UNIVERSITAT POLITÈCNICA DE VALÈNCIA

_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DEVELOPMENT OF A VIRTUAL ENVIRONMENT FOR FULL-BODY MOTOR TRAINING

Yari Mirko Anoffo

Tutor: Beatriz Rey Solaz

Cotutor: José María Monzó Ferrer

Trabajo Fin de Máster presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Máster en Ingeniería Telecomunicación

Curso 2018-19

Valencia, 03 de julio de 2019

Escuela Técnica Superior de Ingeniería de Telecomunicación
Universitat Politècnica de València
Edificio 4D. Camino de Vera, s/n, 46022 Valencia
Tel. +34 96 387 71 90, ext. 77190
www.etsit.upv.es

VLC/ CAMPUS
VALENCIA, INTERNATIONAL CAMPUS OF EXCELLENCE

EMAS
Gestión ambiental verificada
REG.NO. ES-CV- 000030

Abstract

The purpose of this project is to create a motor training system combining virtual reality, provided by Oculus Rift, and body tracking, provided by Kinect v1 sensor. Using an immersive experience in a virtual environment, the patient will complete exercises with a simple system, in which the trajectory to be followed will be guided by the appearance of a set of spheres. Patients will have a first person view in this application and they will also see their avatar in a mirror in front of them to look how they are performing the exercise and to correct themselves in any moment.

Therapists will be able to add new exercises and new training sessions with a description in a text file without any programming skill. In this way, it will be possible for them to create custom training protocols based on a specific rehabilitation therapy or also to adapt them for a specific patient. The system will automatically recognize text file changes and it will update the exercises in the application accordingly.

Keywords – virtual reality, body tracking, motor training.

Resumen

El propósito de este proyecto es crear un sistema de entrenamiento motor que combine la realidad virtual, por medio del sistema Oculus Rift, y el seguimiento del cuerpo, por medio del sensor Kinect v1. Usando una experiencia inmersiva en un entorno virtual, el paciente completará los ejercicios con un sistema simple, en el que la trayectoria a seguir vendrá guiada por la aparición de una serie de esferas. En esta aplicación, los pacientes tendrán una vista en primera persona y también visualizarán a su avatar en un espejo frente a ellos para ver cómo están realizando el ejercicio y para corregirse en cualquier momento.

Los terapeutas podrán agregar nuevos ejercicios y nuevas sesiones de entrenamiento con una descripción en un archivo de texto sin ser necesaria ninguna habilidad de programación. De esta manera, podrán crear protocolos de entrenamiento personalizados basados en una terapia de rehabilitación específica o también adaptarlos para un paciente ado. El sistema reconocerá automáticamente los cambios en los archivos de texto y, en consecuencia, actualizará los ejercicios en la aplicación.

Palabras claves: realidad virtual, seguimiento del cuerpo, entrenamiento motor

Resum

El propòsit d'aquest projecte és crear un sistema d'entrenament motor que combine la realitat virtual, per mig del sistema Oculus Rift, i el seguiment del cos, per mig del sensor Kinect v1. Utilitzant una experiència immersiva a un entorn virtual, el pacient completarà els exercicis amb un sistema simple, en el qual la trajectòria a seguir vindrà guiada per l'aparició d'un conjunt d'esferes. En aquesta aplicació, els pacients tindran una vista en primera persona i també visualitzaran el seu avatar a un espill situat en front d'ells per tal de veure com estan realitzant l'exercici i per a corregir-se en qualsevol moment.

Els terapeutes podran agregar nous exercicis i noves sessions d'entrenament amb una descripció en un arxiu de text sense ser necessària cap habilitat de programació. D'aquesta manera, podran crear protocols d'entrenament personalitzats basats en una teràpia de rehabilitació específica o també adaptar-los per a un pacient donat. El sistema reconeixerà automàticament els canvis en els arxius de text i, en conseqüència, actualitzarà els exercicis en l'aplicació.

Paraules claus: realitat virtual, seguiment del cos, entrenament motor

Index

Abstract 3
Introduction 8
Chapter 1. Objectives 11
Chapter 2. Methodology 13
2.1 Project Management 13
2.2 Task distribution 13
2.2.1 State of the art 13
2.2.2 Study of devices and software 13
2.2.3 Body tracking application 14
2.2.4 Avatar creation 14
2.2.5 Mirror creation 14
2.2.6 Virtual Reality implementation 14
2.2.7 Rehabilitation VR application development 14
2.2.8 Report writing 15
2.3 Temporal diagram 15
Chapter 3. Development and results 17
3.1 Development 17
3.1.1 State of the art 17
3.1.2 Study of devices and software 18
3.1.2.1 Oculus Rift 18
3.1.2.2 Kinect sensor 19
3.1.2.3 Blender 21
3.1.2.4 Adobe Fuse CC 21
3.1.2.5 Makehuman 21
3.1.2.6 Unity 22
3.1.2.7 Microsoft Visual Studio 23
3.1.3 Body tracking application 23
3.1.4 Avatar creation 25
Avatar bones implementation 26
Unity implementation 28
3.1.5 Mirror creation 30
3.1.6 Virtual Reality implementation 31

3.1.7 Rehabilitation VR application 33
JSON file 34
Exercise execution 35
Character selector 35
Sphere 39
Collision 40
3.2 Results 40
Chapter 4. Conclusions and future work 46
Future works 47
Bibliography 49
Annexes 51

Introduction

Injuries, accidents or incorrect posture can cause physical traumas that will affect people health. Motor rehabilitation is a procedure designed to help patients to return to an optimal physical condition. Usually, this procedure consists in custom exercises that depend on the specific rehabilitation therapy and it can involve one or many body parts. Depending on the trauma, it can be a slow procedure.

Previous research has developed different applications to perform motor rehabilitation with full body tracking or virtual reality.

KiReS [1] is one example of motor rehabilitation that uses Kinect to perform exercises. Users can see an avatar performing the exercise and the patient has to replicate that. It's also possible to add another exercise recording user movements.

ReaKinG (Rehabilitation using Kinect-based Games) [2] is another Kinect based game developed to improve elderly people health by means of physical training. The idea is to create the environment as a videogame because it makes users get motivated and get more involved during game execution [3]. This application uses Kinect v2 and users can perform exercise to improve or rehabilitate their strength or aerobic capability. Exercises can be sent remotely by the therapist to the patient computer to update training sessions.

These applications use a non-immersive form of virtual reality to perform rehabilitation.

There are virtual reality applications combined with Kinect that are applied to perform motor rehabilitation after stroke using virtual scenarios to improve balance, strength and other cognitive and physical capabilities [4]. Most applications are focused on the use of virtual reality for upper limb or stroke rehabilitation[5-6-7].

A virtual reality application for motor rehabilitation that offers full body tracking can help users to perform this procedure in a different way than usual. It offers the opportunity to change scenario based on patient preferences and to have a more comfortable virtual place to perform the exercises. In a virtual environment, there are also infinite opportunities to create new exercises, based on the rehabilitation therapy chosen for a specific problem. These exercises can be proposed as videogames, thus including aspects such as the change of scenario, selection of different characters to play, object interaction and achievement of specific goals.

The creation of an automatic and repetitive system allows users to independently perform exercises or training sessions and to repeat them as many times as he or she wants. In this way, the therapist's workload is reduced and he or she can help more patients. On the other hand, patients can carry out their therapy session monitored by the system but also have the opportunity of performing the exercises at home in a more comfortable environment [1]. They can also perform exercises every day without reducing meetings with the therapist and also reducing their displacements, which is very important in some types of traumas.

Most common devices in the market, such as Oculus Rift[8], offer head and hand tracking, optimal display definition and a high refresh rate, essential characteristics to get a free from motion sickness virtual experience and to have a flowing application execution. HTC Vive[9], another virtual reality device present in the market, with similar features of the Oculus Rift, can

perform body tracking of different body parts using additional tracking devices[10]. However, using devices such as Microsoft Kinect v1 [11], which uses depth images to track human body position, it is possible to overcome this gap without using additional sensors on the patient's body. Besides, this is also a non-invasive way to perform exercises, as patients only have to wear a head mounted display to enter inside the virtual world but they don't have to wear anything to detect their bodies. This will be the approach for full-body tracking that will be analyzed in the present project.

First person view will help to create a more immersive experience for patients that will recognize themselves as avatars inside the virtual world generating a funnier and more realistic version of the exercise.

The document is organized in four chapters that talk about:

1. A short view about project objectives
2. Methodology chosen and task subdivisions
3. Project development and obtained results
4. Conclusions and future implementations

Chapter 1. Objectives

The project, focused on creation of a virtual reality application for motor rehabilitation, can be subdivided in three main goals:

1. To explore virtual rehabilitation based techniques
2. To develop a full body tracking system integrated with Oculus Rift
3. To define and develop a rehabilitation VR based application

The first goal is to review previous research or commercial applications that use virtual reality or full-body tracking in rehabilitation therapies to analyze the approaches these previous works are using.

This evaluation of the state of the art is necessary to have a knowledge about existing systems, which will allow us to create a new one based on this previous knowledge taking into account aspects that can be improved in previous applications..

Oculus Rift doesn't provide any kind of full body tracking system at the moment. It can track only user head and hands without giving any data of other body parts. The second objective is focused on finding solutions to reduce this gap combining it with other compatible devices that can be used in Unity, the graphical editor software, chosen to develop the virtual reality application.

The full body tracking system is closely related with the creation of avatars with the right structure to be used in virtual scenes. Avatars will have to reproduce user movements that are recognized with the full body tracking system.

The last objective is focused on the creation of a rehabilitation application based on the full-body tracking system. This application should allow the execution of different kinds of exercises. It must be an intuitive system for users of all ages, avoiding complex actions or interactions with the system. All exercises will be created following the therapist advice, not only to know the right way to develop them but also to provide an easy way for patients to feel immediately comfortable with this type of training system.

Chapter 2. Methodology

In this chapter the methodology used to develop the application will be described, indicating how the work is subdivided in different tasks and detailing the time schedule of the project.

2.1 Project Management

The project is subdivided in tasks. All these tasks together will create the final version of the application. There are also chosen deadlines for each task to create a time schedule to follow. If there are problems during development, this time schedule will be updated.

Every week, a meeting with the supervisors will be conducted to know the state of the project, evaluate the advances, analyze new ideas or look for solutions to any problem that has appeared during the week.

The methodology chosen to develop the project is the throwaway prototyping that will bring a first version to the user that will give a feedback about his or her experience. Thanks to this feedback, the developer can change the application based on the user report and create a better version after every meeting. Following this methodology, during this project, the developer tries to achieve main goals with prototypes. Then, he asks user feedback, in this case, the supervisors, because they can give many data about development and help to improve the final application.

Exercise development follows the specifications given by the supervisors of the project and a neurologist from the Servicio de Neurología of the Hospital Universitari i Politècnic La Fe.

2.2 Task distribution

2.2.1 State of the art

In this first task, related to the first objective of the project, bibliography and commercial applications will be reviewed in order to detect applications that perform motor rehabilitation with virtual reality, using Oculus Rift or other similar devices, and Kinect. This task will help to decide the final requirements of the project based on the state of the art, providing new ideas and complementing previous approaches.

2.2.2 Study of devices and software

Project development starts with the study of all the devices and software that are required to achieve the second and the third goal of the project. These will be Oculus Rift, Kinect v1, their respective Software Development Kit (SDK) and Unity.

Regarding Oculus Rift and Kinect, it is necessary to know how they capture and elaborate data and which documentation is available develop applications with these devices.

Unity [12] and its documentation is needed to create all the virtual environment. This software can create three-dimensional objects, manage colors, interactions and animations using software tools or also using C# scripts.

2.2.3 Body tracking application

This task is dedicated to full body tracking application development using only the Kinect without any virtual reality implementation yet. Developing this application the second objective of the project will be satisfied. The goal of this application is that an avatar reproduces user movements according to the full body tracking performed by Kinect. This task is performed using Unity to create the three-dimensional environment and to manage the avatar or other objects inside the Unity scene.

2.2.4 Avatar creation

The idea is to create a personalized avatar for each user. Avatars will be used as user representation. It is important to understand how built these avatars to work with the full body tracking application and move accordingly inside a virtual environment. Another important aspect to analyze is how to create the avatar to recreate fluid movements.

2.2.5 Mirror creation

Users have a first person view during training execution but they also need to see the avatar body to see themselves and correct their position or movements. This mirror will reflect all the objects of the scene in front of it and it will be used in the body tracking application and in the final application version with virtual reality.

2.2.6 Virtual Reality implementation

Oculus Rift can't perform body tracking because it can track only user head and hands position inside a tracked area. It can also detect head rotation that allows user to see around himself in the virtual environment. On the other hand, Kinect sensor can perform body tracking without any data about user position in the space. The purpose of this task is to create an application that combines Oculus Rift and the Kinect body tracking application to achieve a virtual body tracking representation.

Thanks to this implementation, the user will be able to have a vision of the virtual environment in a first-person view. He or she will be free to move their real bodies and see how avatar can reproduce their movements inside the virtual scene, obviously with devices limitations.

2.2.7 Rehabilitation VR application development

This task corresponds to the final rehabilitation application, that is also the last objective to achieve, which includes the exercise execution. The idea is to create a route made of spheres to guide a patient to perform an exercise. Before the exercise starts, there is a short description of

the exercise to make the user understand which body part or body parts will be trained, and how the movements should be performed

From a main menu, the patient will choose which training session wants to perform and also the avatar to be used. After this, the training will start. As decided with the neurotherapist, the application must create a path to help user to perform the exercise. The application needs also to be repetitive so at the end of a training session the patient can repeat it from the main menu.

2.2.8 Report writing

Each task is described by a report about all activities done, problems and how they are solved. These reports are updated during all project to have a complete knowledge about work development. The present document is based on all the partial reports that have been generated during the project.

2.3 Temporal diagram

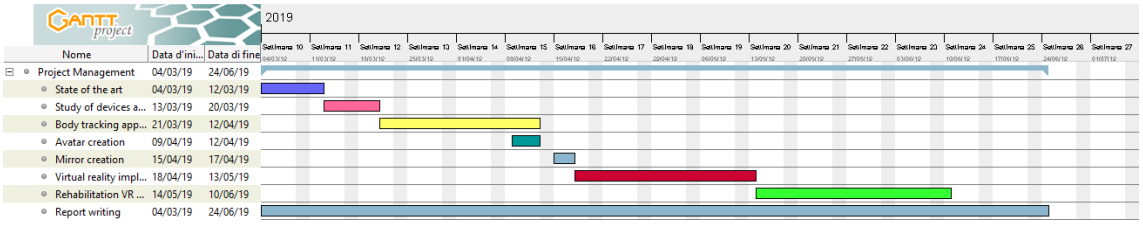

Figure 1. Gantt diagram created with Gantt project.

Chapter 3. Development and results

In this chapter, the development of the project in each phase is explained.

It is subdivided in two simple subsections to define all development phases and the devices and software used during each phase, the approach to create the application and, at the end, the results of the work. Unity documentation [13] and Oculus developer documentation [14] have been used during all the development phases.

3.1 Development

This section describes all the development phases to achieve the project conclusion. Subcategories of this chapter have the same name of tasks.

3.1.1 State of the art

Reading in bibliography about other virtual reality applications for rehabilitation [1,2,3,5,6,7] was the first stage of the project. Many of the previous applications use no-immersive approaches or are developed only with one device, Kinect or just a virtual reality device. The Kinect sensor is usually combined with a big display where the user can see his or her avatar and move it. These types of applications use only body track detection to perform exercise without create an immersive environment for the user [1,2].

Kires [1] and ReaKinG [2] are two examples of these application, thought for patients of all ages. Both of them use Kinect sensor v2 to detect patient body. The idea is to develop non-invasive application to perform rehabilitation exercises. A therapist can store new exercises recording them in front of the Kinect. The application has a database that recognize exercises and compare them with its data set. This evaluation is done with an algorithm that compares user initial position, final position and trajectories of joints.

ReaKinG [2] uses an architecture thought to be a distributed system. Therapist can send exercise directly to patient station, also at home, and then receive data. The exercise here are developed with the purpose to train aerobic or strength skills of the patient. In the first case, users have to walk at place and try to catch coins that will appear in the scene. In the second case, patients will perform exercises like in a gym. All data are then stored and the therapist is informed about that.

Kinect application can also used for rehabilitation after hemiplegic stroke [3]. Patients are more motivate seeing feedback valuation during their training or hearing therapist motivation. In this case, the application provides training based on sports such as:

- Football to move lower extremity and improve the balance
- Table tennis where patients use limbs to catch the ball, improve their balance and also trunk rotation

Data are then collected and evaluated with different scales for motor rehabilitation and balance. In the first case, patients didn't demonstrate any improvement unlike balance scale where they demonstrate a good improvement [6].

Virtual reality systems have also been developed, but, usually, only the upper limb can be trained and this excludes the legs [4,7]. Patients have a different perception performing exercise inside a

virtual environment. In this case the application uses also custom gloves that track hands and reproduce their movements. Exercises performed in this application are very simple, such as reaching a point with the hand. As a result, patients appreciate the new type of training but without perceiving a higher sense of presence in the scene.

Other applications can use both of these technologies and create virtual environments that can be used adding real objects like bars or delimited path in real world [5]. Body tracking data detected from Kinect sensor in this case are sent to HMD for virtual reality that can reproduce user body movements. One example is an exercise to improve balance where the user has to walk on a sort of suspended bridge. In the real world, a similar structure is set to help user to achieve the exercise. In this case, virtual reality is mixed also with real objects to accomplish the exercise.

The application created in this project wants to use only virtual elements and create a first person view to give users the feeling of being inside the game. This higher sense of presence can help not only physical rehabilitation but also the treatment adherence.

3.1.2 Study of devices and software

In this point, all devices and software to be used during project development were evaluated. In the following paragraphs, we provide a summary of the results of this evaluation.

3.1.2.1 Oculus Rift

It's a virtual reality headset, a head-mounted display (HMD), used to have an immersive virtual experience. This isn't a standalone device, it also needs two camera sensors and two joypad called Oculus Touch. Oculus offers 6 degrees of freedom (6DoF), which means that not only head's movements but also user position is tracked. In figure 3, Oculus Rift HMD and Oculus Touch are shown.

HMD has a pair of vibrant OLED displays with a resolution of 1080x1200 pixel for each eye, so the total display has a resolution of 2160x1200. The refresh rate is 90 Hz, important characteristic that consists in the speed to refresh the image to visualize on the displays. If the refresh rate is too slow, the virtual experience can cause motion sickness, headache and nausea because images and head movement aren't matched and this causes problems to users.

Using a value of 90 Hz or higher is a good compromise to obtain good virtual reality experience.

Oculus' field of view (FOV) is 110° and the HMD structure covers completely human eyesight.

Humans have a FOV of 190-195° so Oculus Rift, but also other virtual reality devices, can't cover all of it. To solve this problem, images shown in Oculus Rift lenses are scaled and elaborated to be adapted to human eyes FOV.

Each eye has the same image but shifted to left or right, depending on the eye, and this permits our brain to interpret images with depth like in the real world.

The two camera sensors, figure 2, are used to track devices inside an area, at least 1.5x3.4 meters. Tracking system is called "Constellation" because each tracked device, HMD and Oculus Touch, has its own infrared LED constellation under the plastic. Infrared are invisible to the human eye but the two camera sensors have an infrared camera that detects only infrared LEDs and sends

frames to the workstation that elaborates them to detect the devices' position. That's possible because each infrared LED has a different frequency to be identified.

Figure 2. Oculus Rift camera sensors.

Oculus Touch are motion tracked controllers which use the same system of the HMD to be tracked from base stations. Thanks to sensor mounted on them, they can track and reproduce a few hands' gestures like fist or indicate something. It's possible to use an Xbox One Controller instead of the Oculus Touch [8].

Figure 3. Oculus Rift and Oculus Touch.

3.1.2.2 Kinect sensor

The Kinect used is the v1, figure 4, a compatible version with Xbox 360, which is no longer produced. This device is composed of one RGB camera, one infrared camera and a laser projector with a wavelength of 830 nm. It needs an USB 3.0 to work well with Windows 10 and a DC power supply.

The infrared camera detects infrared patterns, created from the laser projector, to reproduce the object or the body. RGB camera adds image colors to the detected objects.

Figure 4. Kinect sensor v1.

The field of view of the depth camera is 57° horizontally and 43° vertically with a resolution of 320x240 pixels, an average of 5x5 per degree.

Kinect can track a full human body thanks to IR (infrared) reflection. IR projector produces an IR pattern that is captured by IR camera, the system measures the time flight to send and receives IR reflection to know the position of each point. With this data, a depth map is created and every pixel contains information of point distance from camera. From this image, the software tries to recognize 20 joints that define human body. This is done using different human body shapes and sizes taken from a large database.

While RGB camera has a field of view of 62° horizontally and 49° vertically with a resolution of 640x480 pixels at 30 frame per seconds, an average of 10 x 10 pixels per degree.

The minimum distance supported from the depth camera is 40 cm, the maximum is 4.5 meters but to have good performance is better to stay between 1 and 3.5 meters. More distance means also less accuracy. If there isn't much space in the room, there is also a pair of plug and play lenses, called Zoom, that can reduce the play range required of 40% [15]. The room used in this project hasn't much space so this device was used to improve body detection provided by Kinect.

Figure 5. Zoom for Kinect sensor

Microsoft provides also an SDK, Software Development Kit, where a skeleton app is present. This app uses 20 joints to track the human body, hands and thumbs excluded, with a maximum of two skeletons. Kinect camera, as said, creates a depth map and from that can recognize a human shape that it is calculated as a skeleton with 20 joints. Each frame depth map changes and also

joints position. Kinect system uses this changes to know new joints position and can elaborate skeleton gestures based on this data, creating in this way fluid skeleton movements[3,16].

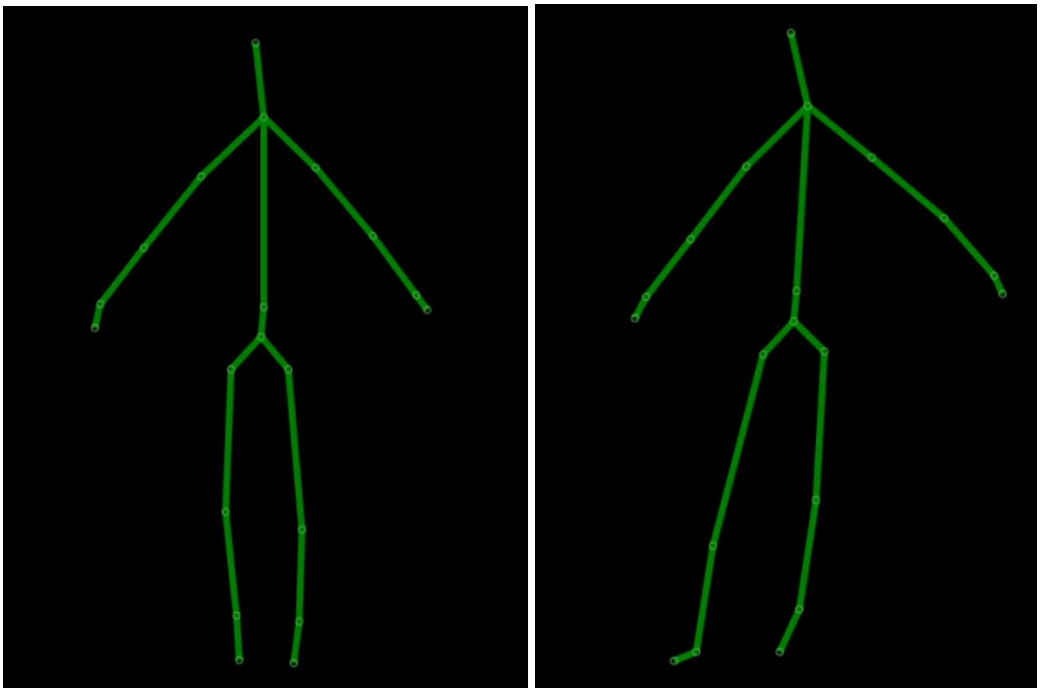

Figure 6. Skeleton detection performed with Kinect SDK.

3.1.2.3 Blender

It is a free and open source software to create 3D applications. It's cross platform so it could be used on Linux, Windows and Mac [17]. To use it, 8 GB RAM, a HD display or better and a 64-bit quad core CPU are recommended. It's also possible combine various scenes into a unique file with a *".blend"* extension. But it also provides compatibility with other extension (obj, fbx, Nan, etc).

Thanks to Blender, it is also possible to model, animate, rig, simulate and render scenarios, avatar or objects. The big community that supports Blender offers a lot of information and help about the use of Blender and its tools. An example, also used in this master thesis, is "Rigify" a tool to create avatar bones.

3.1.2.4 Adobe Fuse CC

Software provided by Adobe that simplifies avatar creation. It's possible to choose many features for avatars and then import them in 3D formats (obj, fbx, etc). Still in beta version, this software can also generate a preview of avatar movements such as walk, jump, run and others. [18]

3.1.2.5 Makehuman

Open source software to create avatars. It can be used in any operating system. It is programmed in Python.

To create avatars, Makehuman uses a morphing method: starting from a standard human avatar, it is possible to modify its physiognomy, its clothes, its gender and its age. The avatar creator can

also try different poses and muscular movements for his avatar. It's also possible add to a skeletal structure directly here and export it into different 3D extensions [19].

Figure 7. Example of avatar.

3.1.2.6 Unity

Developed by Unity Technologies, Unity is a cross-platform game engine used to develop games in two or three dimensions. It is also used to create virtual reality environments. In this project, version 2018.2.15f1 is used. It is possible to use Unity for free and, thanks to its community, it is possible to share ideas and find various solutions for your own project. It supports many file formats, for example OBJ, FBX, CAD and other three-dimensional formats.

It is a cross platform engine, so games or applications developed with Unity can be used in different game consoles, different operative systems or mobile devices with any operating system.

Every object in Unity is called "gameobject". These are like containers which need properties to become a character, a special effect or something that can be used in the scene. These additional properties are called components.

.A group of gameobjects is contained in a scene. Each scene is like a different level in the final application. Scenes are useful to subdivide work and split into different parts. Thanks to Unity, it is possible to create and animate objects inside a scene, add audio or video effects, create 2D or 3D objects. It manages a mesh renderer that is used to show objects colors or hide them. It also manages colliders, which are abstract volumes used to detect collision between two or more objects during game execution. To interact with the application it is possible to create panels for user's interactions. All these actions can be done with simple native tools already existing in Unity or using programming code written in C#, Java or UnityScript and attaching them to the objects inside the scene [12].

3.1.2.7 Microsoft Visual Studio

It is an IDE, Integrated Development Environment, created by Microsoft. It supports 36 programming languages such as C++, C#, JavaScript, XML or HTML.

This IDE usually is used to develop parts of websites, web apps or mobile apps. In combination with Unity, Microsoft Visual Studio allows to create scripts that manage objects actions or interactions inside a scene. It isn't properly open source but freemium, it is possible to use that for free but to add more features the developer needs to pay [20].

3.1.3 Body tracking application

After having finished the state of the art and the evaluation of hardware and software required for the project, the development phase started. In the first step, the body tracking application with Kinect and Unity had to be developed.

Kinect sensor is used to develop body tracking applications and it was necessary to know if it is compatible with workstation and Unity software.

To be able to use Kinect, it's necessary to download its SDK from Microsoft website [21]. Another requirement is to use an USB 3.0 port. Kinect SDK offers native applications to try how the device works, one of them is the skeleton body tracking that is used in this project.

As explained in section 3.1.2.2, Kinect sensor uses depth images that are elaborate by the systems every frame to know user body movements. Kinect camera can perform a full body tracking of a maximum of two users in front of it and there is no recognition between user front side or back side. All the replicated actions are taken from the frontal part.

Now this device is no longer produced, but in the web there are a lot of open source applications created for Kinect. One of them, compatible also with Unity, is called MS-SDK [22], created by Rumen Filkov, and it is available in Unity's Asset Store. This application recalls Kinect skeleton detection functionalities using C# scripts.

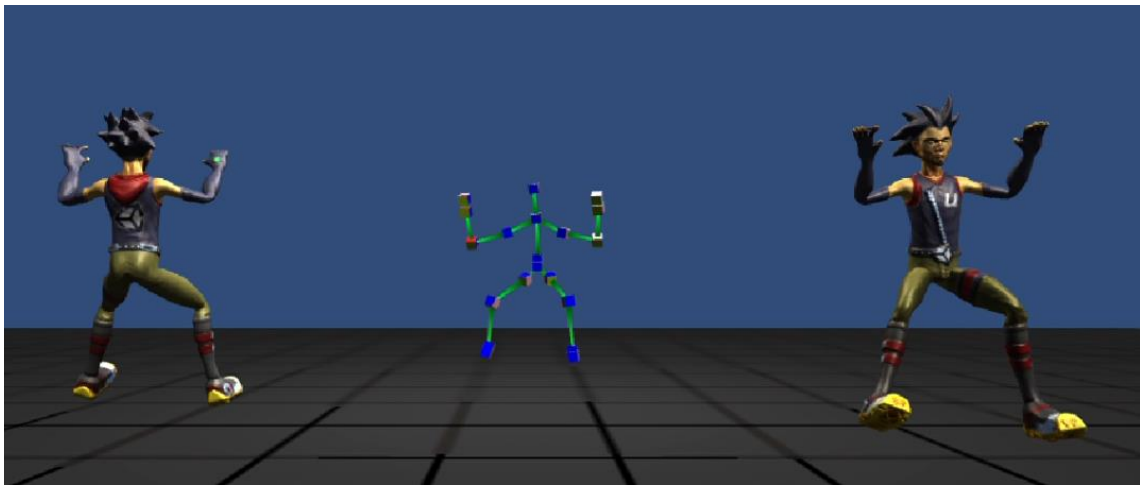

Figure 8. MS-SDK test application.

When application starts, the avatar is mapped to be recognized by scripts and then, the system detects user's movements with Kinect and uses joints to know which body part is being moved. The system calculates the difference between the positions of two joints. If it is higher than a given threshold, it will be detected as a movement and avatar will reproduce that. To be sure of moving the right bone there is a matching between avatar's bones and Kinect joints. All gestures are saved in a script and there is also the opportunity of adding new gestures following the same logic of default gestures.

As said, Kinect cannot detect user front side or back side so, to show both of them, in the figure 8 two identical avatars are used. In the center of figure 8 there is a "Cubeman" that is an avatar composed by lines and cubes. These cubes represent Kinect joints used to detect user movements.

It is also possible to set other parameters, such as:

- Minimum and maximum distance to detect user
- Kinect's highness from the ground
- Detect the closest user
- If one or two users should be detected.
- Kinect' sensor angle, in degrees
- Compute and display user map to see user's depth image
- Compute and display color map to see user's representation with RGB camera
- Display skeleton lines to see the skeleton associated to the user
- Force the avatar to do only chosen gestures

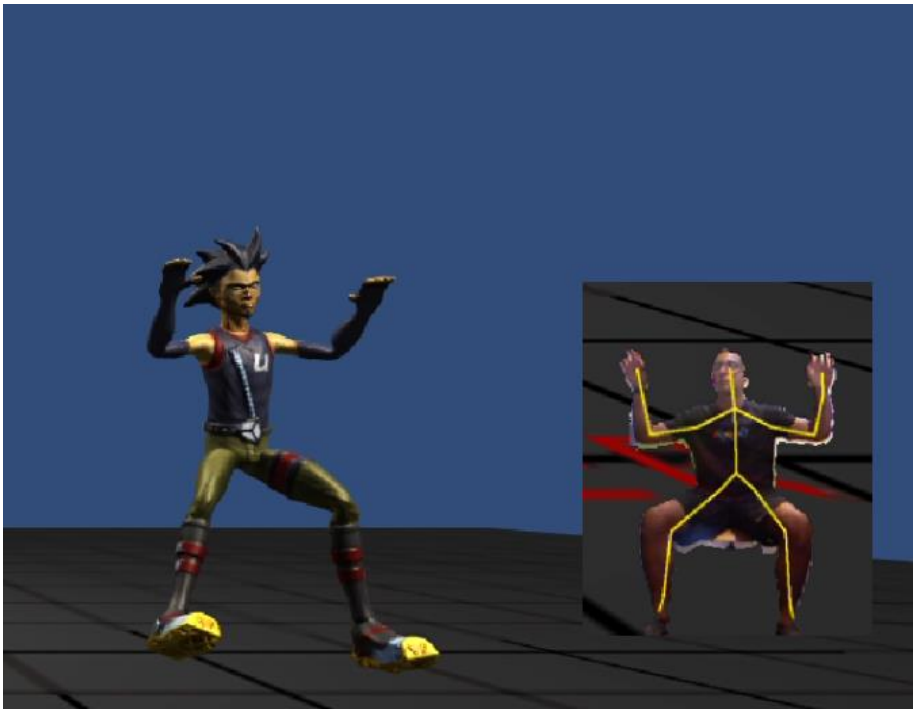

Figure 9. Color map and skeleton lines showed during application execution.

All these parameters can improve application's execution and display how Kinect sees users. A best practice for users is to be in front of the Kinect at a distance of 1,5-2 meters. This distance is considered the ideal range to be seen "well" from the Kinect without losing user's body tracking.

The system was tested in a laboratory room of the Graphical Engineering Department in the UPV, Universitat Politècnica de Valencia. This room has a limited space to perform test, around 1.5 x3 meters but to reduce the play range a Zoom is used on the Kinect. Thanks to this device, the user can stay at a distance of 1 meter from Kinect, not more than 1.5-2 meters, and he/she can be detected without problems because the Zoom reduces the space needed to play by 40%.

Each avatar has the same structure:

- Skin mesh
- Skeleton

Skin mesh is what the user sees, such as face, hands or clothes, but it doesn't perform any action. To move the avatar, the important thing is the skeleton built with gameobjects called bones. As already mentioned, Kinect uses joints to detect user's movements and each of these are associated to their respective bone by MS-SDK' script. With this association, when user moves his arm, Kinect recognizes arm's joint and moves avatar's arm thanks to their correlation.

3.1.4 Avatar creation

To create a personal avatar there are many softwares to create their skin with custom features for example:

- Adobe Fuse CC beta
- Makehuman

"Adobe Fuse CC beta" software offers a large amount of realized characters but also opportunity to create new avatar by choosing all its features (eyes color, hair's color, weight, arm length, etc.). The avatar created with this software has only the skin but no bones and it can be exported in many file formats. In a first moment, this software was chosen to create avatars but it is still in beta version and there was a bug that after the first use there were no chance to open the application again.

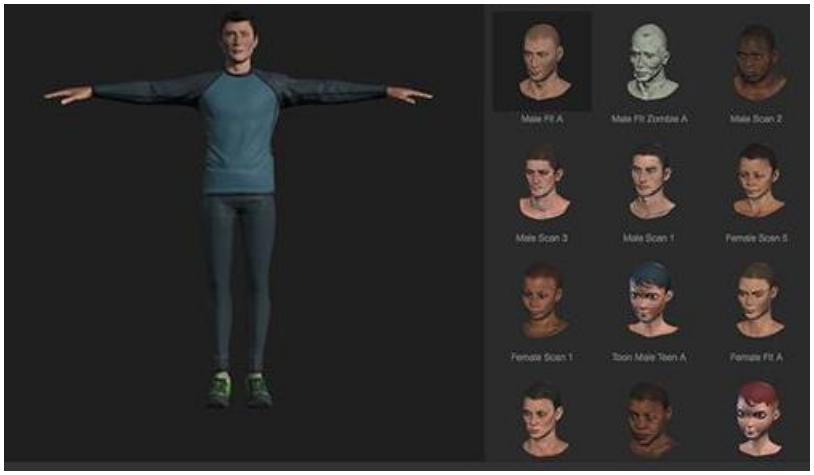

Figure 10. Adobe Fuse CC avatar creation.

For this reason "Makehuman", an open source software, very similar to Adobe Fuse CC, was used. It created many different avatars and the opportunity to create new ones. It is possible to choose also minimal features such as arm length or feet length, for example, which are used to

create avatars as similar as possible to user. With this purpose, users can see an animated version of themselves and be more comfortable during a training session. Besides, Makehuman can export the final result in many file formats.

Figure 11. Makehuman avatar creation.

Blender supports *.obj* file so avatars are exported in this format. Both avatar creation software produce an avatar composed with many meshes that are unified in Blender to have a single mesh.

Avatar bones implementation

To be used with MS-SDK, avatar also needs a skeleton that is created with Blender using an additional plugin called “Rigify” [23]. This plugin adds in “Armature” menu few pre built skeletons, one of this is called “Basic human (Meta-Rig)” and it is added in Blender workspace. There is also the opportunity to create a more complex skeleton with hands and face bones but in this case, the final application does not need so much precision.

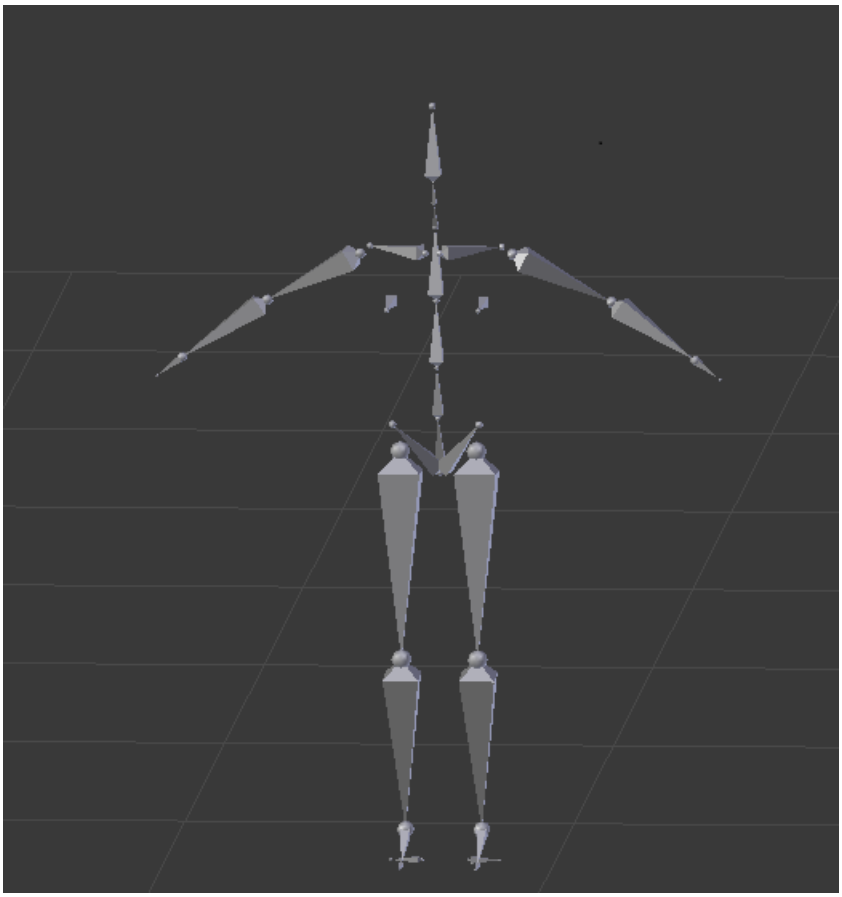

Figure 12. Blender basic-human Meta-rig.

Avatar and skeleton are scaled to the same size to be subsequently unified. Thanks to option "X-Axis Mirror" provided by Blender, a developer can work only in one part of the skeleton and the other one will follow the same changes. To be sure that skeleton and mesh match exactly, it is possible to use the "X-Ray" option to see the skeleton inside the avatar. This is a manual work and all the bones have to be manually adjusted. When this work is completed, skeleton and mesh are merged together with three options:

- With empty groups
- With automatic weights
- With envelope weights

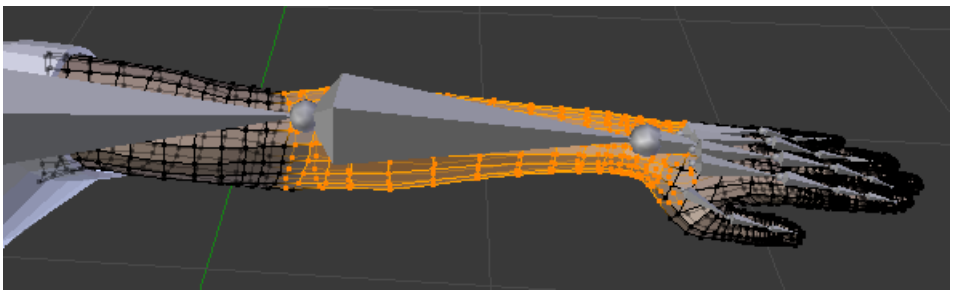

Figure 13. Example of mesh assignment to a bone.

The first one merges avatar and skeleton together without any particular action, so bones are not associated to any particular skin mesh part.

The second and third options try to associate to each bone a portion of skin mesh. Option "with automatic weight" is more precise than "with envelope weights" but to have a good association between skin mesh and bones it's better to do a manual revision.

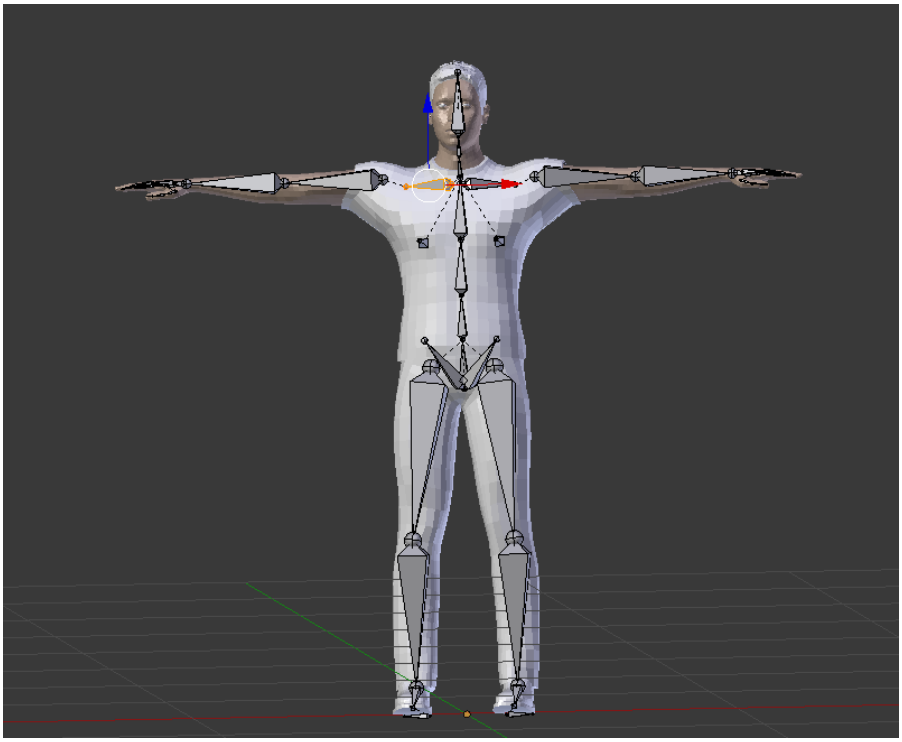

Figure 14. Avatar mesh and skeleton with X-Ray option.

Moving, rotating, scaling or other bone deformations will be also reflected on the skin mesh.

To complete the procedure you must "Generate the Rig". In fact, "Basic human (Meta-Rig)" is composed by bones and bone junctions that are not considered as a unique skeleton. Using the option "Generate Rig", a Meta-Rig is created. It is an assembly of bone-chains that connects all bones creating the final skeleton. In this way, using the "Pose Mode", used to test bones movements, or also tools to try avatar movements, all bones will move as a unique identity and not all separated.

Unity implementation

From Blender, avatars can be exported in ".fbx" format, used to save 2D or 3D project. This type of file is well supported by Unity.

When the avatar is imported in Unity, before using it in the game scene, it r has to be set as "humanoid" in "Rig" options to recognize skeleton and mesh and be ready to animate it. It's possible to configure humanoid type from an existing model or configure a new one. In configuration window, avatar has to be in T pose and bones are automatically mapped to assign each of them to humanoid body part.

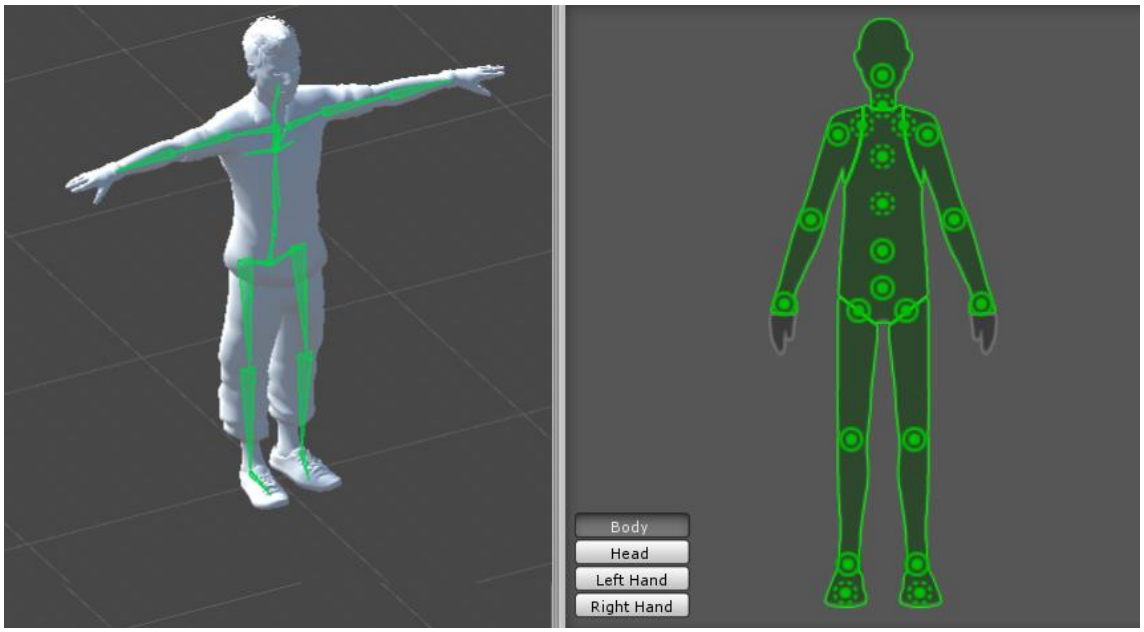

Figure 15. Humanoid T-pose and bones mapping.

It's also possible to test avatar movements and "muscles" flexibility to see how avatar reacts and if meshes deformation works properly as shown in figure 16a and 16b.

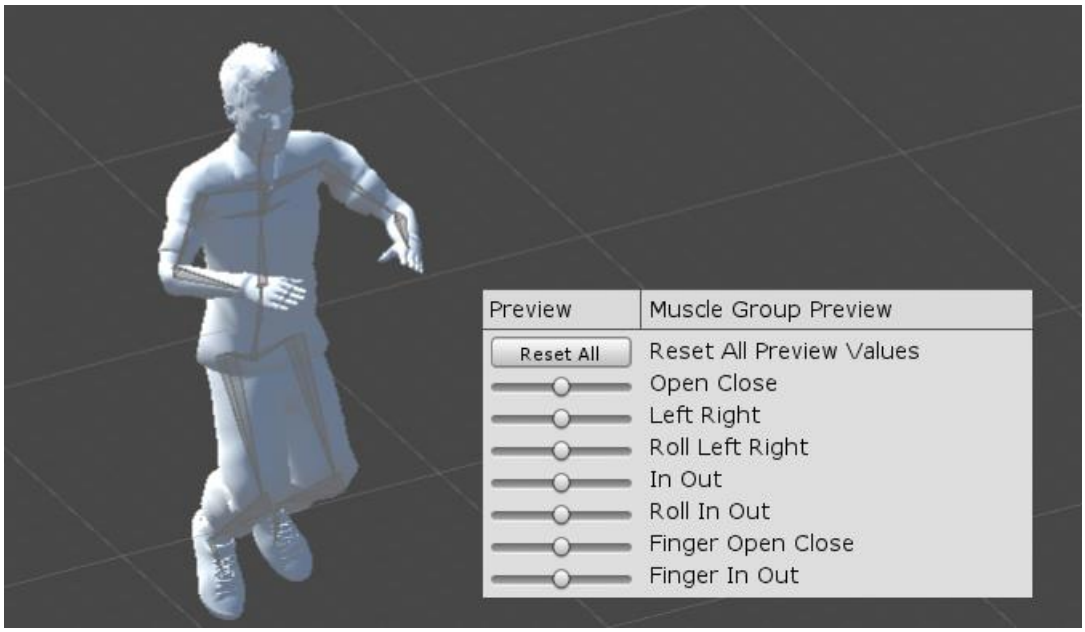

Figure 16a. Avatar muscle group with default settings

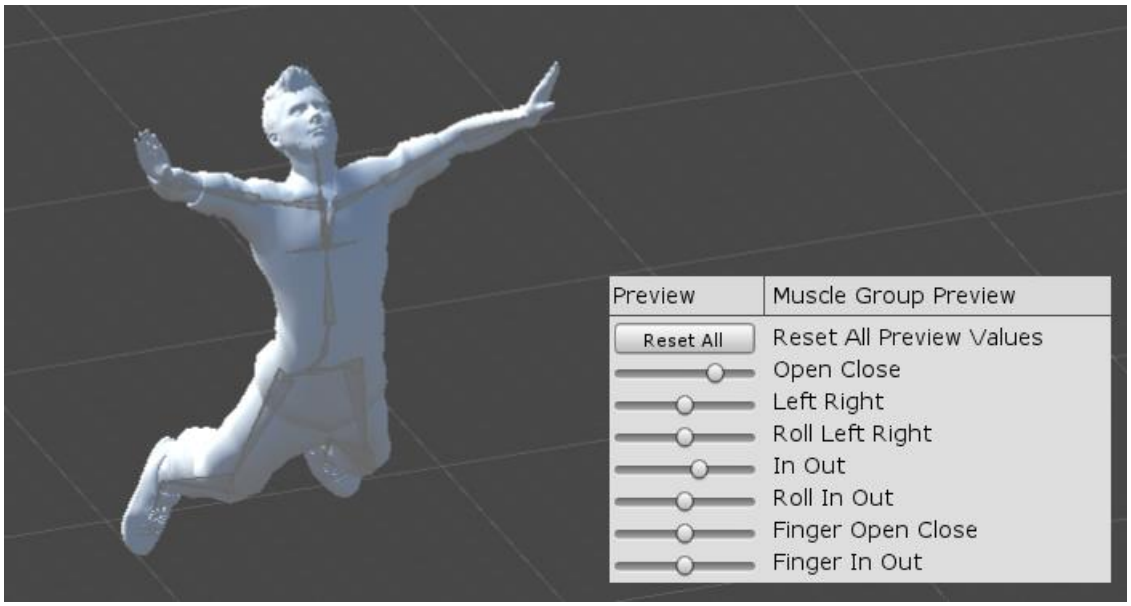

Figure 16b. Avatar muscle group with changed values.

Besides, hands can be mapped with all the fingers but using a basic meta-rig for the project this isn't necessary. Moreover, Kinect can't detect hands.

When configuration is done, everything is saved and to use avatar with MS-SDK it needs to have "Avatar controller" script to follow user's movements. This script takes note of the current position of the bones and each frame update their position based on user movements detected by Kinect using its 20 joints.

3.1.5 Mirror creation

In this stage of the developments, even without using virtual reality yet, it could be created a first person view application. This could be simply done with the main camera inside the avatar's head, being careful of not showing part of the skin avatar. This camera doesn't still follow user's head rotation so it is a static camera on avatar's head but it gives a primitive sensation of first person view.

From the user's point of view, it is useful to see his or her movements recreated by the avatar. To achieve this, inside the scene a mirror is created where all user's movements are reflected.

The mirror shape is created with two planes. To perform image reflection, a camera is used in the middle of the mirror with a render texture, a Unity element that is updated at run time and can render camera's view in one plane. Thanks to a simple script, the camera inside the mirror changes its rotation values dependently on the player camera position, creating in this way a mirror effect. When the player camera moves, the mirror camera rotates showing the correct angle of view for the user. There is also a prefab called globe mirror that performs the same action but the avatar needs to stay in a certain area and out of that nothing appears in the mirror. Consequently, the first option was chosen.

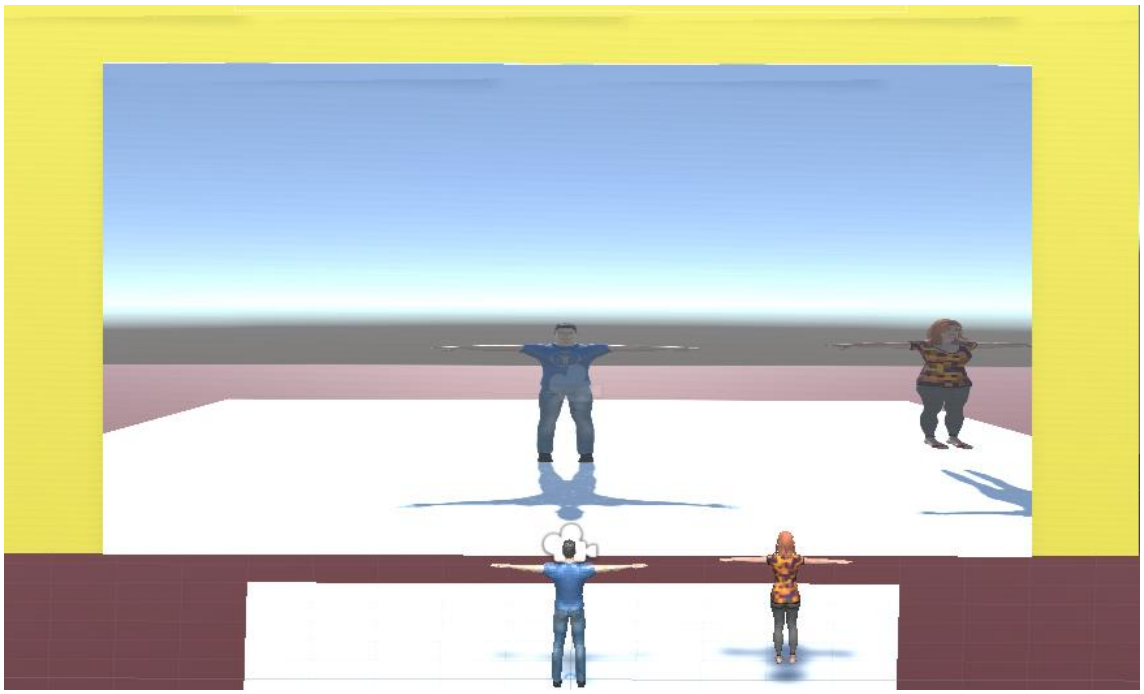

Figure 17. Mirror example.

At this point, the application can perform body tracking but not in a virtual world. User will see his or her avatar reproducing their movements from a desktop. Virtual reality implementation allows users to have a different point of view of the application and stay inside the virtual environment. This is possible because virtual reality devices track head position and rotation allowing a more realistic view and a different perspective of the environment. To cover this gap and to implement virtual reality it is necessary to use Oculus Rift with Unity.

3.1.6 Virtual Reality implementation

In Unity's asset store there is a plugin provided by Oculus Company to use this device. This plugin offers many prefabs to use and also many sample scenes for developers to understand better how it works.

Obviously, the scene created before without virtual components was modified in this step.

The main camera was replaced with Oculus prefab "OVRCameraRig" that will be the first display during game mode. This prefab performs the new user's view with Oculus and it will move inside the scene based on HMD position and rotation.

OVRCameraRig has various gameobject in his hierarchy to achieve is function:

- Tracking space
    - LeftEyeAnchor
    - RightEyeAnchor
    - CenterEyeAnchor
    - TrackerAnchor
    - RightHandAnchor
        - RightControllerAnchor

- LeftHandAnchor
  - LeftControllerAnchor

This list represents OVRCameraRig hierarchy. Each element is called gameobject and each element inside the hierarchy is a "child" of OVRCameraRig that is the "parent". Child follows parent position and scripts attached to parent can affect child gameobjects. A child can also become a parent and have children in its hierarchy, like Tracking space that has many children.

Tracking space is the area where all the gameobjects that will follow Oculus Rift actions are included.

The left, right and center eye anchor GameObjects define the position of the image during the application's execution and the Center eye anchor is chosen to be the main camera in this application.

Right and left hand are Oculus Touch tracker. They can't perform any action because they don't have any instruction but they follow joypads'position.

The first test was to put the OVRCameraRig as child of the avatar. However, it didn't work because as a child of another gameobject the camera loses its prefab mode and becomes a simple gameobject. It is no longer recognized from the system and trying to use Oculus the user is flying in an infinite space and not in the virtual environment.

In the second test, the avatar was put as a child of CenterEyeAnchor. In this way, the conditions were similar to the previous test but without using Oculus, using CenterEyeAnchor as the eyes of the avatar. However, this way also didn't work because while the users move themselves in real life, the avatar didn't follow the OVRCameraRig movements.

In the third test, the avatar was a child of "Tracking space" and the OVRCameraRig was placed inside avatar's head manually to create a first-person view sensation. In this way, the avatar follows OVRCameraRig movements around the space and thanks to kinect, it replicates also user's gestures. With this structure, shown in figure 18, is possible to achieve body tracking and also virtual reality at the same time. Kinect must be put between Oculus base stations to cover better Oculus game area.

Figure 18. OVRCameraRig final hierarchy.

Oculus Touch and avatar hands are different objects in the scene and trying to use LeftHandAnchor or RightHandAnchor as avatar hands' children is not possibile because they are prefabs and they lose their features. It was also tried to allign avatar hands and Oculus Touch just using an avatar with similar measures of real user but, also in this case, there is no matching. Finally, it was also tried to change the avatar's hand to move like a virtual hand used

by Oculus Rift. However, it was not possible because this hand has its own animator that manages all its movements but the avatar also has its own animator and it's not possible to combine two animators in the same gameobject, in this case, the avatar. Kinect, as said before, uses 20 joints to detect user's body, but it cannot natively detect hands so it is also not possible to use just Kinect to detect hand gestures. In conclusion, it isn't a problem to not detect hand gestures in the current application because they are not required for the rehabilitation exercises.

Both Kinect and Oculus use infrared light to detect users but in different ways and with different hardware, so that does not create any interference.

Kinect creates a depth image of the scene each frame and knows depth of each dot, in this way can recognize user shape and also detect depth image changes.

Oculus Rift uses its "Constellation" tracking system where each frame detects only HMD and Oculus Touch thanks to passive IR sensors on them. System knows positions of every IR sensor and can calculate distance and rotation of devices.

Obviously, Oculus Rift that was created in 2016 has a hardware much more advanced than Kinect v1 produced in 2010.

In exercises there are no specific gestures required so it is possible to use avatar hands moved by body tracking detection performed by Kinect. In this way hand tracking is less reliable but the application can allow full body tracking.

3.1.7 Rehabilitation VR application

As said in section 2.1 this works is done combined with neurologist advices. The specifications for this application are in the Annexes section. To achieve this goal, the idea is to create spheres of different colors that the user will have to touch them to complete an exercise's repetition:

- Red: the first sphere
- Yellow: the intermediate spheres
- Green: the last sphere

Every time that the user touches a sphere, it will disappear and the next one will appear until the end of the exercise. All sphere position are saved in a text file. Avatar stays inside a determinate game area and if he/she exits from there, the user is warned to go back inside the game area. This area is delimited by a collider and when the avatar, that has colliders, goes over this area, the system detects the collision and sends a message. In this version, the message is sent to the Unity console but a future work is to add a banner that will appear in the scene.

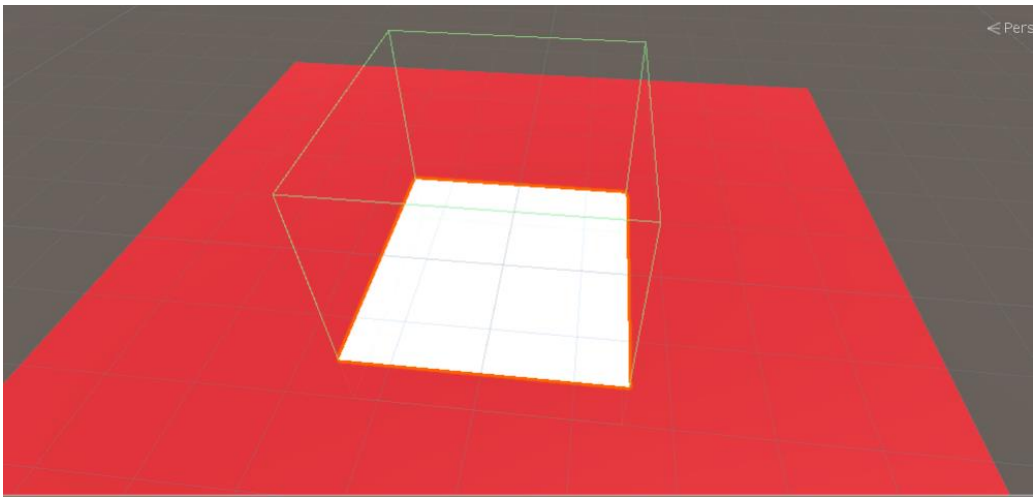

Figure 19. Example of game area delimited by collider.

Before sphere's creation, a matrix is created based on avatar measurements, from right to left hand for x-axis, from foot to head for y-axis and for z-axis a value based on the area in front of the player and a very small portion behind him/her. This is because the exercise will be performed in front of it and not behind. The matrix is created only after avatar selection, in this way it will take measurements of the only avatar inside the scene.

JSON file

Each training session is described inside a JSON because it is very easy add, remove or modify contents of file. JSON's fields are:

- General description of the training session
- The number of exercises
- General description of the single exercise
- Spheres position
- Number of repetitions
- User position
- Time break

With the general description, user will know which part or parts of his body will be involved during the training session. Before starting an exercise, there will also be a small description to explain to the user the movements that will have to be done. With the spheres position, the code knows where the sphere should be created and also the order of appearance of each sphere and with number of repetitions. The code also knows how many times the exercise should be performed. At the end of each exercise, there will be a time break until the last one. User position needs to set a more comfortable virtual environment for the user, based on his position during the exercise, standing or sitting.

User position needs to set a more comfortable virtual environment for the user, based on his position during the exercise. For example, in a future work, a virtual chair can be created where he or she has to perform sitting exercises. This last part was not an objective of the present project but the JSON file is generic and allows this kind of improvements.

All these data are used by the Unity scripts to create and perform all the exercises of a training session.

Unity provides a JSON serialization method to convert object to and from the JSON file. In this case objects are created from JSON files and this needs a script with a serializable classes, necessary to work with JSON serializer, where all the variables are contained with the same names and types of JSON fields. Otherwise, variables won't take any value.

Exercise execution

Exercise execution is composed basically of three scripts:

- **Character Selector,** which manages the main menu where the user can choose a training session and which avatar wants to use. Matrix creation for spheres position and spheres' creation are also performed in this stage.
- **Sphere,** which manages sphere appearance order. When an exercise finishes, it checks if there is another one or not. In the first case, this script recalls "Character Selector" to create new exercise spheres.
- **Collision, which** detects collisions between the avatar and the sphere. It is like a switch to activate functions in "Sphere" script.

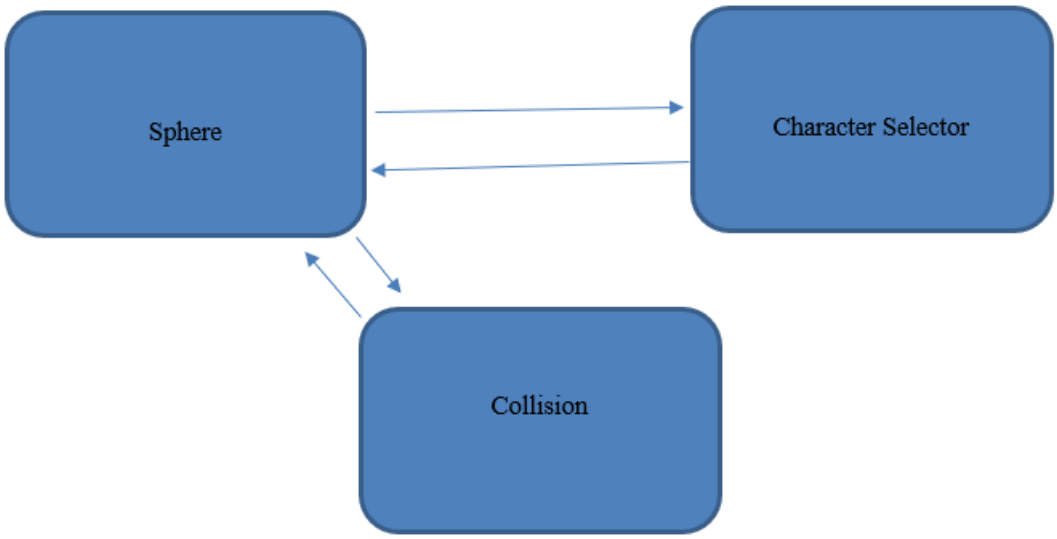

Figure 20. Script diagram and their connections.

Character selector

Character Selector	
SelectEx	Selects training session from JSON file
MatrixCreation	Creates a matrix with all possible spheres positions
CreateSphere	Creates all the necessary spheres with only the first one visible

StartGame	Places in the right position the chosen avatar and recalls Matrix and CreateSphere functions to create exercises

Table 1. Character Selector's functions.

The main menu, to choose training session and avatar, is created using Unity UI (User Interface) and it is composed of panels and buttons, as shown in figure 21.

Figure 21. Main menu example.

Each button has a value which is passed to the Character Selector script to know which elements have to be read from the JSON file and which avatar has to be used.

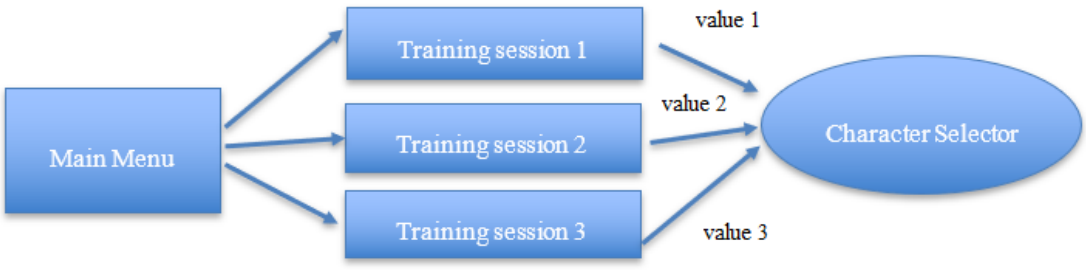

Figure 22. Graphic diagram of main menu work.

In the figure 22 there are three buttons in the main menu to choose different training sessions.

Each training session has different exercises dependently on rehabilitation therapy.

When a user presses one of these buttons, they send a value to Character Selector script. This value corresponds to training session position inside JSON file structure and it is used by "SelectEx" function.

Buttons for choosing training session can be added manually using Unity tools. However, this isn't a good solution because it means that every time a therapist wants to add a training session has to ask to the developer to add it manually.

Therapists haven't programming skills and for this purpose the main menu buttons must be automatically created by script that takes data from JSON file about how many training session there are. In this way the therapist has only to add data to JSON file to create new training session.

To have buttons with same features, it is necessary create a button prefab. This is done dragging a button gameobject from Unity scene to Unity Asset folder, the folder that contains all the items that can be used in the Unity project. Assets can be imported from other files created in other software or also created within Unity.

Using Unity function "Instantiate", it is possible to create a copy of a chosen element, in this case the button prefab. Each new button has a different name text to distinguish them and everyone has a different value associated to one training session. When user press a button, the value goes to "SelectEx" function that finds the exact training session.

Subsequently another panel appears to choose the avatar. This choice sends a value to "StartGame" function to know which avatar has to be spawned in the same position of OVRCameraRig, to perform the first-person view mode. Other avatars disappear from the scene and they will not interfere with application execution.

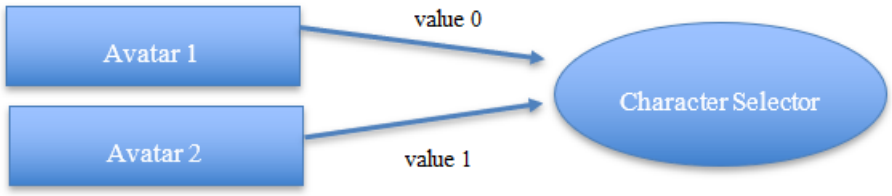

Figure 23. Graphic diagram of avatar selection.

When a character is chosen, "MatrixCreation" function is called to create the matrix for spheres'position using avatar's measures. X-axis and Y-axis are divided in eight parts and this creates nine spheres in each axis, this because body subdivision creates seven spheres. Two more spheres are added for the arms to perform certain types of exercises. There are also two more spheres over the avatar's head. Z-axis is subdivided in four parts and this creates five spheres based on OVRCameraRig position and taking only frontal avatar area and part of its back area because most exercises are performed in front of the avatar.

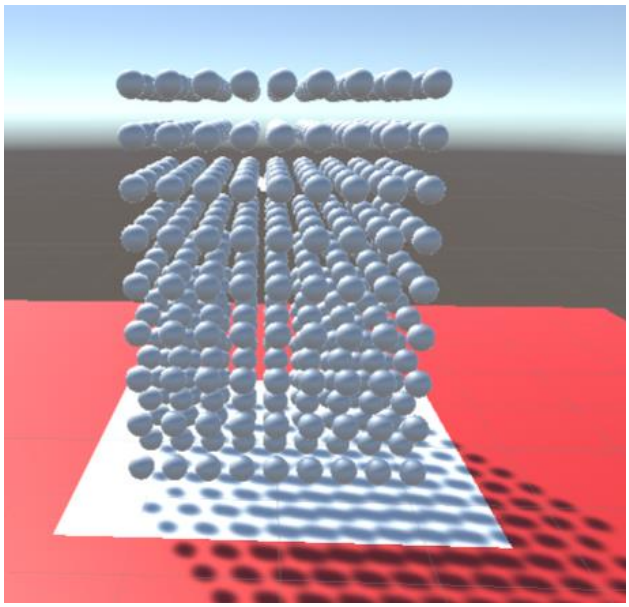

Figure 24.. All the spheres created from matrix.

A kind of invisible grid is generated around the avatar. The function “CreateSphere”, reading information from JSON file, knows which rows of the matrix has to look to create spheres.

Values contained in the JSON field “sphere position” are indices referenced to matrix rows. In each row, there are three values that correspond to space coordinates of the sphere.

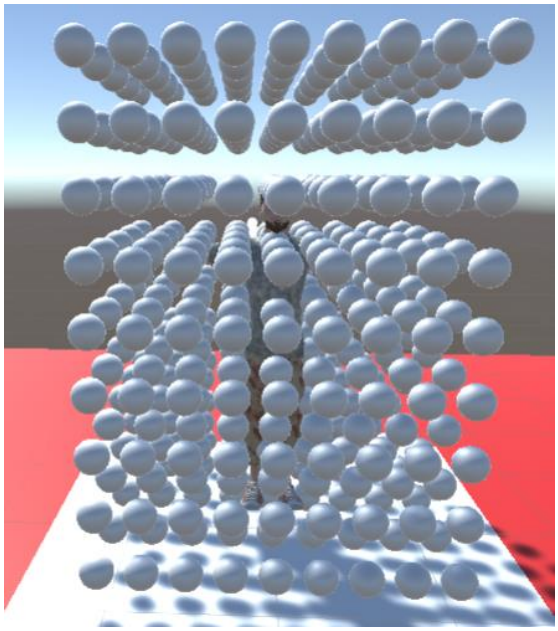

Figure 25. All the spheres around avatar.

Sphere creation is performed using the method ”Instantiate” on a sphere prefab that has mesh and collider. In this way, each new sphere will have a mesh and a collider.

Creating all spheres and deactivating all except chosen spheres creates a problem because if a gameobject is inactive, it can't be found and the next exercise will not be created. Besides, deactivating a sphere after user's touch it's not a good solution because after the first repetition, the system will not find any sphere.

A good solution is to create only chosen spheres and deactivate all their meshes except the first one. In this way, only the first sphere is visible for user. When a sphere is touched, the next one will appear. All the spheres have also a collider that must be activated only for the visible sphere or other colliders could interfere with exercise execution activating counters when it's not necessary.

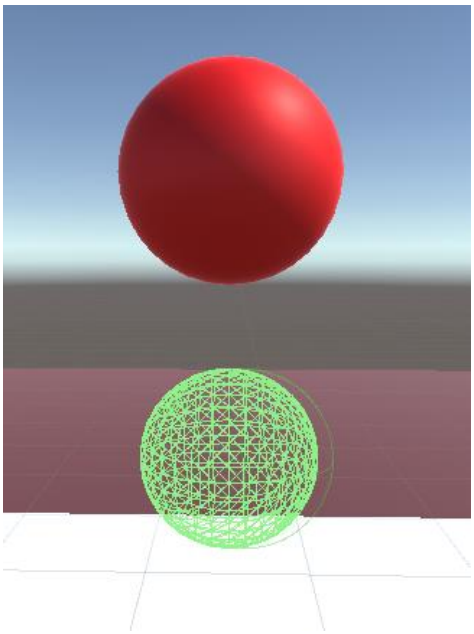

Figure 26. Red sphere active and second sphere waiting for activation.

In figure 26, two spheres are shown to understand how they are hidden:

- Red one is the current sphere with mesh and collider enabled. User has to touch it to continue the exercise
- Second sphere has only Mesh collider enabled to show where it is in the developer mode but mesh render and collider aren't enabled and it doesn't interfere in the scene.

Sphere

Sphere	
CreateNext	Creates the next sphere with its relative color
SphereCounter	Counter to know which sphere will appear and repetitions number
NextEx	When an exercise ends, it looks for the next one, if there are no more exercises it restarts the application

Table 2. Sphere script.

This script recalls CharacterSelector variable to know which training session has to perform.

CreateNext function finds the next sphere and activates its mesh and collider to be used. It also sets the right color: first sphere red, last sphere green and other spheres yellow. It also increases repetitions counter.

SphereCounter checks repetitions number, when the last repetition is done, it resets all the counters, destroys all the spheres and sets the time break counter taken from JSON file. After any exercise, the user should rest for a few seconds and then can start the next exercise.

NextEx is called when all repetitions of one exercise are done. The function checks if there is another exercise or no. In the first case is recalled CreateSphere function to create spheres for the new exercise; otherwise, user comes back to main menu and he or she will be able to choose another training session.

All these scripts are activated based on boolean values given by the Collision script. Thanks to the Update function provided by Unity, during each frame this script checks a boolean value and executes the right one.

Collision

Collision	
OnTriggerEnter	Deactivate current sphere mesh and recall CreateNext function
OnTriggerExit	Deactivate current sphere collider and recall SphereCounter function

Table 3. Collision Script.

This script is a derived class of Sphere script.

Each sphere has a collider. Hands or other body parts also have a collider. When this script detects a collision and a body collider is inside the sphere collider, the sphere mesh is disabled, to be invisible to user, and a boolean becomes true activating CreateNext function that will create the next sphere. When the body collider exits from the sphere collider, the latter is disabled to not interfere with the user actions and a boolean activates the SphereCounter function.

Each body part that can be used for an exercise has a collider and this Collision script has to update counters and perform exercise with more than one body part if requested.

3.2 Results

The final application takes data about training session from the JSON file and creates all the necessary objects to perform exercises.

```
{"motor_rehabilitation":[{


    "general_description": "This session will have two exercises! One for arms and one for legs!",
    "number_of_exercises": 2,
    "exercises": [{
            "exercise_description": "Exercise 1",
            "sphere_position": [12, 16, 17, 16, 12],
            "repetitions": 3,
            "user_position": "standing",
            "pause": 10
        },
        {
            "exercise_description": "Exercise 2",
            "sphere_position": [130, 146, 117],
            "ripetitions": 4,
            "user_position": "standing",
            "pause": 10
        }
    ]
  }
 ]
}
```

Figure 27. JSON file example.

In figure 27 there is a JSON file example that is used to run the application. All training session can be put inside the field motor rehabilitation that contains an array. Each element of the array is a different training session.

The general description in this version isn't shown to user but this is one of the future works to achieve and the same is for the exercise description. The field "number of exercises" is necessary to application to know how many exercises it has to create.

All the following images have been taken from the user's point of view.

The user has to stay in front of the Kinect sensor at a distance of 1 meter, using Zoom, or 1,5-2 meters without Zoom. The Kinect is between the two Oculus Rift camera sensor. With these settings the user is tracked by both devices.

To run the motor rehabilitation application user has to put the Oculus Rift HMD in his or her head and use joypads to press play and start the application.

Figure 28. Training session menu.

When application starts, a user sees the menu to choose what training session perform as in the figure 28.

Using the Oculus Touch the user can select one of them and continue with avatar selection as shown in figure 29.

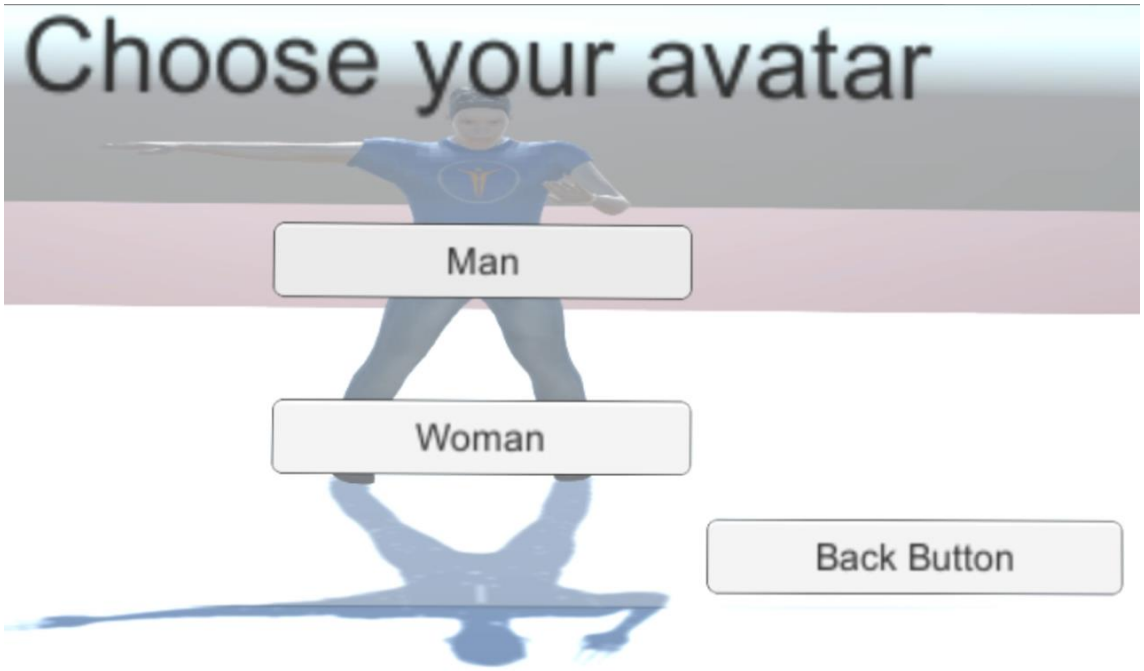

Figure 29. Avatar selection menu.

In this version there are only two avatars, a man and a woman. User can see them from the beginning and with this menu shown in figure 29. he /she can choose which avatar to use. There is also a "Back button" to come back to training session menu and change training session choice.

After that patient chooses an avatar, the other avatar will disappear from the scene and the avatar chosen will stay in front of the mirror with the first sphere.

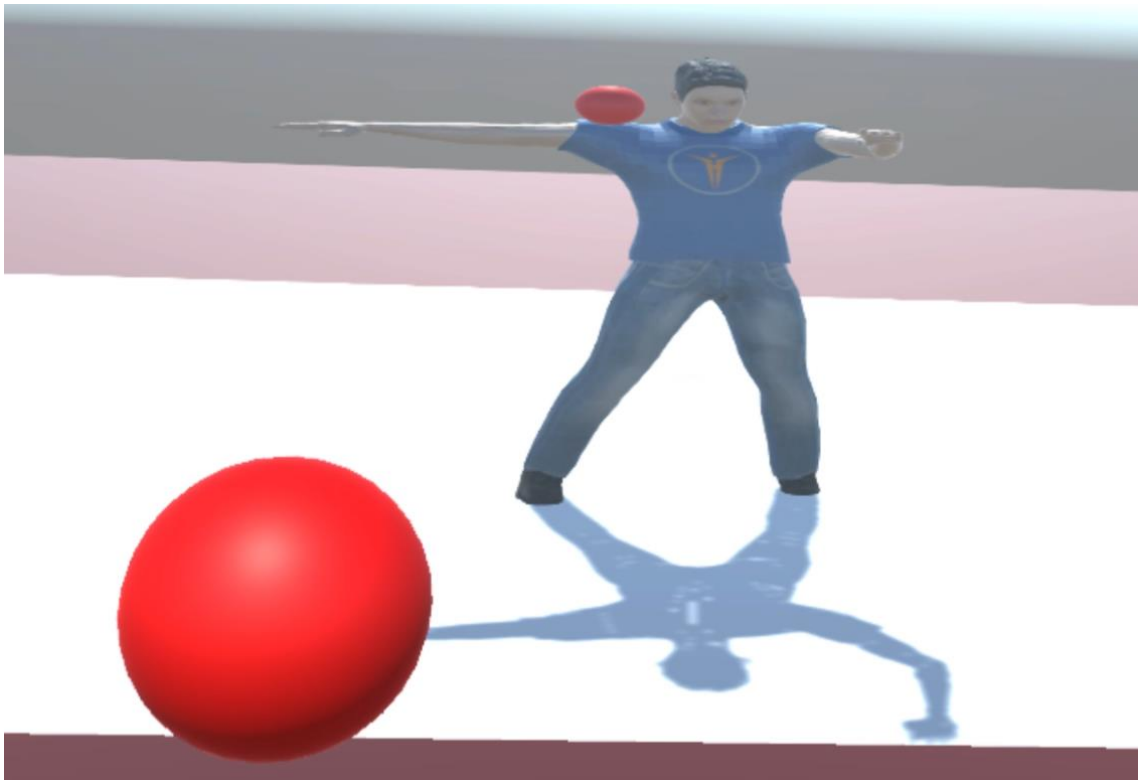

Figure 30. Exercise beginning.

The exercise starts with a red sphere, as in figure 30, that user has to touch. This sphere is in the position provided by the element number 12 of the matrix as written in JSON file in figure 27.

After that user touches the red sphere, a yellow sphere will appear, as in figure 31, based on position written in JSON file.

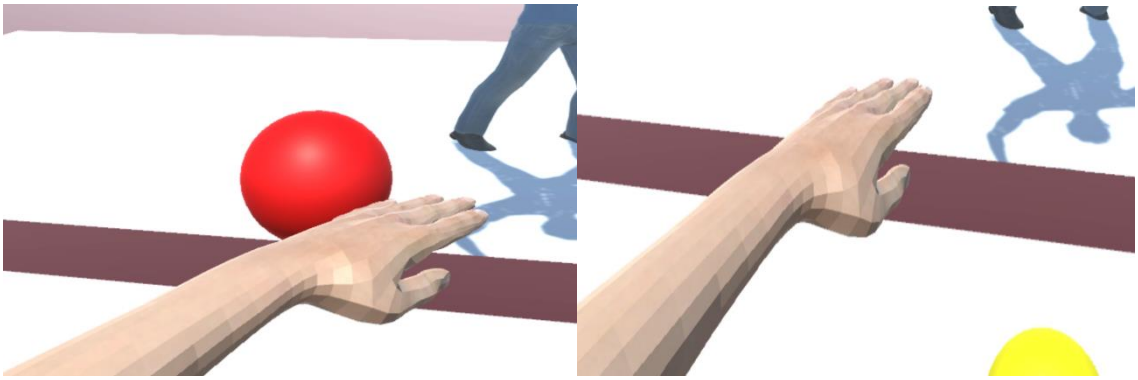

Figure 31. sphere path.

The user sees the yellow sphere and he or she has to touch it. In this way, spheres create a path to perform the exercise for the patient.

Figure 32. Last sphere.

The green sphere is the last one of the exercise, figure 32. After that user touches this sphere, system detects if he or she has other repetitions to do or go to another exercise using the field “number of repetitions” in the JSON file, figure 27.

If after the last repetition the system finds another exercise in JSON file all the sphere are destroyed and the new spheres of the new exercise are created but only after a time break chosen by therapist, figure 27.

Otherwise, if there are no more exercises, after a time break, the application come back to the training session menu where user can do the same training session or perform another one.

Preliminary tests with some volunteers have shown that all the virtual experience is performed without any headache or motion sickness feeling during the application execution.

Chapter 4. Conclusions and future work

The virtual application for motor rehabilitation developed in this project is a first version of full body tracking application combined with virtual reality. Therapists can add new exercises, test them and evaluate them. Creating this automatic application, their workload will be reduced so more patients can be managed. Patients in turn will try a new way to perform rehabilitation that doesn't want to replace classical rehabilitation but want to improve and help that. This kind of motor rehabilitation can also be performed at home with all the required devices and with an appropriate free space. In certain types of rehabilitation therapy, it is an advantage to stay at home and move as little as possible to avoid complications.

As discovered researching in bibliography and in commercial application to achieve the first objective of the project, many virtual rehabilitation applications are created using only the Kinect sensor. There are only few attempts to increase user presence in virtual environment using HMD for virtual reality. This application tries to combine these technologies to create rehabilitation trainings performed in an immersive virtual environment.

Oculus Rift and Kinect work well together without any interference in their tracking system. Both of them as said use infrared light and they can cause delay or interference each other but in this case the application works without any problem of this type.

Kinect as said is in the market since 2009 and in the web but as discovered in the bibliography there are many applications provided for this device. It wasn't necessary to create a body tracking application from scratch but to use one of the open source full body tracking solutions, understand how it works and use that.

In this case, the second objective to develop a body tracking application is achieved using MS-SDK with the possibility for the developer to set many options about body tracking. Avatar is created following Unity documentation to be used in "humanoid" version and, for this purpose, Blender was used to create skeleton structure to combine with avatar. Full body tracking application can detect user movements and move the exact bones to reproduce its movements in a fluid way. If the user is too much near to Kinect sensor this could lose body tracking and a good compromise is to stay at least at 1,5 meters of distance from that.

To achieve the third objective and create a virtual reality rehabilitation application, the virtual reality is improved in full body tracking application following Unity gameobjects hierarchy. Oculus Rift prefab can't be changed deleting gameobject or the system will not work properly but it's possible to add to them gameobjects that don't interfere in their natural work. The avatar controlled by Kinect is a child of the Oculus Rift prefab. In this way, it is possible to add head rotation to body tracking app and also an improvement to move the avatar in the space. In body tracking application performed only by Kinect avatar movements follow user movements but it doesn't detect if the user goes near or far from the Kinect sensor. Another limitation to be careful is that Kinect understands the position of users in front of the camera and not in profile or the back side. So the user has to stay always in front of the camera and also the exercise has to be thought to be performed in front of the camera.

First person view creates a more immersive experience for user that feels to be part of the virtual world. The mirror helps also to have a complete view of the body to have a bigger perception of ourselves in the virtual world.

Using simple objects like spheres, but also cube maybe, it's possible to create a virtual path to guide users through exercise execution. This system is very easy and users learn to use it in few minutes. All these actions are managed with C# scripts that works perfectly with Unity.

All the exercises are described in JSON file and application takes data form here. This means that application can be adapted to many exercises without change all the code but only adding, removing or editing few lines in a text file. In this way, it results very easy for therapist to create new exercises for the application.

Future works

The system can be improved in many ways, for example using Kinect v2 and not Kinect v1[24]. Hardware of Kinect v2 offer a RGB camera and a depth camera with higher resolution than Kinect v1 and Kinect v2 uses 26 joints to detect skeleton, instead of 20.

Matching Oculus Touch with avatar hands moved by body tracking system is another future work. In this way, there are no problems to use laser pointers or similar tools from avatar hands too.

Improving Oculus Touch could be possible create a small menu that appear when a button is pressed. It could appear in front of the avatar or also in avatar hand and decision can be taken with a laser pointer using Oculus Touch or detecting hand position and using that as a mouse. With this menu could be stopped or restarted the game or exit from the current training session and use another one.

Another task could be to create an avatar creator menu to create a custom avatar when application is running in this way a patient could create his or her own avatar. As said, the application is created for people of all ages and create your own avatar resembling yourself or also someone different could reduce pressure during rehabilitation.

This application offers also a no invasive way to perform the exercise but in a future work could be interesting also to use small wearable devices to detect not only user movements but also other parameters as heartbeat or breathing. These devices don't have to interfere with user exercise execution and for each problem it is valuated if it is possible use them or not.

Bibliography

[1] David Antón, Alfredo Goñi, Arantza Illarramendi, Juan José Torres-Unda, and Jesús Seco ,"KiReS: A Kinect-based telerehabilitation system", IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013), 2013.

[2] Miguel Pedraza-Huesoa, Sergio Martín-Calzóna, Francisco Javier Díaz-Pernasa, Mario Martínez-Zarzuelaa, "Rehabilitation using Kinect-based Games and Virtual Reality", Procedia Computer Science 75 (2015) 161 – 168, 2015

[3] Maillot P, Perrot A, Hartley A., "Effects of interactive physical-activity video-game training on physical and cognitive function in older adults", Psychology and aging, Psychology and Aging 2012; 27(3): 589-600.

[4] J.H. Crosbie, S. Lennon, M.D.J. Mcneill, and S.M. Mcdonough, "Virtual Reality in the Rehabilitation of the Upper Limb after Stroke: The User's Perspective", CYBERPSYCHOLOGY & BEHAVIOR Volume 9, Number 2, 2006 © Mary Ann Liebert, Inc, 2006

[5] Sim Kok Swee, Lim Zheng You, Benny Wong Wei Hang,Desmond Kho Teck Kiang,

"Development of rehabilitation system using virtual reality", 2017 International Conference on Robotics, Automation and Sciences (ICORAS), 27-29 Nov. 2017

[6] Parisa Ghanouni, Tal Jarus, Danielle Collette, Rachel Pringle, "Using virtual reality gaming platforms to improve balance in rehabilitation of stroke survivors", 2017 International Conference on Virtual Rehabilitation (ICVR), 19-22 June 2017

[7] Jigna Patel,Gerard Fluet, Alma Merians, Qinyin Qiu, Matthew Yarossi, Sergei Adamovich, Eugene Tunik, Supriya Massood, "Virtual reality-augmented rehabilitation in the acute phase post-stroke for individuals with flaccid upper extremities: A feasibility study", 2015 International Conference on Virtual Rehabilitation (ICVR), 9-12 June 2015

[8] Oculus website, Facebook Technologies, LLC

https://www.oculus.com/

[9] HTC Vive website, HTC Vive

https://www.vive.com/us/

[10] Vive trackers, HTC Vive

https://www.vive.com/us/vive-tracker/

[11] Microsoft Kinect, Microsoft

https://developer.microsoft.com/en-us/windows/kinect

[12] Unity website, Unity Technologies

https://unity.com/

[13] Unity website, Unity Technologies

https://docs.unity3d.com/Manual/index.html

[14] Oculus website, Facebook Technologies, LLC

https://developer.oculus.com/

[15] Zyko website, Zoom for Xbox 360® Kinect®

https://nyko.com/products/xbox-360-zoom-for-kinect

[16] Wenjun Zeng, "Microsoft Kinect Sensor and Its Effect", IEEE, 27 April 2012

[17] Blender website, Blender Foundation

https://www.blender.org/

[18] Adobe Fuse CC, Adobe Systems

https://www.adobe.com/sea/products/fuse.html

[19] MakeHuman, The MakeHuman Team

http://www.makehumancommunity.org/

[20] Visual Studio, Microsoft

https://visualstudio.microsoft.com/it/

[21] Microsoft website, Microsoft Kinect SDK

https://www.microsoft.com/en-us/download/details.aspx?id=44561

[22] Rumen Filkov website, Rumen Filkov

https://rfilkov.com/2013/12/16/kinect-with-ms-sdk/

[23] Blender archive, Blender documentation

https://archive.blender.org/wiki/index.php/Extensions:2.6/Py/Scripts/Rigging/Rigify/#How_Rigify_Works

[24] Oliver Wasenm¨uller and Didier Stricker, "Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision", Computer Vision – ACCV 2016 Workshops: ACCV 2016 International Workshops, Taipei, Taiwan, November 20-24, 2016, Revised Selected Papers, Part II, 2016.

Annexes

VIRTUAL ENVIRONMENT WITH KINECT AND OCULUS RIFT FOR MOTOR TRAINING. SPECIFICATIONS.

Environment

The environment will be composed of a room with a central area where the exercises will be performed. This central area can be defined with a specific element, for example a carpet.
A mirror will be located in front of the avatar.

Application for motor training

There will be three steps in the application:

1. Exercise definition by the therapist
2. Exercise execution by the user
3. Evaluation (automatic evaluation of the user execution for the therapist)

Exercise definition

In order to specify an exercise, thee elements should be defined:

1. Graphical aides to perform the exercise. Three spheres with different colors (red, yellow and green) will be located in the environment to indicate different keypoints that define the trajectory of the exercise. For example, initial hand position, intermediate hand position and final hand position.
2. A text describing the exercise. For example, you should move your arm forwards until your hand touches the red sphere, rotate right until it touches the yellow sphere and return to initial position where the green sphere will be located.
3. The number of repetitions. For example, ten times.

The exercise definition should be expandable so the therapist can add new exercise if needed.

Initially, two exercises (for shoulder and knee) will be defined in an ASCII file that contains for each exercise the following parameters:

- The text describing the exercise
- The position of the three spheres
- The number of repetitions
- User position (standing/sit)

To simplify the position of the three spheres, a fixed matrix of spheres distributed close to the central area of the room will be defined. This matrix will be invisible for the user during exercise execution, only the sphere that should be touched next will be highlighted with the proper color. Each sphere of the matrix will be associated to a number, which will be the way to reference that sphere. This simplifies the definition of the position of the red, yellow and green spheres for each exercise.

Exercise execution

Initially, the application shows the text describing the exercise. Following, the virtual room is shown and the user appears in the central position.
A button should be pressed to start the training. A number will appear on screen showing the repetition number. Then the user will perform the exercise. First, the red sphere will be highlighted. When the user touches the red sphere, the yellow one will appear. Finally, when the user touches the yellow sphere, the final green sphere will be shown. When the user touches it, the repetition counter is increased and all the process will be repeated until the last repetition. When all the repetitions are finished, a "Well done!" message will appear on the screen. The user will also be asked to relax during 10s.
If the user moves during the execution of the exercise, the system should inform the user. In order to return the correct (central) position, a button should be pressed during 3s.
If the system has defined more than one exercise, when an exercise is finished, the described process begins again with the next exercise, until all the exercise have been performed. When all the exercises are performed, the user is informed that the training has finished.

System evaluation
Stored Parameters:

- Total time to perform each repetition
- Total time to perform an exercise (with all repetitions)
- Men time to perform a repetition
- Trajectory (angles and position in each joint)
- Precision to touch the spheres

The results will be stored in an ASCII file.