risk factors and finally, showing some examples of post-stroke rehabilitation evaluation (2.1). Next is exposed the integration of these type of rehabilitation therapies with Virtual Reality (2.2). In chapter 3 the state of art of videogame’s technology is analyzed, including Game Engines (3.1), creation and modelling of 3D objects (3.2), and Motion Capture devices (3.3).

Next chapters assess the materials and methods used in the development of Serious Games and how are Serious Games themselves. In chapter 4 are described the hardware (4.1) and the software (4.2) used for the project fulfilment. Straightaway, in chapter 5 is described how the developed Serious Games operate; how they help to patients (5.1), the design basis for Serious Games development (5.2) and the description of the Serious Games created (5.3).

The thesis ends with the conclusions and future work extracted from conclusions analysis (chapter 6).

2 Target users

This Serious Game project tries to tackle rehabilitation processes dedicated to people that have suffered a neurological disease. One of the physical sequels caused by this kind of diseases is the hemiparesis, which is a unilateral (“hemi”) weakness (“paresis”) of the entire left or right side of the body. To treat this complaint physiotherapists force their patients to use the weakened part of their body.

Hemiparesis affects about 8 out of 10 stroke survivors so for that reason, along this dissertation we will focus in people that have been afflicted by a stroke, especially to patients with reduced mobility and coordination.

2.1 Stroke

In this section we are going to review some important issues of this disease that affects to the locomotor system, and its rehabilitation processes which can be handle with Serious Games.

2.1.1 Definition and classification

A stroke, also called cerebrovascular accident (CVA), occurs when blood flow is interrupted to part of the brain. Without blood to supply oxygen and nutrients, brain cells of the affected area quickly begin to die**.** Depending on the region of the brain affected, a stroke may cause paralysis, speech impairment, loss of memory, coma or death. [3]

Stroke is classified on the basis of its aetiology as either ischaemic (87%) or haemorrhagic (13%). **Ischaemic stroke** is produced by occlusion of a cerebral artery [thrombotic or atherosclerotic (50%), embolic (25%) and microartery occlusion, “lacunar stroke”, (25%)], meanwhile **haemorrhagic stroke** is caused mainly by spontaneous rupture of blood vessels or aneurysms. [4]

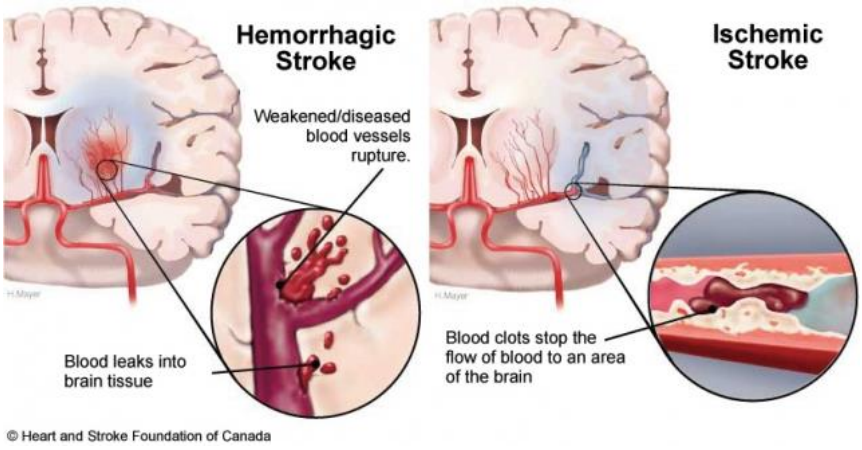

Figure 2.1: Stroke's Classification

In the next subsections we are going to resume some important features about this neurological disease according to [3], with the purpose of know better how stroke can affect us and how we can deal with it.

2.1.2 Causes

Cerebral thrombosis occurs when a blood clot, or thrombus, forms within the brain itself, blocking the flow of blood through the affected vessel. Clots most often form due to "hardening" (atherosclerosis) of brain arteries. Cerebral thrombosis is often preceded by a transient ischemic attack, or TIA, sometimes called a "mini-stroke." In a TIA, blood flow is temporarily interrupted, causing short-lived stroke-like symptoms. Recognizing the occurrence of a TIA, and seeking immediate treatment, is an important step in stroke prevention.

Cerebral embolism occurs when a blood clot from elsewhere in the circulatory system breaks free. If it becomes lodged in an artery supplying the brain, either in the brain or in the neck, it can cause an ischemic stroke. The most common cause of cerebral embolism is atrial fibrillation, a disorder of the heartbeat. In atrial fibrillation, the upper chambers (atria) of the heart beat weakly and rapidly, instead of slowly and steadily, and blood within the atria is not completely emptied. This stagnant blood may form clots within the atria, which can then break off and enter the circulation.

Haemorrhage, or bleeding, occurs when a blood vessel breaks, either from trauma or excess internal pressure. The vessels most likely to break are those with pre-existing defects such as an aneurysm, which is a "pouching out" of a blood vessel caused by a weak arterial wall. Intracerebral haemorrhage affects vessels within the brain itself, while subarachnoid haemorrhage affects arteries at the brain's surface, just below the protective arachnoid membrane.

2.1.3 Risk factors

- **Age and sex**. The risk of stroke increases with age, doubling for each decade after age 55. Men are more likely tohave a stroke than women.
- **Heredity**. Blacks, Asians, and Hispanics have higher rates of stroke than do whites. People with a family history of stroke are at greater risk.
- **Diseases**. Stroke risk is increased for people with diabetes, heart disease (especially atrial fibrillation), high blood pressure, prior stroke, or TIA.
- **Other medical conditions**. Stroke risk increases with obesity, high blood cholesterol level, or high red blood cell count.
- **Lifestyle choices**. Stroke risk increases with cigarette smoking (especially if combined with the use of oral contraceptives), low level of physical activity, alcohol consumption above two drinks per day, or use of cocaine or intravenous drugs.

2.1.4 Treatment

Treating a stroke may involve drugs, surgery, or other therapies. Treatments for stroke vary depending on whether the stroke is caused by a blood clot (**ischemic stroke**) or a brain bleed (**haemorrhagic stroke**). Not matter the type of stroke, acting fast and seeking treatment as quickly as possible is key to reducing the risk of permanent brain damage. [5]

2.1.4.1 Treatment for Ischemic Stroke

With this type of stroke, the goal is to restore blood flow to the brain as quickly as possible. A number of medications may be given at the hospital to help break up the clot and prevent the formation of new clots. These medications may include:

- **Tissue plasminogen activator (tPA, alteplase):** Alteplase or tPA is a thrombolytic medication, often referred to as a "clot buster." They will quickly break up or dissolve blood clots that are blocking blood flow to the brain.
- **Aspirin:** Aspirin won't dissolve existing blood clots, but it will help to prevent new clots from forming. Doctors may give aspirin within 48 hours of the start of stroke symptoms.
- **Anticoagulants:** Anticoagulants, such as heparin, may be used to help prevent more blood clots from forming.

2.1.4.2 Treatment for Haemorrhagic Stroke

Treatment for haemorrhagic stroke will depend on the cause of the bleeding and what part of the brain is affected. Bleeding around the brain is often caused by abnormally formed blood vessels, called aneurysms.

Non-surgical treatments for haemorrhagic stroke may include:

- Controlling blood pressure.
- Stopping any medications that could increase bleeding (e.g., aspirin).
- Blood transfusions with blood clotting factors to stop ongoing bleeding.
- Measuring pressure within the brain using a ventriculostomy.

Surgical treatments for haemorrhagic stroke may include:

- **Endovascular treatment:** A long tube is slid into a blood vessel in an arm or leg, and passed all the way up to the blood vessels in the brain, where a coil or clip is placed to prevent further bleeding.
- **Aneurysm treatment:** This may involve removing a small piece of the skull to locate the aneurysm and put a small clamp around it to stop the bleeding. An aneurysm may also be treated by placing a small tube or catheter into a blood vessel in the groin. The catheter is then guided through the blood vessel to the location of the aneurysm and a small coil is placed within the aneurysm to block blood flow and prevent it from rupturing again
- **Decompressive craniotomy:** If a patient's life is in danger, the doctor may consider opening the skull to remove blood and release pressure on the brain.

2.1.5 Rehabilitation

Rehabilitation is another part of treatment. It helps the person keep abilities and gain back lost abilities to become more independent and it usually begins while the patient is still in acute care. But for many patients, it continues afterward, either as formal rehabilitation program or as individual rehabilitation service. [6]

Some people do not need rehabilitation after a stroke because the stroke was mild or they have fully recovered. Others may be too disabled to participate. However, many patients can be helped by rehabilitation. There are several kinds of rehabilitation programs:

Hospital programs

These programs can be provided by special rehabilitation hospitals or by rehabilitation units in acute care hospitals. Complete rehabilitation services are available. The patient stays in the hospital during rehabilitation. An organized team of specially trained professionals provides the therapy. Hospital programs are usually more intense than other programs and require more effort from the patient.

Nursing facility (nursing home) programs

As in hospital programs, the person stays at the facility during rehabilitation. Nursing facility programs are very different from each other, so it is important to get specific information about each one. Some provide a complete range of rehabilitation services; others provide only limited services.

Outpatient programs

Outpatient programs allow a patient who lives at home to get a full range of services by visiting a hospital outpatient department, outpatient rehabilitation facility, or day hospital program.

Home-based programs

The patient can live at home and receive rehabilitation services from visiting professionals. An important advantage of home programs is that patients learn skills in the same place where they will use them.

2.1.6 Outcome Measures in Stroke Rehabilitation

In [7] Van der Putten et al. pointed out that measuring the outcome of health care is a “central component of determining therapeutic effectiveness and, therefore, the provision of evidence-based healthcare”. So following this, is really important to outcome measurements in stroke rehabilitation in order to check the effectiveness of stroke rehabilitation. Each of these measurements usually are evaluated in terms of appropriateness, reliability, validity, responsiveness, precision, interpretability, applicability and feasibility. [8]

Some important outcomes in stroke rehabilitation are the Fugl-Meyer Assessment of Motor Recovery after Stroke (FMA) and the Action Research Arm Test (ARAT).

2.1.6.1 Fugl-Meyer Assessment of Motor Recovery after Stroke

The Fugl-Meyer Assessment (FMA) is a disease-specific impairment index designed to assess motor function, balance, sensation qualities and joint function in hemiplegic post-stroke patients. The scale comprises five domains; motor function (in the upper and lower extremities), sensory function, balance (both standing and sitting), joint range of motion and joint pain.

Scale items are scored on the basis of ability to complete the item using a 3-point ordinal scale where 0=cannot perform, 1=performs partially and 2=performs fully. The total possible scale score is 226. Points are divided among the domains as follows: 100 for motor function (66 upper & 34 lower extremity), 24 for sensation (light touch and position sense), 14 points for balance (6 sitting & 8 standing), 44 for joint range of motion & 44 for joint pain.

It is not uncommon for the sections of the FMA to be administered separately. However, it should take approximately 30 – 45 minutes to administer the total FMA. Assessments are completed by direct observation on a one-to-one basis and should be performed by a trained physical therapist.

The Fugl-Meyer assessment is widely used and internationally accepted. The motor assessment is grounded in well-defined, observable stages of motor recovery. But on the other hand, though a trained therapist should be able to administer the test in approximately 30 - 45 minutes, it may take considerably longer, forcing therapists spend much time with only one patient. [8]

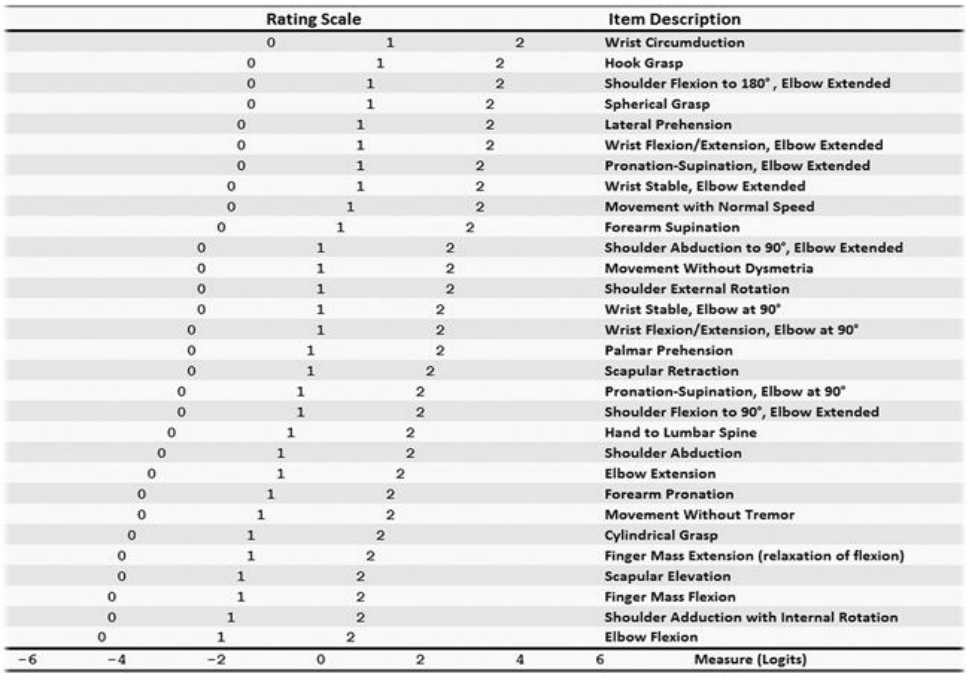

Figure 2.2: Example using Fugl-Meyer Assesment-Upper Extremity with post stroke patients [9]

2.1.6.2 Action Research Arm Test

The Action Research Arm Test (ARAT) is an observer-rated, performance-based assessment of upper extremity function and dexterity.
The ARAT comprise only 19 items, which are grouped into 4 subsets. These subsets include: grasp (6 items), grip (4 items), pinch (6 items) and gross movement (3 items). All items are rated on a 4-point ordinal scale ranging from 0 to 3 where 0 represents no movement possible and 3 represents normal performance of the task.

Within each subset, the first item is the most difficult and the second is the easiest. The remainder of the items are ordered by ascending difficulty. Successful completion of a particular task or item implies that subsequent, easier tasks can also be successfully completed. For each subset, the most difficult task is attempted first, and, if successful (i.e. 3 points awarded), full points for that subsection are awarded. If the item is not completed successfully (i.e. <3 points were awarded), the next (easiest) item is attempted. If the patient receives a score of 0 on the easiest item, no points are awarded for that subsection and no further items are attempted. If the patient receives a score greater than 0, all remaining items within the subset are assessed.

The ARAT is a relatively short and simple measure of upper limb function that provides assessment of a variety of tasks over a range of complexity.

The test covers most aspects of arm function, including proximal control and dexterity. But also has some limitations such as in more impaired individuals, testing time can extend to 20 minutes or more, test administration requires a fairly long list of materials, or in patients with severe impairments or near normal function, the scale may not be sensitive enough to detect changes in performance. [8]

Action Research Arm Test Activity	Score
Grasp 1. Block, wood, 10 cm cube (If score = 3, total = 18 and go to Grip) Pick up a 10 cm block 2. Block, wood, 2.5 cm cube (If score = 0, total = 0 and go to Grip) Pick up 2.5 cm block 3. Block, wood, 5 cm cube 4. Block, wood, 7.5 cm cube 5. Ball (Cricket), 7.5 cm diameter 6. Stone 10 x 2.5 x 1 cm Coefficient of reproducibility = 0.98 Coefficient of scalability = 0.94	
Grip 1. Pour water from glass to glass (If score = 3, total = 12, and go to Pinch) 2. Tube 2.25 cm (If score = 0, total = 0 and go to Pinch) 3. Tube 1 x 16 cm 4. Washer (3.5 cm diameter) over bolt Coefficient of reproducibility = 0.99 Coefficient of scalability = 0.98	
Pinch	
1. Ball bearing, 6 mm, 3rd finger and thumb (If score = 3, total = 18 and go to Grossmt) 2. Marble, 1.5 cm, index finger and thumb (If score = 0, total = 0 and go to Grossmt) 3. Ball bearing 2nd finger and thumb 4. Ball bearing 1st finger and thumb 5. Marble 3rd finger and thumb 6. Marble 2nd finger and thumb Coefficient of reproducibility = 0.99 Coefficient of scalability = 0.98	
Grossmt (Gross Movement) 1. Place hand behind head (If score = 3, total = 9 and finish) 2. (If score = 0, total = 0 and finish 3. Place hand on top of head 4. Hand to mouth Coefficient of reproducibility = 0.98 Coefficient of scalability = 0.97	

Table 2.1: Action Research Arm Test (ARAT) Evaluation [10]

2.2 Virtual Reality in rehabilitation

Standard rehabilitation (ie. physiotherapy and occupational therapy) helps improve motor function after stroke, but it presents some important limitations that are outlined in Table 2.2.

Time–consuming
Labor and resource intensive
Dependent on patient compliance
Limited availability depending on geography
Modest and delayed effects in some patients
Requires transportation to special facilities
Initially underappreciated benefits by stroke survivors
Requires costs/insurance coverage after the initial phase of treatment

Table 2.2: Limitations of conventional rehabilitation [11]

So for this reason, Virtual Reality (VR) is becoming a consistent alternative to standard rehabilitation, innovating in the different therapeutic areas and enhancing the rehabilitation process for the patients.

VR is a computer-based technology that allows users to interact with a multisensory simulated environment and receive "real-time" feedback on performance. VR exercise applications have the potential to apply relevant concepts of neuroplasticity (ie. repetition, intensity, and task-oriented training). Also classified as VR are a variety of non-immersive video game systems developed by the entertainment industry for home use, making this technology less costly and more accessible to clinicians and individuals. Several of these games have been adopted by clinicians as rehabilitation interventions although they have not been especially designed to meet rehabilitation goals. [11]

Several companies have seized the potential of VR in rehabilitation processes, launching different products and softwares designed to create a virtual and interactive environment that facilitates to patients their rehabilitation programs. Below are listed some companies that commercialize these VR rehabilitation's products for hospitals: [12] [13]

2.2.1 BRU™ (Balance Rehabilitation Unit™)

Medicaa™ is a company committed through the use of technology, to help and treat patients with balance disorders. The research team of Medicaa™ have developed the MBS™, a software platform that provides a set of tools to improve the results of rehabilitation in patients with balance disorders. With this purpose

in mind they have created the BRU™ (Balance Rehabilitation Unit™), an efficient solution for functional assessment and rehabilitation therapy for patients that suffer from balance disorders. The BRU™ uses the MBS™, as software that works by means of virtual reality stimuli recreating real life situations. The BRU™ comprises of three main modules:

- **Functional Assessment**

The Functional Assessment is carried out using the force platform. It is a technique used to diagnose which families of stimuli are more conflictive to the patient in the postural control strategies.

- **Therapeutic using Virtual Reality**

According to the clinical evaluation and Functional Assessment, a customized rehabilitation program is designed, which includes visual stimulation by means of Virtual Reality Headsets. This therapeutic module is based on neurorehabilitaion concepts, which confronts the patient to the conflictive stimuli to unleash mechanisms of neuroplasticity, as to achieve vestibular compensation.

Figure 2.3: Virtual Reality Headset used for BRU™

- **Motor Coordination Retraining™ - MCR™**

The MCR™ module consists of exercises for the training of the postural control using a biofeedback system. The MBS™ recreates controllable interactive exercises in which the patient trains postural strategies under professional supervision. The exercises have different recreational styles and difficulty levels to stimulate the patient’s balance improvement and transform rehabilitation process into an entertaining experience.

Figure 2.4: BRU™ workstation

2.2.2 BioTrak©

BioTrak© is a tool based on virtual reality technology which integrates in a single platform training exercises for the rehabilitation of certain body functions that have been depleted or lost due to various pathologies.

BioTrak©´s technology allows the patients to interact in a virtual environment where they are challenged to fulfil simple tasks by means of their own movements. The system motivates patients in order to improve their adherence to the treatment, and serves them as a very efficient tool in their recovery process. Actually BioTrak© works in exercises designed by physicians for the rehabilitation of musculoskeletal injuries, balance, anosognosia, memory, etc.
BioTrak© needs an Internet connection and a gamut of hardware devices used during the performance of the exercises/exergames, which are listed below:

- Laptop
- Tracking devices (Wii Balance Board, Kinect, Wireless Magnetic Tracker)
- TV connection
- Internet connection

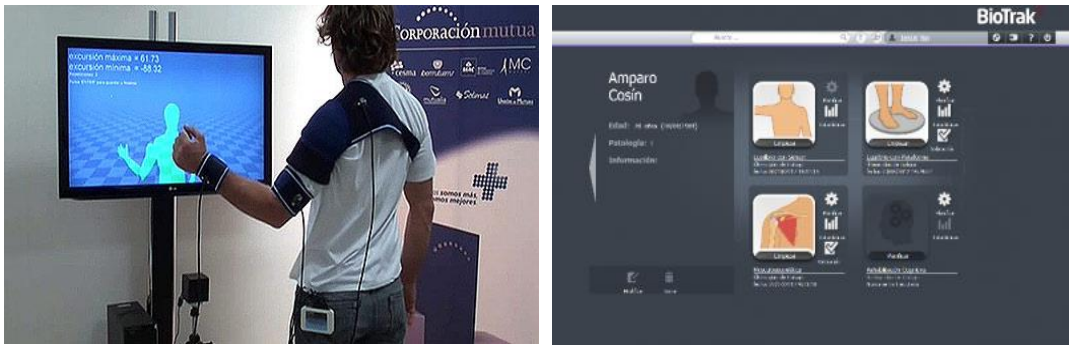

Figure 2.5: BioTrak© workstation and software interface

3 Technologies

In this chapter we will expound information about all the technologies thought-out and used during the development of this project, their differences and reasons of why are technologies that we can use to create our Serious Game.

First we will discuss about the called Game Engines, which let us design and create videogames for different platforms. After that, we will review some modelling and 3D animation softwares, their characteristics and which one fits better to our needs. Finally, at the end of the chapter, we are going to describe several hardware devices used in motion capture together with the plugins that allow us to connect these devices with our personal computer in order to test and design our Serious Game.

3.1 Game Engines

Game engines are software systems that enable game development and execution. These systems can provide features from communication with input and output devices to physics calculations and artificial intelligence. They are like toolkits aimed to provide an abstraction to developers making their project development easier by hiding complex concepts. Game Engines normally are packed with a set of powerful resources (usually described as libraries or APIs) and tools used in the design and in coding stages. [14]

A regular game engine provides different functions: scripting, imagery rendering, artificial intelligence, physics, animation, cinematic, network access and resource management. [15]

- **Scripting:** Let developers to write little pieces of code to control certain parts of the game.
- **Imagery rendering:** It’s the core of visual part of the game. Handles lights, shadows, ray tracing, and rendering of 3D objects.
- **Artificial Intelligence:** Brings the world and characters of the game to life, through a set of routines that makes possible the interaction with the game environment.
- **Animation:** Adds behaviour to objects, through transformations, skeletons, deformations and dynamics.
- **Physics:** Provides realistic interaction between objects and with their environment. Plus, character moment, fluid simulation, and “soft bodies”.
- **Cinematic:** Adds the possibility to include video within the game to capture the attention of the player.

- **Network access:** Provides support to deploy the game in a network environment, be client-server or peer-to-peer.
- **Resource management:** A fundamental issue for the game engines is the efficient use of the computer resources (CPU, graphics card, memory, storage, hardware) and the load of game related resources (animations, shaders, 3d objects, sound, etc.)

Game engines can be classified according to several criteria, being one of them the type of licensing: commercial and freeware or Open Source.

Nowadays exist several Game Engines that could provide all the functions mentioned above, for that reason is not an easy task to choose a Game Engine that better fits to our videogame’s idea. In the next subsections of this chapter we will describe some of the most used Game Engines in 2014 according to [16], and compare between them in order to decide which one is the best one for this project.

3.1.1 Unreal Engine 4

The Unreal Engine 4 (UE4) is the last version of the Game Engine’s family Unreal developed by Epic Games, and first showcased in the 1998 first-person shooter game Unreal. This Game Engine is the base for several known videogames as *Unreal Tournament, Tom Clancy's Rainbow Six: Vegas, Gears of War, BioShock* series*, Star Wars Republic Commando or Mass Effect*. [17] [18]

Figure 3.1: Unreal engine 4 logotype

With its code written in C++, the UE4 features a high degree of portability and is a tool used by many game developers today. It is compatible with different platforms such as:

- Microsoft Windows
- GNU/Linux

- Apple Macintosh (Mac OS)
- Xbox One
- PlayStation 4
- Android
- HTML5
- Oculus Rift

In addition to this, Unreal Engine offers different useful tools for designers helping their artistic work in the creation and visualization of environments.

Unreal Engine 4 is free to use, with a 5% royalty on gross product revenue after the first $3,000 per game. The Unreal engine technology is licensed to many notable entities in the fields of education, training simulation, construction simulation, virtual reality, and CG animation. Licensees include many universities, corporations, the U.S. Army, the U.S. Air Force, NASA, the Federal Bureau of Investigation (FBI), the Michigan Department of Transportation (MDOT), and the U.S Department of Homeland Security (DHS).

For developing with UE4, is recommended a desktop PC with Windows 7 64-bit or a Mac with Mac OS X 10.9.2 or later, 8 GB RAM and a quad-core Intel or AMD processor, and a DX11 compatible video card. UE4 will run on desktops and laptops below these recommendations, but performance may be limited.

Unreal Engine is one of the most important Game Engines that we can found in the Web nowadays, and this reputation begun in 2006 with the launch of the videogame Gears of Wars, created with the second version of this Game Engine. Gears of Wars received several nominations among which stand out “Game of the year 2007” and “Best game for Xbox 360”. So it is a Game Engine that was necessary to take into account its use in this project.

Figure 3.2: Image of the videogame Gears of War

3.1.2 Unity

Unity is a cross-platform game engine developed by Unity Technologies used to develop video games for PC, consoles, mobile devices and websites.

Figure 3.3: Unity Logotype

Unity Technologies was founded at 2004 by David Helgason (CEO), Nicholas Francis (CCO) and Joachim Ante (CTO) in Copenhague, Denmark, after the failure of their first game, *GooBall*. They recognize the potential and value of the Game Engine with its development tools, so for that reason they started to create an accessible engine for anyone.

So following this philosophy, Unity has reached a great success giving importance to the needs of indie game developers, who can’t create their own Game Engine or acquire licenses from other Game Engines. Unity tries to offer and to make accessible to everyone the development of interactive contents and videogames in 2D and 3D environments. [19]

Currently different reports demonstrate that Unity has a participation of 45% of the Game Engine market, being the most popular engine between game’s developers. The proportion of people trusting in Unity as development toolkit undergo in a constant growth. [20]

Figure 3.4: Game engine’s Global market participation [20]

Unity is used both by companies and independent developers. For the second ones this is a great tool, with a lot of functions to develop videogames, easy to use and totally free with the *Personal* license of Unity although with few limitations, but with great results. But Unity also offers other three licenses for companies and professional developers: [21]

- *Plus*: Offers more functions and reports in order to optimize the projects (35$/month)
- *Pro*: Advanced personalization and more flexibility and storage (125$/month)
- *Enterprise*: License offered taking into account the services that you and your company need (Negotiable price)

In March of 2015 the version 5 of Unity was launched, this version allow users develop games for the next platforms:

- Apple Macintosh (Mac OS)
- Windows
- Linux
- Web
- iOS
- Android
- Windows Phone 8
- Blackberry 10
- Playstation 3, 4, and VITA
- Xbox 360 and ONE
- Wii U
- Tizen

In addition to this, Unity is compatible with many design softwares like Blender, Maya, Adobe Fuse, Cheetah3D, etc. This Game Engine lets import objects from all these programs, such as we can use them as elements of the videogame’s environment.

To create and modify the source code of a videogame, we can use *MonoDevelop* or *Microsoft Visual Studio* as code developer tools. This codes, or scripts, can be written in three different programming languages: [15]

- C#, similar to C++, used in a great percentage of the created scripts.
- UnityScript, or JavaScript for Unity.
- Boo, with a syntax similar to Python.

The three of them can coexist with each other, and make us of .Net libraries for database access, regular expressions, XML, etc.

In November of 2010 was launched an important resource for Unity, the called *Asset Store.* The *Asset Store* is an online platform in which Unity users can access to thousands of packages that incorporates a wide collection of game objects and elements including: 3D models, textures, materials, music, sound effects, tutorial, projects, scripts, etc. Some of these assets can be downloaded totally free meanwhile others have to be bought with a price that depend on asset’s creator.

Summing up, Unity is a great Game Engine with several interesting functions for our Serious Game project, relatively easy to use and with the capacity to create Open Source applications, scripts and entire games that other people can modify and use using the *Asset Store*.

3.1.3 CryEngine

CryEngine is a Game Engine designed by the German game developer company *Crytek*. Originally was a demo Game Engine for *Nvidia* enterprise, showing a great potential, and for that reason is implemented for first time in the videogame *Far Cry*. In 2006, all the rights of CryEngine are acquired by the company *Ubisoft.*

Figure 3.5: CryEngine logotype

This Game Engine is one of the most complete and awarded engines available, providing top-notch functionality for the games developed with this technology. It’s considered a next generation solution for game development, able to use scalable computing technology. Is expected for version 3 to be free for outside development, but the licensing and business model has not been disclosure yet. CryEngine technology is built on top of a “sandbox” that permits real time adjustment of parameters and error correction. [22]

CryEngine has a very intuitive interface, which allows closely observing and controlling the event flow in a visual manner, largely avoiding the need for ground zero coding of the application. [15]

The programming languages used for the users to create games with this Game Engine are C++, C# and Lua. Although is a completely free software for non-commercial uses, CryEngine is now available as a pay what you want service, allowing users to set their own price for it, and obtaining access to different features:

- Full Engine Source Code
- Full commercialization for any games created
- 100% royalty-free
- Access to all supported platforms
- Primed for VR development
- Buy and sell assets on the Marketplace
- Access to Learning resources

Other important feature about CryEngine is it compatibility with a great part of videogame’s platforms used nowadays as:

- Windows
- Linux
- Playstation 3 & 4
- Wii U
- Xbox 360 & One
- iOS
- Android

All this flexibility in the code manipulation, access to tutorials and to the Marketplace, where you can obtain assets for your game, makes this Game Engine really interesting from the Open Source perspective. Finally in addition to this, its potential and easy to use interface makes this engine a serious candidate to be used in our project. [23]

3.2 Creation, modelling and animation of objects

As in many videogames, our Serious Game has to be composed of different objects interacting in a motivating environment for the patient. These graphic's components, or "gameobjects", are designed by 3D computer graphics softwares that let you create avatars, game environments, objects or anything required in a videogame.

Some of the Game Engines previously listed incorporates an online "market" where you can download, buy and sell gameobjects created by other companies or users, as could be the *Asset Store* in Unity or the *Marketplace* in CryEngine. The variety of objects, scenarios, and avatars are huge in these places, some of them free and others paying an extra cost.

All these gameobjects can be created by specific graphic design programs like Blender, MakeHuman, Maya, Adobe Fuse, Rhinoceros, etc. The advantage of use these softwares is that many of them are free and easy to use, allowing the export of gameobjects to Game Engines without problems in different 3D modelling formats as could be *.fbx, .blend, .obj, .max*, etc.

For the creation of different models integrated in this project, some 3D modelling programs have been assessed. Blender and MakeHuman are two useful and complete design tools, free and relatively easy to use. These characteristics fit perfectly with our project's philosophy.

3.2.1 Blender

Blender is a professional free and Open Source 3D computer graphics software product used for creating animated films, visual effects, 3D models, and videogames. Blender's features include 3D modelling, texturing, rigging and skinning, camera tracking and rendering.

Figure 3.6: Blender logotype

The Dutch animation studio *Neo Geo* developed Blender as an in-house application in January 1995, with the primary author being software developer Ton Roosendaal. On July 18, 2002, Roosendaal started the *Free Blender* campaign, a crowdfunding precursor. Finally on September 7, 2002, it

was announced that they had collected enough funds to release the Blender source code. Today, Blender is free and Open Source software developed by the community.

The last version of Blender, version 2.77a, is compatible with different operative systems like Windows, Mac OS X, GNU/Linux, Solaris, FreeBSD and IRIX, available in both 32-bit and 64-bit versions. This software contains features that are characteristic of high-end 3D software, among them we can found: [24] [25]

- Support for a variety of geometric primitives, including polygon meshes, fast subdivision surface modelling, Bezier curves, metaballs, icospheres, multiresolution digital sculpting and a new n-gon modelling system called B-mesh.
- Internal render engine with scanline rendering, indirect lightning, and ambient occlusion that can export in a wide variety of formats.
- Integration with a number of external render engines through plugins.
- Simulation tools for Soft body dynamics including mesh collision detection, Lattice Boltzmann methods (LBM) fluid dynamics, smoke simulation, Bullet rigid body dynamics, and ocean generator with waves.
- Python scripting for tool creation and prototyping, game logic, importing and/or exporting from other formats, task automation and custom tools.
- Camera and object tracking

All these characteristics and its high capability to export compatible gameobjects in format *.blend*, make of this software an important tool for the creation of visual environments and gameobjects that decorates our Serious Game in order to entertain and amuse the patient during the Serious Game performing.

3.2.2 MakeHuman

MakeHuman is an Open Source 3D computer graphics software middleware designed for the prototyping of photo realistic humanoids. It is developed by a community of programmers, artists, and academics interested in 3D modelling of characters. [26] [27]

Figure 3.7: MakeHuman logotype

The ancestor of MakeHuman was MakeHead, a python script for Blender, written by Manuel Bastioni, artist and coder, in 1999. A year later, a team of developers had formed, and they released the first version of MakeHuman for Blender. In 2005 MakeHuman was moved outside Blender, and finally in 2009 the team decided to release MakeHuman 1.0 pre-alpha.

MakeHuman is developed using 3D morphing technology. Starting from a standard (unique) androgynous human base mesh, it can be transformed into a great variety of characters (male and female), mixing them with linear interpolation. It uses a very simple GUI in order to access and handle hundreds of morphings. The MakeHuman approach is to use sliders with common parameters, like height, weight, gender, ethnicity and muscularity. The tool is specifically designed for the modelling of virtual humans, with a simple and complete pose system that includes the simulation of muscular movement. The interface is easy to use, with fast and intuitive access to the numerous parameters required in modelling the human form.

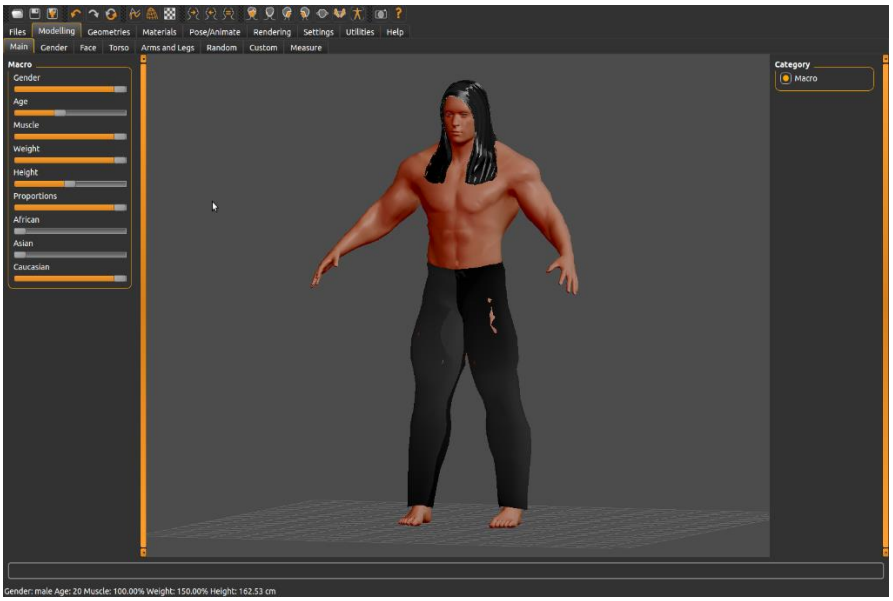

Figure 3.8: MakeHuman interface

MakeHuman is fully Open Source, and the character output of MakeHuman is released to public domain under Creative Commons License, in order to be freely used in commercial and non-commercial projects.

Summing up, MakeHuman is a software that fits perfectly with our project strategy; is easy to use, Open Source and can provide us the avatars we need for our Serious Games.

3.3 Motion Capture devices

We can define motion capture (MoCap) as the process of recording the movement of objects or people. Motion capture and computer animation techniques have made significant progress in game and film industry, detecting movements of people in 3D and displaying it in a 3D virtual scene is a research problem. There are two ways for motion capture, marker based motion capture and markerless motion capture.

The marker based motion capture has many drawbacks, the major drawback is that the performer has to wear a suit with adherent sensors or markers on it, and the process consist of handling multiple cameras placed in a room.

Figure 3.9: Example of Marker based motion capture

In markerless motion capture the performer doesn't have to wear a suit, but still being a challenging task. With this type of motion capture enough good results cannot be obtained using a single ordinary camera, the process still requires a set of multiples cameras placed all over the room, which also increases cost of the overall system.

With the development of depth cameras such as Microsoft Kinect has eased the task of motion capture, without requiring the burden of multiple cameras, hence it decreases the cost of overall system. [28]

Figure 3.10: Example of depth camera motion capture

In the following subsections we are going to analyse some of the principal depth cameras that are nowadays in the market according to Ashish Shingade and Archana Ghotkar [28], which features are overviewed in the incoming table.

Specifications	Kinect Camera	Sony PlayStation Eye	Prime Sense Sensor	Intel's Creative Camera
Illustration				
Viewing angle	43°vertical by 57° horizontal.	56° to 75° field of view.	57.5°to 45° field of View	73° field of view (diagonal).
Device range	Minimum 0.8 meter to maximum 4 meter.	Minimum 0.3meter	Minimum 0.8 meter to maximum 3.5 meter.	Minimum 0.15 meter to maximum 0.99 meter.
Frame rate	12 and 30 frames per second (FPS).	75and 187 frames per second (FPS).	60 frames per second (FPS).	30 frames per second (FPS).
Resolution	1280 x 960 resolution at 12 frames per second, or a 640 x 480 resolution at 30 frames per second.	320 x 240 resolution at 187 frames per second, or a 640 x 480 resolution at 75 frames per second.	640 x 480 resolution.	1280x720resolution.
IR camera	Yes.	No	Yes.	Yes.
Microphone array	Yes.	Yes.	Yes.	Yes.
OS support	Windows.	Windows, Mac OS, Linux.	Windows, Linux.	Windows.

Table 3.1: Survey of different depth cameras with their specifications [28]

3.3.1 Kinect camera

Kinect is a line of motion sensing input devices by Microsoft for Xbox 360, Xbox One and Windows PCs. Based around a webcam-style add-on peripheral, it enables users to control and interact with their console/computer without the need for a game controller, through a natural user interface using gestures and spoken commands. The first-generation Kinect was first introduced in November 2010 in an attempt to broaden Xbox 360's audience beyond its typical gamer base.

Figure 3.11: First-generation Kinect sensor (Xbox 360)

Kinect builds on software technology developed internally by *Rare*, a subsidiary of Microsoft Game Studios owned by Microsoft. On range camera technology by Israeli developer *PrimeSense*, which developed a system that can interpret specific gestures, making completely hands-free control of electronic devices possible by using an infrared projector and camera and a special microchip to track the movement of objects and individuals in three dimensions. This 3D scanner system called *Light Coding* employs a variant of image-based 3D reconstruction. [29]

Kinect sensor is a horizontal bar case connected to a small base with a motorized pivot and is designed to be positioned lengthwise above or below the video display. Inside the sensor case, a Kinect for Windows sensor contains: [30]

- An RGB camera that stores three channel data in a 1280x960 resolution. This makes capturing a colour image possible.
- An infrared (IR) emitter and an IR depth sensor. The emitter emits infrared light beams and the depth sensor reads the IR beams reflected back to the sensor. The reflected beams are converted into depth information measuring the distance between an object and the sensor. This makes capturing a depth image possible.
- A multi-array microphone, which contains four microphones for capturing sound. Because there are four microphones, it is possible to record audio as

well as find the location of the sound source and the direction of the audio wave.

- A 3-axis accelerometer configured for a 2G range, where G is the acceleration due to gravity. It is possible to use the accelerometer to determine the current orientation of the Kinect.

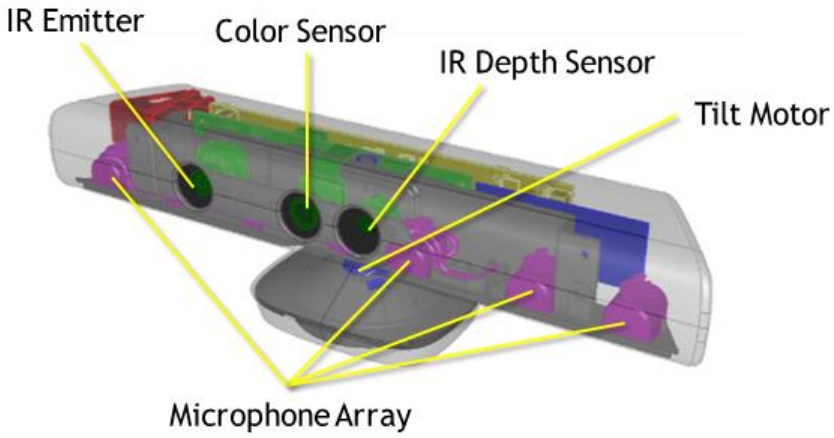

Figure 3.12: Kinect sensor components

The Kinect sensor have some specification about its performance, which are detailed in the following table:

Kinect	Array Specifications
Viewing angle	43° vertical by 57° horizontal field of view
Vertical tilt range	±27°
Frame rate (depth and color stream)	30 frames per second (FPS)
Audio format	16-kHz, 24-bit mono pulse code modulation (PCM)
Audio input characteristics	A four-microphone array with 24-bit analog-to-digital converter (ADC) and Kinect-resident signal processing including acoustic echo cancellation and noise suppression
Accelerometer characteristics	A 2G/4G/8G accelerometer configured for the 2G range, with a 1° accuracy upper limit.

Table 3.2: Kinect sensor specifications

What distinguish Kinect of other previous cameras, which software programs used differences in colour and texture to distinguish objects from their backgrounds, is the invisible transmission of near-infrared light and measures its “time of flight” after it reflects off the objects. Time-of-flight works like sonar: Knowing how long the light takes to return, you know how far away an object is. So with this technology the Kinect camera is able to produce a depth image distinguishing objects’ depth within 1 centimeter and their height and width within 3 mm. [31]

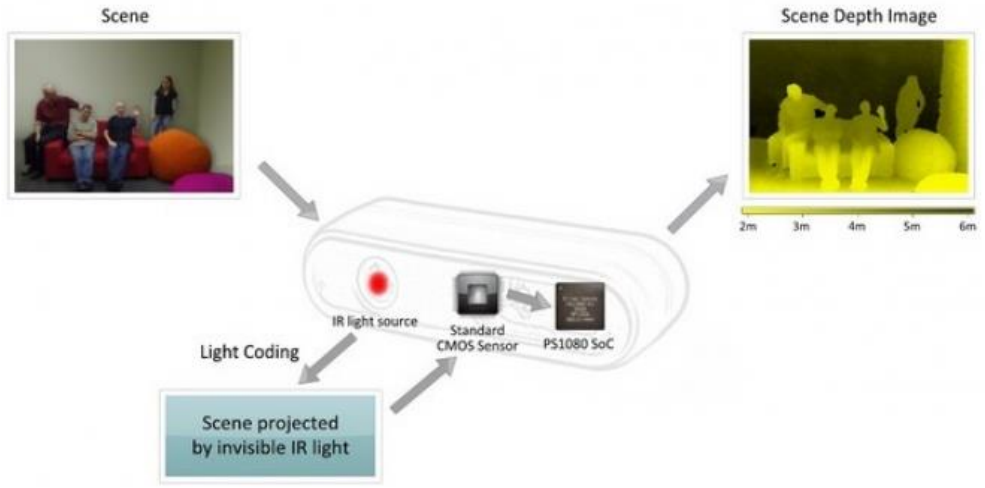

Figure 3.13 Depth image creation [31]

The Kinect has an on-board processor, considerate its firmware or sometimes “middleware”, which is using algorithms to process the data to render the three-dimensional image. The middleware also can recognize people: distinguishing human body parts, joints and movements, as well as distinguishing individual human faces from one another, making possible recognise and distinguish several natural human movements.

In 2011 Microsoft announced that it would release a non-commercial Kinect software development kit (SDK) for Windows, which was released for the use of Kinect cameras in Windows 7 PCs. The SDK included Windows 7 compatible PC drivers for Kinect device. It provides Kinect capabilities to developers to build applications with C++, C#, or Visual Basic by using Microsoft Visual Studio and includes following features: [29]

- Raw sensor streams: Access to low-level streams from the depth sensor, colour, camera sensor, and four-element microphone array.
- Skeletal tracking: The capability to track the skeleton image of one or two people moving within Kinect's field of view for gesture-driven applications.
- Advanced audio capabilities: Audio processing capabilities include sophisticated acoustic noise suppression and echo cancellation, beam formation to identify the current sound source, and integration with Windows speech recognition API.
- Sample code and Documentation.

This SDK have been update from version 1.5 in 2012 to version 1.8 on September of 2013. Finally in 2014 was released version 2.0 of Kinect SDK, necessary for the integration of second generation of Kinect sensors (Kinect for Xbox One) in Windows.

3.3.2 Sony PlayStation Eye

The PlayStation Eye (trademarked PLAYSTATION Eye) is a digital camera device, similar to a webcam, for the PlayStation 3. The technology uses computer vision and gesture recognition to process images taken by the camera. This allows players to interact with games using motion and color detection as well as sound through its built-in microphone array. It is the successor to the EyeToy for the PlayStation 2, which was released in 2003. [32]

Figure 3.14: PlayStation Eye device

There are many features that makes PlayStation Eye different from its predecessor. For a start, PlayStation Eye was engineered from the onset to perform well in low-light conditions. It also has a 2-setting zoom lens (normal and wide angle) with fixed focus. PlayStation Eye gets a sharp, clear picture at 640 x 480 resolution at 60 frames per second. It picks up image and motion in varying light conditions and doesn’t need your help to focus.

It also has a built-in 4-microphone array that let users to enjoy online AV chat (audio visual chat) and in-game voice chat. The camera employs technologies for multi-directional voice location tracking, echo cancellation, and background noise suppression. This allows the peripheral to be used for speech recognition and audio chat in noisy environments without the use of a headset. The PlayStation Eye microphone array operates with each channel processing 16-bit samples at a sampling rate of 48 kilohertz, and a signal-to-noise ratio of 90 decibels.

The PlayStation Eye features a free video editing software, *EyeCreate*, which enables users to capture pictures, video, and audio clips directly to the hard drive of the PlayStation3 console. *EyeCreate* includes a variety of different capturing modes, including stop motion and time-lapse. Through the software, users can edit, save, and share their own custom images, movies, and audio content. [33]

PlayStation Eye enables natural user interface and mixed reality videogame applications through the use of computer vision (CV) and gesture recognition technologies implemented in the software. Though initial PlayStation Eye software has mostly been based on the same general techniques as the EyeToy (e.g. simple edge detection and color tracking, Digimask face mapping), Sony has been promoting a number of other technologies available for the PlayStation Eye. Among these are the Vision Library, which can perform advanced facial recognition/analysis and CV based head tracking, and PlayStation Voice Recognition (PSVR), a speech recognition library intended to support about 20 different languages. According to Sony, the facial technology can identify features such as eyes, mouth, eyebrows, nose, and eyeglasses, read the shape of the mouth and detect a smile, determine the position and orientation of the subject's head and estimate the age and gender of the face. [32]

Figure 3.15: Eye pet™ for PlayStation Eye

3.3.3 PrimeSense Sensor

PrimeSense 3D sensing technology gives digital devices the ability to observe a scene in three dimensions. It translates these observations into a synchronized image stream (depth and color). [34]

Figure 3.16: PrimeSense sensor Carmine 1.08

It then takes those synchronized images and translates them into information such as:

- Identification of people their body properties, movements and gestures
- Classification of objects such as furniture, packages, etc.
- Measurements such as size or volume
- Location of walls and floor

All this functions are called depth sensing, which is made possible through the cutting-edge technology embedded in our sensors and middleware.

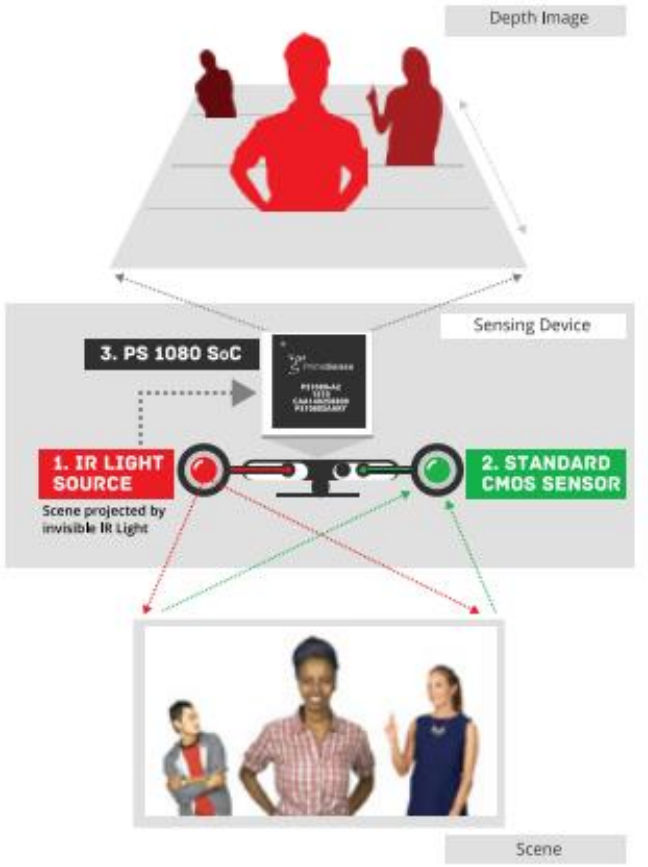

Figure 3.17: PrimeSense Sensor workflow

PrimeSense’s Light Coding patented technology solutions make 3D depth sensing possible. It works by coding the scene with near-IR light, which is invisible to the human eye. The solution then uses a standard off the shelf CMOS (complementary metal-oxide-semiconductor) image sensor to read the coded light back from the scene. This is the process that enables depth acquisition and what makes PrimeSense solutions so accurate.

Using the CMOS image sensor, PrimeSense cameras execute sophisticated parallel computational algorithms to decipher the received light coding infrared patterns, in order to produce a VGA (videographics array) size depth image of a scene. With a USB interface used to pass all data to the host, this cameras have minimal CPU requirements as all depth acquisition algorithms run on the camera itself.

The specifications and main features for the PrimeSense products, Carmine 1.08 and Carmine 1.09, are given in the following table:

PROPERTY		CARMINE 1.08	CARMINE 1.09	UNIT
Operating Temperature		5 – 40	10 – 40	[°C]
Data Interface		USB 2.0 / 3.0	USB 2.0 / 3.0	
Operation Range		0.8 – 3.5	0.35 – 1.4	[m]
Field of View (Horizontal, Vertical, Diagonal)		57.5, 45, 69 (H,V,D)	57.5, 45, 69 (H,V,D)	[deg]
Depth Image Size		640 x 480 (VGA)	640 x 480 (VGA)	[pixel x pixel]
Spatial x/y Resolution (2-Sigma Values)	@2m	3.4		[mm]
	@0.5m		0.9	[mm]
Depth Resolution (2-Sigma Values)	@2m	1.2		[cm]
	@0.5m		0.1	[cm]
Maximal Frames-per-Second Rate		60	60	
Color Image CMOS	@ 30 FPS	640 x 480 (VGA)	640 x 480 (VGA)	[pixel x pixel]
Built-in Microphones		2	2	
Data Format		16	16	[bit]
External Digital Audio Inputs		4	4	[inputs]
Dimensions: Width x Height x Depth		18 x 2.5 x 3.5	18 x 2.5 x 3.5	[cm]
Power Supply		USB	USB	
Maximal Power Consumption		2.25	2.25	[Watt]

Table 3.3: PrimeSense products specifications

All this features make of them one of the best sensors in depth performance, with a thin host, low-power supply and with a competitive price in the market.

4 Materials and Methods

Once explored what technologies are present nowadays related with the creation of videogames, is necessary to select which ones fit better with our purpose of create a Serious Game for post-stroke’s patients. In addition, in this chapter we will describe how work all the technologies used for the accomplishment of this project and how they have been integrated during the design and development of the Serious Game.

4.1 Hardware

4.1.1 Kinect v2

Finally the motion capture device used for the development of this work was the last version of Kinect’s cameras, Kinect for Xbox One or Kinect v2, due to its capability to capture the entire patient’s body and track body’s positions, movements and orientations.

Figure 4.1: Kinect sensor v2

Kinect v2 leads in many features to its predecessor Kinect for Xbox 360, making this device more accurate technologically speaking, and attractive for research purposes. Some of the benefits of using this camera instead of the first generation camera are explained in the following table:

Feature	***Benefits***
Improved body tracking	*The enhanced fidelity of the depth camera, combined with improvements in the software, have led to a number of body tracking developments. The latest sensor tracks as many as six complete skeletons (compared to two with the original sensor), and 25 joints per person (compared to 20 with the original sensor). The tracked positions are more anatomically correct and stable and the range of tracking is broader.*
Depth sensing *512 x 424* *30 Hz FOV: 70 x 60* *One mode: .5–4.5 meters*	*With higher depth fidelity and a significantly improved noise floor, the sensor gives you improved 3D visualization, improved ability to see smaller objects and all objects more clearly, and improves the stability of body tracking.*

1080p color camera *30 Hz (15 Hz in low light)*	*The color camera captures full, beautiful 1080p video that can be displayed in the same resolution as the viewing screen, allowing for a broad range of powerful scenarios. In addition to improving video communications and video analytics applications, this provides a stable input on which to build high quality, interactive applications.*
New active infrared (IR) capabilities *512 x 424* *30 Hz*	*In addition to allowing the sensor to see in the dark, the new IR capabilities produce a lighting-independent view—and you can now use IR and color at the same time.*
Kinect for Xbox One sensor dimensions *(length x width x height)*	*9.8" x 2.6" x 2.63" (+/- 1/8")* *24.9 cm x 6.6 cm x 6.7 cm* *Length: The Kinect cable is approximately 2.9 m long* *Weight: approximately 1.4 kg* *Sensor FOV: 70 x 60*
A multi-array microphone	*Four microphones to capture sound, record audio, as well as find the location of the sound source and the direction of the audio wave.*

Table 4.1: Kinect v2 hardware key features and benefits [35]

The main problem that present Kinect's cameras, is their incompatibility to be connected with other devices apart from Xbox consoles. To sort out this, in 2014 Microsoft launched the *Kinect adapter for Windows*, adapter that lets users to experience the convenience and versatility of the Xbox One Kinect Sensor on Windows PCs, developing interactive apps on PC. The system requirements to use this adapter and, thus, the Kinect sensor v2 in a personal computer are the following:

Processor	*64-bit (x64) processor, physical dual-core 3.10 GHz (2 logical cores per physical) or faster processor*
Operating system	*Windows 8 or 8.1, or Windows Embedded 8*
Memory	*4GB RAM*
Video	*Graphics card that supports DirectX 11*
Other	*USB 3.0 controller dedicated to the Kinect for Windows v2 sensor*

Table 4.2: Kinect v2 system requirements [36]

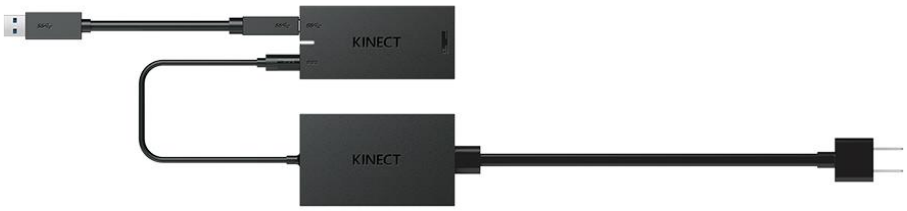

Figure 4.2: Xbox Kinect adapter

A *plug-in* in computing is a software component that adds a specific feature to an existing computer program. When a program supports *plug-ins*, it enables customization. [37]

For Kinect, Microsoft offers a *plug-in* that allow to the users use different development tools to build applications for Windows which can interact with the Kinect v2 hardware. This *plug-in* is the *Kinect for Windows SDK 2.0*, which incorporates many features and functions, together with demos and programs that facilitate the understanding of how Kinect works. Some important features and benefits of using the SDK 2.0 are listed in the table below.

Feature	***Benefit***	***Potential applications***
Improved body, hand and joint orientation	*With the ability to track as many as six people and 25 skeletal joints per person—including new joints for hand tips, thumbs, and shoulder center—and improved understanding of the soft connective tissue and body positioning, you get more anatomically correct positions for crisp interactions, more accurate avateering, and avatars that are more lifelike.*	*New and better scenarios in fitness, wellness, education and training, entertainment, gaming, movies, and communications.*
Windows Store support	*You can now create Kinect enabled Windows apps by using familiar Windows Runtime components.*	*You can list and sell your Kinect enabled Windows apps in the Windows Store.*
Unity Pro support	*For more than just gaming, Unity Pro offers cross-platform rapid prototyping.*	*Build and publish apps by using tools that you already know across multiple platforms.*

Powerful tooling	*Kinect Studio provides enhanced recording and playback, and Visual Gesture Builder lets developers build their own custom gestures that the system recognizes and uses to write code by using machine learning, increasing productivity and cost efficiency.*	*Develop on the go without the need to bring the Kinect sensor with you. Create custom gestures that decrease the time to prototype and test solutions.*
Advanced facial tracking	*Resolution is 20 times greater, enabling the application to create a mesh of more than 1,000 points for a more accurate representation of a person’s face.*	*Build avatars that appear more lifelike.*
Simultaneous multi-app support	*Improved multi-app support enables multiple applications to access a single sensor simultaneously.*	*For example, by allowing a retail app and a business intelligence app to access the same sensor, you can get business intelligence in real time in your retail space.*

Table 4.3: Kinect for Windows SDK 2.0 key features and benefits [38]

This SDK was the key for the implementation of the Kinect Sensor not only with the Windows system, but to use this device within the Game Engine used for the creation of Serious Game.

4.1.2 Computer

In order to develop and play videogames, the specifications of the computer are really important. Its CPU, graphics card, storage, memory, RAM, etc. make the difference in the performance of a videogame. So for that reason, for this project was important to use a computer which specifications are made to fit with videogames requirements.

Finally the computer used for the project development was *MSI GT72 2QE DOMINATOR PRO*, which specifications are listed below:

Figure 4.3: MSI GT72 2QE DOMINATOR PRO

CPU	*4th Generaton Intel® Core™ i7 processor*
OS	*Windows 10*
Chipset	*Intel HM87*
Memory	*DDR3L, up to 1600 MHz, slot *4, max 32GB*
Display	*17" Full HD (1920X1080) , Anti-glare*
Graphics	*GeForce® GTX 980M*
Graphics VRAM	*GDDR5 8GB/4GB*
Storage	*Until 512GB Super RAID 3 + 1TB HDD 7200rpm*
Optical Drive	*BD Writer / DVD Super Multi*
Audio	*Sound by Dynaudio 2.1 channel with 1 Woofer* *Support 7.1 channel SPDIF output* *Exclusive Audio Boost 2 technology* *Nahimic sound technology*

Table 4.4: MSI GT72 2QE DOMINATOR PRO specifications [39]

4.2 Software

Once we have set the hardware, the last step to carry out this project is the election of the software and their integration with the hardware. In this case, the software used had to be a Game Engine in order to develop and design reliable Serious Games for post-stroke rehabilitation.

4.2.1 Unity

Its compatibility with Kinect, its popularity, the set of functions that offers to develop videogames, its ease to use and its availability totally free, makes of Unity the best software for the development of this project.

4.2.1.1 Unity Interface [40]

The main editor window is made up of tabbed windows which can be rearranged, grouped, detached and docked. This means that each user can custom the interface of Unity as he likes. The most common and useful windows are shown below:

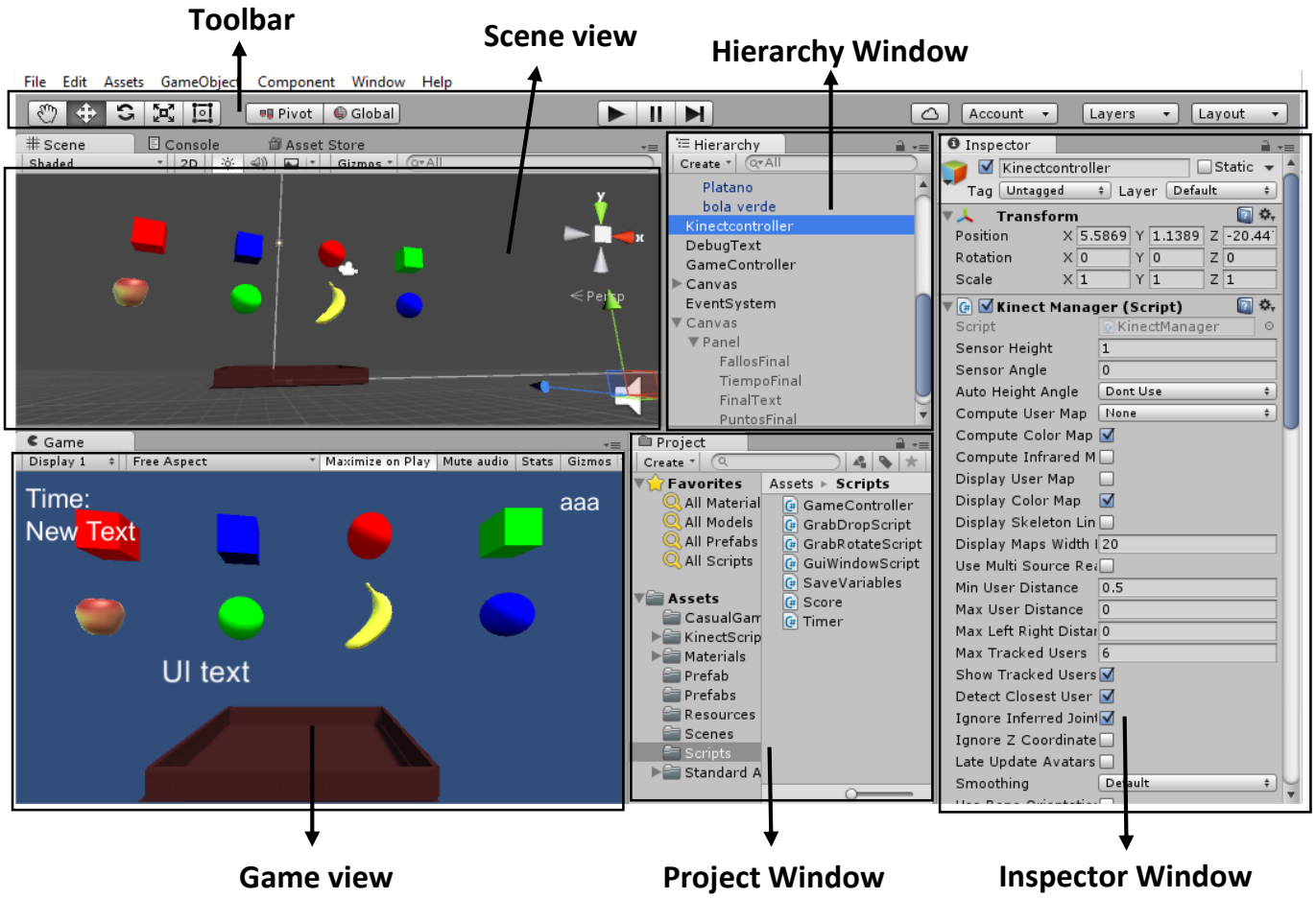

Figure 4.4: Unity User Interface

- **The Project Window**

The Project Window displays your library of assets, or game elements, that are available to use in your project.

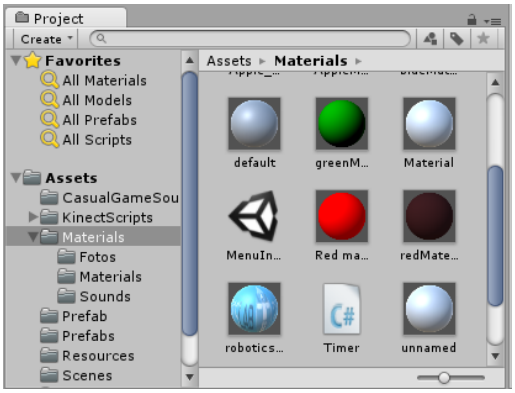

Figure 4.5: Project Window

The left panel of the browser shows the folder structure of the project as a hierarchical list. When a folder is selected from the list by clicking, its contents will be shown in the panel to the right. The individual assets are shown in the right hand panel as icons that indicate their type (script, material, sub-folder, etc). Located at the left side of the toolbar, the Create menu lets you add new assets and sub-folders to the current folder. To its right are a set of tools to allow you to search the assets in your project. Summing up, this window store all the elements that you import or create to develop a videogame.

- **The Scene View**

The Scene View allows you to visually navigate and edit your scene. The scene view can show a 3D or 2D perspective, depending on the type of project you are working on.

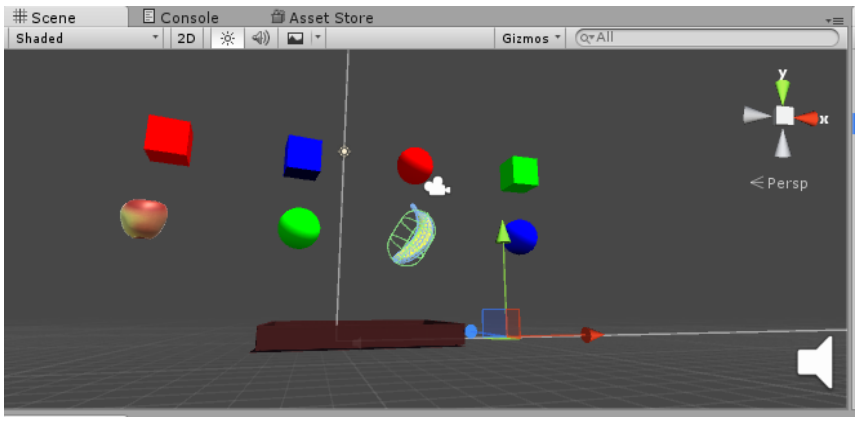

Figure 4.6: Scene view

The Scene View is the view into the game’s world the user is creating. In the Scene View you can select and position scenery, characters, cameras, lights, and all other types of Game Object. By this way, users are able to select, manipulate and modify objects in the scene, changing their transform coordinates, size and orientation.

- **The Game View**

The Game View is produced by the Camera or Cameras of your game. It represents the final view of the game. You will need to use one or more Cameras to control what the future player actually sees when he or she is playing your game. In the upper right part of the Game View panel you can select some options such as the maximizing of the screen while the user tries the game, mute game’s audio, etc.

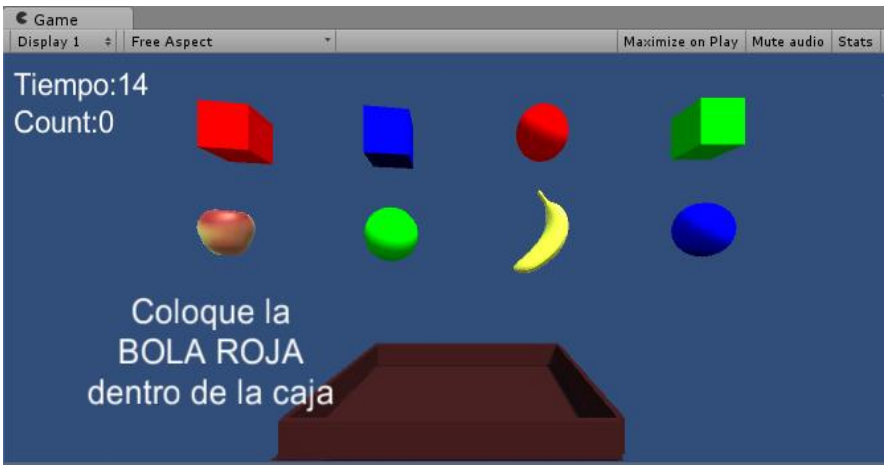

Figure 4.7: Game view

- **The Hierarchy Window**

The Hierarchy Window is represents every object in the scene. Each item in the scene appears in the hierarchy. The hierarchy reveals how objects are attached and related to one another.

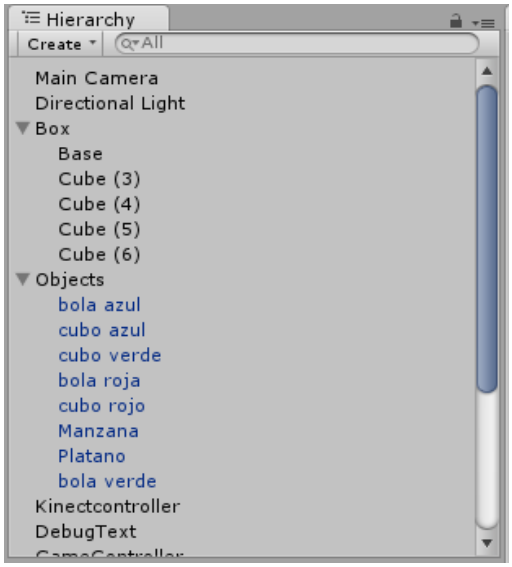

Figure 4.8: Hierarchy window

The Hierarchy window contains a list of every GameObject in the current Scene. As objects are added and removed in the Scene, they will appear and disappear from the Hierarchy as well.

To order and organize the objects in this window, Unity uses a concept called Parenting. When you create a group of objects, the topmost object or Scene is called the “parent object”, and all objects grouped underneath it are called “child objects” or “children”, belonging all to the same group.

- **The Inspector Window**

The Inspector Window allows you to view and edit all the properties of the currently selected object. Because different types of objects have different sets of properties, the layout and contents of the inspector window can vary.

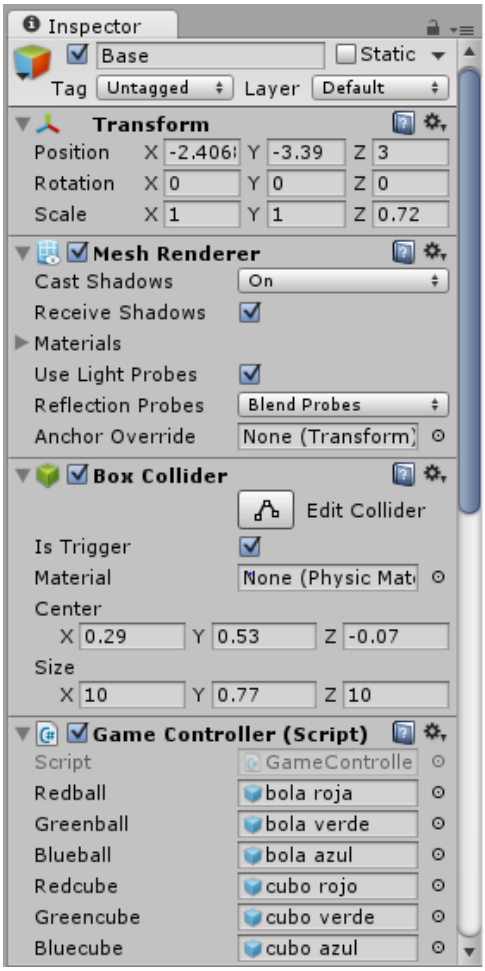

Figure 4.9: Inspector window

The Inspector is used to view and edit the properties and settings of Game Objects, Assets, and other preferences and settings in the Editor.

When you select a GameObject in the Hierarchy or Scene View, the Inspector will show the Transform of the object, Properties of all its Components and Materials on that object, and allows to edit them.

When Game Objects have custom Script components attached, the public variables of that script are also shown in the inspector and can be viewed and edited like the properties of Unity’s components. This allows you to set parameters and default values in your scripts easily without modifying the code.

- **The Toolbar**

The Toolbar provides access to the most essential working features

Figure 4.10: Unity's Toolbar

On the left it contains the basic tools for manipulating the scene view and the objects within it. In the centre are the play, pause and step controls. The buttons to the right give you access to your Unity Cloud Services and your Unity Account, followed by a layer visibility menu, and finally the editor layout menu (which provides some alternate layouts for the editor windows, and allows you to save your own custom layouts).

4.2.1.2 Kinect v2 with MS-SDK

One of the challenges that set out this project was the creation of an Open Source application, this means that people can modify the code and share the app because it is publicly accessible.

So it's important that our Serious Games can be modified or enhanced by anyone who want do it. Unity is not an Open Source Game Engine, but its free accessibility and the freedom that lets us to share Unity projects on the Internet or in the *Unity Asset Store* makes of this software a good tool to create Open Source applications.

This Open Source feature was important during the development of the project. An important part of the software's codes used for the creation of Serious Games, were freely download from the Internet.

The *Asset Store* offers different packages with codes and examples that allow to users use the Kinect hardware in Unity.

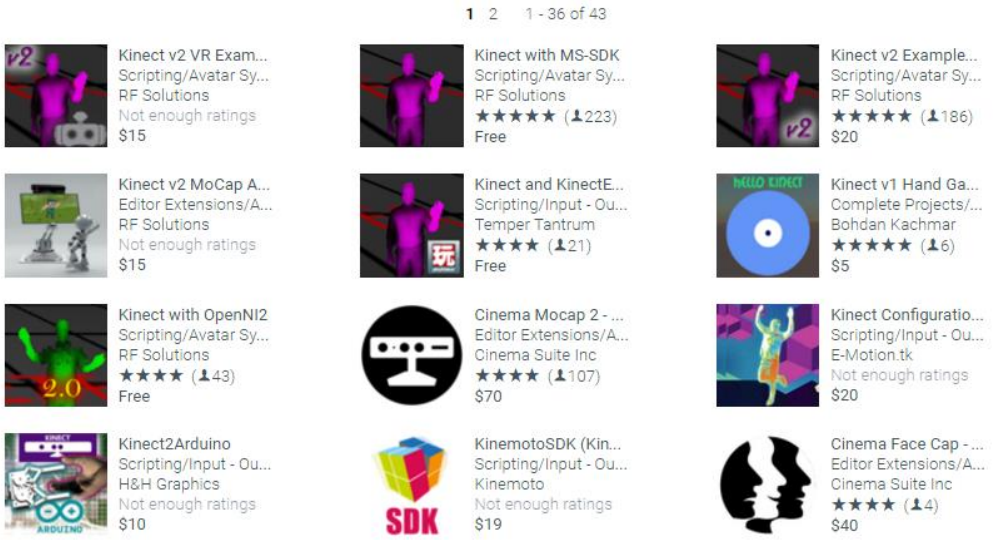

Figure 4.11: Kinect packages available in Unity’s Asset Store

Evaluating several packages, we decided to test and use the ones created by RF Solutions, *Kinect v2 with MS-SDK*, which are the most used and best rated. Clicking on the asset, appears the description of the asset, its contents and its price.

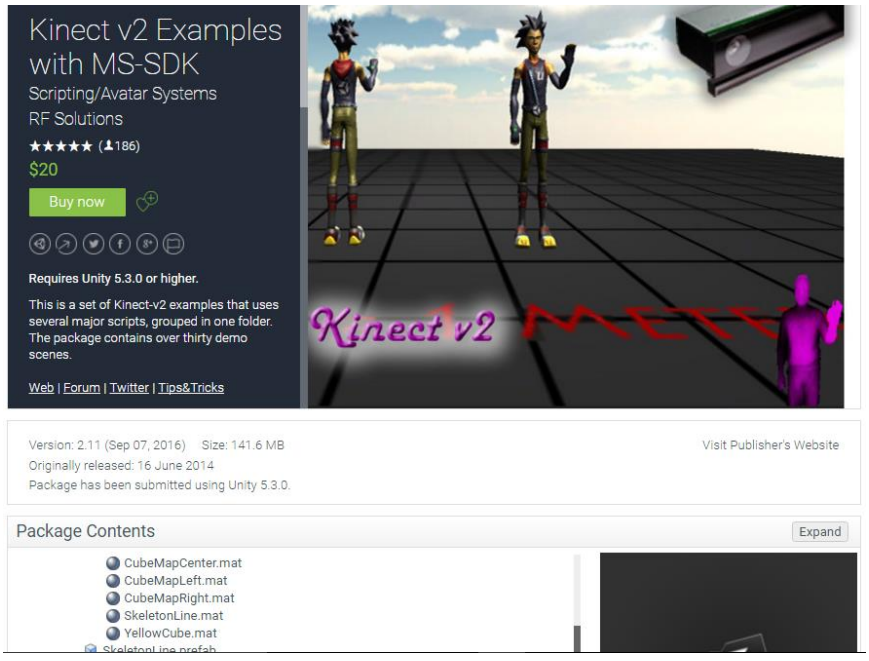

Figure 4.12: Description of Kinect V2 MS-SDK package

Although this package has a price of 20$, the creator of the scripts and Kinect demos offered the asset completely free to users who want it for academic use and not commercial purposes, how is our case.

The developer of the code that links Kinect sensor and Unity is Rumen Filkov, an IT professional that currently works as research assistant at the User Centered Technologies (UCT). All the information about *Kinect v2 with MS-SDK* is posted on his website https://rfilkov.com/.

Kinect v2 Examples with MS-SDK is a set of Kinect v2 examples that uses several major scripts, grouped in one folder. The package contains over thirty demo scenes. The avatar-demo scenes show how to utilize Kinect-controlled avatars in Unity projects. This package works with Kinect v2 and v1, supports Windows 32- and 64-bit builds and can be used in both Unity Pro and Unity Personal editors. [41]

4.2.2 Blender and MakeHuman

Together with Unity, other softwares were used in the creation and model of different GameObjects. Mainly the softwares used for this task were Blender and MakeHuman.

4.2.2.1 Blender

Blender is a free and Open Source 3D creation suite. It supports the entirety of the 3D pipeline: modeling, rigging, animation, simulation, rendering, compositing and motion tracking, even video editing and game creation. The models and objects created in Blender can be exported in *.blend* format, format compatible with Unity. This makes of Blender a perfect tool to create different objects that decorate the game scene and make the game more pleasant for the patients.

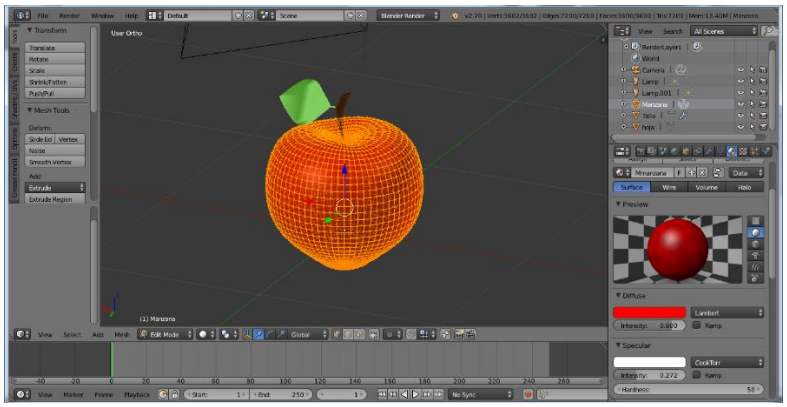

Figure 4.13: Blender interface

4.2.2.2 MakeHuman

MakeHuman is a free and Open Source software, available for Windows, OSX, Linux, to create realistic 3d humans for illustrations, animations, games, etc. Although its basic version includes only simple clothes and accessories to dress and design our character, its simple interface, the election in the number of joints for animations and games, and its compatibility with Unity, makes of this tool perfect for the creation of avatars for our Serious Games.

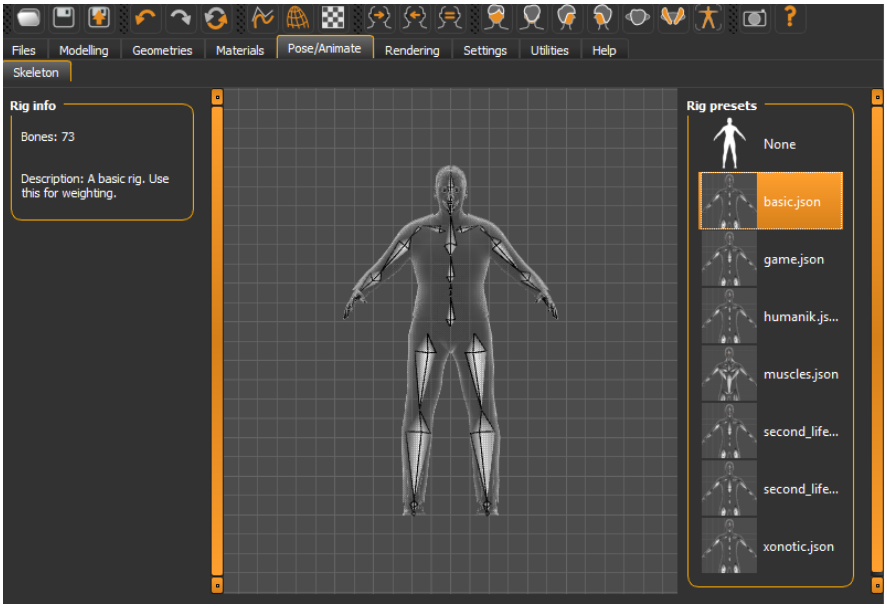

Figure 4.14: Avatar Rigging in MakeHuman

Once the avatar is exported in Filmbox format (.fbx), it can be imported into the Project Window of Unity without problem and used as an avatar gameobject.

5 Serious Games design and Results

This part of the dissertation will describe the Serious Games developed; its features in rehabilitation terms, their basis and how operates the last version of these Serious Games, together with the acquisition of useful data for the physicians such as the performance time, range of patient’s motion, game’s score, etc.

5.1 Serious Games features

A high percentage of stroke survivors suffer serious effects due to the disease. This effects can be classified in different type of deficits: motor deficit, communication deficit, cognitive deficit and emotional deficit.

These Serious Games are focused to treat the motor deficits and the hemiparesis of the patients. There are many simple motor tasks that a post-stroke patient cannot perform, as could be shoulder flexion or abduction, grab objects, stand keeping the balance, etc. So these games try to oblige patients to accomplish these routine actions enhancing their movement accuracy. Taking into account that great part of stroke survivors are old people, the game has to be simple and easy to understand, thus the game will be a motivation and not an obligation. Summing up the Serious Games have to incorporate the following features:

- Registration of patient’s body
- Capture and analysis of patient’s movement
- Gameplay adapted to patient’s goals and limitations
- Simple interface and easy to play
- Reward system for patient’s motivation

5.2 Serious Games design basis

The first thing that we have to set upon we are developing our Serious Games, is to make a videogame compatible with the Kinect sensor, in such a way that the user himself will be the game controller.

To make this possible, we have to analyze the content of the Unity package *Kinect v2 Examples with MS-SDK*. In the first level of the package content, we can observe five different folders: *Readme, KinectDemos, KinectScripts, Resources* and *Standard Assets.*

- *Readme:* This folder contains useful information about the installation and the content of the Unity package, documents in PDF format written by the developer Rumen Filkov.

- *KinectDemos:* As it name indicates, this folder contains several type of demos which can be played in Unity in order to test the different type of videogames we can create with this package. For example in *AvatarsDemo*, we can play with a 3D avatar that mimic the user movements when he plays.

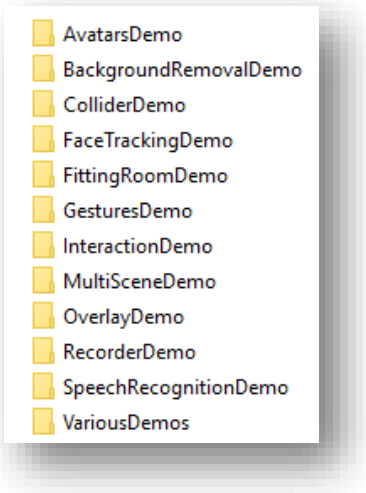

Figure 5.1: Demos from KinectDemos folder

- *KinectScripts*: This folder contains all needed scripts, filters and interfaces to create a Kinect videogame in Unity.

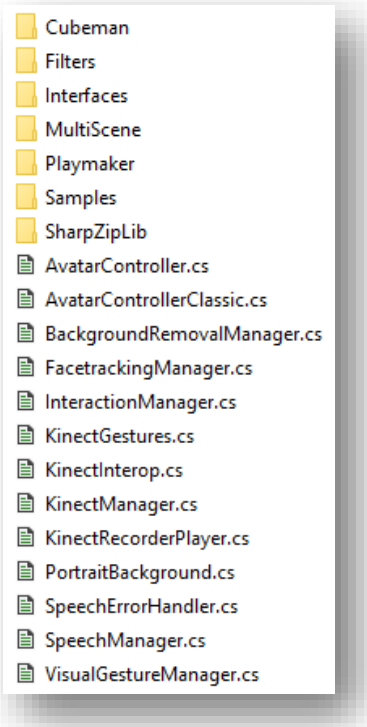

Figure 5.2: KinectScripts contents

- *Resources*: In this folder we can find the needed libraries and resources for the development of our project.

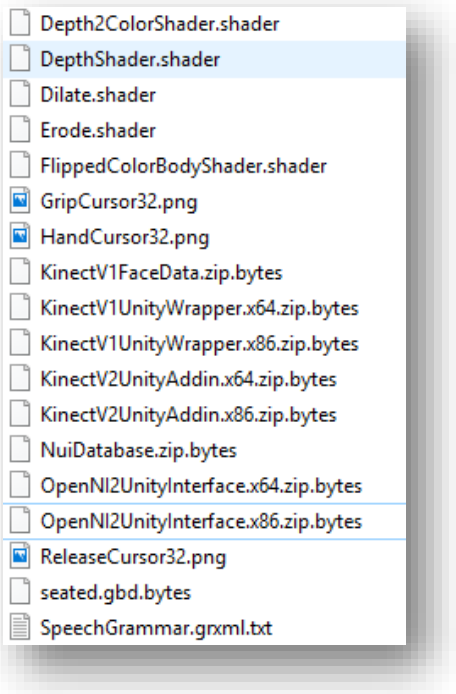

Figure 5.3: Libraries and resources from Resources folder

- *Standard Assets*: It contains the wrapper classes for Kinect v2. A wrapper is data that precedes the main data or program that sets up another program so that it can run successfully. [42]

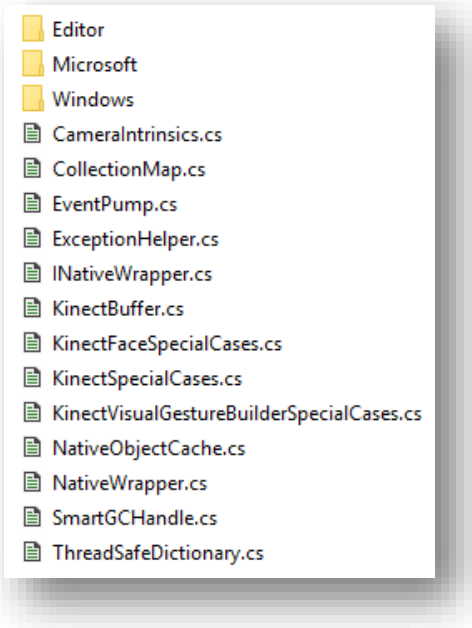

Figure 5.4: Standard Assets contents

So once we know what we have from *Kinect v2 Examples with MS-SDK* package, we can start to build our Serious Game.

5.2.1 First steps

The first step is to open Unity and create a new project with its scene empty. Then we import the *Kinect v2 Examples with MS-SDK* package, displaying the folders previously mentioned on the *Project Window* of the Unity’s interface.

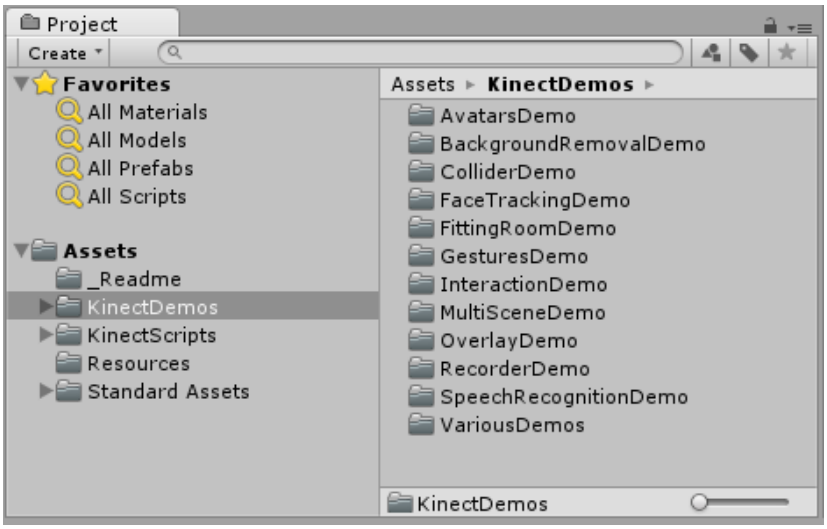

Figure 5.5: Project Window with Kinect v2 Examples with MS-SDK integrated

Then an empty gameobject is created in the Hierarchy Window, in this gameobject we add the script *KinectManager.cs* as component of the object. Once added we can custom different settings that collects this script in the *Inspector Window*, such as the Kinect camera display settings, the number of players of the game, etc. *KinectManager.cs* is the principal code that establishes and manages the connection between the Kinect sensor and our Unity software.

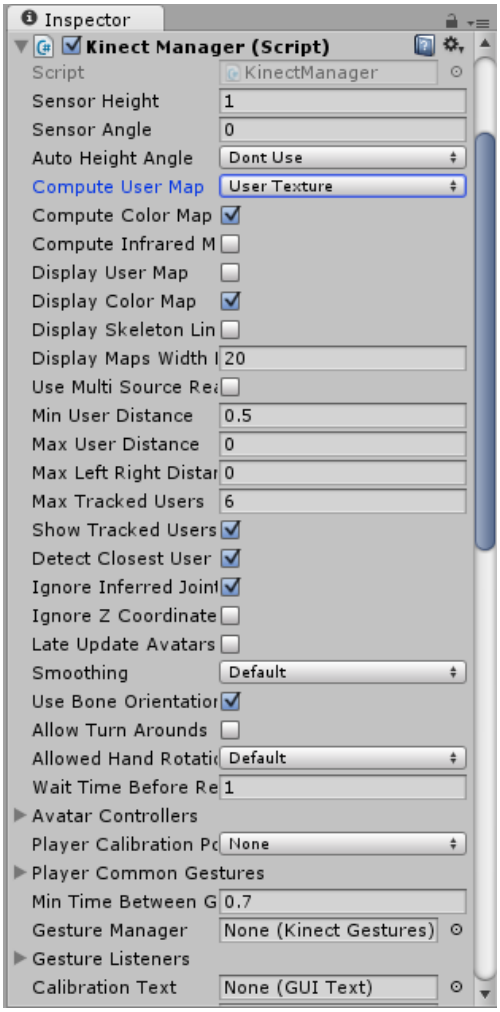

Figure 5.6: Adjustable settings of KinectManager.cs

5.2.2 Game environment and avatar

The next step for the Serious Games design and development, is the creation of a creative environment, interactive gameobjects and the implementation of an avatar, used as gamecontroller in great part of this kind of videogames.

5.2.2.1 Game Environment

To create an environment there are different options: designing it using Unity functions, importing gameobjects and developing the whole look of the game surrounding. Other option is to import directly environments from the *Asset Store* or other webpages.

Once the environment is downloaded and imported to Unity, we can modify it from the *Scene View* and the *Hierarchy Window*.

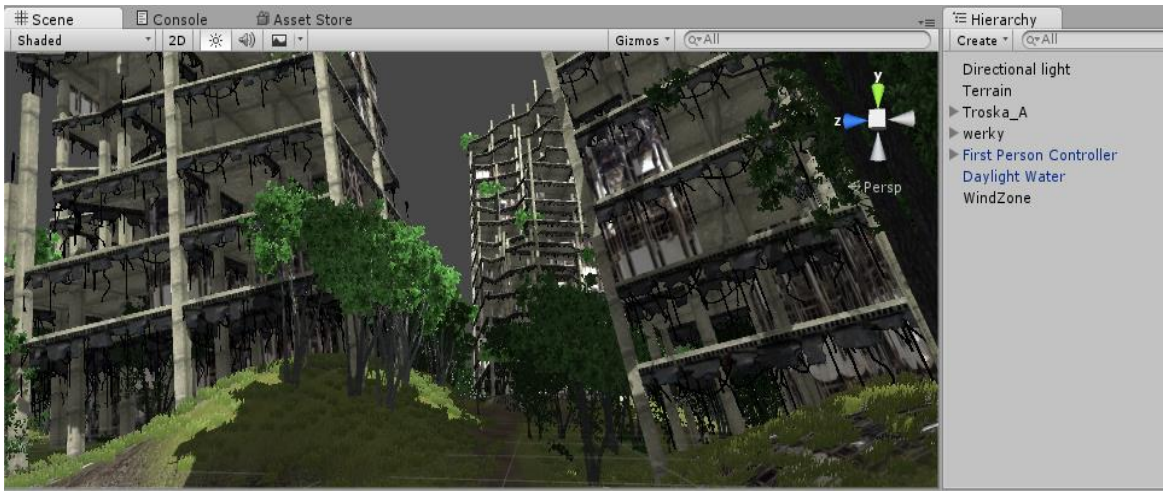

Figure 5.7: Scene view of “Destroyed City” environment

Just as the environment, different 3D modelled objects can be downloaded from the Asset Store or imported from modelling softwares as Blender, Maya, etc.

5.2.2.2 Avatar integration

An avatar is the graphical representation of the user, normally exemplified in a three-dimensional humanoid form. This avatar will be the intermediary between the patient and the game environment, allowing to the user interact with the gameobjects in the digital world.

The avatar can be created by specific softwares such as MakeHuman, or also can be imported from the *Asset Store* or other webpages. Once the avatar is created and imported to our project in Unity, we have to ensure that the avatar’s rig settings are correct from the *Inspector Window.* By default the avatar is composed by “joints”, that are empty gameobjects located in different body parts of the human avatar, representing each one a real human joint.

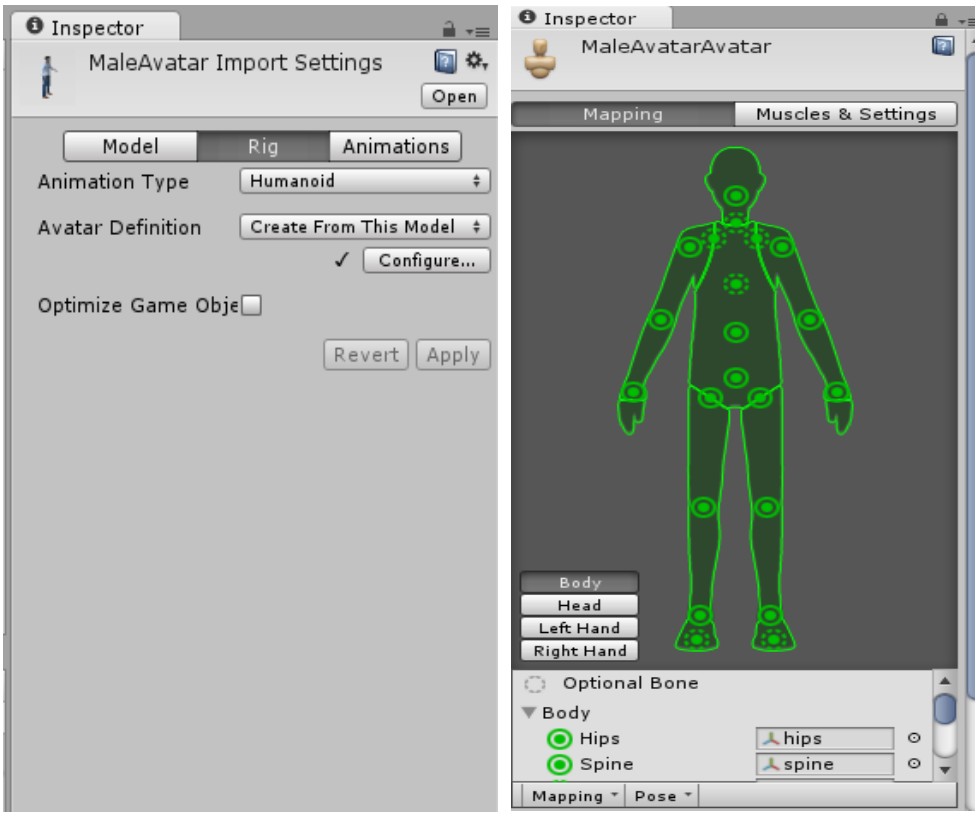

Figure 5.8: Rig settings of the imported avatar

Finally, the script *AvatarController.cs* is added as component of the avatar in the *Inspector Window*. This script is the responsible in the registration of the real human joints acquired by Kinect sensor into the avatar’s joints.

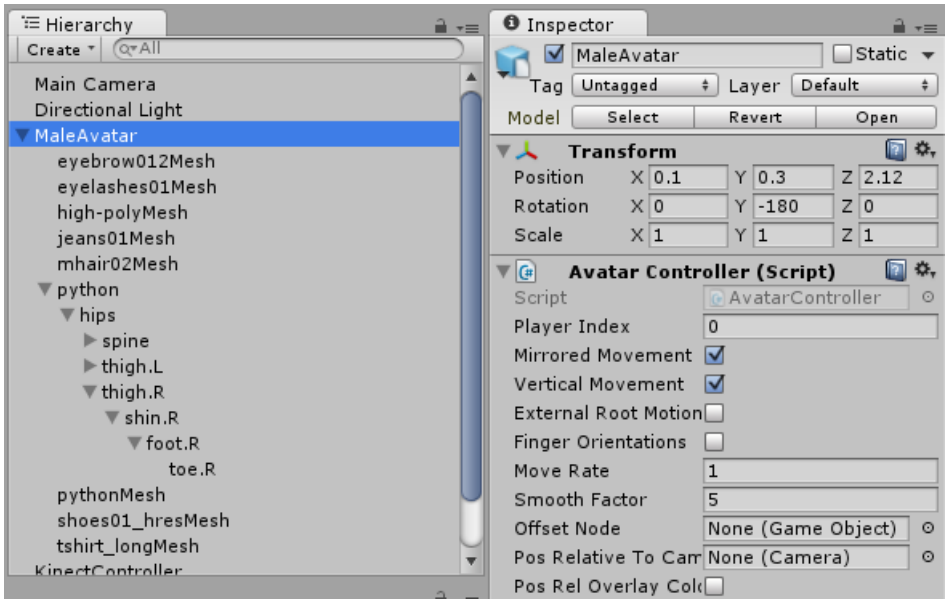

Figure 5.9: Avatar Controller settings

5.2.3 Acquiring and saving data

One of the objectives of this project is to keep patients entertained during the rehabilitation process, but also is important to extract and store useful data for medical analysis.

For the physiotherapist that treats the patient, is important to know the patient’s movement ranges, the time that he spends performing the exercise, how many times the patient fail when he is playing, etc.

In order to acquire the position in Cartesian coordinates (x,y,z) of different joints that are essential to follow during the gameplay, we can use the code of one of the scripts integrated in *Kinect v2 Examples with MS-SDK* package. This script is the *GetJointPosition.cs*, which is showed in the Annex of this dissertation.

This code gets and stores in a vector variable the coordinates of the selected joint, which can be very useful to know the maximum distance that the patient reaches along the vertical or horizontal plane.

Other important function of our Serious Games had to be save all the useful game variables in a file that physicians can have a look easily in their own computers. To do that, we have implemented a piece of code that creates and store all the useful variables into a CSV file, which can be imported easily to Microsoft Office.

```
public string saveFilePath; // This variable contains the PC path in which
the CSV file will be stored
    public float tiempo;
    private int fallos;
    private int puntuacion;
    private string Nombre;
    private string Fecha;


    void SaveData(){

if (!File.Exists(saveFilePath))
 {
     using (StreamWriter writer = File.CreateText(saveFilePath))
     {
         // csv file header
         string sLine = "Nombre,Fecha,Tiempo,Puntuacion,Fallos";
         writer.WriteLine(sLine);
     }
            }
 using (StreamWriter writer = File.AppendText(saveFilePath))
 {
 string sLine = string.Format("{0},{1},{2:F2},{3:F0},{4:F0}", Nombre,
Fecha, tiempo, puntuacion, fallos);

     writer.WriteLine(sLine);
            }
    }
```

Figure 5.10: Code to save variables in CSV format

Each time the patient plays the SeriousGame, the new data from the last session is saved in the CSV file. By this manner, physicians are able to compare the results from different session days.

	A	B	C	D	E
1	Nombre del paciente	Fecha	Tiempo	Puntuacion	Fallos
2	Jaime	14/09/2016	69.82	7	1
3	Jaime	14/09/2016	73.78	7	1

Figure 5.11: Example of data stored and displayed in Microsoft Excel

5.3 Developed Serious Games

The end of this project is turned into two different Serious Games with two specific therapeutic functions, which are showed and described in this part of the dissertation.

5.3.1 Menu module

The first we see when the Serious Game application is run, is a module of menus in which the user, in this case the physician, selects what therapy game want to use and introduces the name and the date of the session, that will be saved together with the results.

The first menu is an interface that displays the videogames available in the application. Currently the menu shows two different Serious Games: the *Reach Game* and the *Balance Game*.

Figure 5.12: Game selection menu

The second menu that appears when the user has selected the game, is a simple interface with two input fields in which the physician can introduce the name of the patient and the date of the session.

Figure 5.13: Input data menu

5.3.2 The *Reach Game*

This game tries to imitate a quotidian grip movement, grabbing an object in order to transfer it to other place.

5.3.2.1 Game interface

The game shows a simple gameview, where a series of figures and objects are floating in the scene and a box gameobject is located at the lower part of the game.

Besides this, the interface is integrated by some texts that show essential data and info for the game performance. In the upper left part of the screen, the scoring and the passed time during the gameplay is displayed. An UI (user interface) text is showed nearby the box, this text shows instructions to the player. Additionally to this text message an audio message is also displayed at the same time in order to enhance the game-player communication.

Finally at the lower right part is displayed a window with the Kinect’s camera vision, showing the user in the “real world”.

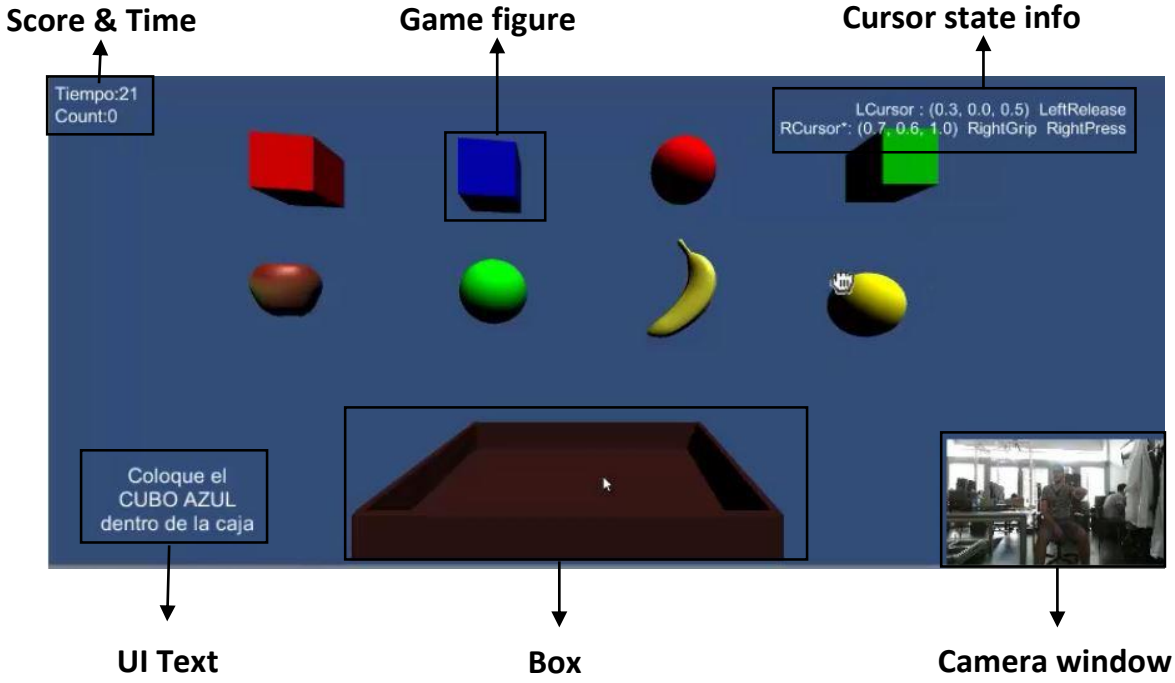

Figure 5.14: Game interface

5.3.2.2 Game Objective

The objective of this game is to grab the correct figure and put it into the box. In order to do it the patient has to place the hand cursor, which follows the real user hand, over the target gameobject and perform the grip action. This grip movement is detected by the Kinect camera and the hand cursor grasp the object such as in the real life.

Finally when the grasped object is placed inside the box the object disappears and one point is added in the scoring. If the player fails and puts the wrong object into the box, one unit is added to the variable “mistake” that is showed at the end of the game at the final menu, together with the scoring and the time the player needed to finish the game in seconds.

Figure 5.25: Final menu

5.3.2.3 Game Software

The software basis of this game is founded in one script of the Unity package *Kinect v2 Examples with MS-SDK.* This script is the *GrabDropScript.cs*, which allows to create a first-person videogame that displays a hand cursor that moves together with the detected user hand. Also detects the hand grab gesture and picks-up the objects that underlie the cursor when it is in “grab” state. The hand cursor that appears on the screen behaves different depending on the user hand’s position. Between the two main hand cursor states we can distinguish the “release” state and the “grab” state. If the Kinect detects the user’s hand palm, it turns the cursor into the “release” state, if detects the user’s fist, turns into the “grab” state

Figure 5.16: Hand cursor main states

5.3.2.4 *Reach Game* Workflow

The *Reach Game* was developed using several scripts with different functions inside the game. This section goes into detail about all the processes that happen while the patient is playing, which can be classified in different functions:

1. Ensure Kinect connection and user detection.
2. Allow to the user to grab and drop different gameobjects.
3. Detect the target gameobject inside the box.
4. Time and scoring.
5. End the game and save results.

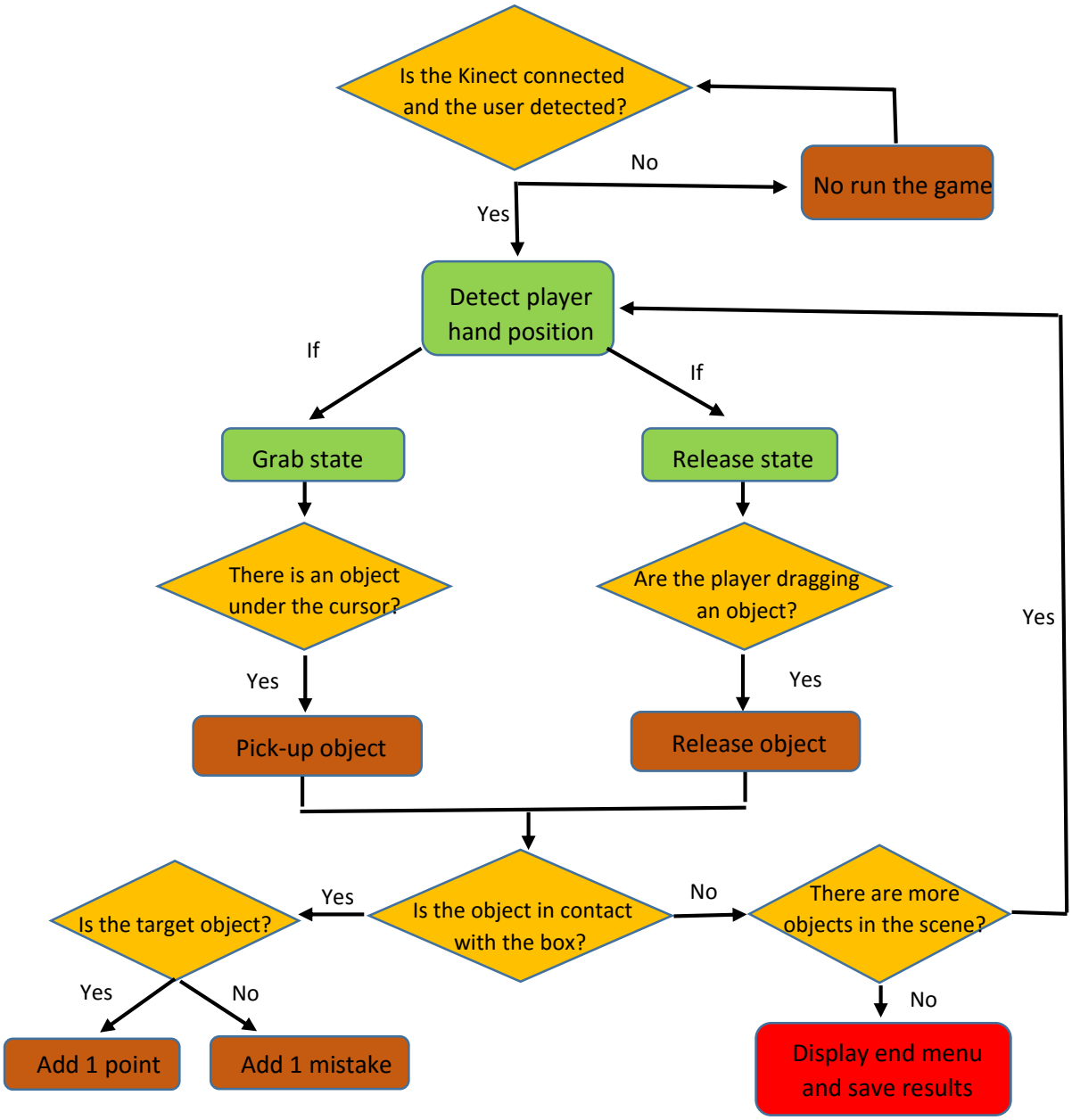

Figure 5.17: Reach Game workflow

5.3.3 The *Balance Game*

This second game obliges the patient to maintain the equilibrium and dodge different hazards waggling the entire body.

5.3.3.1 Game Interface

The game environment is located in a digital forest where an avatar stands facing the user. Behind the main camera there is a “beehive” from which bees go out in the direction of the avatar.

Additionally, a green square lies under the avatar’s feet. This square marks off the area in which the avatar can move in order to dodge the hazards. This area is restricted with the aim of coerce patients to do not move in excess for maintain the equilibrium, enhancing a right posture and balance.

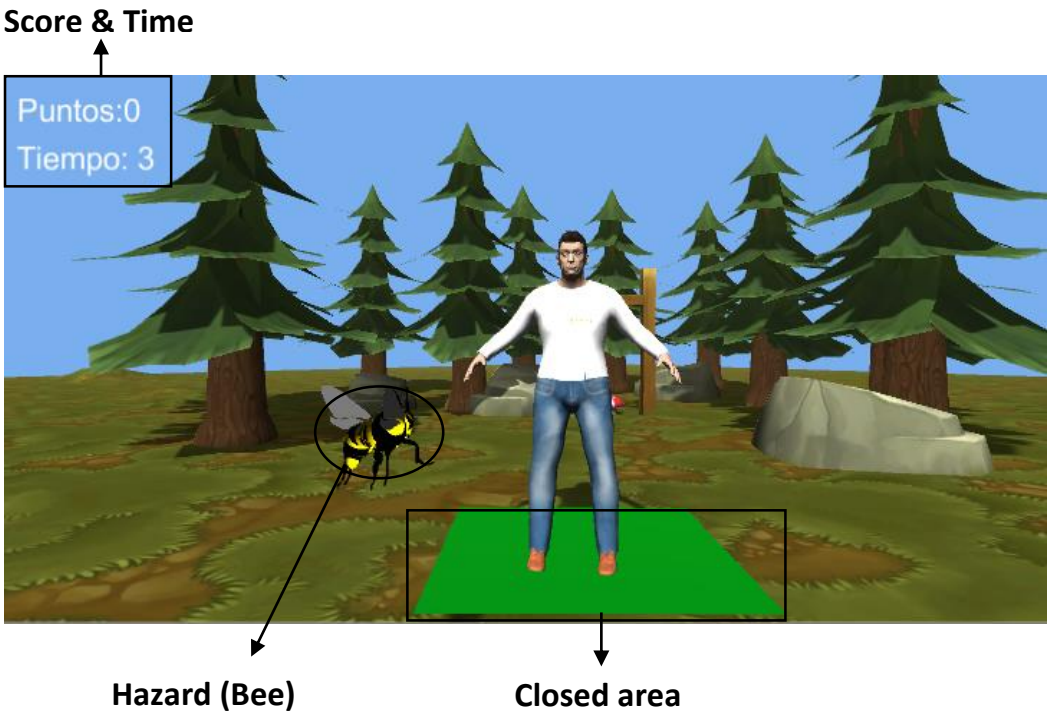

Figure 5.18: The Balance Game interface

5.3.3.2 Game Objective

The aim of this Serious Game is to dodge a wave of bees that fly straight to the avatar. The patient, represented by an avatar that follows the patient’s movements, has to avoid to touch the bees bending the body from one side to other.

If the bee passes the avatar without touching him, one point is added to the scoring, if on the contrary it touches the avatar, the patient is penalized with one mistake.

To avoid the excess of movement by part of the patient to maintain the equilibrium, the green closed area was added underneath the avatar. By this way, when the avatar puts one foot outside the green area, the green square becomes red and the game stops until the patient comes back to the right position.

Figure 5.19: Detection of the avatar outside the square area

5.3.3.3 Game Software

The software basis used in this game, was the script AvatarController.cs, which added as component of the used avatar converts it to a digital mime of user movements.

Avatar controller is the component that transfers the captured user motion to the humanoid model. The script acquire the positions of the different player’s joints and applies them into avatar’s joint, allowing to the avatar move together with the user. At the same time we can decide if the avatar will follow us in a mirrored movement or not, for this game we decided is better that the avatar follow the player with a mirrored movement.

5.3.3.4 *Balance Game* Workflow

Such as the previous Serious Game, game processes and events are made up by different scripts. This part itemize these events listing them and putting all together in a workflow diagram. The script events of this game are:

1. Ensure Kinect detection and the user-avatar connection.
2. Spawn and give movement to the bees.
3. Detect the avatar-bee collision or not collision.
4. Time and scoring.
5. End the game and save results.

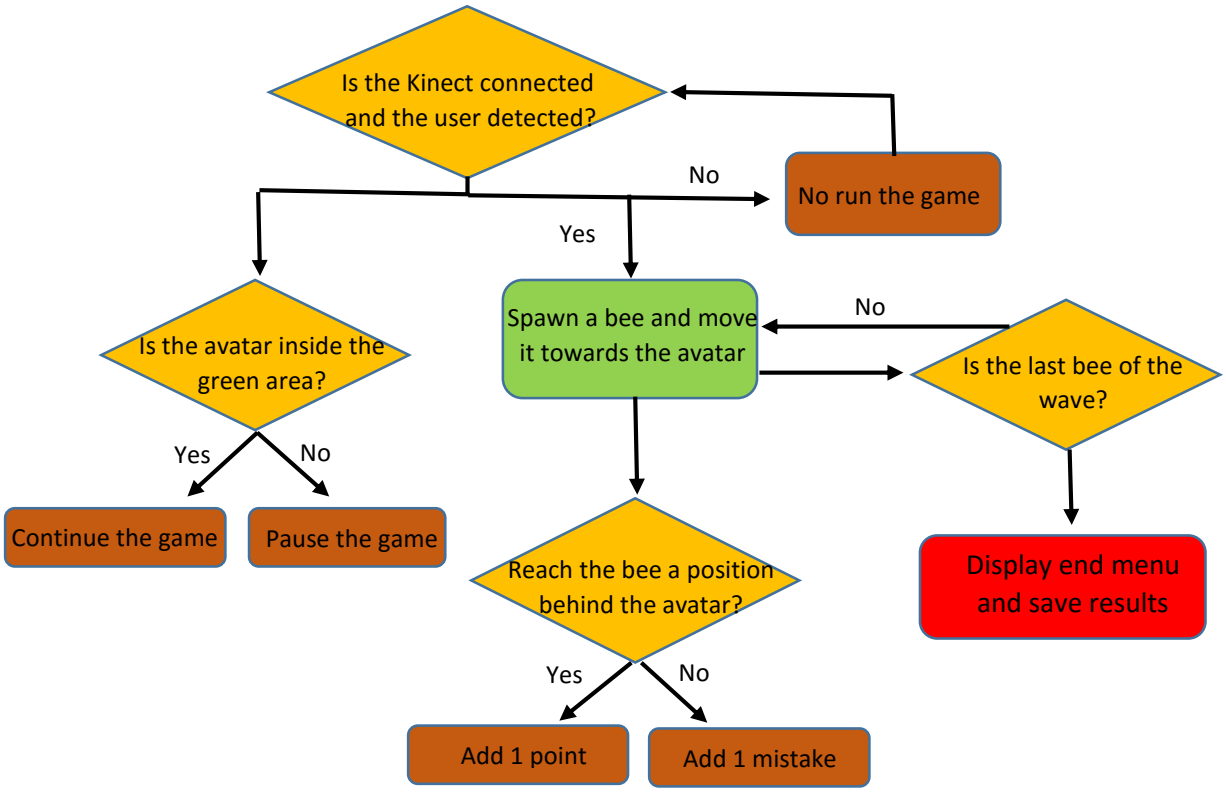

Figure 5.20: Balance Game workflow

6 Conclusions and Future Work

This last chapter analyses and summarizes the work done into the implementation of an Open Source application “Serious Game” for physiotherapeutic purposes, discussing the final result and what objectives have been achieved from the ones initially proposed. In addition to this, several ideas for the future enhancement and research in this topic are showed.

6.1 Conclusions

Regarding the initial objectives, almost all of them have been accomplished, as the creation of an Open Source application, which tracks patient’s movements, integrates this tracking in a dynamic and enjoyable digital environment and lets patients perform motor rehabilitation exercises. However, the designed Serious Games continue being initial versions that need the supervision of the medical team.

The acquisition of patients’ joints coordinates in order to study their position and movements was possible due to the use of Kinect sensor for Xbox One, a low cost sensor designed for the detection of human joints in real time, together with some specific scripts for Unity that extract the coordinates calculated by the Kinect camera during the exercise performance. However, although the extraction of position coordinates directly through the software Unity is relatively accurate, the research group is trying to enhance this task with the use of an alternative software developed with *Matlab* with the unique purpose of extract the patient position information.

Unity is a perfect software for the creation of virtual environments, easy to use and with an intuitive interface. This allows to create funny and dynamic scenes that lead the patients to forget that are in an unpleasant rehabilitation session. The use of a football stadium, a forest or an island as game environment have been already tested along the development of this application, but can be integrated many others.

The motivation of the patients is one of the most important items during a rehabilitation process. To deal with this objective a scoring reward system is implemented in the game together with a timer. This will motivate patients to improve their marks in less time and compete healthily against their rehabilitation mates or friends. In addition, these Serious Games offer to patients a simple way to perform exercises proposed by the physicians, to have fun, and different positive feedbacks such as audio and text messages supporting them.

To store the different results and the patients’ data, finally was developed a Unity script that stores all the useful information in a CSV file. This format file is appropriate because it can be imported to Microsoft Excel, which is an extended software that physicians know how to use, making the data easy to retrieve and check at any moment by them.

The fact that this application is an Open Source app, allows the dissemination of these Serious Games to other people with informatics skills, who can modify and create new games from the created work. Also Open Source means accessible, so anyone can have access to play these Serious Games anywhere, transforming this project in a home-treatment application for neurological disease survivors.

But is true that we can find some limitations in this project, such as the reduced possibility of customize the game by part of the physicians. Also currently this application only is integrated by two Serious Games that not embrace many important rehabilitation exercises

6.2 Experiences and problems occurred during the project development

After some meetings with the work group and the hospital physiotherapists, the first step that the student had to do was the research of a suitable software that fulfilled all the features indicated in this dissertation. It was a long research due to the great number of Game Engines and softwares available for the creation of videogames, but finally Unity software was selected due to its simple interface and workflow.

Once the software was selected, the student learned how to use the Game Engine together with the programming language used for game’s development in Unity, C#. Thanks to the informatics skills that has the student with some programming languages, was not so difficult to learn the basis of this new programming language following tutorials and performing small parallel projects.

One important obstacle was stablish the communication between the Kinect sensor for Xbox One and the software Unity. Finally was an easy task thanks to the possibilities that offers Unity, such as its *Asset Store* that contains different projects made by people, among which the student found the used package *Kinect v2 with MS-SDK* for this project development.

Finally the development of the proposed Serious Games is the final result of developing several demos these months, testing different functions and learning more daily with Unity tutorials.

6.3 Future work

From the conclusions previously mentioned, we can take out different work lines to improve the present project and make of this technology a usual tool in hospitals and rehabilitation centres.

One important issue would be the implementation of a well thought-out interface, that allows to the physician to select the difficulty of the game, custom some game variables as could be the number of repetitions that the patient has to do, include a maximum performance time, etc.

Also is necessary to integrate more Serious Games designed by the physicians that include different therapies for different body parts, in order to create a more professional and versatile rehab application. In addition to this the already created Serious Games can be enhanced improving some functions such as the game-user interaction, add difficult levels and updating them taking into account the physicians opinion and the tests with real patients.

Finally we could bring this application to more public creating a server. This server would allow the access to the Serious Games by part of different hospitals, physicians and patients. Use a server would make of this project, an application more accessible to people, and motivating for the patients with any type of scoring competition between them.

To conclude, we can say that this bachelor thesis is the basis for the development of a complete application that modernizes the actual concept of physiotherapeutic rehabilitation, or at least, can be a useful and a different therapeutic complement for the actual rehabilitation methods using low cost hardware. However this work have to continue with the aim of create a solid and consistent application for something so serious how is the health.

References

[1] D. R. Michael, S. Chen, E. Chen, and L. Chen, *Serious games: Games that educate, train, and inform*. Boston, MA: Thomson Course Technology, 2005. Accessed: Jul. 1, 2016.

[2] J. Wiemeyer and A. Kliem, "Serious games in prevention and rehabilitation—a new panacea for elderly people?" *European Review of Aging and Physical Activity*, vol. 9, no. 1, pp. 41–50, Dec. 2011. Accessed: Jul. 1, 2016.

[3] Farlex, "Stroke," TheFreeDictionary.com, 2003. [Online]. Available: http://medical-dictionary.thefreedictionary.com/stroke. Accessed: Jul. 1, 2016.

[4] R. Wittenauer and L. Smith, "Background paper 6.6 Ischaemic and Haemorrhagic stroke," 2012. [Online]. Available: http://www.who.int/medicines/areas/priority_medicines/BP6_6Stroke.pdf. Accessed: Jul. 1, 2016.

[5] L. Konkel, EverydayHealth.com, 2015. [Online]. Available: http://www.everydayhealth.com/stroke/guide/treatment/. Accessed: Jul. 10, 2016.

[6] "Recovering after a stroke: A patient and family guide," AHCPR Publication, 1995. [Online]. Available: http://www.strokecenter.org/wp-content/uploads/2011/08/Recovering-After-a-Stroke.pdf. Accessed: Jul. 10, 2016

[7] J. J. M. F. van der Putten, J. C. Hobart, J. A. Freeman, and A. J. Thompson, "Measuring change in disability after inpatient rehabilitation: Comparison of the responsiveness of the Barthel index and the functional independence measure," *Journal of Neurology, Neurosurgery & Psychiatry*, vol. 66, no. 4, pp. 480–484, Apr. 1999.

[8] K. Salter, N. Campbell, M. R. Msc, and S. Mehta, "EBRSR [Evidence-Based review of stroke Rehabilitation] 21 outcome measures in stroke rehabilitation," Oct. 2013. Available: http://www.ebrsr.com/sites/default/files/chapter21_outcome-measures_final_16ed.pdf. Accessed: Jul. 26, 2016.

[9] C. A. Velozo and M. L. Woodbury, "Translating measurement findings into rehabilitation practice: An example using Fugl-Meyer assessment-upper extremity with patients following stroke," 2010. [Online]. Available: http://www.rehab.research.va.gov/jour/11/4810/velozo4810.html. Accessed: Jul. 15, 2016.

[10] S. Kokones, K. Carlton, and I. Medical, *Patent WO2009051963A1 - patient programmer with input and sensing capabilities*. Google Books, 2008. [Online]. Available: https://www.google.com/patents/WO2009051963A1?cl=en. Accessed: Jul. 15, 2016.

[11] G. Saposnik and M. Levin, "Virtual reality in stroke rehabilitation: A Meta-Analysis and implications for Clinicians," *Stroke*, vol. 42, no. 5, pp. 1380–1386, Apr. 2011.

[12] "Medicaa balance for life," 2006. [Online]. Available: http://www.medicaa.com. Accessed: Jul. 25, 2016.

[13] "Biotrak". [Online]. Available: http://www.biotraksuite.com/. Accessed: Jul. 25, 2016.

[14] R. V. Rocha, R. V. Rocha, and R. B. Araújo, "Selecting the Best Open Source 3D Games Engines," IX SBGames, Florianópolis, Nov. 2010.

[15] A. Navarro, J. V. Pradilla, and O. Rios, "Open Source 3D Game Engines for Serious Games Modeling," Mar. 2012. [Online]. Available: http://www.intechopen.com/books/modeling-and-simulation-inengineering/open-source-3d-game-engines-for-serious-games-modeling. Accessed: Jul. 3, 2016.

[16] C. Chapple, "The top 16 game engines for 2014," in *http://www.develop-online.net*, 2014. [Online]. Available: http://www.develop-online.net/tools-and-tech/the-top-16-game-engines-for-2014/0192302. Accessed: Jul. 3, 2016.

[17] "Unreal engine," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Unreal_Engine. Accessed: Jul. 4, 2016.

[18] "Unreal engine frequently asked questions," 2004. [Online]. Available: https://www.unrealengine.com/faq. Accessed: Jul. 4, 2016.

[19] "Unity (software)," in *Wikipedia*, Wikimedia Foundation, 2014. [Online]. Available: https://es.wikipedia.org/wiki/Unity_(software). Accessed: Jul. 6, 2016.

[20] U. Technologies, "Unity - fast facts," 2016. [Online]. Available: https://unity3d.com/es/public-relations. Accessed: Jul. 6, 2016.

[21] U. Technologies, "Unity". [Online]. Available: https://store.unity.com/es. Accessed: Jul. 6, 2016.

[22] "CryEngine," in Wikipedia, Wikimedia Foundation, 2016. [Online]. Available: https://es.wikipedia.org/wiki/CryEngine. Accessed: Jul. 26, 2016.

[23] C. GmbH, "CRYENGINE," 2016. [Online]. Available: https://www.cryengine.com/get-cryengine. Accessed: Jul. 26, 2016.

[24] W. authoring, "Blender (software)," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Blender_(software). Accessed: Jul. 28, 2016.

[25] B. Foundation, "Blender.org," blender.org, 2016. [Online]. Available: https://www.blender.org/. Accessed: Jul. 28, 2016.

[26] "MakeHuman," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/MakeHuman. Accessed: Jul. 28, 2016.

[27] "Open source tool for making 3d characters,". [Online]. Available: http://www.makehuman.org/index.php. Accessed: Jul. 28, 2016.

[28] A. Shingade and A. Ghotkar, "Animation of 3D human model using Markerless motion capture applied to sports," *International Journal of Computer Graphics & Animation*, vol. 4, no. 1, pp. 27–39, Jan. 2014.

[29] "Kinect," in *Wikipedia*, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Kinect. Accessed: Aug. 4, 2016.

[30] Microsoft, "Kinect for windows sensor components and specifications,". [Online]. Available: https://msdn.microsoft.com/en-us/library/jj131033.aspx. Accessed: Aug. 6, 2016.

[31] T. Carmody, "How motion detection works in Xbox Kinect," in *Gadget Lab*, WIRED, 2010. [Online]. Available: http://www.wired.com/2010/11/tonights-release-xbox-kinect-how-does-it-work/. Accessed: Aug. 6, 2016.

[32] "PlayStation eye," in Wikipedia, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/PlayStation_Eye. Accessed: Sep. 4, 2016.

[33] S. Stocker, "PlayStation eye, A little more Info…" PlayStation.Blog, 2007. [Online]. Available: http://blog.us.playstation.com/2007/10/10/playstation-eye-a-little-more-info/. Accessed: Sep. 4, 2016.

[34] "PRIMESENSE IS CHANGING THE WAY WE INTERACT WITH DIGITAL DEVICES PrimeSense™ 3D sensors ®".[Online]. Available: http://www.i3du.gr/pdf/primesense.pdf. Accessed: Sep. 4, 2016.

[35] Microsoft, "Kinect hardware," 2016. [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/hardware. Accessed: Sep. 5, 2016.

[36] Rjones, "Buy Kinect adapter for windows," Microsoft Store. [Online]. Available: https://www.microsoftstore.com/store/msusa/pdp/Kinect-Adapter-for-Windows/productID.308803600. Accessed: Sep. 5, 2016.

[37] "Plug-in (computing)," in Wikipedia, Wikimedia Foundation. [Online]. Available: https://en.wikipedia.org/wiki/Plug-in (computing). Accessed: Sep. 5, 2016.

[38] Microsoft, "Developing with Kinect for windows,". [Online]. Available: https://developer.microsoft.com/en-us/windows/kinect/develop. Accessed: Sep. 5, 2016.

[39] M.-S. In. CO, "GT72 2QE Dominator pro | MSI España | laptops - the best gaming laptop provider," msi.com, 2015. [Online]. Available:

https://es.msi.com/Laptop/GT72-2QE-Dominator-Pro.html#hero-specification. Accessed: Sep. 8, 2016.

[40] U. Technologies, "Unity - manual: Learning the interface," 2016. Available: http://docs.unity3d.com/Manual/LearningtheInterface.html?_ga=1.210288797.986860033.1464543497. Accessed: Sep. 8, 2016.

[41] R. F, "Kinect v2 examples with MS-SDK," RFilkov.com - Technology, Health and More, 2014. [Online]. Available: https://rfilkov.com/2014/08/01/kinect-v2-with-ms-sdk/. Accessed: Sep. 11, 2016

[42] "Wrapper function," in Wikipedia, Wikimedia Foundation, 2016. [Online]. Available: https://en.wikipedia.org/wiki/Wrapper_function. Accessed: Sep. 15, 2016.

Annex

A.1 Work Hours Breakdown

The tasks carried out in the project will be divided in three phases to ease the calculation of the total time spent in the development of the project:

1. **Documentation**
   a) Meetings with the work team and the physiotherapists (20 hours)
   b) Learning to use Unity and to program in C# with tutorials (50 hours)
   c) Implementation of Kinect hardware with Unity software (5 hours)

2. **Development of the project**
   a) Development of basic prototypes with essential functions (40 hours)
   b) Development of more complex prototypes of Serious Games (75 hours)
   c) Implementation of the work done to create final Serious Games (45 hours)

3. **Writing the dissertation**
   a) Research of other projects and studies about Virtual Reality rehab (15 hours)
   b) Writing the bachelor thesis memory (100 hours)
   c) Revision and tutor meetings (5 hours)

Task	Hours
Documentation	75
Development of the project	160
Writing the dissertation	120
TOTAL	355

Table A.1: Total thesis hours breakdown

A.2 Project budget

The estimation of the total budget taking into account the direct and indirect costs. The direct costs are those directly related with the development of the project, such as the labor costs, equipment and software tools. On the other hand, the indirect costs are those committed for the project development, but are not directly linked with the development itself. Following the stencil, this costs correspond to the 15% of the total direct costs.

A.2.1 Personnel

In this subsection are itemized the personnel costs. For the calculation of this costs we take into account the participation of only one engineer, working during four months and a half in a part-time journey with a salary of 600€ per month.

Name	Position	Dedication (worker/month)	Cost per month	Total cost
Herreros Díaz, Jaime	Engineer	4,5	2.694,39€	2.700€

Table A.2: Personnel costs

A.2.2 Equipment

In this part are detailed the costs related with the equipment used for the project.

Description	Cost	% Use for the project	Dedication (months)	Devaluation period (60 months)	Imputable[1] cost
MSI GT72 2QE DOMINATOR PRO	1.999€	100	2	60	66,63€
Microsoft Sensor Kinect Xbox One	149,99€	100	4	60	10€
Sensor Kinect Xbox One Adapter	50€	100	4	60	3,33€
				Total	79,97€

Table A.3: Amortization Equipment cost

[1] Formula for the Amortization calculation

$$\frac{A}{B} \times C \times D$$

A = nº of months since the equipment begins to be used
B = depreciation period (60 months)
C= equipment cost (without IVA)
D = % of use during the project development

A.2.4 Estimation of total costs

Using the costs calculated previously, and adding the indirect costs (20%) and taxes (IVA) we calculate the final costs of this project.

Concept	Cost
Personnel	2.700€
Amortization	79,97€
Total direct costs	2.779,97€
Indirect costs	377,21€
Costs without IVA	3157,18€
IVA (21%)	663,01€
Total	3820,19€

Table A.4: Estimation of total costs

A.3 *GetJointPosition.cs*

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

namespace HutongGames.PlayMaker.Actions
{
	[ActionCategory("Kinect Actions")]
	[Tooltip("Allows you to get a user's joint position from the Kinect.")]

	public class GetJointPosition : FsmStateAction
	{
		public enum JointType : int
		{
			HipCenter = 0,
			Spine,
			ShoulderCenter,
			Head,
			ShoulderLeft,
			ElbowLeft,
			WristLeft,
			HandLeft,
			ShoulderRight,
			ElbowRight,
			WristRight,
			HandRight,
			HipLeft,
			KneeLeft,
			AnkleLeft,
			FootLeft,
			HipRight,
			KneeRight,
			AnkleRight,
			FootRight
		}

		// Setup to use the enum udpate
		[RequiredField]
		[Tooltip("Which joint you want to track.")]
		public JointType kinectJoint;

		// store the name of the gesture called
		[UIHint(UIHint.Variable)]
		[Tooltip("Store the position of the joint selected.")]
		public FsmVector3 jointPosition;

		private KinectManager manager;
		private int kinectJointIndex;


		// called when the state becomes active
		public override void OnEnter()
		{
			kinectJointIndex = (int)kinectJoint;
			getKinectJointPos();
		}

		// called before leaving the current state
		public override void OnExit ()
		{
```

```
		}

		public override void OnUpdate()
		{
			//if (updateCall == PlayMakerUpdateCallType.Update)
			{
				getKinectJointPos();
			}
		}

		// Update is called once per frame
		private void getKinectJointPos()
		{
			if(manager == null)
			{
				manager = KinectManager.Instance;
			}

			if(manager != null && KinectManager.IsKinectInitialized() &&
manager.GetPrimaryUserID() > 0)
			{
				long userId = manager.GetPrimaryUserID();
				jointPosition.Value = manager.GetJointPosition(userId,
kinectJointIndex);
			}
		}
	}
}
```