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

TRABAJO DE FIN DE GRADO

Grado en Ingeniería Biomédica[*1]

Grado en Ingeniería Mecánica[*2]

APLICACIÓN DE REALIDAD VIRTUAL PARA LA REHABILITACIÓN

Memoria y Anexos

Autor:	Laura Daniela Pedreros Almeciga[*1]
	Martí Fabregat Pous[*2]
Director:	Jordi Torner
Co-Director:	Gil Serrancoli Masferrer
	Francisco Alpiste Penalba
Convocatoria:	Enero 2019

● Resum

La Implementació de tecnologies digitals com la realitat virtual en el camp de la medicina ha augmentat considerablement en els últims anys, tot i això, aquesta implementació és un tema que està en fase de desenvolupament. A través aquest projecte busquem l'oportunitat de contribuir en aquest camp d'investigació mitjançant el desenvolupament d'una aplicació mèdica per a tractaments de rehabilitació.

El propòsit d'aquesta aplicació és que pugui ser utilitzada per a la rehabilitació motriu de persones que han patit un Ictus. Degut a que la rehabilitació abasta un aspecte molt ampli, vam decidir centrar-nos en la rehabilitació de l'equilibri, ja que és una de las afectacions més comunes en aquests pacients i una de les que més impacte tenen en la seva vida quotidiana.

El treball s'enfoca a treballar aquest tipus de rehabilitació a partir d'un joc on l'objectiu és recollir la major quantitat de monedes mentre s'esquiven els diferents obstacles inclinant el cos lateralment. Tot això a partir d'una interfície immersiva i entretinguda per a que el pacient aconsegueixi rehabilitar-se sense ser conscient i així , no només poder canviar el concepte de rehabilitació a una activitat més entretinguda , sinó també facilitar el treball dels fisioterapeutes ja que podran veure els progressos del pacient de forma quantitativa.

Per tal de poder dur a terme aquest treball s'ha utilitzat la plataforma de creació d'aplicacions computacionals Unity3D, on es realitza tota la programació de l'entorn virtual, un sensor de captura de moviment Kinect amb el qual s'extreu valors d'interès del cos durant la realització de la teràpia, unes ulleres de realitat virtual (HTC Vive) i el software phpMyAdmin per a la creació de bases de dades.

Per poder avaluar la funcionalitat de l'aplicació s'han realitzat proves amb 22 persones (sense tenir discapacitat motriu) formades per un rang d'edat d'entre 20 y 72 anys, gràcies a aquestes proves s'ha pogut evidenciar y analitzar les dades capturades en cada sessió de rehabilitació i veure si els factors com l'edat influeixen a l'hora de realitzar teràpies en un entorn virtual.

Paraules clau: fisioteràpia, ictus, immersiva, medicina, pacient, realitat virtual, rehabilitació.

Resumen

La implementación de tecnologías digitales como la realidad virtual en el campo de la medicina ha venido en aumento en los últimos años, sin embargo, esta implementación es un tema que aún se encuentra en desarrollo. A través de este proyecto vemos la oportunidad de contribuir en este campo mediante la continuación del desarrollo de una aplicación médica para tratamientos de rehabilitación.

El propósito de esta aplicación es que pueda ser utilizada para la rehabilitación motriz de personas que han sufrido la enfermedad de Ictus. Debido a que el tema de rehabilitación abarca un espectro muy amplio, decidimos centrarnos en la rehabilitación del equilibrio, ya que es una de las afectaciones más comunes en estos pacientes y una de las que más impacto tienen en su vida cotidiana.

El trabajo se enfoca a trabajar este tipo de rehabilitación por medio de un juego donde el objetivo es recoger la mayor cantidad de monedas mientras se esquivan los diferentes obstáculos, realizando inclinaciones laterales. Todo esto por medio de una interfaz inmersiva y entretenida para que el paciente logre rehabilitar en un entorno lúdico que favorezca la motivación a realizar la terapia y así no solo poder cambiar el concepto de rehabilitación favorablemente, sino también facilitar el trabajo de los fisioterapeutas, ya que estos podrán ver el progreso de su paciente de forma cuantitativa.

Para poder llevar a cabo este trabajo se ha utilizado la plataforma de creación de aplicaciones computacionales Unity3D, en donde se realizó toda la programación para el entorno virtual, un sensor de captura del movimiento Kinect con el cual logramos extraer valores de interés del cuerpo durante la realización de la terapia, unas gafas de realidad virtual (HTC Vive) y el software phpMyAdmin para la creación de bases de datos.

Para poder evaluar la funcionalidad de la aplicación se realizaron pruebas con 22 personas (sin discapacidad motriz) que se encuentran en un rango de edades entre los 20 y los 72 años, gracias a estas pruebas se pudo evidenciar y analizar los datos capturados en cada sesión de rehabilitación y ver si factores como la edad influyen a la hora de realizar terapias en un entorno virtual.

Palabras clave: fisioterapia, ictus, inmersiva, medicina, paciente, realidad virtual, rehabilitación.

● Abstract

The current application of digital technologies such as virtual reality in fields for medical purposes has been increasing in the last years. Nevertheless, this implementation is yet to be developed. This project seeks to continue and contribute to the development of this field through an application to treat and attend the rehabilitation in the most practical way.

The purpose of this application is to be able to be used for the motor rehabilitation of people who have suffered from a stroke. Because rehabilitation covers a wide spectrum of exercises and practices, this project is focused on the recovery of equilibrium and balance due to its commonness and the dilemmas it brings against their everyday life.

This project focuses its efforts in reinstating the patients through a game where the goal is to recollect the most coins while evading and dodging different obstacles leaning the body sideways. All of this over an immersive and enjoyable interface therefore the patient accomplishes his/her target without realizing while being in a lucrative environment that motivates and upraise the patients' moral. This way not only changes the concept of rehabilitation to something entertaining and lively but also facilitates the work of the physiotherapists when it comes to evaluate the progress of their patients with a more quantitative and analytical result.

To be able to carry out this project we used the platform Unity3D to create computational applications and programming of all the virtual environment, a Kinect sensor to capture the movement and extract the values of interest while being rehabilitated, a set of virtual reality goggles (HTC Vive) and the software phpMyAdmin to create the database.

To be able to evaluate the functionality of the application a test with 22 people was conducted. These people did not have any type of incapacity or disability and represented a small group from age 20 to 72 years. Thanks to this test it was possible to analyze the collected data and study the impact age has (amongst other impacts) while doing a rehabilitation therapy in a virtual environment.

Key words: physiotherapy, stroke, immersive, medicine, patient, virtual reality, rehabilitation.

● Agradecimientos

Durante el desarrollo de nuestra tesis pasamos por 4 meses de trabajo arduo, pero durante este tiempo contamos con personas que estuvieron ayudándonos y apoyándonos para hacer de este proyecto una realidad.

Es por esto que nos gustaría agradecer en principio a la Universidad Politécnica de Cataluña (UPC) por brindarnos la oportunidad de desarrollar este proyecto, gracias a todos los profesores y estudiantes de esta institución que de forma directa o indirecta fueron participes de nuestro proceso, ya que su aporte, grande o pequeño se ve reflejado en la culminación de este proyecto.

De la misma manera queremos realizar un agradecimiento a nuestro director de tesis Jordi Torner y nuestros codirectores de tesis Gil Serrancolí y Francesc Alpiste, por su acompañamiento durante este proceso, por brindarnos las herramientas necesarias para sobrellevar los obstáculos con los que nos íbamos encontrando y por estar siempre dispuestos a solucionar nuestras dudas.

~ Laura Daniela Pedreros Almeciga y Martí Fabregat Pous

Principalmente quiero agradecer a la Escuela Colombiana de ingeniería Julio Garavito y a la Universidad del Rosario por hacer parte de mi formación, por brindarme las bases necesarias para ser una buena profesional y por darme la oportunidad de realizar mi doble titulación en ingeniería Biomédica con la Universidad Politécnica de Cataluña. Gracias a mis profesores y amigos de la universidad que participaron en el desarrollo y crecimiento de mi formación profesional y personal.

Cabe recalcar el agradecimiento que tengo hacia mis padres Gloria Amparo Almeciga y Alfonso Pedreros por su apoyo incondicional, por confiar en mí y en mis aptitudes, gracias a ellos por enseñarme a trabajar para lograr mis sueños y por cada una de sus palabras que me han ayudado a ser quien soy en este momento.

Por último, quiero agradecer a Miguel Angel Rivera por siempre estar siempre ahí apoyándome cuando lo necesitaba, por siempre estar ahí animándome para seguir adelante cuando se presentaron dificultades en el desarrollo de este trabajo, gracias por la ayuda y por los aportes no solo para el desarrollo de la tesis, sino también para mi vida.

~ Laura Daniela Pedreros Almeciga

Así mismo, me gustaría dedicar este trabajo a mi familia Fabregat Clotet, Pous Andrés y Fresco Díez por el apoyo incondicional durante toda mi educación, seguir todos mis proyectos conmigo y ayudarme a ser creativo y diferente. También agradecer a mi madre Maria del Mar Pous Andrés que, aun no pudiendo estar presente, ha dejado un legado que me sigue educando hasta el día de hoy.

Agradezco la colaboración de todos aquellos que hicieron posible mi educación, tanto dentro como fuera de la escuela, este es el fruto de vuestro trabajo.

~Martí Fabregat Pous

● Glosario

RV: Realidad Virtual

VR: Virtual Reality

TFG: Trabajo de Fin de Grado

ADFO: Asociación de Disminuidos Físicos de Osona

UPC: Universidad Politécnica de Cataluña

EEBE: Escuela de Ingeniería Barcelona Este

BD: Base de Datos

DB: Database

ÍNDICE

1. Prefacio

La rehabilitación juega un papel muy importante a la hora de la recuperación de un paciente, muchas veces este proceso se asocia a ejercicios monótonos o repetitivos, pero con este proyecto queremos mostrar una forma moderna y complementaria de la fisioterapia convencional. El presente trabajo de grado está enfocado en la creación de una aplicación de realidad virtual inmersiva para el tratamiento de personas con movilidad reducida, con el fin de mejorar no solo los procesos de rehabilitación sino también la obtención de datos para su posterior análisis.

1.1 Origen del trabajo

El campo de la medicina y la tecnología siempre han estado muy unidos ya que cuando una avanza, le da terreno a la otra para seguir innovando. Es por eso que, viendo los avances que hay en la tecnología de hoy en día, sobre todo en el campo de la RV y similares, se ha visto necesario la creación de una aplicación inmersiva para su utilización en el campo de la rehabilitación física.

Este proyecto es la continuación de 2 trabajos anteriores. En el primer TFG, realizado por los exalumnos Joel Guerrero y Alexis Santaeulalia [1], se consiguió conectar el sensor Kinect (un controlador de juego libre y entretenimiento) con Unity (un motor de videojuego multiplataforma). Esto conlleva el posible control de un personaje o avatar con el movimiento del cuerpo y sin necesidad de mandos o botones. No solo eso sino también se puede extraer datos con gran exactitud de la persona que lo está controlando. Un segundo TFG, realizado por otros exalumnos Cindy Vega y Oscar Mayorgas [2] consiguió desarrollar la aplicación con un primer ejercicio de rehabilitación y extraer los datos de los pacientes y fisioterapeutas a una base de datos.

Así pues, este TFG se centrará en el desarrollo de estos ejercicios de rehabilitación de la parte inferior del cuerpo, haciéndolos más divertidos y gratificantes al mismo tiempo que se extraen más datos “frame a frame” y no solo los valores máximos de desplazamiento, como hasta ahora.

Al mismo tiempo, otro grupo, compuesto por Elodie Medina y Víctor Rofes [3], avanzó los ejercicios de la parte superior del cuerpo, así como una base de datos para almacenar los datos. Nuestros compañeros Iván y Dani se encargan del desarrollo de esta parte en su propio TFG.

1.2 Motivación

Desde un principio nuestra principal motivación fue la posibilidad de trabajar en el campo de la rehabilitación utilizando tecnologías muy poco exploradas que pueden contribuir a la recuperación de los pacientes.

Además, otra de las razones que nos motivó fue el hecho de poder mejorar el proceso de rehabilitación y diseñar un ambiente entretenido, en donde el paciente no pierda la motivación ni el interés por la terapia que se le está realizando, ya que en muchas ocasiones el realizar ejercicios repetitivos no sólo es tedioso sino desmotivador al no ver avances, es por esto que por medio de esta aplicación se podrá evaluar qué tanto progreso se logró de una terapia a otra.

Otro aspecto que nos motivó a llevar a cabo este proyecto es el trabajar sobre el ejercicio de equilibrio ya que es una de las afectaciones más comunes y de la que más impacto tiene en su vida cotidiana, todo esto fue posible ya que nos apasiona el saber que gracias a nuestro trabajo podamos no solo tener un impacto positivo en los pacientes sino también poder contribuir con futuras investigaciones.

1.3 Requerimientos previos

Para poder alcanzar los objetivos propuestos en este proyecto fue necesario adquirir conocimientos previos en lenguajes de programación como C#, PHP y MySQL. En temáticas como la rehabilitación, la utilización de la plataforma para la creación de videojuegos Unity3D, la creación de bases de datos, y la biomecánica.

1.4 ADFO

La ADFO es una Asociación de Disminuidos Físicos de Osona. Se trata de una organización sin ánimo de lucro que da soporte a la integración de sus asociados en diferentes ámbitos: familiar, laboral, formación y deporte [46].

2. Introducción

La realidad virtual es usada principalmente para el entretenimiento, pero en los últimos años cuenta con un crecimiento notable en el sector de las ciencias de la salud, implementando así estas herramientas como parte interactiva entre la relación médico paciente, ayudando a facilitar diagnósticos, planear procedimientos médicos con mayor detalle y dar soluciones a problemáticas preexistentes.

Dentro de esta categoría de las ciencias de la salud se puede encontrar la fisioterapia que es conocida por brindar un tratamiento terapéutico y de rehabilitación usado para la prevención, tratamiento y diagnóstico de síntomas de dolencias tanto crónicas como agudas. En ocasiones los ejercicios de rehabilitación suelen ser repetitivos y tediosos, para solucionar esto se implementó esta nueva técnica vinculada con la tecnología mediante la cual se quiere lograr un mejor impacto en la perspectiva y resultados que ven los pacientes; para ello se desarrolló esta aplicación de rehabilitación por medio de la realidad virtual.

2.1 Objetivos del trabajo

2.1.1 Objetivo General

- Implementar un juego/programa con su respectiva interfaz en realidad virtual a una aplicación médica, para ser utilizado en el ámbito de la rehabilitación de personas con discapacidad en las extremidades inferiores.

2.1.2 Objetivos Específicos

- Implementar un juego/programa con su respectiva interfaz en realidad virtual a una aplicación médica, para ser utilizado en el ámbito de la rehabilitación de personas con discapacidad en las extremidades inferiores.
- Diseñar un ambiente entretenido de realidad virtual que sea utilizado para la rehabilitación de las extremidades inferiores de pacientes que han sufrido la enfermedad de ictus.
- Desarrollar tres niveles diferentes en donde se puede ajustar la dificultad según el avance del paciente.
- Extraer datos como ángulos de caída y valores de posición de los pacientes durante el proceso de rehabilitación para el posterior análisis de estos. Cada uno de estos ficheros tendrá el respectivo DNI de cada paciente, la fecha y la hora de la terapia realizada.
- Generar por medio de los ficheros obtenidos una gráfica en tiempo real de los ángulos de equilibrio de cada paciente.
- Desarrollar una interfaz llamativa, práctica y eficaz, que sea fácil de usar tanto por el paciente como por el fisioterapeuta.

- Realizar primeras pruebas en pacientes sanos con la aplicación diseñada.
- Comparar los datos obtenidos en diferentes pacientes sanos para validar la funcionalidad de aplicación creada.

2.2 Alcance del trabajo

Este trabajo tiene como antecedentes tres proyectos desarrollados en Trabajos Final de Grado (TFG); *"VIRTUAL MIRROR Aplicación de Realidad Virtual"* [1]*, "VIRTUAL REALITY REHABILITATION APP (UPPER BODY)"* [3] ambos elaborados el semestre anterior (primavera 2019-2020) y *"REHABILITATION APP"* [3] elaborado el semestre de otoño 2018-2019.

De ahí, teniendo como base estos proyectos, este TFG trata de avanzar y ampliar las funciones de esta aplicación de rehabilitación. Por esa razón, se ha incrementado el campo de ejercicios con un juego basado en la rehabilitación de la parte inferior del cuerpo. De este modo, se consigue que las sesiones de rehabilitación sean más motivadoras, entretenidas y amenas al mismo tiempo que se extrae ficheros detallados de sus movimientos y ángulos de interés. Estos ficheros son muy fáciles de manejar y al mismo tiempo visuales, ya que se pueden graficar con mucha facilidad.

Estos juegos de rehabilitación se adaptan dependiendo del paciente a través de niveles con diferente dificultad y entorno. Asimismo, también se puede regular la velocidad de cada nivel con el menú de opciones (de entre muchas otras funciones que se mencionan en el apartado 6.2.3).

Además, con la intención de ver su funcionalidad y saber qué líneas seguir en un futuro, se ha llevado a cabo un estudio estadístico con personas sin problemas de movilidad para ver sus resultados y poder ajustar la aplicación acorde a éstos. Así pues, se puede recalibrar la aplicación y reajustarla para su funcionamiento.

3. Estado del arte

3.1 Realidad virtual

La realidad virtual tiene sus inicios en los Estados Unidos el 1935. Cuando un joven llamado Stanley G. Weinbaum publicó un libro con el nombre de "***Pygmalion's Spectacles"*** [4]. En este corto libro de ciencia ficción se describe por primera vez un aparato que, citando el libro*: "But listen--a movie that gives one sight and sound. Suppose now I add taste, smell, even touch, if your interest is taken by the story. Suppose I make it so that you are in the story, you speak to the shadows, and the shadows reply, and instead of being on a screen, the story is all about you, and you are in it. Would that be to make real a dream?",* y continúa con una descripción más física: *"In his room Ludwig fumbled in a bag, producing a device vaguely reminiscent of a gas mask. There were goggles and a rubber mouthpiece […]"* (Weinbaum, 1935).

Más adelante, el cineasta multimedia Morton Heilig a mediados de los años 50, para ser más concreto el 1955, publicó en un periódico llamado "***The Cinema of the Future***" (Robinett, 1994) su idea llamada "Experience Theater", donde explicaba que el teatro era una actividad capaz de abarcar todos los sentidos de forma eficaz e inmersiva para el espectador. Por esa razón, el 1962 construyó una máquina que denominó como Sensorama, ésta permitía envolver 4 de los 5 sentidos con pantallas 3D estereoscópicas, sonido estéreo, ventiladores, generadores de olor y una silla que vibraba.

Figura 3.1: Sensorama (Fuente: mortonheiling.com)

Mientras tanto, en 1961, dos ingenieros de la empresa "Philco Corporation" desarrollaron el primer precursor del sistema HMD (Head Mounted Display), tal y como los que hay hoy en día. El dispositivo se llamó "Headsight". El artilugio tenía dos pantallas independientes para cada ojo i un sistema de seguimiento de movimiento magnético, el cual estaba conectado a una cámara de circuito cerrado. El instrumento no fue creado con fines lúdicos, sino que tenía como objetivo el entrenamiento militar a partir de situaciones difíciles y peligrosas. El circuito cerrado de la cámara, mencionada anteriormente, permitía al usuario poder ver sus alrededores ya que, al moverse, la cámara también se movía.

Al cabo de 4 años, en 1965, Ivan E. Sutherland describió un concepto llamado "The Ultimate Display" [5]. El último parágrafo de este breve e interesante informe contiene las fundaciones de la VR de hoy en día: *"The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked."* (Sutherland, 1965).

De ahí, Ivan Sutherland y su estudiante, Bob Sproull, crearon en 1968 el primer HMD que, a diferencia del "Headsight" de Philco Corporation, se conectaba a un ordenador. Este invento se llamó "Sword of Damocles". Aun siendo muy pionero en su tiempo, era tan grande y pesado que se precisaba de una estructura en el techo para poder ser utilizada (véase en la Figura 3.2).

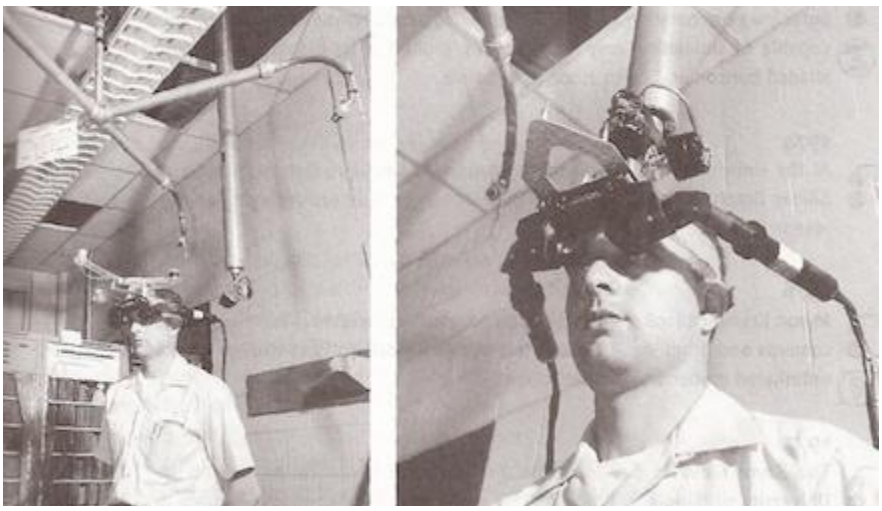


Figura 3.2: Sword of Damocles (Fuente: mortonheilig.com)

Aún ser un campo en desarrollo y con un gran crecimiento, el término "Virtual Reality" no fue utilizado hasta el 1987, cuando Jaron Lanier, fundador del VPL (visual programming lab) realizó muchos avances tecnológicos para el campo de la RV como, por ejemplo: el "Dataglobe" y el "EyePhone". Esta empresa fue la primera en vender productos de RV para ocio.

Figura 3.32: Dataglobe y EyePhone HMD (Fuente: mortonheilig.com)

A partir de entonces, el progreso de la RV fue en aumento y sus recursos, capacidades y aplicaciones fueron creciendo a medida que pasaban los años. En 1991 se empezaron a crear juegos arcade con VR. En 1993, la empresa japonesa SEGA quiso lanzar al mercado sus "SEGA VR headset". Dos años más tarde, en 1995, la empresa Nintendo también quiso enseñar al público su producto llamado "Virtual Boy". Aunque no triunfaron por problemas técnicos en el desarrollo de las gafas, cada vez se tenía más claro que la RV sería una realidad en poco tiempo.

No fue hasta entrar en el siglo XXl que la RV tuvo un gran impacto en la sociedad y pudo dar los frutos que se buscaba en los años 90. En el 2010, Palmer Luckey creó el primer prototipo de sus famosas gafas "Oculus Rift". Estas gafas permitían un campo de visión de 90º, lo que interesó mucho a los diseñadores de juegos y devolvió el interés en la RV.

Figura 3.4: Gafas de RV HTC Vive con sensores base y mandos (Fuente: https://www.xataka.com)

A partir de entonces, la RV subió exponencialmente en el mercado y atrajo a grandes empresas como Sony Entertainment, Google, Facebook (quien compró la empresa de Palmer Luckey) y Samsung entre muchos otros. Todos ellos sacaron sus propios HMD con la mejor tecnología al mercado entre 2012 y la actualidad.

3.1.1 Tipos de realidad virtual

La RV se puede distinguir por el grado de inmersión en la que se encuentra el usuario. Estos grados son: sistemas no inmersivos ("Desktop"), sistemas de proyección semi inmersivos y sistemas totalmente inmersivos HMD ("Head-Mounted Dispay" o dispositivos montados en cabeza, en castellano).

- **Sistemas no inmersivos:** Tal y como sugiere el nombre, este tipo de RV es la menos inmersiva que se puede utilizar. El entorno virtual se ve a través de una pantalla o ventana como la de un ordenador. La interacción entre el mundo virtual y el usuario puede ser totalmente convencional, por ejemplo: usando teclados o ratones; o se puede mejorar con el uso de un dispositivo de interacción 3D como lo puede ser el "SpaceBall" o "DataGlove" [18].

Figura 3.5: Ejemplo de RV no inmersiva con controles convencionales. (Fuente: N/A)

Figura 3.6: Control 3D "SpaceBall" (Fuente: https://www.pinterest.com/pin/442267625886170548/)

La ventaja que tienen los sistemas no inmersivos es que no requieren el más alto nivel de rendimiento para los gráficos y no necesita de hardware especial o complicado y eso hace que sean más económicos.

- **Sistemas de proyección semi inmersivos:** Los sistemas de proyección semi inmersiva son la combinación de tecnología de RV y tecnología desarrollada para los simuladores de vuelo. Este sistema necesita un buen rendimiento de la tarjeta gráfica para proyectar en una gran pantalla, sea de ordenador o varios o sobre un fondo liso como puede ser el caso de un proyector convencional.

Figura 3.7: El uso de la RV semi inmersiva es muy utilizada para el entrenamiento de pilotos de vehículos de carreras o Comerciales.

Figura 3.8: Un uso bastante acertado es la práctica de tiro para los policías y agentes en pruebas.

Su mayor problema es el coste que requiere el obtener y mantener esta tecnología en uso. Así que para utilizar una proyección semi inmersiva se debe analizar bien la causa de su uso y las ventajas que esta tecnología aporta.

- **Sistemas totalmente inmersivos:** Este tipo de sistema es muy comúnmente conocido. Los dispositivos que permiten una inmersión total se llaman "Head-Mounted Display" (anteriormente comentados como HMD). Estos dispositivos tienen una pantalla por cada ojo y permiten la recreación en 3D de un mundo virtual en el que el usuario puede jugar siendo totalmente envuelto en la experiencia y no tener conocimiento alguno del mundo exterior.

La pega de los HMD es su gran demanda en hardware y software por parte del sistema que lo hace funcionar. Es por eso que tienden a ser bastante costosos.

A continuación, en la Tabla 3-1, se puede observar una comparativa entre estos tres tipos de RV, tanto sus ventajas como desventajas.

CARACTERÍSTICAS PRINCIPALES	RV NO INMERSIVA	RV SEMI INMERSIVA	RV INMERSIVA
RESOLUCIÓN	ALTA	ALTA	BAJA-MEDIA
PERCEPCIÓN (ESCALA)	BAJA	MEDIA - ALTA	ALTA
SENSACIÓN DE INMERSIÓN	NINGUNA - BAJA	MEDIA - ALTA	MEDIA - ALTA
SENSACIÓN DE CONOCIMIENTO DEL ESPACIO	BAJA	MEDIA	ALTA
RETRASO (LAG)	BAJA	BAJA	MEDIA - ALTA
SENSACIÓN DE OBSERVACIÓN	BAJA	MEDIA	ALTA

Tabla 3-1: Diferencias entre los 3 tipos de inmersión en la RV.

3.1.2 Realidad virtual aplicada a la rehabilitación

Las aplicaciones que tiene la RV son muy vastas. Ya que permite distorsionar la realidad y engaña a los sentidos para crear espejismos u otras realidades. En 1992, mientras Jaron Lanier diseñaba el "Dataglobe" y el "EyePhone", Brett Leonard dirigió la película "The Lawnmower Man" donde expandió la RV al mundo de la rehabilitación y el tratamiento de pacientes discapacitados. En esta película un científico usaba las gafas de RV con un paciente con discapacidad mental.

Al no tener la tecnología de la RV desarrollada hasta tal punto que pudiera ser útil, no se empezó a utilizar hasta hace unos pocos años. Aun así, desde el 2007, con el lanzamiento del Wii Fit, se ha empezado a aplicar el juego con la rehabilitación física [6].

Figura 3.9: Paciente con discapacidad física estimulando sus habilidades motoras a partir del Wii Fit.

Según los expertos: *"Los estímulos que provienen de la posición de nuestras articulaciones y su relación con el del medio, que en este caso sería un ambiente virtual, inducen cambios en las células de la corteza y subcorteza cerebral (neuronas). Los estímulos de cierta manera "despiertan" a las células motoras que no realizaban ninguna actividad, y organizan a todas las células involucradas en el patrón de movimiento para mejorarlo. La realidad virtual guía la ejecución del movimiento: si perdemos en el juego es porque no conseguimos hacer el patrón de movimiento bien, eso despierta la competitividad de modo que lo repetimos hasta hacerlo bien y ganar."* (Martínez, 2016).

Actualmente, el mundo de la fisioterapia está actualizándose y cambia sus formas tradicionales de rehabilitación a otras más inmersivas y amenas para el paciente. Universidades como "Universidad Carlos lll" han estudiado el uso de la RV dentro de la fisioterapia obteniendo unos resultados excelentes. Esta aplicación se centra en los movimientos de la aducción y la abducción del hombro simulando un

portero de fútbol. El sistema de RV usado en el estudio realizado por la Universidad Carlos lll utiliza el sensor de movimiento "*Intel RealSense*" y las gafas de RV "*Oculus Rift DK2*". El prototipo diseñado se presentó en a una revista llamada "*Procedia Computer Science*" y a un congreso científico internacional sobre la aplicación de nuevas tecnologías en el ámbito de la salud "*HCist*" [7].

Por otro lado, la "Universitat de Barcelona" (UB) publicó, en 2017, una tesis doctoral de M. Matamala Gómez - titulada: "***The use of immersive virtual reality in neurorehabilitation and its impact in neuroplasticity***" - que estudió el impacto que tiene la RV inmersiva en la neurorehabilitación y su impacto en la neuroplasticidad. El él se tiene como objetivo ser un estudio cuantitativo de la efectividad de la RV inmersiva en rehabilitación neurológica y ortopédica y, explorar la utilidad que tiene ésta para el tratamiento del dolor neuropático crónico (discapacidad cognitiva o "*cognitive impairment*"). Para llevar a cabo estos objetivos, diseña, implementa y testea todo un seguido de ejercicios de RV inmersiva para la rehabilitación el brazo (troco superior o "*Upper Limb*") [27]. La creación dé las escenas virtuales será desarrollado mediante la plataforma Unity3D.

Los ejercicios y testeos hechos por M. Matamala Gómez consiguieron extraer unos resultados favorables a la implementación de la IVR ("*Immersive Virtual Reality*" o RV inmersiva) con una terapia de observación activa. También concluye que el uso de la IVR acelera el proceso de rehabilitación y ayuda considerablemente en la recuperación de reaprendizaje motor tanto en el troco superior como en el tronco inferior ("*Lower Limb*") [27].

A pesar de ser un campo en investigación, aún no hay un gran número de aplicaciones y/o dispositivos que te permitan hacer la rehabilitación desde un sitio remoto y, por lo tanto, los hospitales (mayoritariamente privados) tienen los dispositivos de RV o de rehabilitación inmersiva dentro de sus instalaciones.

Para concluir, este proyecto en específico quiere poder extraer los datos con una precisión milimétrica para poder tener un seguimiento del paciente extraordinario, mientras éste juega a un juego divertido y trabaja en su objetivo sin que se dé cuenta o sufra descontentos por la repetitividad de las sesiones.

3.2 Unity

Unity tiene sus inicios en la Conferencia Mundial de Desarrolladores de Apple en el año 2005. Unity es una plataforma diseñada por Unity Technologies Este programa fue diseñado para funcionar y crear proyectos exclusivamente en equipos soportados por la plataforma Mac. A partir de ahí, Unity ganó un gran éxito y continuó con el desarrollo de la plataforma.

El 2010, se lanzó Unity 3. En esta versión de Unity se empezó a introducir gran cantidad de herramientas utilizadas por los grandes desarrolladores con fin de captar el interés de éstos.

A continuación, empezaron a sacar más versiones como Unity 4, la cual tuvo muchas mejoras respecto a la versión anterior; Unity 5, lanzado el 2015 y con una gran sorpresa: en su sexta actualización, versión 5.6, se introdujo el 2D Toolkit y el soporte de la RV y la RA (realidad aumentada).

A partir de entonces no hay muchas mejoras a destacar y Unity no se utiliza en grandes juegos por su poca capacidad de operación cuando hay una gran cantidad de gente trabajando en un solo proyecto.

La última versión que hay de Unity en la actualidad es "*Unity 2019.1*".

3.2.1 Unity aplicado a la rehabilitación

El programa Unity 3D ha sido una de las plataformas favoritas para los diseñadores de juegos. Esta plataforma está bien para hacer juegos simples de forma rápida pero efectiva. Los juegos de rehabilitación suelen ser sencillos para que el paciente no se distraiga con facilidad u obtenga más información de la necesaria. Es por eso que Unity 3D es un gran programa para este tipo de aplicaciones.

Actualmente, tanto las clínicas como los centros privados centrados en la rehabilitación están implementando, desde hace ya algunos años, juegos no inmersivos o semi inmersivos para la realización de sus pacientes. Ha sido una práctica que ha mostrado ser muy exitosa ya que consigue resultados y los pacientes parecen estar más contentos con la experiencia ya que son más amenas y no frustran al ser tan repetitivas.

En varias Universidades y empresas se han llevado a cabo varias aplicaciones y de RV que permiten la rehabilitación de paciente, como es el ejemplo de Carlos lll, donde consiguieron el buen recuperamiento de un paciente con un problema de movimiento motor del hombro.

Figura 3.10: Juego de rehabilitación desarrollado con Unity3D (Fuente: https://www.researchgate.net/publication/319024764_An_adaptive_self-organizing_fuzzy_logic_controller_in_a_serious_game_for_motor_impairment_rehabilitation/figures)

Un proyecto de rehabilitación mediante la RV llamado "***Virtual Reality Interface Built Using Unity3D for Rehabilitation with BCI Systems Based on Motor Imagery***" presentado en el "*2018 IEEE Biennial Congress of Argentina*" utilizó el Unity 3D para crear su mundo virtual ya que: *"Este motor de videojuegos se caracteriza por su intuitivo manejo y por la gran variedad de recursos, herramientas, tutoriales y trabajos disponibles. Asimismo, permite crear videojuegos compatibles con numerosas plataformas en 2D o 3D, utilizando lenguaje Java o C#. También permite la comunicación con aplicaciones externas, [...]"* [22].

La interfaz permite rehabilitar tanto la parte superior del cuerpo como la inferior después de haber padecido un accidente cerebrovascular (Ictus). Este estudio trata de promover la plasticidad neuronal "brindando realimentación" al paciente de forma que permita el reaprendizaje motor [22].

Otro estudio desarrollado por *JHI* ("Journal of Health informatics") y titulado: "***Serious Games for Stroke Rehabilitation Employing Immersive User Interfaces in 3D Virtual Environment***" [23] tiene como a objetivo la reducción de costos y recursos humanos que presentan hoy en día las terapias de rehabilitación para pacientes que han sufrido accidentes vasculares y/o derrames. Este informe potencia la posibilidad de hacer rehabilitaciones domésticas para rebajar costes aplicando la RV. Para la realización del proyecto utilizan tanto Unity 3D para la UI y el sensor Kinect para captar los datos del paciente y almacenarlo en un servidor PHR [23]. Aunque apuntan que la Kinect no puede ser usada en la parte del tronco superior a corta distancia; el proyecto valida y confirma el prototipo diseñado mediante una serie de pruebas a pacientes obteniendo unos resultados exitosos.

A causa de tener una gran comodidad de uso y una gran facilidad de adquisición, Unity3D es un programa de desarrollo de videojuegos que ha tenido más impacto y aplicaciones en el mundo de la rehabilitación. En muchos proyectos [23] [26] inmersivos o semi inmersivos de RV se ha utilizado esta plataforma para la creación de los espacios, actividades y ejercicios de rehabilitación físicos y de estímulos cerebrales [22] [23] [26].

3.3 Kinect

El dispositivo llamado "*Kinect*" tiene sus inicios el junio de 2009 con el nombre de "*Project Natal*". Fue una herramienta que podía traquear a 4 personas a la vez. La Kinect se presentó como a un controlador de juego libre para la consola Xbox 360 (lanzada al público el año *2010*). Durante el diseño de la Kinect, los desarrolladores de videojuegos empezaron a adaptar y experimentar con los controles de los juegos para poder incluir el nuevo sensor.

No fue hasta junio de 2010 que en la conferencia "*E3 2010"* se quiso hacer un evento llamado "*World Premiere 'Project Natal' for Xbox 360 Experience*". A partir de ese día, la empresa Microsoft anunció que el controlador de juego se llamaría "*Kinect*" (formado por las palabras *kinetic* y *connect*).

Un año más tarde, en junio de 2011, Microsoft anunció el lanzamiento del SDK (*"software development kit"*) dando pie a desarrollar aplicaciones no comerciales que usaban Microsoft Windows 7, la Kinect y la Xbox 360.

Figura 3.11: Kinect para Windows [24].

Aun tener un *Guinness World Record* por ser la pieza de electrónica más vendida (35 millones de unidades), Microsoft no consiguió capitalizar el sistema de control libre como lo hizo *Nintendo Wii*. Por esa razón, en 2017 la empresa confirmó la descontinuación del producto.

Actualmente Kinect tiene dos versiones "*V1*" y "*V2*" pero gracias a la nueva aplicación que tiene en la RV, se habla de una posible "*V3*" [24] y las utilidades que tendría en este campo.

Figura 3.12: Kinect aplicada a los controles de la RV [24].

3.3.1 Kinect aplicado a la rehabilitación

La Kinect ha tenido un gran papel en el desarrollo de aplicaciones para la rehabilitación. Al ser un sensor de juego libre, permite captar el cuerpo humano y posteriormente permite su análisis a partir de unas librerías creadas para programar en Visual Studio (creada por la empresa Microsoft). Al poder extraer datos de la posición de puntos de interés, se puede pedir al programa Visual Studio que haga cálculos para obtener ángulos, velocidades, desviaciones o muchas más operaciones.

Para poner un ejemplo, hay un informe llamado "***Towards Pervasive Physical Rehabilitation Using Microsoft Kinect***" [19] creado por varios investigadores y publicado por el IEEE ("Institute of Electrical and Electronics Engineers") en 2012. En este estudio se compara el uso de la Kinect y otro sensor de juego libre más sofisticado, pero más caro y con un hardware mucho más complicado, llamado OptiTrack. Para la creación del espacio virtual se usó el desarrollador Unity3D. El informe final concluye analizando la reducción de costes que supone aplicar la RV i la Kinect en las sesiones de rehabilitación ya que permiten un uso doméstico y su coste no es muy elevado [19].

Otro estudio que hay sobre la relación que tienen la Kinect y la fisioterapia es un artículo llamado "***The emerging role of Microsoft Kinect in physiotherapy rehabilitation for stroke patients***" publicado en el portal/foro de fisioterapia "*Phisiopedia*". En él se comenta el gran crecimiento de la Kinect en el mundo de la rehabilitación y el nuevo uso que se dio a este fallido sensor creado para el desarrollo de juegos de ocio descontinuado en 2016. Este proyecto utilizó la Kinect para interaccionar con el sistema en un espacio 3D semi inmersivo. Este artículo enseña muchas aplicaciones en el mundo de la rehabilitación, ya sea para un paciente con un hueso roto o para un accidente cerebrovascular (ictus). Todos los pacientes parecen tener una buena progresión y las sesiones se transforman en lúdicas. La primera frase de su conclusión resume el artículo de arriba a abajo: "*Microsoft Kinect is a device which has been*

utilized to allow virtual rehabilitation differing from more conventional therapy in its ability to provide more innovative and exciting ways to rehabilitate" [20].

Figura 3.13: Rehabilitación mediante el uso de la Kinect y juegos semi inmersivos [20].

El nuevo uso de la Kinect junto con la RV abre un gran abanico de posibilidades en el mundo de la fisioterapia y la rehabilitación. Tal es el desarrollo de estos dos campos que, hoy en día, la gran mayoría de estudios que aplican la RV a la fisioterapia, utilizan la Kinect como a controlador de movimiento.

En 2018, Richard J. Adams, miembro *Senior* del *IEEE* ("Institute of Electrical and Electronics Engineers"), utilizó los sensores de la Kinect en su estudio: "***Virtual Activities of Daily Living for Recovery of Upper Extremity Motor Function.***" para investigar la recuperación de la capacidad motora en extremidades superiores mediante el uso de la RV [28].

Existen juegos que, tan sólo usando la Kinect v2 ya pueden funcionar para las sesiones de fisioterapia y rehabilitación. "***Validation of a Kinect V2 based rehabilitation game*"** [25] es un trabajo que estudia la aplicación y la utilidad que tiene la Kinect. En este proyecto también se estudian los datos extraídos de la Kinect (mediante el SDK v2 de la Kinect para Windows) y se discuten los resultados obtenidos. Fijarse en las ecuaciones de la Figura 3.14.

$$Extent\ of\ reach = \sqrt{(h_x - s_x)^2 + (h_y - s_y)^2 + (h_z - s_z)^2},$$

$$Hand\ Velocity = \frac{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2 + (z_{i+1} - z_i)^2}}{t_{i+1} - t_i},$$

Figura 3.14: Cálculos realizados para extraer la posición y la velocidad. La posición de la mano está representada por el vector (h_x, h_y, h_z), el hombro por (s_x, s_y, s_z), t simboliza el tiempo y el vector (z_i, y_i, z_i) la posición de la mano [25].

Estas ecuaciones, extraídas del proyecto mencionado anteriormente [25], son algunas de las que se utilizan en este proyecto para determinar la posición de las extremidades inferiores y de la cadera. Para concluir con este estudio, parece interesante introducir una parte del estudio que llama la atención en el momento de obtener los datos: "*Overall, the average difference values of maximum extent of reach*

were less than 3 cm across all six trials and the percentage error was less than five percent" [25]. Esta afirmación nos dice que el sistema de tracking utilizado por la Kinect es fiable y los datos extraídos son válidos para su posterior análisis.

Un año más tarde, en 2019, Han Suk Lee, investigador en el campo de la terapia física en la Facultad de Ciencias de la Salud de la República de Corea, utilizó el sensor Kinect para hacer un estudio sobre los

Figura 3.143: Rehabilitación de la extremidad superior mediante el control de la Kinect [29].

efectos de la RV aplicado a la terapia en pacientes que han padecido de ictus o infarto: "***The Effects of Virtual Reality Training on Function in Chronic Stroke Patients: A Systematic Review and Meta-Analysis***" [29]. H. Suk Lee concluyó en su trabajo que la rehabilitación mediante la RV ayuda a mejorar a los pacientes que padecen de ictus crónico. El uso de la RV ayuda tanto a la parte superior del cuerpo (*upper limb*) como a la parte inferior (*lower limb*). Afirma que la rehabilitación requiere al menos de 8 semanas (de manera más casual) y que éste es más eficaz que un tratamiento diario. El estudio también afirma que el efecto de la RV ayuda al equilibrio en pacientes con problemas cerebrovasculares: "*This study has a moderate evidence to support the effect of VR on lower extremity function in patients with chronic stroke. Therefore, VR training would be helpful in improving functional outcomes with chronic stroke patients such as gait […], balance […], lower limb movement […], lower limb strength, and lower limb muscle tone*." [29].

3.4 Bases de datos

Una base de datos (BD) es la colección organizada de data. Es decir, un banco que permite guardar datos de cualquier tipo. Normalmente se accede a ellas mediante ordenadores. Para hacer funcionar una BD se utiliza un software llamado "*DataBase Management System*" (DBMS). Éste permite la interacción con usuarios, aplicaciones y la misma BD para capturar, analizar y almacenar los datos.

Las BD se han utilizado desde la Antigüedad. Las bibliotecas y los registros fueron las primeras BBDD que existieron. Se utilizaban para recoger información sobre cosechas y censos [30]. Obviamente, la eficacia de esas BBDD era muy poca o nula. Tanto para la creación como para la búsqueda de datos. Posteriormente, con la necesidad de almacenar grandes datos o información se empezaron a desarrollar las BBDD.

En 1884 Herman Hollerith creó una máquina automática de tarjetas perforadas [30]. Este invento le otorgó el puesto a primer ingeniero estadístico de la historia. Esta máquina fue desarrollándose para implementarla en censos y otras sesiones con información abundante.

Figura 3.144: Máquina perforadora (Fuente: https://histinf.blogs.upv.es/files/2011/01/hh-tabulator.gif).

No fue hasta la década de los 50 que se inventó la cinta magnética. Este invento automatizó la información y respaldó en el tratamiento de la información para las nuevas industrias. La desventaja que tenía el dispositivo era que la información solo se podía guardar de forma secuencial.

En la siguiente década, los ordenadores bajaron su precio considerablemente para que las empresas grandes pudieran adquirir uno y popularizar su uso.

Los discos magnéticos también obtuvieron mucha fama debido a su gran facilidad de consulta. Es decir, por primera vez se podía consultar información sin tener que estar directamente en la misma ubicación. Así mismo, en la misma época, se iniciaron las primeras BBDD de red y jerárquicas ya que era posible guardar la información de forma estructurada [30].

Un gran avance para las BBDD fue la cooperación de las empresas IBM y American Airlines para desarrollar SABRE. Éste es un sistema operativo capaz de manejar reservas de vuelos, transacciones y la información sobre los pasajeros.

Posteriormente, en 1967, un consorcio de industrias informáticas llamado CODASYL ("*Conference on Data Systems Languages*") creó el IDS. Esto supuso la creación de un nuevo tipo de sistemas de BD conocido como modelo en red que permitió la creación de standard en los BD debido a la creación de nuevos lenguajes en los sistemas de información [31].

Una vez se entró en la década de los 70, el científico informático Edgar F. Codd publicó un artículo titulado "***A Relational Model of Data for Large Shared Data Banks***" [32]. En este artículo definió el modelo relacional al mismo tiempo que una serie de reglas para los sistemas de datos relacionales. Con este artículo dio paso al nacimiento a una nueva generación en los sistemas gestores de BD.

A consecuencia de esto, Lawrence J. Ellison, a partir del artículo y el trabajo hecho por el señor Codd [32], desarrolló el *"Relational Software System"*, hoy conocido como "*Oracle Corporation*". Dando así un sistema de gestión de BD relacional [30].

En el año 1986, con una gran comercialización y auge por los sistemas relacionales, *SQL* ("*Structured Query Language*") comenzó ser el estándar de la industria. Su sistema de tablas (filas y columnas) dieron mucha competencia a las BD de jerarquía y de red. Su nivel de programación era sencillo y su nivel de programación bajo. El SQL es un lenguaje de consultas y/o declarativo de acceso a BD relacionales que permite efectuar búsquedas y consultas con el fin de recuperar información.

La década de los 90 dio paso a la tercera generación de BD. Esta generación giró en torno a los objetos o, mejor dicho, a la BD orientadas a éstos. Así es como se desarrollaron los paquetes "*Access*" y "*Excel*" de la empresa Microsoft.

Otro evento importante a destacar en la década de los 90 fue el nacimiento del "*www*" ("*World Wide Web*") lo que facilitó la consulta a las BBDD.

Figura 3.161: Logo SQL (Fuente: https://histinf.blogs.upv.es/files/2011/01/imagesCAHW1TUV.jpg).

3.5 Enfermedad de Ictus

El ictus o más comúnmente conocido como infarto cerebral o embolia, es una enfermedad cerebrovascular [21]. Ésta se produce cuando se obstruye el flujo sanguíneo del cerebro por causa de rotura u obstrucción de un vaso sanguíneo. Al no recibir suficiente sangre, las células dejan de funcionar por falta de oxígeno.

Esta enfermedad es muy común, i según los datos recopilados por el **"Grupo de Estudios de Enfermedades Cerebrovasculares de la Sociedad Española De Neurología (GEECV-SEM)"** [21] alrededor de 650.000 personas mueren en Europa a causa del ictus.

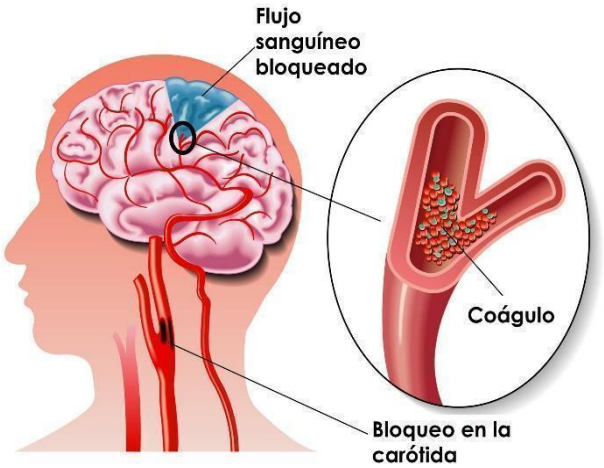


Figura 3.18: Una de las posibles causas del ictus. (Fuente: https://rithmi.com/que-es-un-ictus/#iLightbox[gallery1683]/0).

3.5.1 Tipos y clasificación

- **Isquémico o infarto cerebral:** Este tipo de ictus ocurre cuando una arteria queda taponada. Ya puede ser por un coágulo o un trombo como se muestra en la Figura 3.18.
- **Hemorrágico:** Este tipo de caso es menos frecuente y tan solo se ve en un 10 o 15% de los casos. Tal y como sugiere el nombre, la causa del ictus se trata de la rotura de un vaso sanguíneo que deriva en una hemorragia en el cerebro. Este caso es muy agresivo ya que hay más probabilidades de morir por falta de oxígeno en las células cerebrales.
- **Fibrilación auricular:** Esta causa también se conoce como el tipo más frecuente de arritmia. Supone un gran problema de salud y aumenta el riesgo de sufrir paradas cardíacas

3.5.2 Síntomas y grados de discapacidad

Esta enfermedad conlleva varios síntomas que pueden llegar a ser muy frustrantes. Al ser un ataque al cerebro, puede quedar dañado y/o producir lesiones en poco tiempo.

Dependiendo de la zona que haya sido afectada, se pueden producir distintos síntomas como, por ejemplo:

- Debilidad muscular en la cara, brazos o piernas.
- Hormigueos musculares.
- Dificultad a la hora de hablar, leer o entender palabras.
- Vista distorsionada en uno o ambos ojos.
- Dolores intensos de cabeza.
- Pérdida de equilibrio y coordinación.

Hasta un tercio de las personas que han sufrido un ictus van a tener una discapacidad física o psíquica. Esto hace que su vida cotidiana sea un tanto difícil, ya que en bastantes casos se tendrá que dejar el trabajo y recolocarse en otro sitio más adecuado para la persona afectada.

En todos los casos, el ictus es una enfermedad muy reconocida y el Ayuntamiento accede a proveer a la gente con discapacidad a tener una subvención y ayudas sociales dependiendo del carácter y grado de discapacidad que presente la persona afectada.

Figura 3.19: Síntomas del ictus cerebral (Fuente: https://ictusfederacion.es/infoictus/codigo-ictus/).

3.5.3 Equilibrio y cómo tratarlo

Como se ha mencionado anteriormente, uno de los síntomas de tener un ictus es el desequilibrio y la pérdida de coordinación. Este proyecto tiene como objetivo la implementación de una aplicación totalmente inmersiva para la rehabilitación de este síntoma en concreto.

Hasta ahora, para poder recuperar el equilibrio y la coordinación de todos los componentes dañados se han usado métodos convencionales con la ayuda de fisioterapeutas como se muestra en la Figura 3.20 y Figura 3.21.

Figura 3.20 y Figura 3.21: Sesiones de rehabilitación convencionales para tratar los síntomas del ictus. (Fuente: https://ictusfederacion.es/infoictus/rehabilitacion/)

Todos los ejercicios actuales para la rehabilitación son repetitivos y sin una distracción que haga las sesiones más amenas y divertidas.

3.6 Recopilación de trabajos referenciados para la investigación del estudio

REF.	INSTITUCIÓN	HERRAMIENTAS USADAS	ACTIVIDAD VIRTUAL	OBJETIVO DE REHABILITACIÓN	RESULTADOS
[7]	Universidad Carlos lll	"*Intel RealSense*" "*Oculus Rift DK2*"	Portero de fútbol	Aducción y abducción del hombro (*upper limb*)	El paciente mejoró la capacidad motora del hombro y el tiempo de rehabilitación disminuyó.
[19]	IEEE	"Kinect" "Unity3D"	Coger cápsulas en un mundo virtual	Abducción del brazo (*upper limb)*	Implementar la RV y la Kinect en la fisioterapia reduce los costes y da la posibilidad a una terapia a distancia.
[20]	Physiopedia (artículo)	"Kinect"	Varios juegos de rehabilitación	Todo (*upper limb, lower limb y cognitive impairment*)	Los usuarios se recuperan de una forma rápida y no tan dolorosa.
[22]	IEEE	"Unity3D" "Virtual Reality" "Sistema BCI"	Movimientos cotidianos	Plasticidad neuronal después de padecer un Ictus (*cognitive impairment*)	El usuario presentó mejoras notables, así como una estimulación de la plasticidad neuronal.
[23] [26]	JHI	"Unity 3D" "Kinect" "Touch-Screen" "PHR Server"	Lanzamiento de objetos i el movimiento de un avatar	Reducción de costes en pacientes de accidentes vasculares y/o derrames. Rehabilitación de todo el cuerpo	Confirman la viabilidad de un proceso de rehabilitación doméstica y su reducción de coste.
[25]	University of Missouri – Columbia	"Unity3D" "Kinect"	Varios ejercicios para ejercitar todo el cuerpo	Estudiar el uso de la Kinect comparado con otro sensor de movimiento más sofisticado	La Kinect proporciona unos datos muy similares al otro sensor y es mucho más económico
[27]	UB	"IVR" "Unity3D"	Varios ejercicios de la parte superior del cuerpo	La rehabilitación neuronal a partir de la rehabilitación física	Implementar la IVR reduce el tiempo de rehabilitación y mejora las habilidades tanto cognitivas como físicas.
[28]	IEEE	"Kinect" "SaeboVR"	Actividades de la vida cotidiana	Estudiar la recuperación del movimiento de las extremidades superiores (*upper limb*)	Los resultados muestran una significativa mejora en el control de las actividades motoras.
[29]	Faculty of Health Science (Eulji University)	"Kinect"	Sin especificar	Estudiar la utilidad de la RV en pacientes que han padecido ictus (*upper* y *lower limb*)	La RV incrementa la utilidad y el éxito (de forma modesta) el balance, la fuerza y las funciones motoras.

Tabla 3-2: Recopilatorio de los proyectos, estudios, informes y tesis que han utilizado una o más herramientas usadas en el desarrollo de este proyecto.

4. Planificación del proyecto

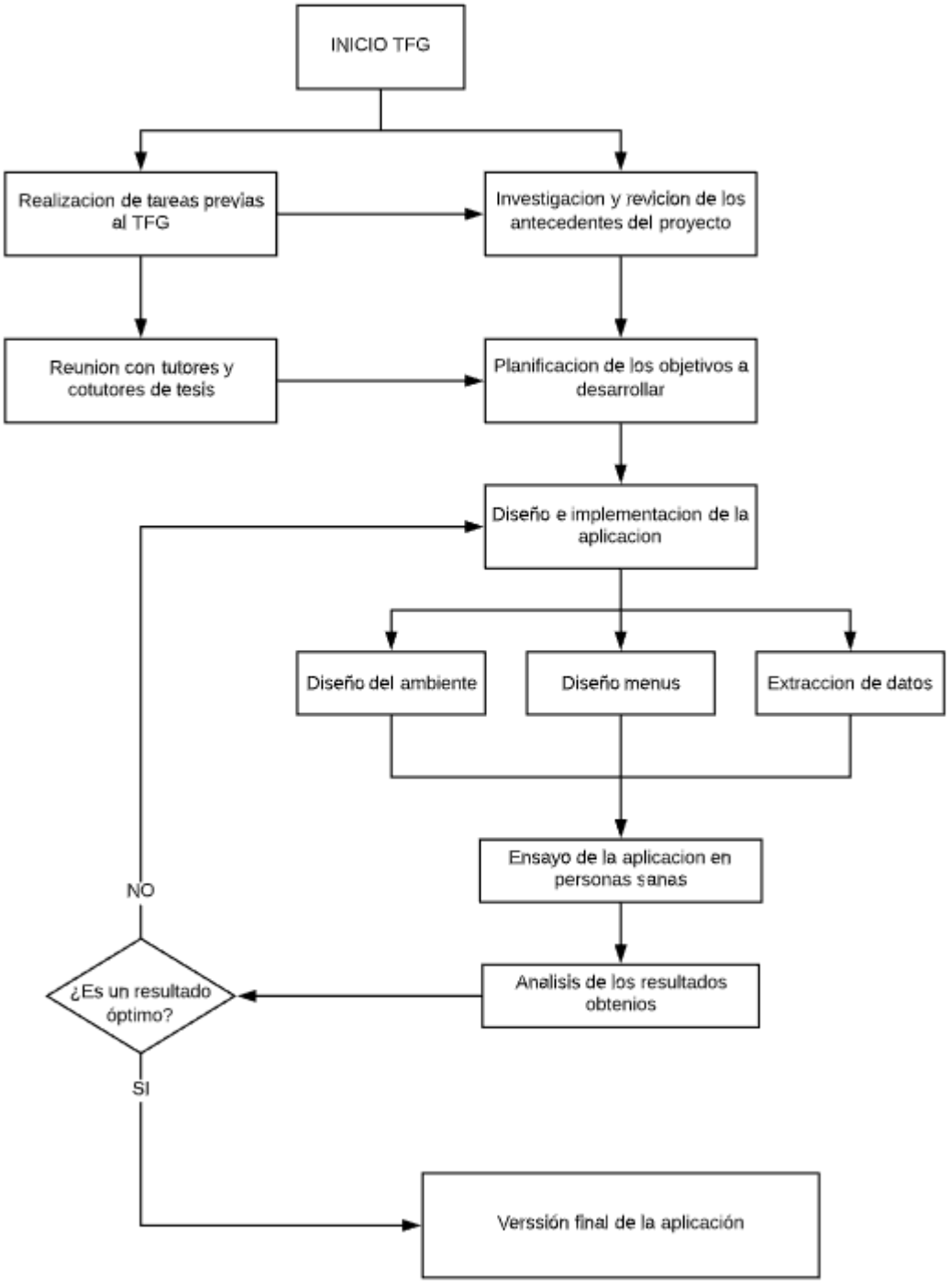


Figura 4.1: Diagrama de flujo de la planificación

4.1 Metodología

Para poder llevar a cabo este proyecto, debimos adquirir conocimientos previos para así lograr un mejor desempeño durante nuestro trabajo, es por esto por lo que se realizó el "Máster en Programación de Videojuegos con Unity 2018 y C#" de la plataforma de aprendizaje Udemy, donde se nos explicaban los conocimientos básicos de la creación de videojuegos en la plataforma de Unity3D.

Una vez realizado el máster, se concretó una reunión con los tutores donde se definieron unas tareas previas al inicio del proyecto (unificar versiones, extraer valores frame a frame de los ángulos e implementar el GIT), pero para llevarlas a cabo lo primero que hicimos fue estudiar y entender los trabajos de grado anteriores [1][2], para así saber que habían logrado y como lo habían hecho. Además, para lograr un entendimiento total del trabajo nos reunimos con uno de los grupos que realizan el TFG anterior y aprovechamos esta reunión para despejar todas las dudas que teníamos acerca de la aplicación.

Después de haber realizado las tareas previas, fueron asignados a cada grupo los temas de TFG en una segunda reunión con los tutores y una fisioterapeuta de la ADFO, seguidamente se definieron los objetivos puntuales que queríamos alcanzar en este proyecto, cada uno de los integrantes planteó ideas para lograrlo.

Posterior a esto se organizaron las tareas propuestas con el fin de estructurar el trabajo, proponiendo fecha de inicio y un estimado de duración para cada una de ellas. Una vez estructurado el plan de trabajo se comenzó con la fase de implementación, la cual se dividió en 4 partes:

1. Creación de tres ambientes para la rehabilitación del equilibrio por medio de un juego de realidad virtual, donde la dificultad del nivel depende del progreso del paciente.

2. Creación e implementación de una interfaz gráfica eficaz y práctica.

3. Extracción de datos para el posterior análisis

4. Implementación de gráficas en tiempo real para brindar un seguimiento constante al paciente

Estos puntos serán explicados a detalle más adelante, sin embargo, todas las tareas y reuniones estarán descritas en el diagrama de Gantt.

Por último, al terminar con la implementación de la aplicación, se evaluó qué tan viable y eficaz era esta; para ello, se probó en personas sanas y los resultados obtenidos fueron utilizados para evaluar la calidad de la aplicación por medio de un análisis estadístico el cual será presentado más adelante.

4.2 Diagrama de Gantt

Trabajo a realizar	*Inicio*	*Duración*
Primera Reunion	18-sep.	0
Integracion del GIT	2-oct.	5
Unificacion de Versiones (Upper y Lower)	19-sep.	1
Extraccion de los Valores Frame a Frame	26-sep.	7
Personalizacion de los Valores Temporales	1-oct.	7
Segunda Reunion (Asignacion de proyecto)	9-oct.	0
Antecedentes del Proyecto	10-oct.	15
Revision y Entendimiento del TFG Pasado	26-sep.	5
Reunion con los Integrantes de TFG Pasado	25-sep.	0
Planteamiento del los Objetivos a Desarrollar	10-oct.	2
Diseño del Ambiente Inmersivo en Unity3D	12-oct.	30
Programación de las interacciones del ambiente en V.R	14-oct.	30
Programacion de los sistemas de control de movimiento por V.R	14-oct.	16
Programacion del sistema de control del personaje por medio de Kinect	15-dic.	2
Diseño de la interfaz grafica de la aplicacion	10-nov.	35
Reconocimiento de las extremidades del cuerpo por medio del Kinect	13-oct.	3
Calcular el angulo de equilibrio del pasciente	27-oct.	1
Calcular valores de poscicion del pasciente	28-oct.	1
Extraer valores temporales para ser analizados	30-oct	0
Realizar pruebas en personas sanas	20-dic	4
Analisis de los datos obtenidos	20-dic	19
Realizacion de la memoria	01-oct	65

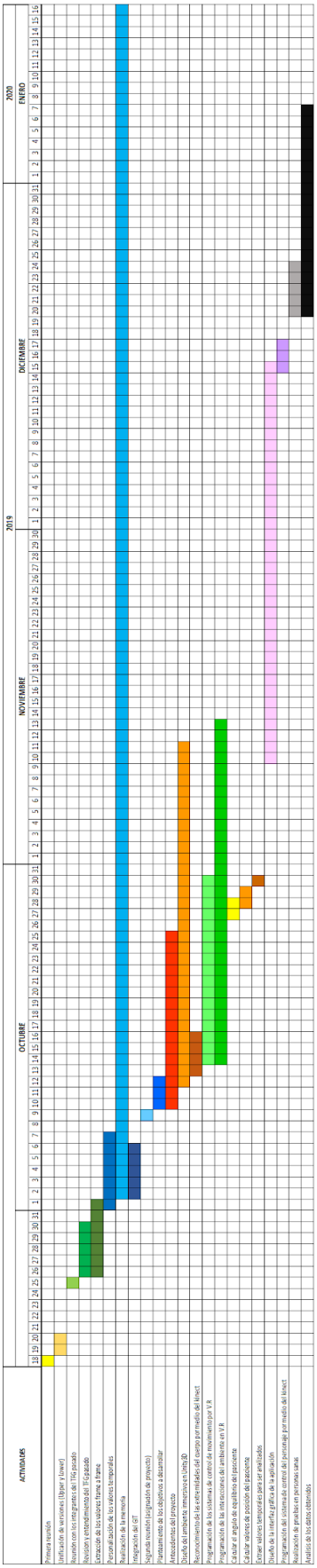


Figura 4.2: Diagrama de Gantt

4.3 Requerimientos

4.3.1 Requerimientos Funcionales

- Registro correcto de la sesión de cada paciente
- Poder escoger el ejercicio diseñado (rehabilitación del equilibrio de forma didáctica)
- Poder escoger el nivel dependiendo del grado de rehabilitación que se necesite
- Poder modificar la dificultad de cada nivel adecuándose a las necesidades del paciente
- Poder modificar el sistema de control de movimiento entre Kinect o gafas HTC Vive dependiendo de las preferencias del paciente
- Acceso a las instrucciones del ejercicio de rehabilitación
- Poder eliminar o ingresar un paciente (lo realiza el fisioterapeuta)
- Extracción de ángulos de equilibrio muestreados (derecha e izquierda)
- Extracción de datos de posición tomados durante la realización del ejercicio
- Posibilidad de volver atrás en todo momento para finalizar sesión
- Acceso al historial de cada paciente

4.3.2 Requerimientos No Funcionales

- Alta velocidad de procesamiento para el muestreo de datos
- Rendimiento (tiempo de respuesta a las acciones del usuario menor a un segundo)
- Hardware (Kinect, equipo HTC Vive)
- Software (Unity, Microsoft Kinect for Windows SDK, XAMPP, MySQL)
- Lenguaje de programación (C#, PHP)
- Interfaz gráfica amigable con el usuario
- Tarjeta gráfica de última generación para tener una alta calidad de imagen
- La versión de Windows debe ser 7 o superior
- Posibilidad de ser utilizada en varios computadores al tiempo
- Posibilidad de funcionamiento sin internet
- Protección de datos, debido a que estos se guardan en un servidor local

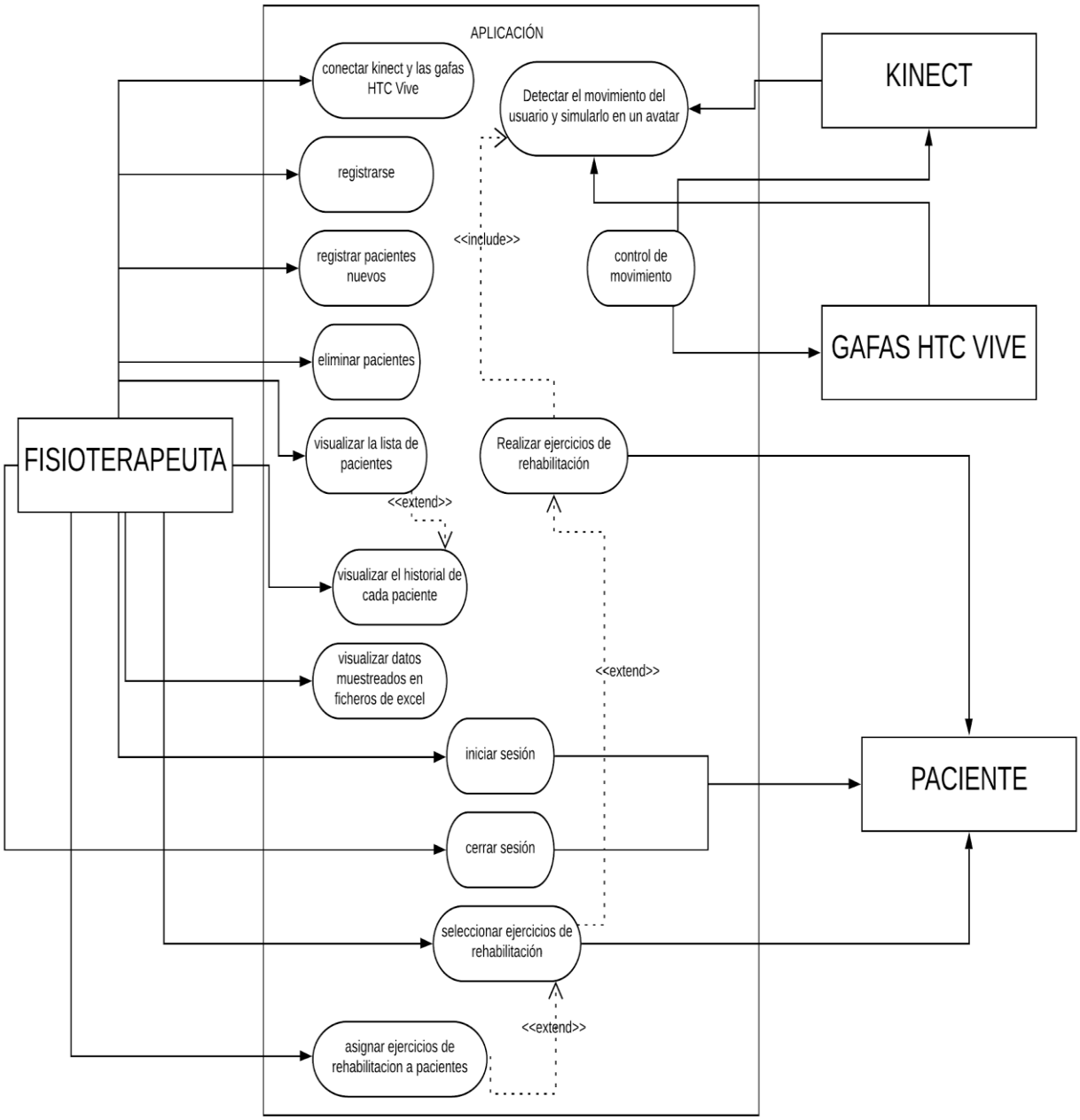


Figura 4.3: Diagrama de casos uso

5. Material utilizado

Para el desarrollo de esta aplicación se ha utilizado herramientas como el motor de juegos “Unity”, el sensor de juego libre “Kinect” y unas gafas de RV “HTC Vive” todo ello proporcionado por la UPC conjuntamente con la empresa tecnológica Visyon360.

5.1 Unity

Unity es un motor de videojuegos multiplataforma, creado por la empresa “Unity Technologies”. Esta plataforma cuenta con un soporte de compilación con una grandísima cantidad de otras plataformas, por ejemplo: WebGL, Linux, Windows, OS X, iOS, Android, Samsung TV, Play Station, Xbox, Wii U, Oculus Rift, Google Cardboard, HTC Vive y muchos más.

Esta plataforma es muy utilizada tanto en grandes empresas, como “Bragg Communications Inc.” [8], como en pequeñas empresas y hasta individuales de manera gratuita si no se supera un ingreso de 100.000 € al año

5.1.1 Interfaz

Figura 5.1: Interfaz de Unity (Fuente: http://deusexmachina.es/taller-empezando-con-unity-3d/)

La interfaz de Unity se divide en 5 pestañas que se pueden desplazar y modificar según los requisitos que tenga el desarrollador. Como se puede observar en la Figura 5.1, en la parte superior se encuentra el "Toolbar" o barra de herramientas. En el Toolbar se puede cambiar el tipo de deformación que se quiere aplicar a un objeto representado en la pantalla "Scene". Asimismo, también se pueden cambiar los puntos de referencia de tal objeto de local a global dependiendo de las necesidades. Finalmente, una de las principales funciones que tiene el Toolbar es el testeo del juego que se está diseñando mediante el "Play" que se encuentra en el centro de la parte superior (véase la Figura 5.2).

Figura 5.2: Vista expandida de la barra de herramientas (Fuente: researchgate.net/figure/The-User-interface-of-Unity-3D-game-engine_fig1_326434680)

Siguiendo con la Figura 5.1, se aprecia una ventana llamada "Scene" o escena. En ella se lleva a cabo el diseño de la aplicación, juego o interfaz del usuario (UI) que se quiera hacer. Esta ventana sirve para ver visualmente el proyecto y hacer las modificaciones en el espacio, dividiendo le espacio con líneas de medida y determinando que se ve en el juego final y que no.

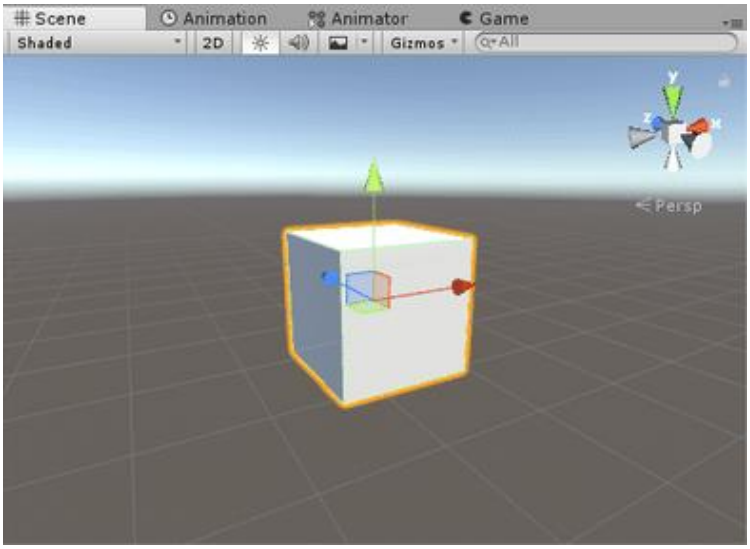


Figura 5.3: Vista expandida de la ventana de Escena (Fuente: researchgate.net/figure/The-User-interface-of-Unity-3D-game-engine_fig1_326434680)

También se puede ver que hay otras ventanas en la parte superior e izquierda en la Figura 5.3 llamadas "Animation", "Animator"y "Game". Las dos primeras ventanas mencionadas son para la creación de Animaciones dentro de un juego o UI y en este proyecto tan solo se usarán para temas estéticos como transiciones. Aun así, la ventana de "Game", juego en castellano, sirve para poder testear el transcurso del proyecto o juego.

Seguidamente, en la parte derecha de la Figura 5.1 tenemos una ventana denominada "Inspector". La función que tiene es la de cambiar los parámetros de los objetos creados. Se puede cambiar el tamaño,

posición y escala, así como introducir sonidos, vídeos, imágenes, texturas, scripts, etc. Tiene muchas funcionalidades que permite que se pueda crear todo un mundo virtual a medida y personalizarlo hasta donde se quiera.

A continuación, en la parte inferior de la Figura 5.1 hay la ventana llamada “Project” o proyecto donde se puede observar y ordenar todos los archivos que tiene una escena o proyecto. Haciendo un símil, se puede decir que su función se asemeja a las carpetas “Explorer” de Windows (ver Figura 5.4 y Figura 5.5).

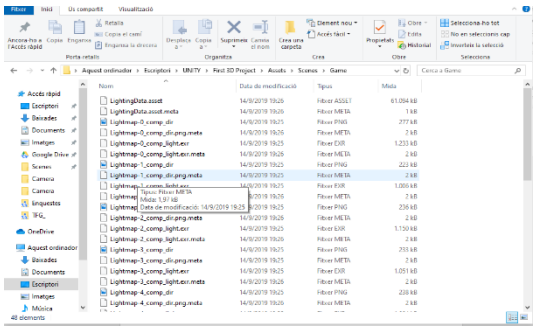


Figura 5.4: Explorador de Windows

Figura 5.5: Ventana “Project” de Unity (Fuente: Figura 5.1)

Para terminar, tenemos la ventana de “Hierarchy” o jerarquía en castellano. Esta ventana nos enseña los “objetos físicos” que tenemos en una escena. Estos pueden tienen unos parámetros y características que se pueden modificar mediante la ventana “Inspector”. Asimismo, esta ventana sirve para ordenar esos objetos y marcar un orden en su situación e importancia respecto a los otros.

Como ya se ha comentado anteriormente, estas ventanas pueden tener múltiples disposiciones y maneras de ser ordenadas para adaptarse a las necesidades de cada proyecto o persona.

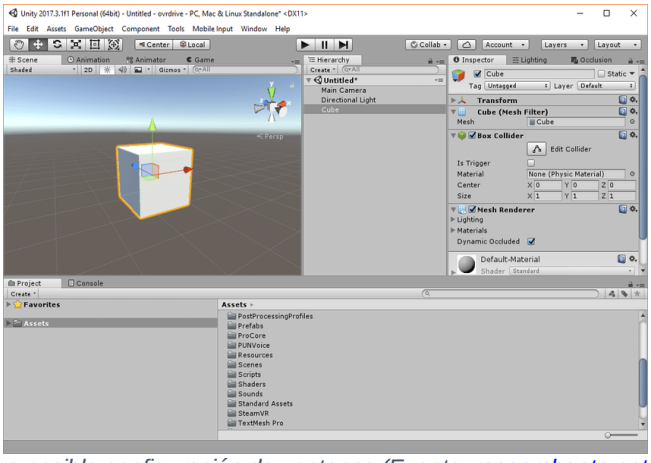


Figura 5.6: Ejemplo de otra posible configuración de ventanas (Fuente: researchgate.net/figure/The-User-interface-of-Unity-3D-game-engine_fig1_326434680)

5.1.2 Requisitos mínimos de uso

Para poder usar y desarrollar con esta plataforma es necesario obtener un ordenador o dispositivo con las siguientes características dependiendo de qué sistema se use (Tabla 5-1).

SISTEMA		REQUISITOS MÍNIMOS
Sistema operativo		**Windows:** 7 SP1+, 8, 10, 64-bit versiones **macOS:** 10.12+ **Linux:** Fijado a: Ubuntu 16.04, 18.04 y CentOS 7 Versiones en servidores de Windows y OS X están sin probar.
CPU		Capacidad para soportar instrucciones SSE2
GPU		Tarjeta gráfica con capacidades DX10 (shader model 4.0)
DISPOSITIVOS	iOS	Ordenador Mac con macOS 10.12.6 y Xcode 9.4 o mayor.
	Android	Android SDK y Java Development Kit (JDK). IL2CPP scripting "backend" requiere Android NDK.
	Windows	Windows 10 (64-bit), Visual Studio 2015 con el componente "C++ Tools" o mayor y Windows 10 SDK.

Tabla 5-1: Requerimientos mínimos para la funcionalidad de Unity como a desarrollador [9].

También es necesario tener unos requisitos mínimos para poder cargar la aplicación en el dispositivo desde el que se quiera usar (ver la Tabla 5-2).

SISTEMA		REQUISITOS MÍNIMOS
ESCRITORIO	Sistema operativo	**Windows:** 7 SP1+ **macOS:** 10.12+ **Linux:** Ubuntu 16.04+
	CPU	Capacidad para soportar instrucciones SSE2
	GPU	Tarjeta gráfica con capacidades DX10 (shader model 4.0)
iOS	iOS 9.0+	
Android	OS 4.1+ ARMv7 CPU con soporte NEON o Atom CPU OpenGL ES 2.0+	
WebGL	Cualquier versión actualizada de Firefox, Chrome, Edge o Safari.	
Windows	Windows 10 con tarjeta gráfica con capacidades DX10 (shader model 4.0)	

Tabla 5-2: Requerimientos mínimos para la funcionalidad de Unity como a aplicación [9].

5.2 Kinect

La herramienta Kinect es un controlador de juego libre, lo que significa que no hay necesidad de estar sujetando un mando u otro sistema de control a pesar de tu propio cuerpo. Este dispositivo fue creado por el estadounidense Alex Kipman y desarrollado por Microsoft para la videoconsola Xbox 360 y, más adelante, para ordenadores con sistema operativo Windows 8 [10]. La Kinect permite controlar e interactuar con la consola mediante una interfaz que reconoce a la persona que tiene a la vista y hasta tiene reconocimiento de voz.

Aun siendo una gran pieza tecnológica, los usuarios no la usaban y en el 2017 se descontinuó su producción por falta de popularidad.

5.2.1 Especificaciones técnicas

La Kinect está formada por cuatro componentes, los cuales permiten su buen funcionamiento. Una cámara RGB, un sensor de profundidad, un emisor de infrarrojos y varios micrófonos como se puede ver en la Figura 5.7.

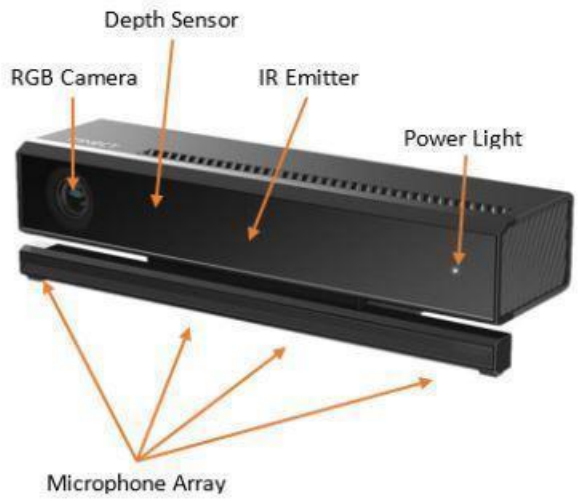


Figura 5.7: Partes que componen la Kinect (Fuente: projectabstracts.com/21631/workplace-posture-assessment-and-biofeedback-with-kinect-project.html)

- **Cámara RGB:** Una cámara RGB (Red Green Blue) es un dispositivo que capta los tres colores básicos para poder detectar y formar una imagen en dos dimensiones de lo que está captando.

- **Emisor de infrarrojos:** La Kinect detecta la profundidad de los objetos que la rodean mediante un emisor de rayos infrarrojos que son captados por el sensor de profundidad y dependiendo del tiempo que tarden, se calcula la profundidad a la que están tales objetos (Figura 5.8).

- **Sensor de profundidad:** Este sensor capta los rayos infrarrojos y determina la profundidad a la que están los "obstáculos" mediante el sistema "Time of Flight" [11]. Así pues, es capaz de crear una imagen 3D de los alrededores en el que se encuentra.

- **Grupo de micrófonos:** No solo puede crear una imagen 2D y 3D sino también puede determinar de dónde viene el ruido que captan los micrófonos ya que tiene varios para saber desde qué posición llega el sonido.

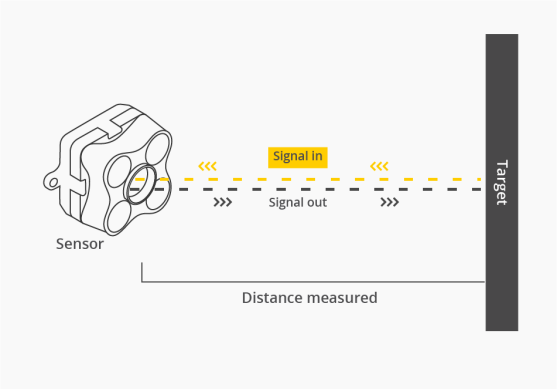


Figura 5.8: Principio "Time of Flight" (Fuente: https://www.terabee.com/time-of-flight-principle/)

También cabe comentar que la Kinect con la que se realiza esta aplicación es la Kinect V2, la cual fue diseñada especialmente para los "developers" (término que se utiliza para llamar a los desarrolladores en inglés) y contiene ciertas mejoras respecto a la otra versión (véase en la Tabla 5-3).

Kinect XBOX	Kinect V2
Campo de visión de 57º en horizontal y 43º en vertical	Campo de visión de 70º en horizontal y 60º en vertical
Resolución 640 x 480	Resolución 1920 x 1080
Rango de profundidad: 0,5 a 4,5m	Rango de profundidad: 0,5 a 4,5m
	Permite calcular la fuerza muscular y medir el ritmo cardíaco

Tabla 5-3: Diferencias entre la Kinect XBOX y la Kinect V2 [12].

5.2.2 Kinect para Windows SDK 2.0

El Kit de Desarrollo de Software (SDK) 2.0 de Kinect para Windows permite a los desarrolladores a crear aplicaciones que permiten el reconocimiento de gestos y voz usando el sensor Kinect y ordenadores con sistema operativo Windows. Estas aplicaciones se pueden programar gracias a las librerías que provienen de este SDK (ver Anexo B).

Este SDK es totalmente gratuito y se puede conseguir en su página web [14]. Este ejecutable te muestra lo que el sensor Kinect ve en todo momento por su cámara RGB y su sensor de infrarrojos gracias a la aplicación Kinect Studio (Figura 5.9 y Figura 5.10)

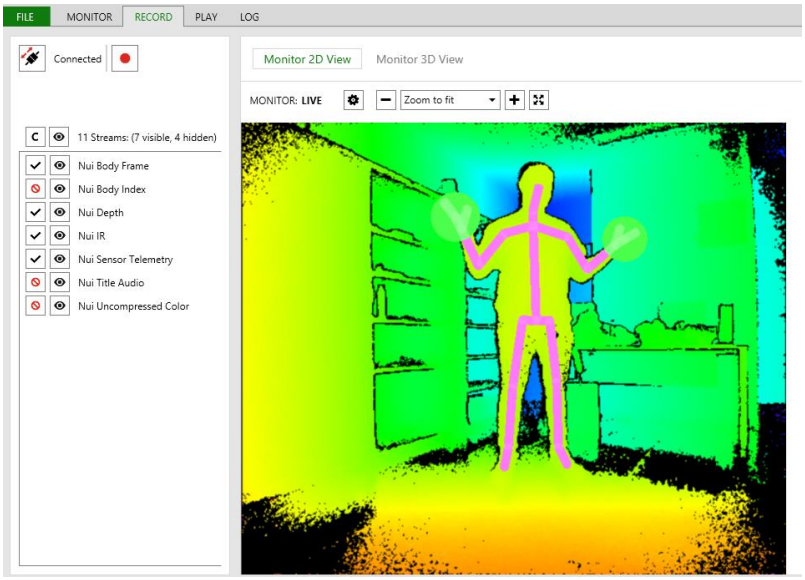


Figura 5.9: Kinect Studio representando lo que captura la cámara RGB.

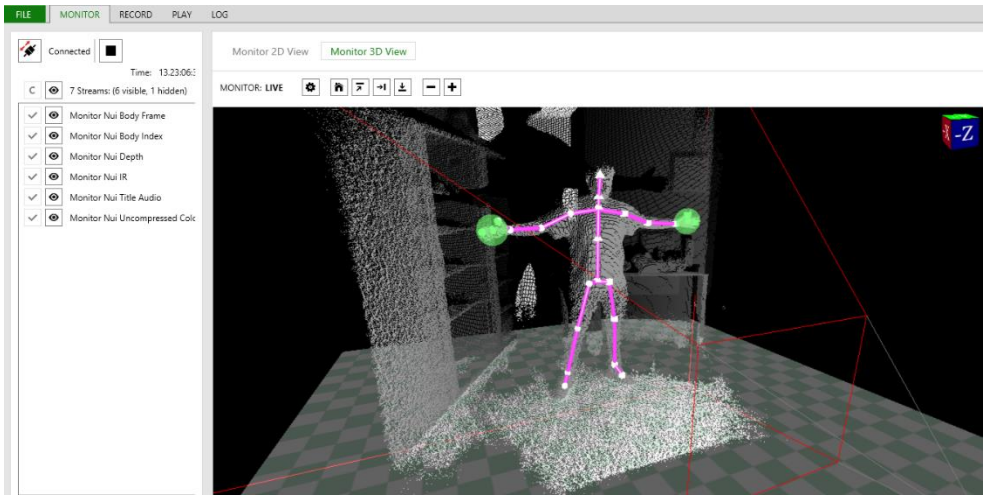


Figura 5.10: Kinect Studio representando lo que captura el sensor de profundidad con los IR.

El dispositivo Kinect detecta “joints” (juntas en castellano) con la cámara RGB y a partir de ellos traza un enlace entre los *joints* para que simule el cuerpo humano [15]. Cada *joint* tiene 11 propiedades: color (x,y), profundidad (x,y), cámara (x,y,z) y orientación (x,y,z,w). Todas estas propiedades se calculan y se tienen en cuanta para mostrar la posición de ese punto desde que se enciende el sensor hasta que se apaga o se desconecta.

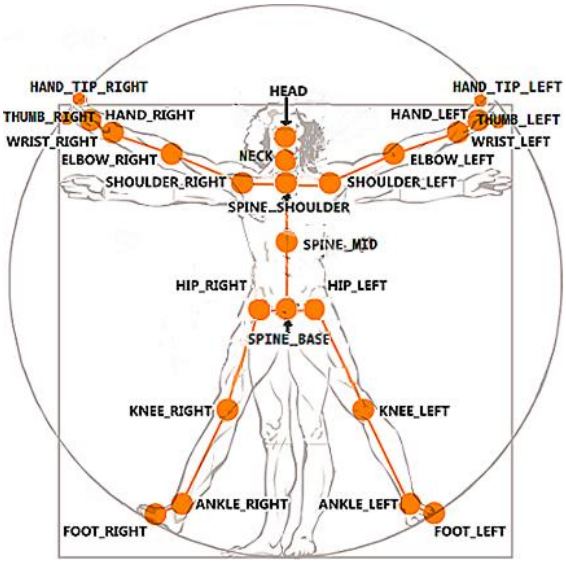


Figura 5.11: Sistema de "joints" que crea el software de la Kinect para simular un cuerpo humano [15].

5.2.3 Requisitos mínimos de uso

Para poder conectar la Kinect con Windows es necesario poseer un seguido de requisitos [13]. Todos ellos son bastante fáciles de cumplir y en la actualidad muchos de los ordenadores ya tienen todos estos requisitos (tanto el sistema operativo, hardware y software).

- **Sistema operativo**: Es necesario que el ordenador tenga Windows 8 o superior como sistema operativo.

- **Hardware del sistema**: Procesador de 64 bits (x64), procesador de doble núcleo 3.2 GHz o más, bus USB 3.0 y 2 GB de memoria RAM.

- **Software del sistema**: Microsoft Visual Studio 2013 o superior, .NET Framework 4.0 y Microsoft Speech Platform SDK v11.

5.3 HTC Vive

HTC Vive son unas gafas de RV fabricadas por las empresas HTC i Valve el año 2016 (ver Figura 3.4). Este dispositivo permite al usuario sumergirse en un mundo virtual y poder andar libremente por la habitación o recinto en donde se encuentre. Este dispositivo permite moverse por un espacio abierto gracias a su tecnología de "sensor-base tracking" (seguimiento de sensores base), ya que estos detectan la posición y orientación de las gafas en todo momento para recalibrar y controlar el espacio virtual que ve el usuario.

5.3.1 Especificaciones técnicas HTC Vive

Esta pieza de tecnología tiene muchos avances tecnológicos para poder crear un mundo virtual cómodo y lo más parecido a la realidad posible en términos de vista, oídos y hasta interacción. Por ello, tiene muchas especificaciones en varios campos como se puede ver a continuación.

Especificaciones de las gafas de RV	
Pantalla	Dual AMOLED 3.6'' diagonal
Resolución	1080 x 1200 píxeles por pantalla (2160 x 1200 píxeles combinado)
Ratio de actualización	90 Hz
Campo de visión	110 grados
Componentes de seguridad	Fronteras de campo de juego
Sensores	SteamVR Tracking, G-sensor, giroscopio y proximidad
Conexiones	HDMI, USB 2.0, stereo 3.5 mm auriculares, Power y Bluetooth
Inputs	Micrófono integrado
Vista	Distancia a la pupila y distancia de lentes ajustable
Especificaciones de los controles	
Sensores	SteamVR Tracking
Inputs	Multifunction trackpad, Grip buttons, dual-stage trigger, System button, Menu button
Uso de carga	Aproximadamente 6 horas
Conexiones	Micro-USB puerto de carga

Tabla 5-4: Especificaciones HTC Vive [16].

Figura 5.12: Hardware que compone las gafas HTC Vive (Fuente: http://i.imgur.com/hCCSaDB.jpg)

5.3.2 HTC Vive, SteamVR y Unity3D

Todas las gafas de RV (hardware) precisa de un software que permita la lectura y posterior funcionalidad de éstas. En el caso de HTC Vive funciona con SteamVR, la cual es una plataforma de distribución digital de videojuegos desarrollado por Valve (cofabricador de la HTC Vive). Este servicio provee aplicaciones para las gafas de RV (sean o no HTC).

Figura 5.13: Sistema de reconocimiento de hardware de SteamVR

Figura 5.14: UI que proyecta SteamVR (Fuente: https://holographica.space/news/valve-steamvr-update-4978)

Unity VR es una extensión (integrada en el "Build Settings") de Unity que te permite enlazar dispositivos de RV sin la necesidad de *plug-ins* externos o adicionales [17].

Para habilitar el uso de RV en Unity se debe ir a “Project Settings”, dentro de la pestaña “Edit”. Una vez dentro de “Project Settings” buscar la opción de “Player”. Allí, buscar “XR Settings” para habilitar el soporte de la RV tal y como muestra la Figura 5.15 y Figura 5.16.

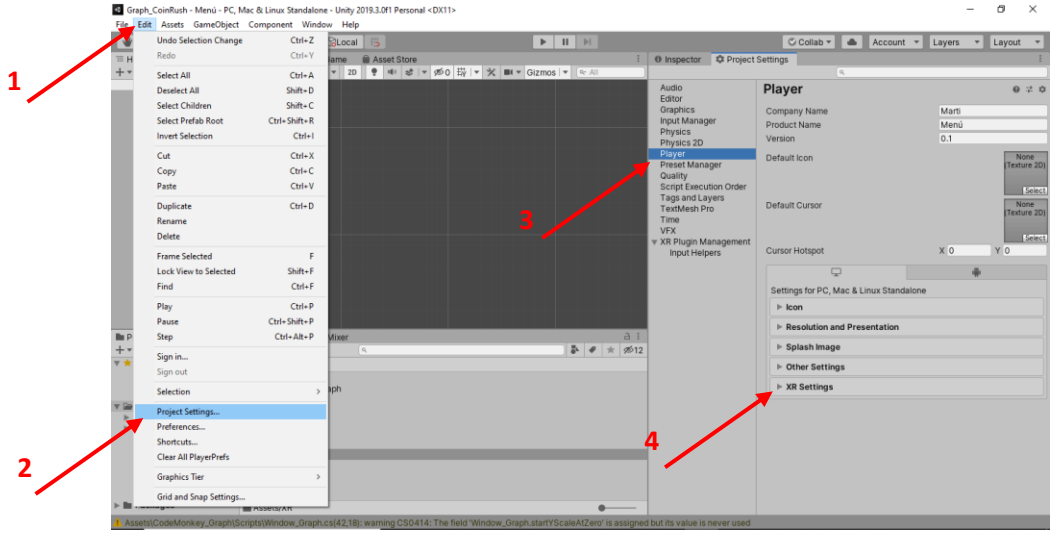


Figura 5.15: Pasos a seguir para la habilitación de la RV.

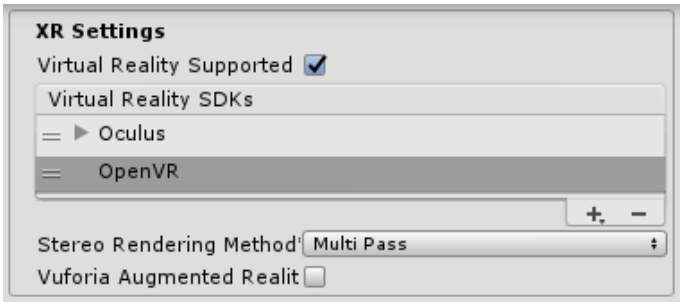


Figura 5.16: Ventana para habilitar el uso de RV en Unity [17].

5.3.3 Requisitos mínimos de uso

Este dispositivo de RV tiene algunos requisitos por parte del hardware del ordenador utilizado ya que es una maquinaria un poco más pesada de lo habitual (véase en la Tabla 5-5).

Tarjeta gráfica	NVIDIA® GeForce® GTX 1060 o AMD Radeon™ RX 480, equivalente o mejor.
Procesador	Intel® Core™ i5-4590 o AMD FX™ 8350, equivalente o mejor.
Memoria RAM	4 GB o más.
Output Video	HDMI 1.4, DisplayPort 1.2 o mejor.
Puerto USB	USB 2.0 o mejor.

Sistema operativo	Windows® 7 SP1, Windows® 8.1 o Windows® 10

Tabla 5-5: Requerimientos para el uso de las gafas HTC Vive [16]

Por otro lado, es necesario que la habitación o el espacio en donde se lleve a cabo la actividad sea de unas dimensiones de entre 1.98 x 1.5 metros y 3,5 x 3,5 metros para que los sensores base puedan captar bien las gafas de RV.

5.4 XAMPP

Esta herramienta es un paquete de software libre. Su función principal es la gestión de bases de datos "*MySQL"*, el servidor web "*Apache"* y los programas informáticos (también llamados *intérpretes*) para lenguajes de script "*PHP"* y "*Perl"* [33].

Esta herramienta fue utilizada por V. Rofes Pujol y E. Medina Ourselin en su proyecto [3] con el fin de enlazar la aplicación con una BD (desarrollada por ellos) y poder registrar y/o acceder como paciente/fisioterapeuta y guardar los valores máximos, ejercicio, fecha y hora de las sesiones de rehabilitación. C. Vega Pinzón y O. Mayorgas también utilizaron esta BD para el registro, acceder como paciente/fisioterapeuta y almacenar los datos de sus sesiones [2].

En este proyecto se usará la BD para registrar pacientes y fisioterapeutas mediante los scripts (PHP y C#) que enlazan las plataformas *phpMyAdmin* y *Unity3D* [3]. Estos scripts están enlazados entre sí mediante el lenguaje de programación C#.

El uso principal de este software libre en este proyecto es la gestión de la BD MySQL e interpretar los scripts con el lenguaje PHP para su posterior uso. Una vez se haya descargado e instalado el software XAMPP se debe empezar ("*Start*") el servicio "*Apache*" y "*MySQL*". Al iniciar el "*Admin*" (Administrador) del "*MySQL*" se abrirá una página web llamada "*phpMyAdmin*" (ver Figura 5.17).

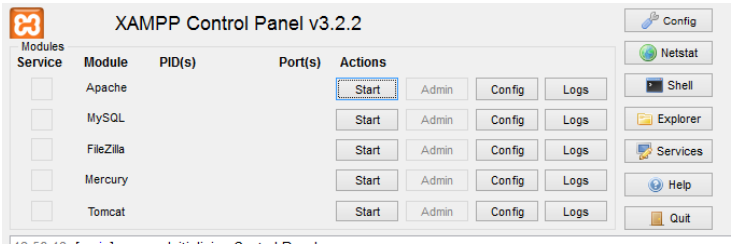


Figura 5.17: Panel de control del software XAMPP.

5.4.1 phpMyAdmin

Este software gratuito está desarrollado con el lenguaje PHP y controla la administración de la BD MySQL en la red. Este software es capaz de soportar un rango muy amplio de operaciones tanto en MySQL como en MariaDB [34]. Las operaciones más usadas de este software son el manejo de las BD (tablas, relaciones entre ellas, índices y permisos). Todas estas operaciones (y más) pueden usarse mediante la UI del programa.

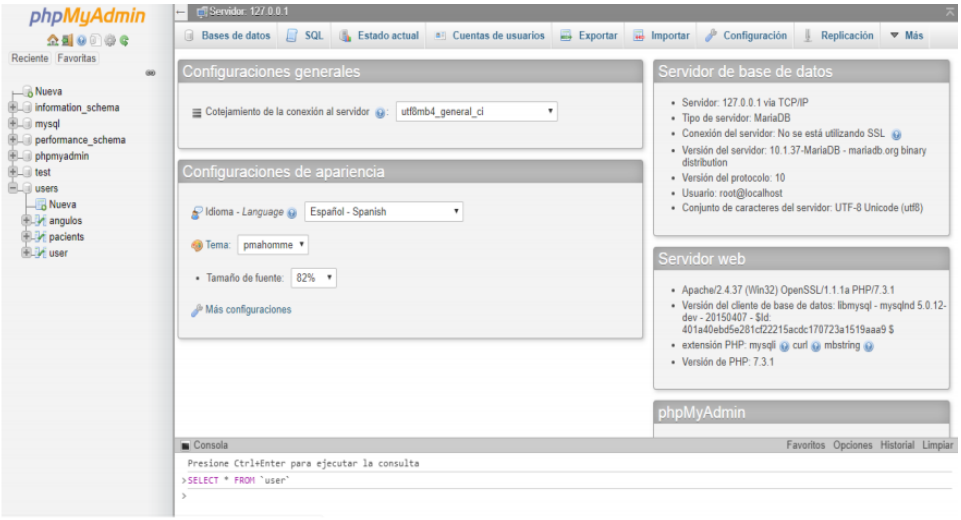


Figura 5.18: Interfaz de phpMyAdmin [3].

En esta interfaz (Figura 5.18) se pueden observar los datos almacenados, crear más BD y enlazarlas con las otras si es necesario.

5.4.2 MySQL

Este software es un sistema de gestión de BD relacionales de código abierto [36]. Hoy en día, MySQL es la BD más popular en todo el mundo por su rendimiento, confiabilidad y su fácil uso [35] [36].

Para poder hacer consultas, recuperar y administrar información se debe escribir en el lenguaje SQL ya que es el lenguaje declarativo que permite hacer todas estas operaciones. Es decir, MySQL es el sistema que gestiona la BD, pero para poder acceder y mandar órdenes se debe indicar usando su lenguaje (SQL) [3].

5.4.3 Requisitos mínimos de uso

Para el uso de XAMPP, phpMyAdmin o MySQL no es necesario ningún hardware o software complicado. Con tal de tener un ordenador que funcione con Windows, Linux o Mac y poseer un mínimo de 256 MB de memoria RAM y 85MB de almacenaje, el programa ya funciona. Los requerimientos del software básicamente abarcan los programas/software que se usará (ver Tabla 5-6).

REQUISITOS DEL SOFTWARE	
MySQL	5.0.41
PHP	5.2.2
phpMyAdmin	2.10.1

Tabla 5-6: Requerimientos para el uso del software XAMPP [37].

5.5 Mapa de relaciones entre los materiales utilizados

Figura 5.19: Mapa de relaciones entre los materiales utilizados para el desarrollo de la aplicación (Fuentes: [34] / https://upload.wikimedia.org/wikipedia/commons/0/03/Xampp_logo.svg/https://commons.wikimedia.org/wiki/File:Unity_Technologies_logo.svg/http://deeprealities.ca/kinect-for-windows-hackathon//http://www.unioncosmos.com/htc-vive-pro-realidad-virtual/)

Mientras todos los programas reciben y envían datos de unos a otros, la Kinect tan solo envía los datos de posición y profundidad que detecta sin recibir ningún tipo de dato a cambio.

6. Diseño de la aplicación

En este apartado se expondrá el paso a paso que realizamos para el diseño de la aplicación, mostrando las diferentes escenas creadas para el diseño de esta, junto con su respectiva descripción.

6.1 Metodología del diseño

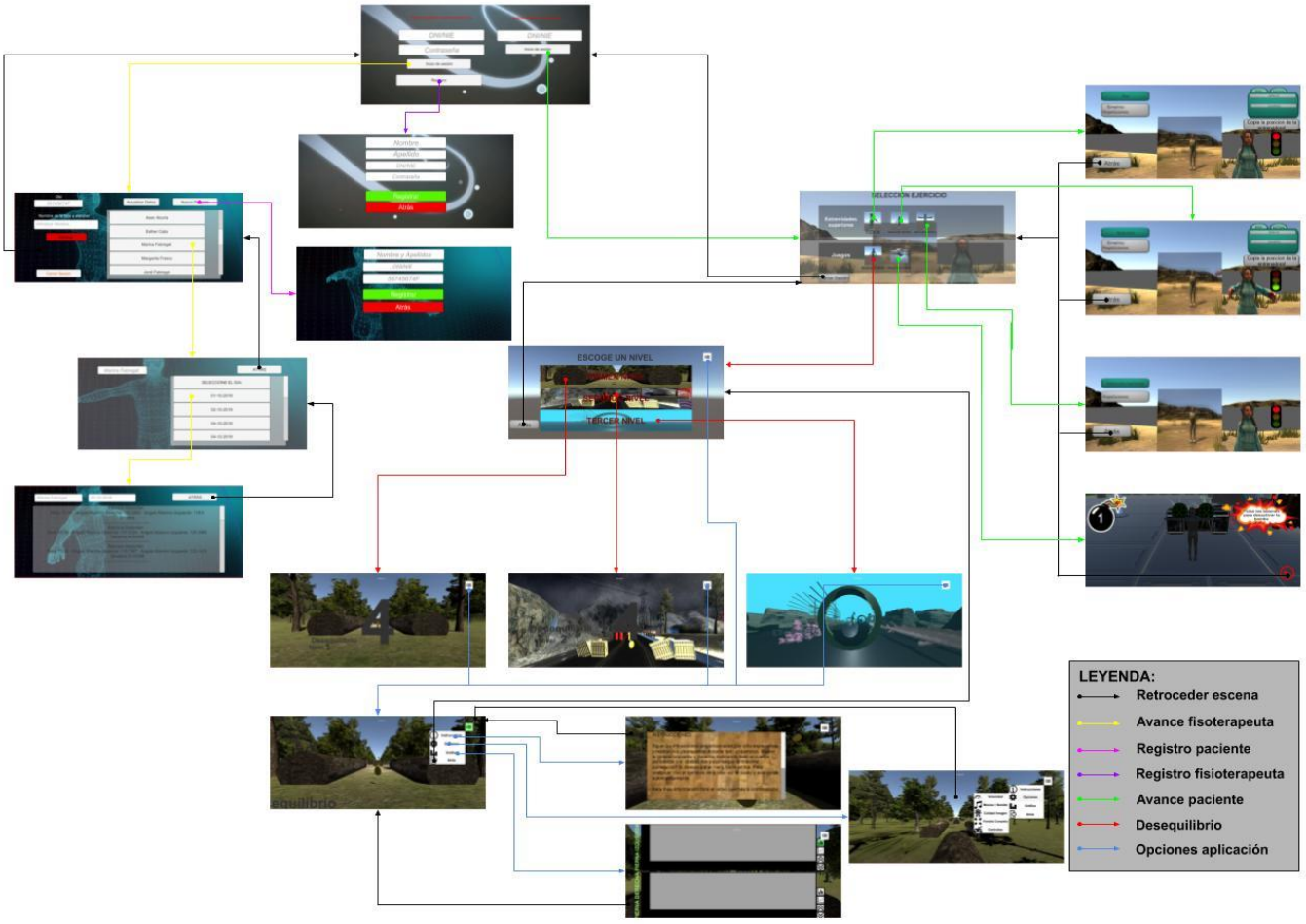


Figura 6.1: Diagrama visual de las funciones de la UI (User Interface).

Figura 6.2: Diagrama de flujo del software desarrollado

6.2 Diseño de la interfaz gráfica

Unity 3D permite al desarrollador diseñar UI ("User Interface" o interfaz de usuario en castellano) para poder crear menús, letras o créditos, botones, controles, etcétera. Para esta aplicación se ha utilizado esta herramienta (localizada en el Toolbar). Permite crear todo tipo de objetos interactivos como se muestra en la Figura 6.3. Estos objetos permiten crear una aplicación personalizable con varias acciones y permiten al usuario a escoger y tener un abanico más grande de oportunidades.

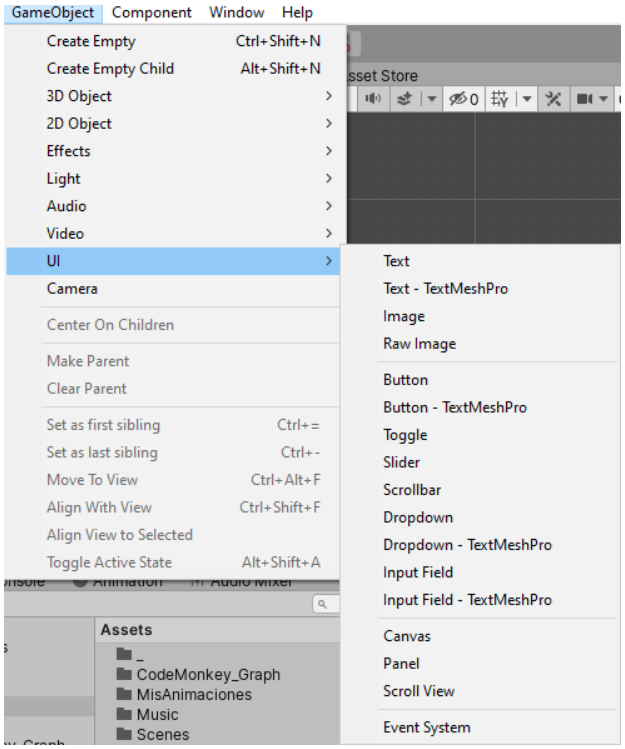


Figura 6.3: Herramientas para la creación de una interfaz del usuario.

Para este proyecto es muy importante la aplicación que tiene la interfaz gráfica ya que la rehabilitación se tiene que adaptar a una cantidad muy grande de personas diferentes, con varias edades, cualidades y diferentes dificultades. Es por eso que a través de la interfaz se podrá personalizar el ejercicio para que sea útil a la mayor cantidad de pacientes.

6.2.1 Diseño menú inicial "Escena Movement" (Selección del ejercicio)

Esta escena fue creada por V. Rofes y E. Medina como parte de su TFG [3]. Esta escena tiene 2 partes. La primera se trata de escoger la actividad que se quiere realizar (véase la Figura 6.4). La segunda parte sucede al seleccionar el ejercicio, la pantalla UI desaparece y procede a realizar el ejercicio en el entorno virtual creado (para conocer más sobre el proyecto y el entorno anterior recomendamos leerse los trabajos previos a este [1], [2] y [3]).

Figura 6.4: Escena Movement (Selección ejercicio)

Para introducir el botón (marcado en rojo en la Figura 6.4) en el menú, se tuvo que utilizar la herramienta UI y cambiar las propiedades mediante la ventana "Inspector". Las características que tienen estos botones es que te permiten ver una pequeña animación del ejercicio (mediante un objeto que representa una fisioterapeuta. Para hacer tal animación, es necesario utilizar las herramientas "*Animator*" y "*Animation*".

Figura 6.5: Animación hecha para la representación de la actividad

Una vez se obtuvo la animación, se introdujo un parámetro (mediante la ventana "Inspector") que pudiera reproducir un video en el momento de empezar la aplicación (mediante el botón de "Play").

Este botón se ubicó en un separador nuevo al que se le llamó "Juegos" para distinguirlo de los otros ejercicios de rehabilitación [3].

6.2.2 Diseño menú del juego “Escoge un Nivel”

Este menú se creó para escoger el nivel de dificultad y/o ambiente de la sesión de rehabilitación. Para entrar en estos ambientes (mencionados en el punto 6.3 de este TFG), se diseñó un menú donde se pudiera ver una imagen del ambiente y un título que sugiere el nivel de dificultad según el ambiente.

Figura 6.6: Menú “Escoge un nivel”

Este menú constituye muchos “*GameObjects*” (así es como se llama cada objeto, sea imagen, texto u objeto físico en la plataforma Unity) de carácter UI. Al ser un menú, está hecho meramente de botones interactivos y textos que nos indican cuál es su función (ver Figura 6.6).

6.2.3 Diseño de una aplicación personalizable

Uno de los grandes objetivos de esta aplicación es que los pacientes puedan recuperarse físicamente. Aun así, cada paciente es una persona muy distinta a la anterior. Es por eso que se ha decidido hacer un menú (usando UI) en el que se pueda cambiar varios parámetros del juego, así como leer las instrucciones de uso, observar en directo los datos que recoge la Kinect y salir del juego.

Figura 6.7: Menú del juego

- **Instrucciones de uso:**

Este apartado de la lista sirve para entender cómo se debe utilizar el juego y como se debe hacer para realizar bien el ejercicio. Obviamente, estas instrucciones deben ser cumplimentadas por un experto fisioterapeuta.

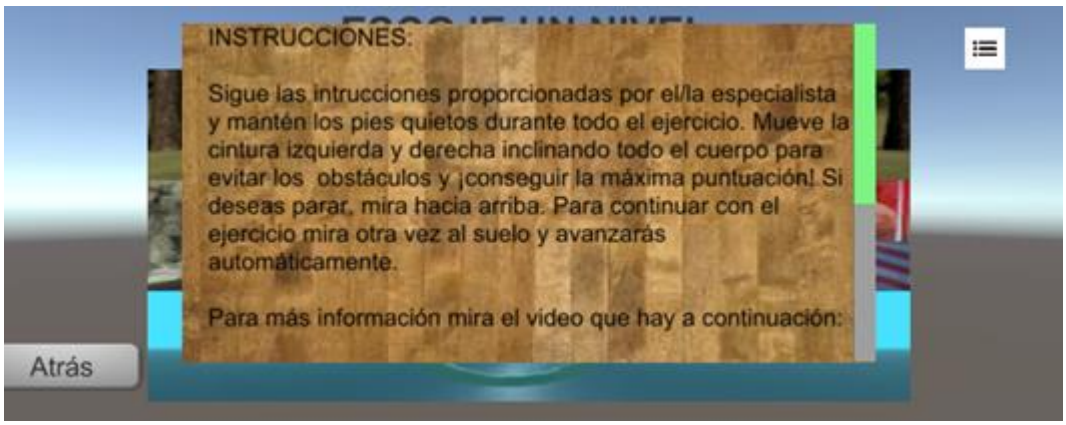


Figura 6.8: Menú del juego

En la Figura 6.8 se puede observar la ventana de “INSTRUCCIONES”. Esta ventana está diseñada con la herramienta llamada “Scrollbar” (procedente de la UI) para mostrar un texto más extenso que la pantalla.

- **Opciones:**

El menú de opciones permite al usuario/paciente/fisioterapeuta cambiar una gran cantidad de parámetros que influyen en el juego, por ejemplo: Velocidad, Volumen de los sonidos y la música, calidad de la pantalla, la posibilidad de cambiar a pantalla completa y cambiar los controles del juego. Véase en la Figura 6.9.

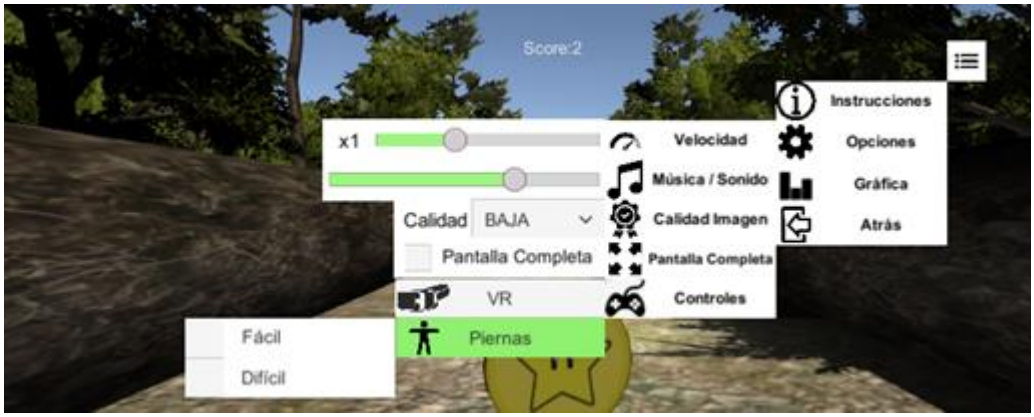


Figura 6.9: Opciones personalizables según el paciente/fisioterapeuta

Para poder crear esta comunicación entre el usuario y la aplicación se ha usado varios elementos de la UI.

- ***Slider* o deslizante:** Este mecanismo permite variar el valor o magnitud de una variable dependiendo de la distancia en la que esté. Como se puede ver en la Figura 6.9, la opción de "Velocidad" y "Música / Sonido" están reguladas por este sistema de control.

- ***Dropdown* o desplegable:** Esta herramienta deja escoger entre varias opciones predeterminadas y con características diferentes, por ejemplo: la Figura 6.9, muestra como la opción "Calidad Imagen" tiene un desplegable, éste con tres opciones distintas ligadas a distintas calidades de imagen (gráficos para ser exactos).

- ***Toggle* o marcador:** Este booleano se utiliza para marcar si un enunciado o situación es verdadera o falsa (sistema booleano). Este mecanismo es utilizado en esta aplicación tanto por "Pantalla Completa" como para determinar la función que dictará la velocidad de desplazamiento en el control "Piernas". (Ver Figura 6.9).

    - La opción "Fácil" llama a una función constante de desplazamiento igual a:

$$Desplazamiento = desplazamiento \pm 0{,}05$$

    - La opción "Difícil" trata con una función de carácter exponencial sin llegar a grandes magnitudes:

$$Desplazamiento = desplazamiento \pm \frac{0{,}05}{\acute{a}ngulo\ pierna^2}$$

- ***Button* o botón:** Esta herramienta es la base de todos los menús interactivos ya que permite llamar funciones (procedentes de Scrpits en C#) que actúan según hayan sido desarrolladas. Todas las imágenes que se pueden ver en la Figura 6.9 y no tienen ningún mecanismo nombrado anteriormente son botones que tienen como función desplegar uno de esos mecanismos u otras ventanas como "Instrucciones" y/o "Gráfica" o salir del juego usando el botón con el texto "Atrás".

- **Gráfica:**

Esta opción crea unos objetos (*GameObjects*) que, a partir de un Script (ver en Anexo B), se sitúan relativamente entre ellos para formar dos gráficas (una por cada pierna) que muestran, a tipo real, los ángulos que crean las piernas respecto a un plano vertical.

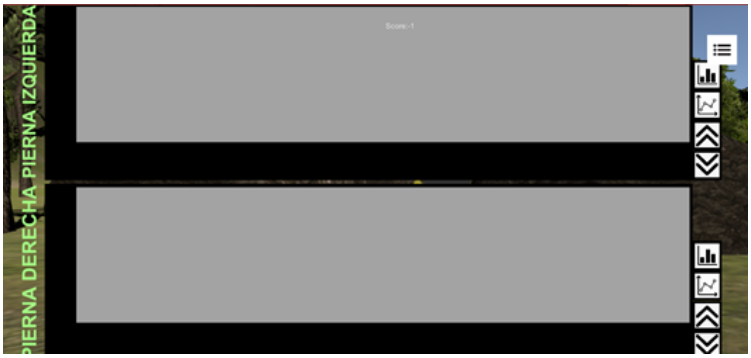


Figura 6.10: Gráficas del menú de juego

Estas gráficas tienen varias opciones como el número de muestras que enseña en la ventana y el tipo de gráfico que representa (ya sea de barras o de puntos de dispersión).

Observar la Figura 6.10 para ver la distribución de las gráficas.

Los objetos que crean la gráfica son meras imágenes situadas de tal forma que representan el *background* (fondo) de los datos que se muestran. Los datos también son representados por imágenes; en el caso de un gráfico de barras se representan con un cuadrado y, en el caso de ser un gráfico de puntos de dispersión se representan con círculos. Por otro lado, las propiedades de estos valores, como su tamaño, posición, color, etcétera; son determinados por un *Script* (llamado "*Window_Graph"*, ver en Anexo B) que los modifica dependiendo de los valores que tiene la lista de ángulos sacados de la Kinect.

- **Atrás:**

Por último, este botón permite al usuario volver atrás al menú "Escoge un Nivel" (Figura 6.4). Sea por error o porque el juego terminó, para salir del ambiente virtual se debe presionar este botón.

6.3 Creación del ambiente

Para este apartado partimos de la idea de que se quería que el ejercicio de rehabilitación pudiera ser realizado de una manera interactiva y didáctica en un ambiente inmersivo para el paciente, mostrando de forma eficaz los resultados ya sea para que el paciente vaya notando su progreso como para facilitar el seguimiento que le realiza el fisioterapeuta a sus pacientes.

Como se verá a continuación se realizaron 3 ambientes diferentes, no solo para ampliar las opciones del lugar de realización del ejercicio a preferencias del paciente, sino también para adaptar la dificultad de cada uno de estos dependiendo del grado de discapacidad del paciente.

El objetivo a cumplir en cada uno de los niveles es recolectar la mayor cantidad de monedas mientras son esquivados los obstáculos que se presentan en cada nivel, debido a que la idea en concreto es rehabilitar el equilibrio del paciente los obstáculos solo se podrán evitar al realizar movimientos de desequilibrio tanto a la derecha como a la izquierda.

Una vez seleccionado el nivel en el cual el paciente desea realizar la terapia, en la pantalla aparecerá un conteo regresivo, en donde el paciente se preparará para iniciar el juego una vez el contador llegue a cero.

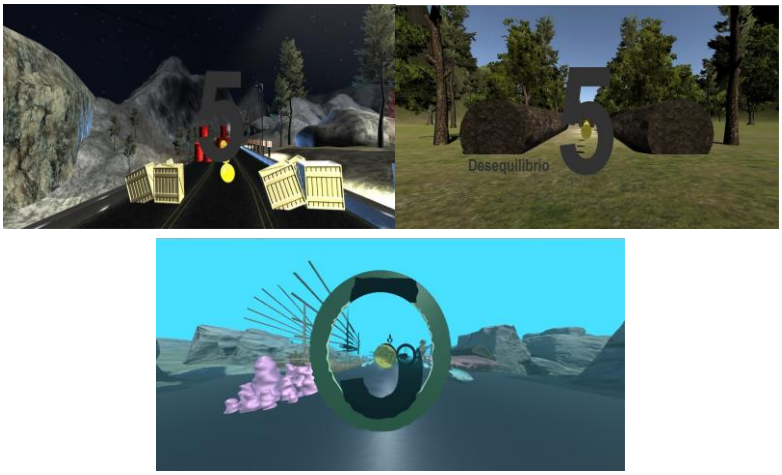


Figura 6. 11. conteo regresivo para dar inicio al juego/terapia

Una vez iniciado el juego el paciente podrá avanzar en la escena sin la necesidad de moverse más que inclinándose hacia la derecha e izquierda, para seguir el camino mientras recolecta las monedas doradas.

A la hora de desarrollar el ambiente se pensó en la implementación de diversas características, para así brindar una experiencia completamente inmersiva en todos los niveles, estas son:

- Sonido para la ambientación del nivel
- Marcador de monedas recogidas
- Control de velocidad
- Cambio en el sistema de control

6.3.1 Primer Nivel escena "PRIMERNIVELJCG"

En esta escena encontramos el primer nivel, donde el paciente al ponerse las gafas de realidad virtual entra a un espacio ambientado como un bosque donde podrá interactuar con el ambiente, y cumplir el objetivo que es llegar a la línea de meta con el mayor número de monedas recolectadas esquivando los obstáculos que aparezcan en el camino.

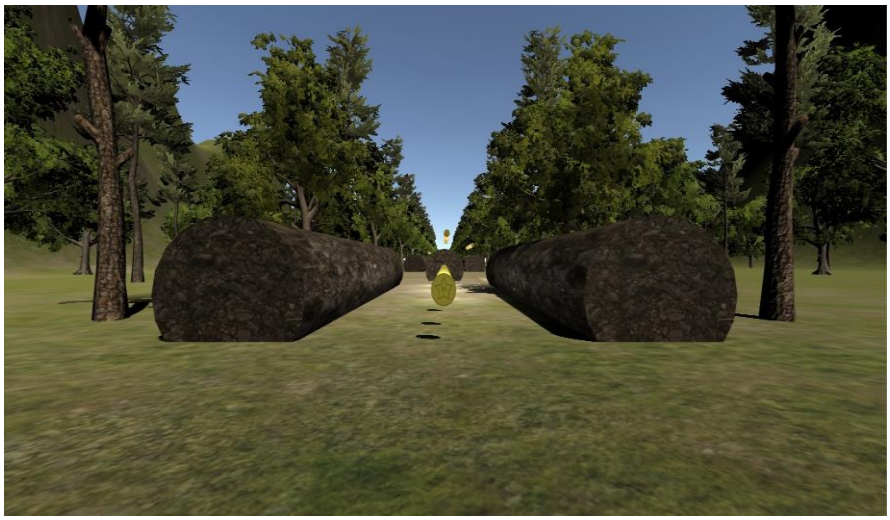


Figura 6.12. inicio primer nivel ejercicio de desequilibrio

Este nivel se ambientó en un bosque puesto que se quería asemejar una caminata al aire libre, donde el paciente mientras juega podrá disfrutar de ver y escuchar de la fauna y flora creada.

Como fue mencionado anteriormente cada nivel cuenta con diferentes grados de dificultad predeterminados, este al ser el primer nivel se caracteriza por ser el menos difícil, por cumplir con factores como lo son la velocidad, ya que en este nivel esta es mínima y los sistemas de control son sensibles a una inclinación menor, es por esto que no es necesario generar una gran inclinación para cambiar de carril.

6.3.2 Segundo nivel escena “SEGUNDONIVELJCG”

En esta escena encontramos el segundo nivel donde el paciente al ponerse las gafas de realidad virtual entra a un espacio ambientado como una pista de carros en la noche, donde podrá interactuar con el ambiente y cumplirá el mismo objetivo el cual es llegar a la línea de meta con el mayor número de monedas recolectadas.

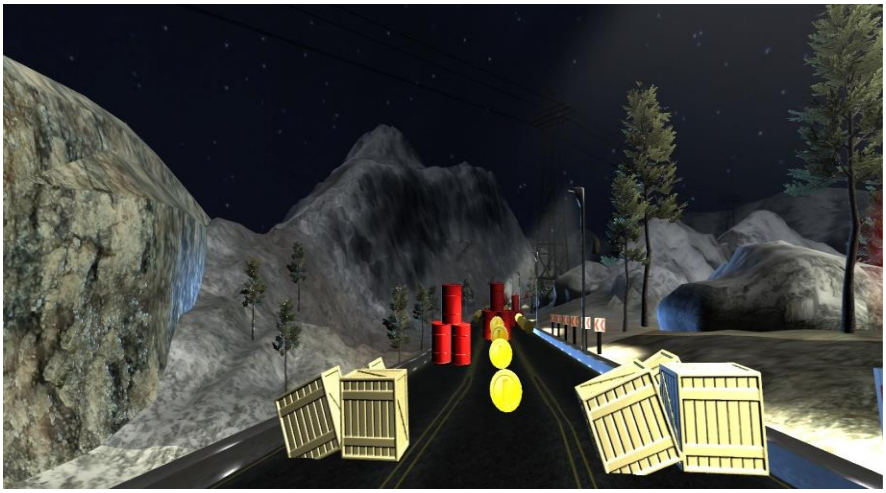


Figura 6.13. Inicio segundo nivel ejercicio de desequilibrio

En este nivel encontraremos un grado de dificultad predeterminado más alto referente a: sistemas de control menos sensibles a ángulos pequeños lo que conlleva a que la inclinación hacia los lados tenga que ser mayor para esquivar los obstáculos, además en este nivel existen una mayor cantidad de obstáculos en comparación con el primer nivel y es por esto que la interacción con el ambiente es mucho mayor, al igual que en el nivel anterior este nivel también cuenta con sonidos y efectos que hacen que la experiencia sea más real.

6.3.3 Tercer nivel escena “TERCERNIVELJCG”

En esta escena encontramos el tercer nivel donde el paciente al ponerse las gafas de realidad virtual entra a un espacio ambientado en una experiencia de buceo en la que podrá interactuar con el ambiente y cumplirá el mismo objetivo donde tendrá llegar a la línea de meta con el mayor número de monedas recolectadas mientras cruza la mayor cantidad de aros.

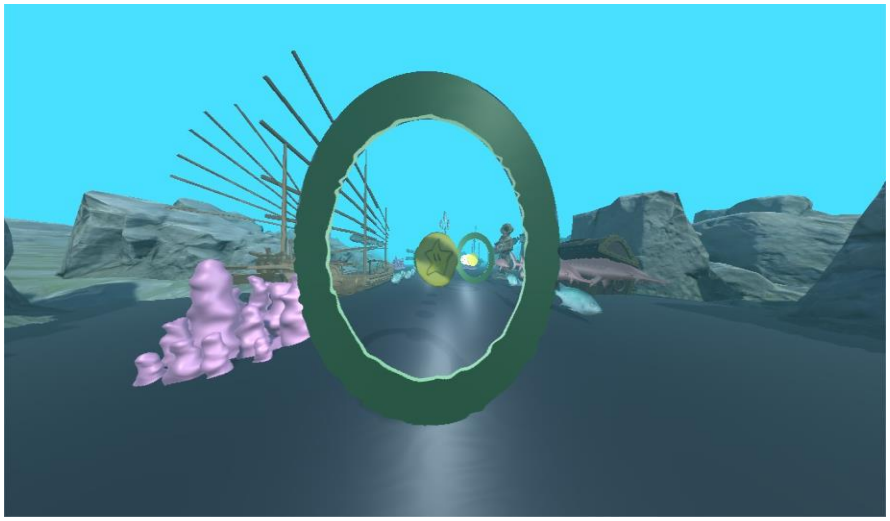


Figura 6.14. Inicio tercer nivel ejercicio de desequilibrio

Este nivel al ser ambientado bajo el mar, se realizó como una experiencia de buceo donde mientras se avanza a lo largo de la escena, no sólo puede coger monedas y pasar por en medio de los aros, sino que también se pueden ver peces nadando a tu lado, barcos hundidos y todo esto mientras escuchas y vives una experiencia muy real.

Al hablar del grado de dificultad predeterminado de este nivel, se puede decir que este, es el más difícil ya que el sistema de control es muy sensible al cambio y es por esto que se tendrán que realizar inclinaciones muy seguidas para colocarse en el carril correcto, otro de los factores que hace que este nivel sea más difícil es la velocidad ya que es el nivel en donde el paciente avanza más rápido a través de la escena.

6.3.4 Scripts utilizados para el diseño del ambiente

1. **Puntos**
    - **Recoger:** Por medio de este script le damos un valor a las monedas que se encuentran en el juego, además por medio de la función OnTriggerEnter

cuando el jugador pasa por la moneda, esta desaparece y genera un sonido para asemejar que el jugador la está recolectando.

- **Ll:** En este script se encuentra el contador de las monedas, además por medio de la función OnGui este contador o score aparece en la pantalla del juego.
- **RotarCoin:** Por medio de este código hicimos que la monada rotara en su eje Z, dándole un valor a la fuerza de rotación y multiplicando por Time.deltaTime.

2. Obstáculos

- **Destroybythecontact:** Por medio de este script desaparecen los objetos que toque el jugador en este caso los obstáculos, generando efectos y sonidos para que la experiencia sea más realista.

3. Player

- **VR Look:** Por medio de este script hacemos que el personaje camine dentro del juego sin que el jugador se mueva, este código nos permite definir velocidades y un ángulo para caminar o quedarse quieto dependiendo de hacia donde esté mirando el jugador, lo ideal es que el jugador esté mirando hacia el frente para jugar, pero si este quiere detenerse lo puede hacer en cualquier momento mirando hacia arriba.
- **Flym:** Por medio de este Script se controlan los movimientos laterales por camara. Se definen ángulos a alcanzar tanto en derecha como en izquierda, estos ángulos varían en cada nivel para aumentar la dificultad.
- **ControlPiernas:** Por medio de este script se controlan los movimientos laterales por medio del kinect.

4. Ambientación

- **FEfct:** Por medio de este código se crea un efecto en el foco de la cámara para dar la apariencia de estar bajo el agua.
- **UnderWaterEfect:** por medio de este código se crea un efecto de estar bajo el agua controlando factores como el ruido, la frecuencia y profundidad.

Estos codigos anteriores son complementados con script llamadas Shader en donde el programa realiza cálculos gráficos como el coloreado de píxeles o la transformación de vértices, entre otros.

6.4 Reconocimiento de las extremidades del cuerpo

Uno de los objetivos a alcanzar en este proyecto es el poder brindar al fisioterapeuta de una forma tangible y práctica datos asociados con las terapias de rehabilitación realizadas a sus pacientes, para que así, sea más fácil realizar el seguimiento y progreso de estos.

Para lograr esto se siguieron los siguientes pasos:

1. Búsqueda de la enfermedad, tratamiento para el desequilibrio (apartado 3).
2. Investigación biomecánica del equilibrio para conocer datos relevantes para el estudio del equilibrio en pacientes con ictus (apartado 3).
3. Importancia del conocimiento de la posición del paciente en cada momento para evaluar qué grado de desequilibrio presenta. (apartado 3).
4. Utilización del Kinect para reconocimiento de extremidades necesarias para el cálculo de los datos de interés.
5. Trabajar con el Kinect acoplarlo con Unity por medio de la script BodySourceManager. (Este script ha sido escrito por el grupo del primer TFG [1]).

A continuación, se expondrán los datos de relevancia calculados, explicando cómo se sacaron estos datos y por qué fue necesario su cálculo.

6.4.1 Cálculo de ángulos de caída (Izquierda y Derecha)

El ángulo de caída, es de los datos más importantes para el seguimiento del progreso de un paciente que está trabajando en rehabilitar el equilibrio. Para el cálculo de estos ángulos se realizó un script "BodySourceView", este capta los movimientos del usuario a través del Kinect para reproducirlos en Unity mediante el avatar en primera persona, este script fue descargado de las librerías del Kinect y modificado según lo necesitado, para este caso se modificó el apartado Update donde gracias al reconocimiento de las extremidades mediante el Kinect, se crearon los vectores asociados a los puntos conocidos Spin Min, Ankle Right y Ankle Left para posteriormente encontrar los ángulos deseados como se muestra en la Figura 6.15**.**

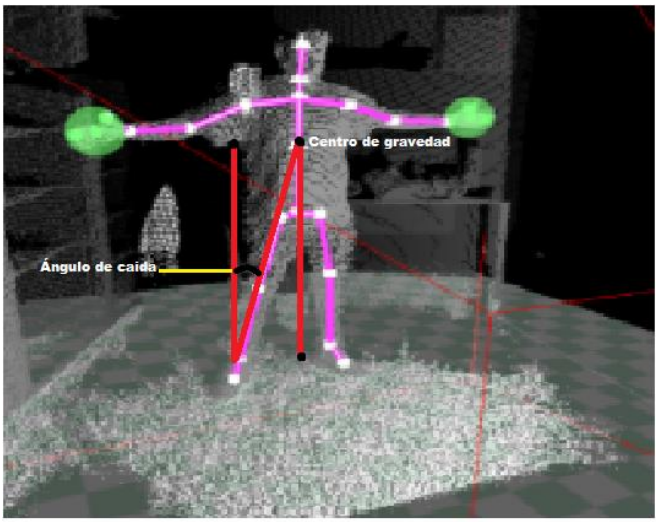


Figura 6.15: Vectores creados para hallar ángulo de caída

Una vez creados los vectores se procede a calcular el ángulo entre estos por medio del arco- cotangente de la magnitud del vector (adyacente) sobre la magnitud del vector (hipotenusa); todo esto se realizó por medio de una función ya definida en Unity como se ve en la Figura 6.16.

```
Vector3 vector1 = new Vector3(x1 - x0, y1 - y0, z1 - z0);
Vector3 vector2 = new Vector3(x0 - x0, y1 - y0, z1 - z0);
float angulocaidaR = Vector3.Angle(vector2, vector1);
```

Figura 6.16: Código para el cálculo del ángulo de caída

Posterior a esto se procede a realizar el cálculo del ángulo de caída máximo en derecha e izquierda, por medio de una función mostrada en la Figura 6.17 la cual compara constantemente los valores de los ángulos encontrados quedándose con el máximo realizado.

```
public void MAXDES(float angulocaidaR, float angulocaidaL)
{

    if (angulocaidaR > maxR)
    {
        maxR = angulocaidaR;


    }
    if (angulocaidaL > maxL)
    {
        maxL = angulocaidaL;
    }

    MAXR = maxR;
    Debug.Log("Angulo maximo R:" + MAXR);
    MAXL = maxL;
    Debug.Log("Angulo maximo L:" + MAXL);
}
```

Figura 6.17: Cálculo de valores máximos de ángulo de caída

6.4.2 Cálculo de Desviación referente a los ángulos calculados

La desviación es una medida de dispersión cuando mayor sea la dispersión mayor es la desviación estándar. Con esta medida de dispersión es posible representar que tanto varían los datos respecto a un valor de referencia [47]. Por consiguiente, decidimos que el mostrar los valores de desviación con respecto al punto de equilibrio inicial nos ayudaría a ver que tanto se alejan los datos del paciente de este punto y si estos concuerdan con los movimientos realizado durante la terapia.

El cálculo de la desviación se realizó, en un principio proyectando el punto Spine Mid o centro de gravedad al piso, esta primera muestra PI0 (Figura 6.18) se define como punto de equilibrio inicial, como requisito se debe cumplir que el paciente inicie el ejercicio en equilibrio total para obtener tener unos valores de desviación certeros. Posterior a esto se calculó la magnitud del vector formado por los puntos PI0 y PIa (punto Spine Mid proyectado al suelo en movimiento).

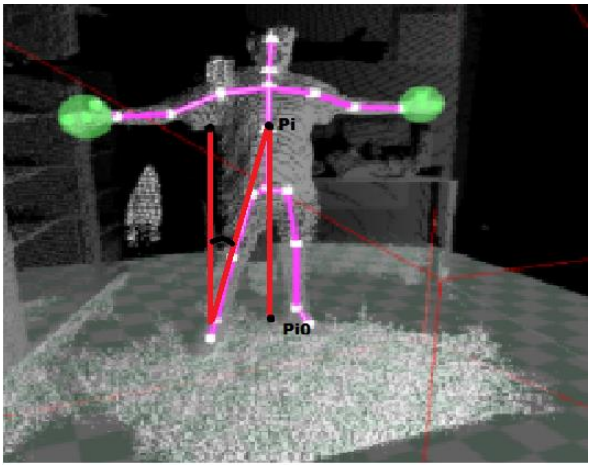


Figura 6.18: Puntos utilizados para la desviación estándar.

6.4.3 Cálculo de datos de posición

El conocer la posición del paciente en cada instante mientras se está realizando el ejercicio, nos brinda información muy importante ya que podemos saber si además de desequilibrio lateral el paciente presenta desequilibrios anteroposteriores.

Para realizar el cálculo de los datos de posición del paciente lo que hicimos fue coger los vectores ya creados (Figura 6.15) y reducirlos a dos dimensiones (quitar la altura), para así poder apreciar los datos mediante una vista de techo para una comprensión mayor de estos.

6.5 Extracción de datos

Como se ha mencionado anteriormente en el apartado *"5. Materiales usados"* y "*6.4 Reconocimiento de las extremidades del cuerpo*". La Kinect recibe y calcula los datos de las piernas respecto a un plano vertical. Estos datos son extraídos mediante un *script* que permite la creación de directorios y ficheros en formato "*.xls*". De esos formatos se pueden crear gráficas y analizar el comportamiento del paciente, así como su rendimiento y evolución.

Para la exportación de estos ficheros se utiliza la herramienta de *script* que usa Unity. Estos *scripts* están en lenguaje C# y se realizan en la plataforma "Visual Studio" de la empresa Microsoft Corp.

Antes de exportar algún fichero, primero se crea un seguido de directorios para guardar y diferenciar los ficheros que se extraen, tanto el paciente como el día y la hora. Es decir, si se analiza el siguiente código se puede ver que, justo al iniciar el juego, se crean dos directorios; uno para el paciente (si no lo tiene hecho) y otro para la hora y el día en que se hace la actividad.

```
public void Start()
{
    GameObject.Find

    #region Nuevo directorio paciente (angulos)
    DNI = GameObject.FindGameObjectsWithTag("DNI");
    nombre = DNI[1].name;

    dataDia = System.DateTime.Now.ToString();
    dataDia = dataDia.Replace('/', '_');
    dataDia = dataDia.Replace(':', '.');

    if (!Directory.Exists(Application.dataPath + "/Seguimiento/" + nombre))
    {
        Directory.CreateDirectory(Application.dataPath + "/Seguimiento/" + nombre);
    }

    if (!Directory.Exists(Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia))
    {
        Directory.CreateDirectory((Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia));
    }
    #endregion
}
```

Figura 6.19: Script para la creación de los directorios de cada paciente.

Si analizamos el código a fondo se ve que se encuentra dentro de la función "*Start*". Eso significa que tan solo empezar se creará un directorio si, y sólo si se introduce un DNI de paciente, ya que éste será el nombre del directorio. En la parte superior del código se sustituye varios caracteres que vienen por defecto en la fecha y la hora (ya que las barras pueden dar problemas al crear carpetas y/o directorios). Seguidamente, en el comando *"if"* se busca si existe un directorio con el DNI del paciente. Si la respuesta es afirmativa, el programa lo reconoce y sigue a la siguiente línea sin necesidad de crear nada. Si, por lo

contrario, no encuentra ningún directorio con el DNI introducido, éste lo crea y sigue leyendo las líneas de código. Una vez tiene seguro que el directorio existe, busca si hay una carpeta con el mismo horario (al ser un horario distinto, no existe ninguna) y al no encontrar una coincidencia, crea un directorio con la fecha y la hora como nombre (así, cada vez que se haga un ejercicio se creará un fichero nuevo).

6.5.1 Exportación datos ángulos de caída (Izquierda y Derecha) y la desviación estándar

El sistema que se utiliza para exportar los datos que recoge la Kinect se parece mucho al sistema para crear nuevos directorios. Pero en este caso son ficheros ".xls" y se tiene que añadir los datos cada vez que la Kinect (con su Temporizador) los recopila. Para ello, se tiene que crear un camino para el fichero y un contenido que añadir cada "*Update*".

```
#region Creación .xls
//Ángulo Derecho
string path = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia + "/Desequilibrio_AngleR" + ".xls";
string content = angulocaidaR + "\n";   //Contenido del .xls

if (!File.Exists(path))
{
    File.AppendAllText(path, "Data Angulo Derecho\nLogin date: " + dataDia + "\n");
}
File.AppendAllText(path, content);
//Ángulo Izquierdo
string path2 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia + "/Desequilibrio_AngleL" + ".xls";
string content2 = angulocaidaL + "\n";   //Contenido del .xls

if (!File.Exists(path2))
{
    File.WriteAllText(path2, "Data Angulo Izquierdo\nLogin date: " + dataDia + "\n");
}
File.AppendAllText(path2, content2);
```

Figura 6.20: Script para la creación de los archivos ".xls" de los ángulos izquierdo y derecho.

Como se muestra en el trozo de código de la Figura 6.20 primero se crea un camino para que el archivo sepa dónde debe ir. Obviamente, al haber creado un directorio para éste, se introduce el camino del directorio anteriormente creado y así los ficheros quedan ordenados para su posterior análisis. La segunda línea de código pertenece al contenido que se añade al fichero. Para el ángulo derecho se introduce la variable "*angulocaidaR"* y para el ángulo izquierdo la variable "*angulocaidaL*". Al final de la variable se le suma un "*n*", esto es una orden que se le da al fichero para saltar de línea o, en este caso, de casilla. De este modo se asegura de que el fichero queda limpio y preparado para analizarlo sin tener que ordenarlo posteriormente. Para terminar, el código busca si el fichero que quiere crear existe (mediante el comando *if*) y si no lo encuentra le ordena que cree un fichero "*.xls*" en el camino definido y con la fecha y la hora. Una vez creado, cada vez que la Kinect obtenga un dato, éste lo guarda en el fichero en una casilla nueva. Es muy importante que esta parte del código esté en la función "*Update*" ya que, si no es así, tan solo se guardará la fecha y la hora.

```
string path4 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia + "/Desequilibrio_Desviación" + ".xls"; //
string content4 = Magnitud + "\n";   //Contenido del .xls

if (!File.Exists(path4))
{
    File.AppendAllText(path4, "Data Desviación\nLogin date: " + dataDia + "\n");
}
File.AppendAllText(path4, content4);
```

Figura 6.21: Script para la creación de los archivos ".xls" de la desviación estándar

Siguiendo con la extracción de datos, para obtener los datos de la desviación estándar el procedimiento es exactamente igual al de los ángulos de las piernas. Tal y como se puede ver en la Figura 6.21.

6.5.2 Exportación datos de posición en los ejes X, Y y Z

Por otro lado, para crear el fichero de la posición se debe crear un contenido más ordenado. Es decir, la posición se representa con 3 datos distintos (x,y,z) y para poderlos ordenar bien se tiene que escribir un código un poco distinto para poder saltar de columna y fila cada vez que se guarde una coordenada.

```
string path3 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia + "/Desequilibrio_SpineMid" + ".xls"; /
string content3 = x1 + "\t" + y1 + "\t" + z0 + "\n";   //Contenido del .xls

if (!File.Exists(path3))
{
    File.AppendAllText(path3, "Data SpineMid\nLogin date: " + dataDia + "\nEje X\tEje Y\tEje Z\n");
}
File.AppendAllText(path3, content3);
```

Figura 6.22: Script para la creación de los archivos ".xls" de la posición en los ejes X, Y y Z

Como se muestra en la Figura 6.22, la base es la misma que en las otras extracciones. El pequeño cambio es a la hora de organizar las variables que se quieren introducir en el fichero "*.xls*". Para poder crear nuevas columnas se tiene que sumar el elemento "*t*" entre las variables que se quieren separar. Este comando hará que en el momento que se escriba el fichero las variables se separen en distintas columnas. Al terminar de guardar todas las variables se salta de fila con el comando "*n*".

6.6 Enlace con la base de datos

Para el almacenamiento de los datos personales de los pacientes/usuarios se utilizará MySQL (administrada por el software phpMyAdmin). Ésta BD es totalmente local (*offline*) por lo que solo se puede acceder a ella desde el ordenador en el que se encuentran los datos. Esta BD permite almacenar objetos (texto, números, listas, cadenas, etc.) y tiene una gran capacidad para guardar datos.

Para que phpMyAdmin y MySQL funcionen y exista una conexión es necesario iniciar los módulos "*Apache*" y "*MySQL*" en la plataforma XAMPP (ver Anexo A).

La parte derecha de la interfaz contiene toda la información referente al servidor (MySQL) y phpMyAdmin mientras que en el centro se encuentran las configuraciones más generales (véase Figura 6.23).

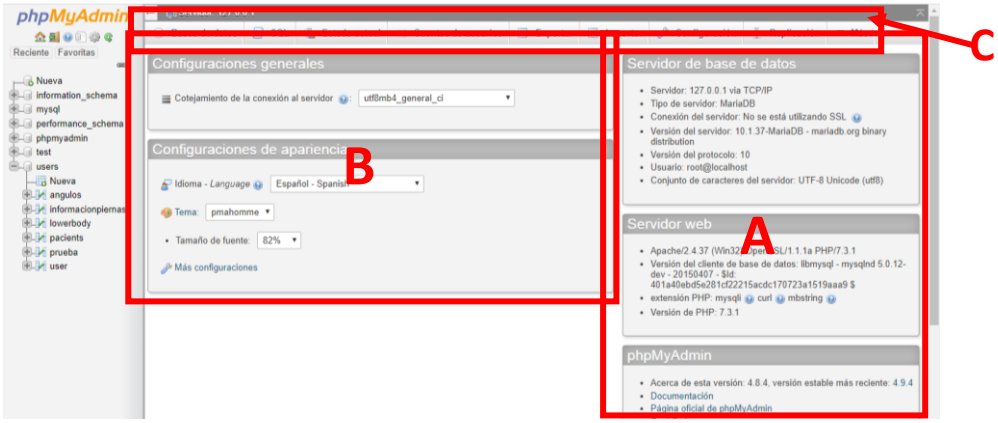


Figura 6.23: Página principal de phpMyAdmin. A – información sobre MySQL y phpMyAdmin. B – configuraciones generales. C – Barra de herramientas.

Una vez entramos en el software phpMyAdmin se puede empezar a crear BBDD, modificar las existentes, relacionarlas entre ellas, introducir datos, etcétera.

6.6.1 Creación de una nueva tabla en la base de datos relacional "*users*"

Para poder almacenar los datos más significativos como: valor máximo de desequilibrio, el paciente que lo ha realizado, el nivel de la escena, fecha y hora, entre otros; se debe crear una nueva tabla en la BD ya existente y enviar los datos listados que se extraen de la Kinect y Unity (versión 5.4.1).

Crear una tabla nueva se ejecuta mediante la barra de herramientas de la página principal de phpMyAdmin (Figura 6.23). Entre las opciones disponibles existe una llamada "Bases de Datos" (Figura

6.24). En ella se encuentran todas las BBDD creadas para el funcionamiento y almacenamiento de los datos tratados en este proyecto (y, consecuentemente, los que le preceden).

Figura 6.24: Listado de las BBDD existentes actualmente.

El proyecto que precede a este estudio [3] crearon el enlace entre Unity y BD haciendo todas las tablas en la BD llamada "*users*". La nueva tabla tiene que estar en la misma BD ya que se necesitará enlazarla o, mejor dicho, relacionarla con las otras. En la Figura 6.24 se aprecia que la BD comentada es la última en la lista. Dentro de esta base hay 5 tablas (6 contando la que se crea para el proyecto) como se puede ver un la Figura 6.25 (A).

Figura 6.25: Esquema de las tablas y sus propiedades. A Tablas que componen la BD. B Columnas que forman la tabla "lowerbody" y sus propiedades.

Para crear la tabla se tiene que introducir las columnas que van a representar los datos almacenados y las propiedades de los datos y su trato dentro de ésta. Estas propiedades se pueden ver en la Figura 6.25 (B).

6.6.2 Creación del enlace mediante claves primarias/foráneas

Una vez se tiene creada la tabla con sus propiedades, es necesario indicar cual será la clave primaria. Ésta está simbolizada como una llave dorada, tal y como se ve representada en la Figura 6.25 (B). Este símbolo otorga la propiedad de exclusividad, es decir, el dato que esté almacenado en esta tabla no puede estar dos veces. Por ejemplo, la columna llamada "*ID*" tiene esta propiedad y un A_I ("*Auto_Increment*"); el A_I, como su nombre indica, incrementa el valor de ID una unidad cada vez de forma automática. La llave dorada indica que los valores de ID representan solamente a una entrada de datos por sesión/ejercicio.

Por otra parte, la clave foránea está representada por una llave plateada (representada en la Figura 6.25). Es imprescindible tener claves primarias para poder relacionar las tablas entre sí, pero sin las claves foráneas no se pueden completar las relaciones. La clave foránea identifica la clave primaria (si no es primaria no se puede enlazar) de la otra tabla y así se relacionan las tablas entre ellas. En la Figura 6.26 se pueden ver las propiedades de las claves primaria y foránea de la tabla "*lowerbody*".

Nombre de la clave	Tipo	Único	Empaquetado	Columna	Cardinalidad	Cotejamiento	Nulo	Comentario
PRIMARY	BTREE	Sí	No	ID	0	A	No	
DNI _Paciente	BTREE	No	No	DNI _Paciente	0	A	No	

Figura 6.26: Claves primaria y foránea para la tabla de datos "lowerbody.

Para hacer la relación entre claves de distintas tablas se debe acceder al "Diseñador" pulsando en el botón "Más" de la barra de herramientas (Figura 6.27). Una vez allí se pueden crear todas las relaciones que sean necesarias entre tablas.

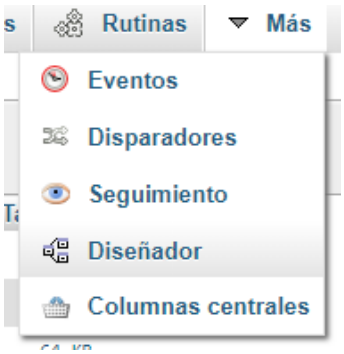


Figura 6.27: Acceso al "Diseñador".

Para poner un ejemplo, en la Figura 6.28 se aprecia la relación entre todas las tablas que hay en la BD. Siguiendo con la tabla "*lowerbody*", la clave foránea *DNI_Paciente* reconoce la clave primaria *DNI* de la tabla "*pacients*" y crea la relación.

Con esa relación, las tablas se pueden enviar información unas a otras sin tener que duplicar datos innecesariamente.

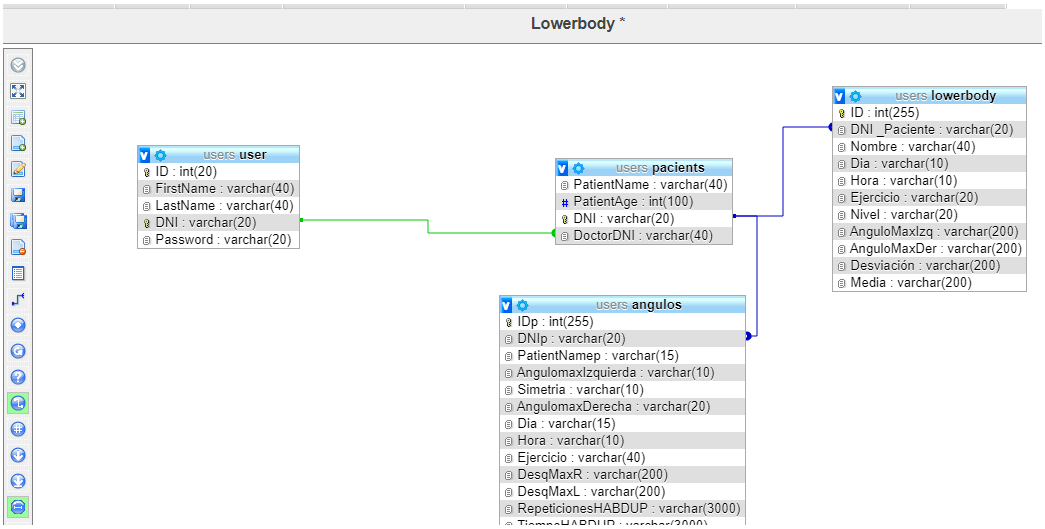


Figura 6.28: Relaciones existentes entre las tablas que componen la BD.

6.6.3 Envío y almacenamiento de datos de Unity a MySQL

Al terminar con la construcción de la nueva tabla "*lowerbody*" se procede al envío de información. Esto se lleva a cabo con dos *scripts* uno que calcula y va guardando los datos de forma temporal llamado "*Envioangulolowerbody"* escrito con el lenguaje C# (soportado por la plataforma Unity). A este *script* se le enlaza un camino en el código para que pueda localizar el segundo *script* llamado "*AngleInsertDes*" (ambos *scripts* disponibles en Anexo B). El segundo *script* está escrito con el lenguaje PHP, para realizar la conexión con la BD y, en lenguaje SQL para insertar los datos, recopilados en el primer *script,* a la tabla correspondiente.

Para que el *script* "*AngleInsertDes*" no envíe los datos automáticamente, se construyó un *trigger* al final del ejercicio. Éste, al ser activado, da paso al *script* "*Envioangulolowerbody*" a recoger los datos y enviarlos al segundo *script.* Así, solo se envían los datos al terminar un ejercicio entero y no cada vez que el programa hace un "*update*".

Finalmente, en la interfaz phpMyAdmin se pueden consultar los datos almacenados tal y como muestra la Figura 6.29.

Figura 6.29: Tabla de datos "lowerbody".

7. Resultados y Análisis de Resultados

Para la obtención de resultados se realizaron pruebas a 22 personas sanas con rango de edad de 20 a 60 años. Estas pruebas consisten en realizar una terapia de desequilibrio por medio de la aplicación de realidad virtual diseñada.

Figura 7.1. Prueba de la aplicación

Como se expone a lo largo del proyecto fueron diseñados 3 niveles ambientados de forma distinta y con una dificultad predeterminada diferente; todos los niveles fueron probados para evaluar su funcionamiento, pero por cuestiones prácticas para el análisis de datos, solo se cogieron los datos arrojados por la prueba del primer nivel.

Partiendo del hecho de que al probar la aplicación el programa crea automáticamente una carpeta para cada paciente en donde se guardaran los resultados de cada sesión como se explica en el apartado 6.2, es por esto que se obtuvieron 22 carpetas cada una con 4 ficheros en donde se guardan los resultados de los ángulos de caída pierna derecha, ángulo de caída pierna izquierda, posición del paciente en cada momento durante la realización del ejercicio y valores de varianza.

Como se muestra en la Figura 7.2. vemos que los ficheros se guardan en la carpeta llamada seguimiento con el número del DNI de cada paciente.

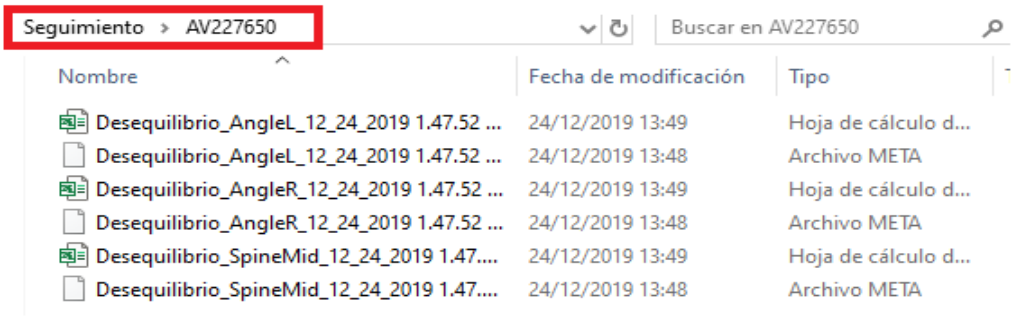


Figura 7.2. Carpeta de seguimiento del paciente

Al analizar los resultados obtenidos en los 22 ficheros observamos que no en todas las gráficas encontramos un comportamiento ideal, sin embargo, es posible reconocer tanto el estado en desequilibrio como el estado en equilibrio, a pesar de esto muchas de las gráficas presentan un comportamiento similar óptimo como podrá ser observado en el ANEXO C.3**,** es por esto que para el análisis se hará seguimiento solo a uno de los 22 ficheros obtenidos.

En base a lo anterior los resultados obtenidos fueron los siguientes:

- **Ángulos de Caída**

Al observar las gráficas de la Figura 7.3 podemos notar que cuanto más desequilibrio hay hacia uno de los lados, el ángulo de ese lado va a ser cada vez más cercano a cero, haciendo que el ángulo del lado contrario se haga grande en la misma proporción.

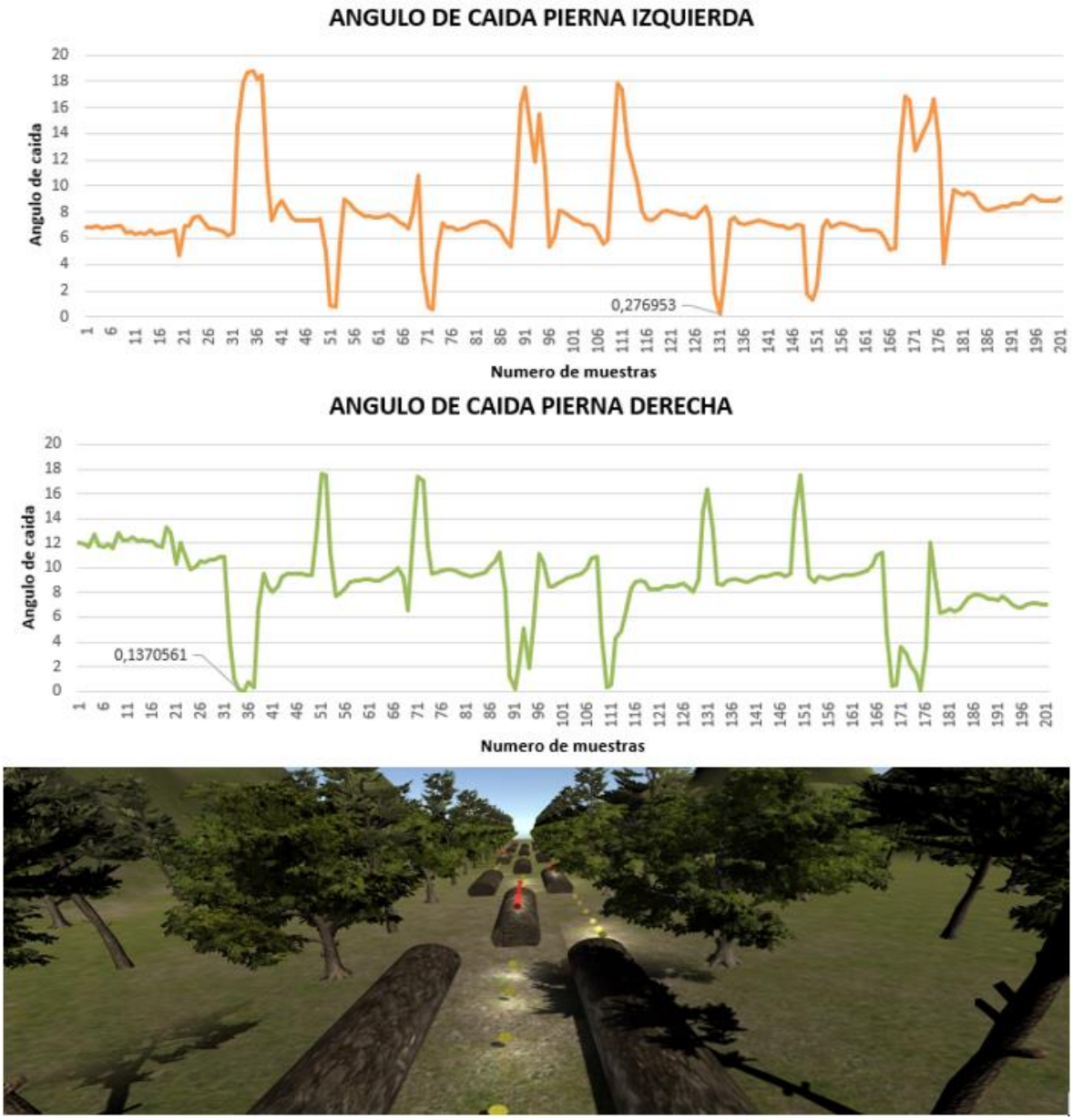


Figura 7.3 a. Gráfica de Ángulo de caída pierna izquierda, b. Gráfica de Ángulo de caída pierna derecha, c. Recorrido a realizar en el primer nivel.

Al observar los gráficos obtenidos y compararlos con el recorrido que debe realizar cada paciente, vemos que las gráficas concuerdan con dicho recorrido, para su mejor análisis este recorrido se ha separado por pasos:

1. El paciente inicia en posición de equilibrio, esto lo vemos en las gráficas donde el valor del ángulo es casi constante.

2. El paciente genera un desequilibrio a la derecha para recoger las monedas, esto se ve en las caídas de la gráfica (Figura 7.3.b), puesto que al desequilibrar a la derecha el valor de este ángulo es cercano a cero, mientras que el ángulo de la pierna izquierda aumenta (Figura 7.3.a).
3. El paciente se mantiene en equilibrio unos segundos antes de volver a generar un desequilibrio, como se muestra en las gráficas en los momentos donde no se presentan ni subidas ni bajadas.
4. Como se muestra en las gráficas el paciente realiza 8 desequilibrios en total, esto se evidencia si se cuentan los picos generados por estas (Figura 7.3.b y 7.3.a).
5.

Otros puntos que pueden ser analizados al observar los gráficos de la Figura 7.3 son:

a. Se obtuvo el valor de máximo desequilibrio para la pierna derecha el cual se observa en el pico de caída más grande de la gráfica verde siendo este (RMAx=0.13°).
b. Se obtuvo el valor de máximo desequilibrio para la pierna izquierda el cual se observa en el pico de caída más grande de la gráfica naranja siendo este (LMax=0.27°).
c. Se puede observar que el ángulo de equilibrio de esta persona se encuentra más o menos en 7°.
d. Al comparar la gráfica de ángulo de caída de la pierna izquierda con el de la pierna derecha se puede concluir que estas gráficas son correctas debido a que notamos que una es el complemento de la otra. (Figura 7.4)

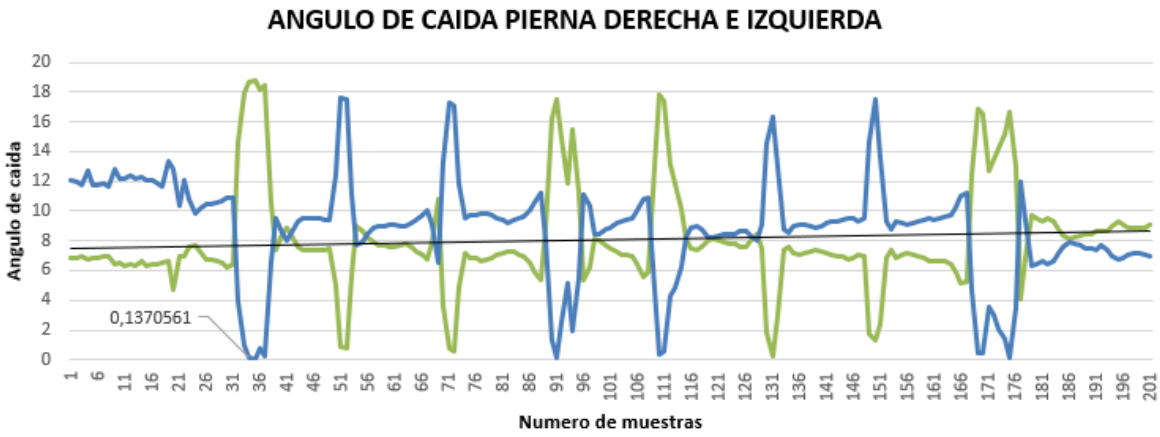


Figura 7. 4. Gráfica de comparación de ángulo de caída pierna izquierda (Verde), con ángulo de caída pierna derecha (Azul)

Por otro lado, al conocer el número de muestras totales y el tiempo de muestreo podemos conocer el tiempo total de juego, que en este caso fue de 1.6 min.

$$Tiempo\ total\ de\ juego\ = \frac{201 * 0.5seg}{60\ seg} = 1.6\ min$$

Además de esto por medio de la gráfica podemos encontrar el tiempo en desequilibrio total a la derecha, el tiempo en desequilibrio total a la izquierda y tiempo total en equilibrio.

$$Tiempo\ total\ en\ desequilibrio\ a\ la\ derecha\ =\ 36 * 0.5seg = 18\ seg$$

$$Tiempo\ total\ en\ desequilibrio\ a\ la\ izquierda\ =\ 24 * 0.5\ seg = 12\ seg$$

$$Tiempo\ total\ en\ equilibrio\ =\ 100.5seg - (18\ seg\ +\ 12\ seg) = 75.5\ seg\ \simeq\ 1{,}25833\ min$$

De los datos anteriores se puede concluir que la mayor parte del juego/terapia el paciente se encuentra en estado de equilibrio, lo cual se corrobora al analizar el recorrido (figura 7.3.c) donde vemos que solo por momentos se generan desequilibrios hacia los lados.

- **Valores de Posición**

Los valores de posición fueron sacados del punto Spine Mid de cada persona mientras esta, se encuentra realizando la terapia de rehabilitación por medio de la aplicación. Las gráficas que se pueden sacar de estos datos son las siguientes:

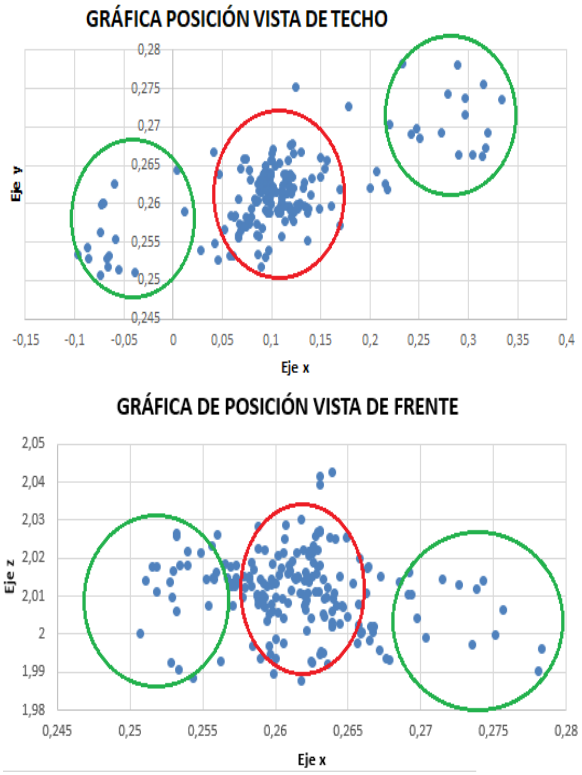


Figura 7. 5. a. Gráfica posición vista de techo, b. Gráfica posición vista de frente.

Al analizar en la figura 7.5 vemos tanto en el gráfico de vista de techo como en el de vista de frente el estado de equilibrio se encuentra en donde vemos la mayor aglomeración de puntos, esto lo podemos corroborar con la información hallada anteriormente en donde vemos que la mayor parte del tiempo de juego/terapia el paciente se encuentra en estado de equilibrio.

Si observamos el gráfico de vista de techo podemos ver que esta persona no presenta un desequilibrio anteroposterior ya que al analizar estos puntos en el (eje y) estos se encuentran muy cercanos entre ellos, además de esto al analizar las dos gráficas vemos que esta persona presenta un desequilibrio mayor hacia el lado izquierdo.

Otro aspecto a analizar es que es notorio que, en la gráfica de vista de techo, se ve girado en diagonal esto se pudo deber a que la persona que realizó el ejercicio no estaba alineada en un 100% con el Kinect, sin embargo, en las dos gráficas es posible observar claramente cuando la persona se encuentra en estado de equilibrio (círculos rojos) y cuando se encuentra en estado de desequilibrio (círculos verdes).

- **Valores de Desviación Estándar**

Como fue explicado en el apartado 6.4.2, para el cálculo de los valores de desviación se coge como valor de referencia el primer punto de proyección al piso del punto Spine Mid, este punto se obtiene cuando la persona inicia el juego/terapia, la gráfica obtenida de estos datos fue la siguiente:

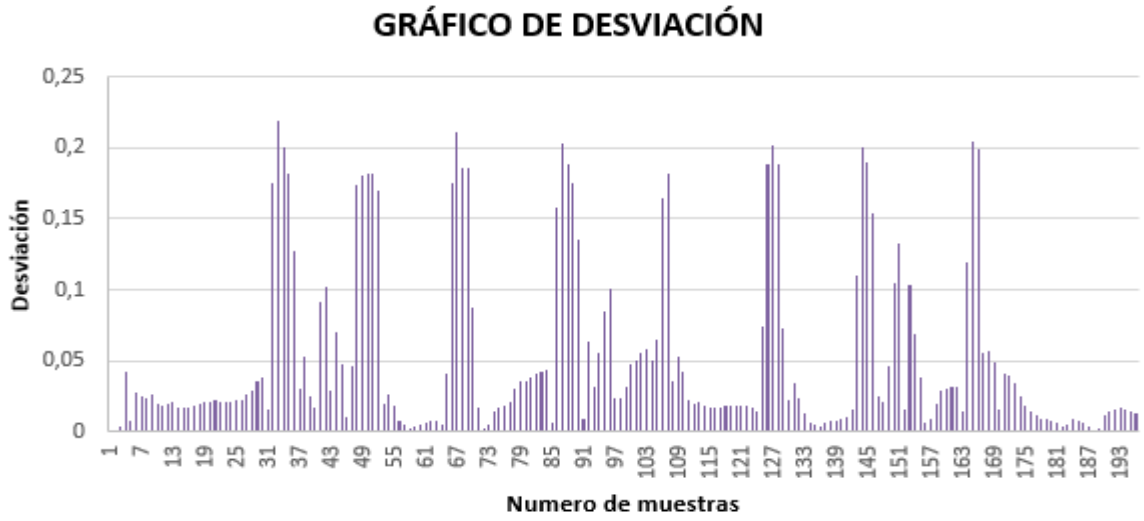


Figura 7. 6. Gráfica de desviación estándar

Al analizar los datos obtenidos podemos observar que el número de picos es igual al número de desequilibrios que se realizan durante el juego/ terapia (8 desequilibrios). Además de esto se puede

observar que cuando la persona se encuentra en equilibrio el valor de desviación es muy cercano a cero (al punto de equilibrio).

Por otro lado, para este proyecto se diseñó un menú en donde el fisioterapeuta puede encontrar y variar diferentes parámetros como se explica en el apartado 6.2.3, uno de estos botones sirve para ver la gráfica en tiempo real del paciente mientras este se encuentra realizando la terapia (ver Figura 7.7).

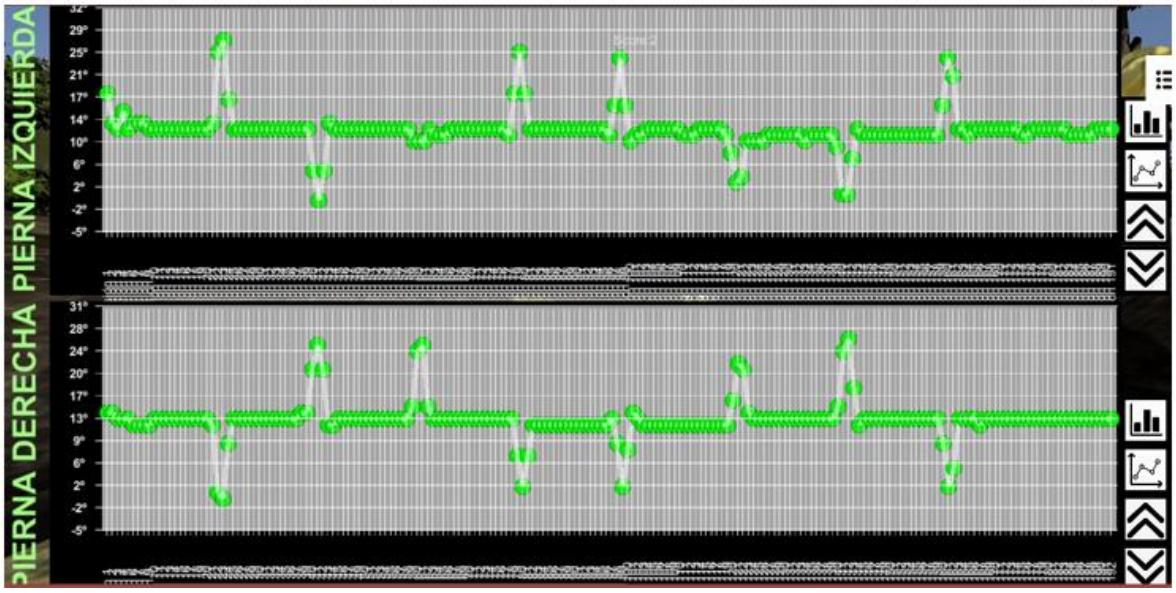


Figura 7.7. a. Ángulo de caída pierna izquierda, b. Ángulo de caída pierna derecha.

Las gráficas mostradas en la Figura 7.7 aparecen en la interfaz de Unity en tiempo real, gracias a esto es posible que el terapeuta genere correcciones instantáneas al paciente, además el terapeuta puede verificar que el paciente esté realizando el ejercicio de forma correcta desde un principio y así lograr que este obtenga resultados en menos tiempo.

Además en la Figura 7.7 podemos observar la gráfica de una persona que tuvo mucho contacto con la aplicación ya que estuvo constantemente ayudándonos a realizar las pruebas, si comparamos esta gráfica con la obtenida en la Figura 7.8 vemos que la gráfica de la Figura 7.7 sería una gráfica ideal a obtener de personas sanas, creemos que no todas las gráficas se obtuvieron de esta forma ya que al no tener contacto anterior con la realidad virtual, puede que el realizar la terapia por primera vez resulte confuso mientras se genera una adaptación, sin embargo la mayoría de los resultados obtenidos son favorables ANEXO C.3.

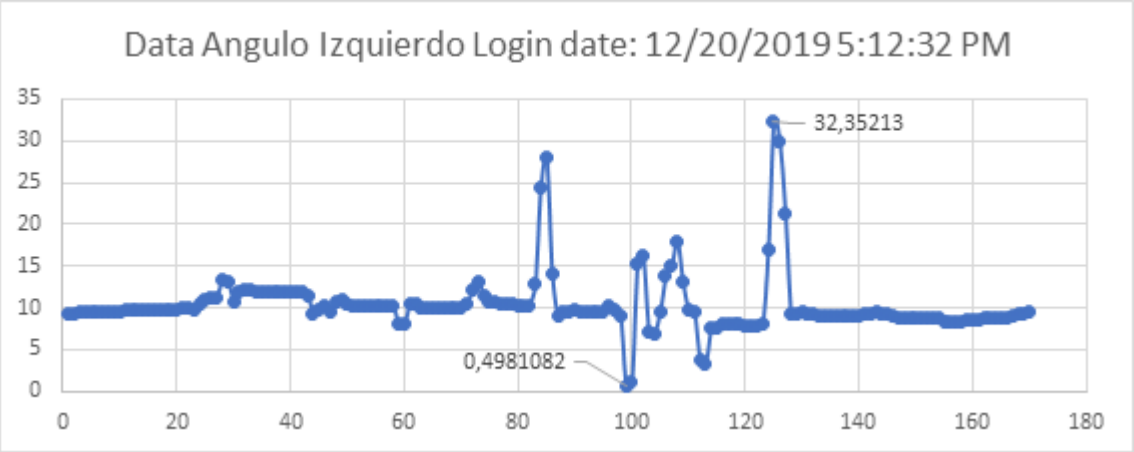


Figura 7.8. a. Ángulo de caída pierna izquierda, sacado de los ficheros de excel ANEXOC.3

Debido a que se quería obtener una retroalimentación de la aplicación diseñada se realizó una encuesta a las 22 personas que probaron la aplicación (ver ANEXO C.2) los resultados obtenidos se analizarán a continuación:

Queríamos analizar es si la edad era un factor de influencia a la hora de realizar la terapia puesto que se tiende a creer que el tema de la realidad virtual lo manejan de mejor forma los jóvenes, es por esto que esta prueba fue realizada a personas con diferentes rangos de edad y estos fueron divididos de la siguiente forma: 20-45, 46-60 y mayores de 60 obteniendo los siguientes resultados:

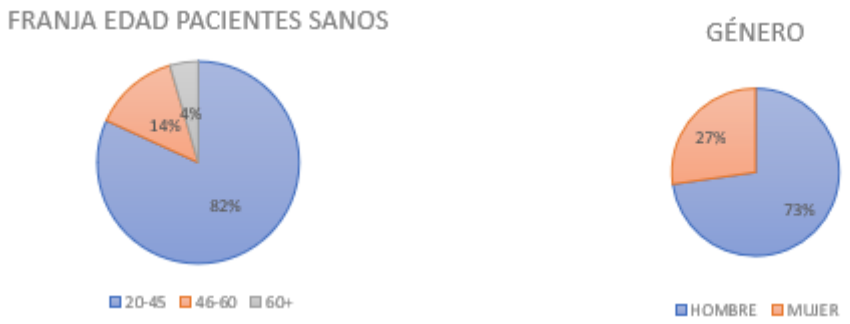


Figura 7.9. a. gráfico de edad, b. gráfico de género.

Al analizar las respuestas de las personas que evaluaron la aplicación en la encuesta vimos que no presentaron mucha dificultad a la hora de realizar la terapia y además todas calificaron la experiencia entre 4 y 5 siendo 5 muy buena experiencia como puede observarse en la Figura 7. 10.

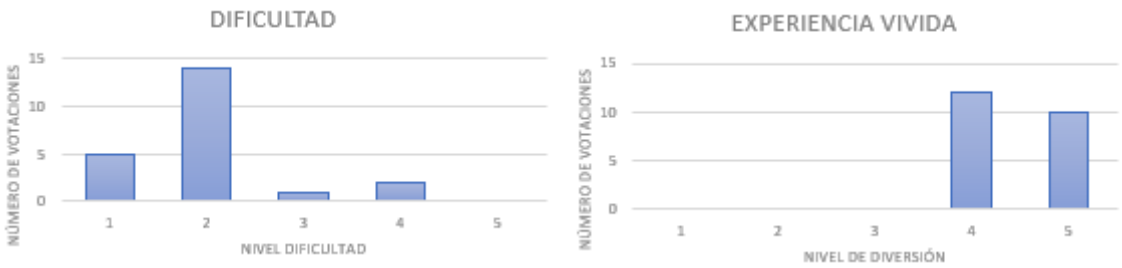


Figura 7. 10. a. gráfico de dificultad del juego, b. gráfico de calificación de la experiencia

Por otro lado, también quisimos saber qué tan útil les parece esta aplicación y si habían sentido o no mareo al realizar la terapia, los resultados obtenidos fueron los siguientes:

Figura 7. 11. a. gráfico de opinión, b. gráfico de presencia de Mario durante la experiencia

Como se puede observar un 95% de las personas piensa que la aplicación puede ayudar a mejorar una experiencia de rehabilitación y un 68% no presentaron mareo a la hora de realizar la terapia, de ellos muchos dijeron que el mareo solo era al comenzar mientras se adaptan a la experiencia.

Por último, ya que realizamos dos sistemas para el control de movimiento queríamos saber cuál de ellos era con el que las personas se sentirían más cómodas realizando la terapia los resultados obtenidos se muestran en la imagen a continuación en donde vemos que un 63% prefiere el sistema de control por piernas.

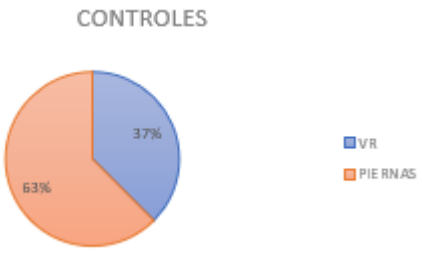


Figura 7. 12. gráfico de preferencia de controles de mando

8. Análisis del impacto ambiental

Para analizar el impacto ambiental que tiene este proyecto se debe tener en cuenta todo el ciclo de vida de los elementos que lo forman. Desde la extracción del material para utilizarlo al final de la vida útil de éste.

Si se incluyen números para el hardware del ordenador (torre y pantalla), se requieren entre 5040 y 5600 millones de Julios de energía para el proceso de producción y 115 vatios de potencia para el consumo de uso diario además una emisión de entre 52 y 234 gramos de CO_2 [38] [39].

Aunque son números altos en comparación a otros dispositivos o productos, los dispositivos utilizados para el proyecto tienen una vida útil prolongada pudiendo superar hasta 8-10 años de funcionamiento (sin contar con posibles accidentes físicos). Esto hace que el impacto ambiental más elevado que tiene el estudio sea el coste energético de su funcionamiento. Pudiendo emitir una cantidad de CO_2 de entre 880,6 y 1.872 kg al largo de su vida útil [41] [42]. Esta emisión puede ser menor dependiendo del método de adquisición de la energía. Es decir, si la energía se obtuviera de energías renovables el impacto ambiental y las emisiones de CO_2 serían considerablemente menores.

Otro impacto que tiene la electrónica sobre el medio ambiente son los residuos desechados. Los dispositivos electrónicos usados en el proyecto tienen metales pesados como mercurio, plomo, níquel y muchos más [40].

Figura 8.1: Símbolo que indica la recogida selectiva de aparatos eléctricos/electrónicos (Fuente: https://www.osakidetza.euskadi.eus/informacion/que-puedo-hacer-con-mis-residuos/ab84-sercon/es/).

Para que el desecho de dichos materiales y aparatos electrónicos sea controlado y no termine contaminando el subsuelo o las aguas cercanas como ríos, mares o lagos, la Unión Europea obligó a la gestión y recogida de los aparatos eléctricos y electrónicos.

La nueva y revisada Directiva Europea 2012/19/UE sobre residuos de aparatos eléctricos y electrónicos (transpuesta en España a través del Real Decreto 110/2015) obliga a la gestión y recogida de los aparatos, eléctricos y electrónicos, destacando los electrodomésticos y aparatos electrónicos de gran consumo una vez dejan de ser utilizados y se convierten en residuos [43].

Estos productos tuvieron que ser marcados con un símbolo que indicaba un cubo tachado y la necesidad de crear un sistema de recogida y reciclado de los residuos [43]. El símbolo se puede observar

en la Figura 8.1. Todos los productos hardware que se emplean para el desarrollo del proyecto tienen el símbolo de la Figura 8.1 y cumplen así con la normativa de recogida y reciclaje marcada por la Unión Europea (UE).

No obstante, según han mostrado algunos estudios, la RV se puede usar para la concienciar y mentalizar a la gente sobre la naturaleza y el impacto que está teniendo el cambio climático y el mal tratamiento de los residuos [44] [45]. En estos estudios se observa el desarrollo de los hábitos de una persona tras vivir una experiencia (de carácter catastrófico) en RV. De este modo se intenta suscitar y cambiar la mentalidad de las personas para mostrar empatía y cercanía con el medio ambiente [44].

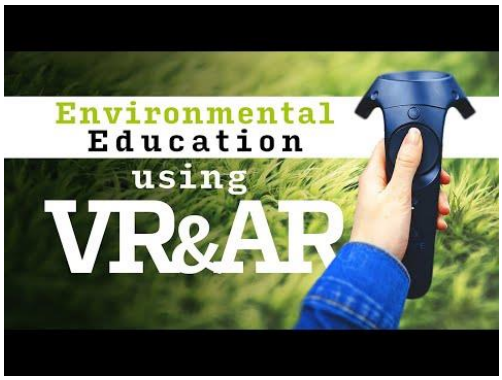


Figura 8.2: Vídeo de YouTube sobre el uso de la RV para preservar el medio ambiente (Fuente: https://www.youtube.com/watch?v=WUV9SydDvYo).

9. Conclusiones

- La implementación de la RV en las terapias de la rehabilitación es favorable y ameniza las sesiones. En este proyecto se desarrolló, de forma favorable, hasta 3 entornos de realidad virtual para restaurar el equilibrio de pacientes que tuvieron un accidente cerebrovascular.

- Se pudo diseñar 3 ambientes con niveles de dificultad y ambientación distintos para que el usuario pueda ver un progreso lineal mientras realiza las sesiones y así incentivar al paciente a seguir con la rehabilitación.

- Según los datos y opiniones recolectadas, la terapia por medio de la RV puede ser una forma muy entretenida y distraída de rehabilitación y, su aplicación, un avance muy útil en el mundo de la fisioterapia.

- Actualmente mediante la aplicación diseñada es posible la extracción de datos de interés como: ángulo de caída, puntos de posición y desviación por medio de ficheros en Excel.

- La aplicación de la RV en la fisioterapia combina los avances tecnológicos con las técnicas convencionales de los fisioterapeutas. Este proyecto contribuye al desarrollo y a la evolución de las ciencias de la salud.

- La inmersión del paciente en primera persona en la aplicación mediante la RV y el control de un avatar mediante la Kinect, hacen que esta sea una manera más efectiva en la rehabilitación del equilibrio ya que, según informes recopilados, reduce el tiempo de rehabilitación.

- Esta aplicación permite adaptarse al usuario y hacer ejercicios más personales y enfocados a cada persona/paciente. El control mediante las piernas se presenta como una alternativa al control mediante las gafas de RV para disminuir el mareo.

- El uso de métodos gráficos para el análisis de resultados hace más comprensible la interpretación del progreso tanto para el fisioterapeuta como para el paciente.

- Con los datos extraídos de las pruebas realizadas a personas sanas se ha podido evaluar la funcionalidad de la aplicación, el hecho de obtener resultados tan parecidos apunta a que la aplicación está bien desarrollada y que aporta datos precisos.

- La aplicación de la RV inmersiva también ayuda a economizar el proceso de fisioterapia ya que se puede realizar desde un domicilio y así, ahorrar tanto en el gasto monetario como en recursos humanos. Aun así, siempre será necesaria la ayuda de fisioterapeutas para el seguimiento y, si el caso lo requiere, una forma de fisioterapia convencional.

● Líneas Futuras

Este proyecto aún está en su fase "*beta*" y hay muchos caminos posibles para mejorar esta aplicación:

1. Poder escoger entre una variedad de entornos para realizar los ejercicios sea cual sea el nivel. Así el paciente se puede involucrar más en las sesiones y mostrar más interés.
2. Hacer más ejercicios con un personaje dinámico para que no parezca muy repetitivo el ejercicio. Es decir, hacer que el avatar se mueva en una dirección predeterminada mientras se hace el ejercicio.
3. Cambiar el servidor/BD (actualmente "*offline"*) a uno "*online*". Con este cambio sería posible hacer una aplicación que pueda ser usada desde un domicilio o en otro sitio que no fuera el hospital o similar (tanto para el fisioterapeuta como para el paciente).
4. Desarrollar un método para controlar la aplicación mediante la Kinect, sin necesidad de controles. Así el paciente sería capaz de entrar y hacer el ejercicio solo.
5. Poder escoger el tiempo o duración del ejercicio para adaptarlo mejor al paciente y hacer una rehabilitación más personal.
6. Mejorar el sistema de control por piernas y que los ángulos iniciales sean los ángulos iniciales del paciente y no unos predeterminados. Para ello se deben almacenar los ángulos iniciales que recoge la Kinect y resetear los ángulos predeterminados por ésos.
7. Crear un nivel para aquellas personas que sean incapaces de aguantar el equilibrio o moverse y hacer una representación del movimiento ampliada para simular un avance mínimo y animar al paciente.

● Análisis Económico

En este apartado se especificarán los costes asociados a la elaboración de la aplicación desde diferentes aspectos los cuales se mencionan a continuación:

- Costos del Personal:

Para este apartado se tiene en cuenta que el proyecto final de grado tiene 24 ECTS y se realizan 25 horas de trabajo por cada ECTS, es por esto que se dedicaron 600h/ ingeniero para la culminación del proyecto. Este proyecto se comenzó el día 18 de septiembre y finalizó el 14 de enero dedicando aproximadamente 7.5 horas diarias.

PERSONAL	TIEMPO (h)	COSTE POR HORA (€)	COSTO TOTAL (€)
1	600,00	30,00	18.000,00
1	600,00	30,00	18.000,00
TOTAL			36.000,00

Tabla A-1. Costo del Personal

- Costos de Consumo Eléctrico:

Para el cálculo del consumo eléctrico de busco el consumo en KW de los materiales utilizados para la realización del proyecto (véase Tabla A-2)

MATERIAL	CONSUMO (KW/ud)	CANTIDAD (ud.)
CPU Core i7	0,091	1
GPU GTX Windforce	0,12	1
Memoria RAM DDR4	0,005	1
HDD+SSD	0,011	1
Placa Base	0,01	1
Ventiladores de 120mm	0,003	3
Ratón	0,002	1
Gafas HTC Vive	0,017	1
Kinect	0,016	1
Pantalla	0,045	1
Total	0.471 (KW)	

Tabla A-2. Costo del Consumo eléctrico

Según lo encontrado [47] el precio de una tarifa general de consumo es de 0.110 €/KW*hora, al día de hoy, es por esto que el costo del consumo eléctrico se calcula de la siguiente forma:

$$Precio\ €KW\ *\ \#\ horas\ de\ trabajo\ *\ \#\ Total\ de\ KW * Hora = Costo\ del\ Consumo\ Eléctrico$$

$$0.110\ €/KWh * 600h\ *\ 0.471KW\ = Costo\ del\ Consumo\ Eléctrico$$

$$31{,}0860\ = Costo\ del\ Consumo\ Eléctrico$$

- Costos de Hardware

Para el cálculo relacionado con los costos de Hardware se tuvo en cuenta la vida útil de cada material utilizado, la fórmula utilizada para encontrar el coste real fue la siguiente:

$$\frac{Meses\ de\ Uso}{Vida\ Útil\ (meses)} * 100 = Depreciación\ \ \ \ * Precio$$

MATERIAL	PRECIO (€/u)	CANTIDAD (ud.)	VIDA ÚTIL (meses)	DEPRECIACIÒN (%)	ENTIDAD	COSTE REAL (€)
Kinect V2 Sensor	85,00	1	36	16,67	UPC	14,17
Ordenador	1.500,00	1	48	12,5	Visyon 360	187,5
Gafas HTC Vive Full Kit	1.074,00	1	42	14,29	Visyon 360	153
Adaptador de Windows	40,00	1	36	16,67	UPC	6,67
Costes totales del Hardware	-	-	-	-	-	361,77

Tabla A-3. Costo del Hardware

- Costos de Software

En los costos del software se tuvo en cuenta que los dos integrantes del proyecto pagaron por el Máster online de formación en Unity.

MATERIAL	Precio por Unidad ((€/u))	Unidades (€)	Total (€)
Curso formación unity	9,00	2	18,00
Microsoft Kinect for Windows SDK	0,00	1	0,00
Microsoft Visual Studio	0,00	1	0,00
Steam VR	0,00	1	0,00
Unity Version 5.3.0f3	0,00	1	0,00
Coste total del Software	0,00	3	18,00

Tabla A-4. Costo del Software

- Costos Totales

Finalmente, en la Tabla A-5 se observan lo costos totales que reúnen los costos explicados en las tablas anteriores, mostrando como presupuesto final un resultado de 446,77 €.

COSTE	PRECIO (€/u)
PERSONAL	36.000,00
CONSUMO ELÉCTRICO	31,0860
HARDWARE	361,77
SOFTWARE	18,00
TOTAL	446,77

Tabla 5. Costos totales.

Bibliografía

[1] A. Santaeularia and J. Guerrero, "REHABILITATION APP" no. enero, 2019.

[2] C. Vega y O. Mayorgas, "VIRTUAL MIRROR Aplicación de Realidad Virtual" no. junio, 2019.

[3] E. Medina and V. Rofes, "VIRTUAL REALITY REHABILITATION APP (UPPER BODY)", no. junior, 2019.

[4] "Pygmalion's Spectacles by Stanley G. Weinbaum - Free Ebook." [Online]. Disponible: http://www.gutenberg.org/ebooks/22893. [Accedido: 19-diciembre-2019].

[5] Ivan E. Sutherland, "The Ultimate Display". [Online]. Disponible:http://worrydream.com/refs/Sutherland%20-%20The%20Ultimate%20Display.pdf [Accedido: 5-enero-2020].

[6] Abigail Martín, "Una terapia divertida: Juegos de realidad virtual y fisioterapia". [Online]. Disponible: https://www.fisioterapia-online.com/articulos/una-terapia-divertida-juegos-de-realidad-virtual-y-fisioterapia [Accedido: 5-enero-2020].

[7] Angela Bernardo, "La realidad virtual llega a la medicina: así puede mejorar los ejercicios de rehabilitación". [Online]. Disponible: https://blogthinkbig.com/la-realidad-virtual-llega-a-la-medicina-asi-puede-mejorar-los-ejercicios-de-rehabilitacion [Accedido: 5-enero-2020].

[8] "Companies using Unity". [Online]. Disponible: https://enlyft.com/tech/products/unity [Accedido: 5-enero-2020].

[9] "Unity". [Online]. Disponible: https://docs.unity3d.com/Manual/system-requirements.html [Accedido:6-enero-2020].

[10] "Microsoft Windows 8 Features Leaked: Instant-On, Facial Recognition, New Technologies.". [Online]. Disponible: https://web.archive.org/web/20101103071614/http://www.xbitlabs.com/news/other/display/20100629221134_Microsoft_Windows_8_Features_Leaked_Instant_On_Facial_Recognition_New_Technologies.html [Accedido: 6-enero-2020].

[11] "Time of Flight Principle" [Online]. Disponible: https://www.terabee.com/time-of-flight-principle/ [Accedido: 6- enero-2020].

[12] "Características Kinect 2" [Online]. Disponible: http://www.kinectfordevelopers.com/es/2014/01/28/caracteristicas-kinect-2/ [Accedido: 6-enero-2020].

[13] "Configurar el sensor Kinect para Windows v2 o un sensor Kinect de Xbox con el Adaptador de Kinect para Windows" [Online]. Disponible: https://support.xbox.com/es-ES/xbox-on-windows/accessories/kinect-for-windows-v2-setup [Accedido: 7-enero-2020].

[14] "Kinect for Windows SDK 2.0" [Online]. Disponible: https://www.microsoft.com/en-us/download/details.aspx?id=44561 [Accedido: 7-enero-2020].

[15] "Understanding Kinect V2 Joints and Coordinate System". Disponible: https://medium.com/@lisajamhoury/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16 [Accedido:7-enero-2020].

[16] "VIVE Specs & User Guide"[Online] Disponible: https://developer.vive.com/resources/knowledgebase/vive-specs/ [Accedido:7-enero-2020].

[17] "VR overview" [Online]. Disponible: https://docs.unity3d.com/Manual/VROverview.html[Accedido: 7-enero-2020].

[18] "Types of VR system" [Online]. Disponible: http://www.agocg.ac.uk/reports/virtual/37/chapter2.htm [Accedido: 7-enero-2020].

[19] Chien-Yen Chang, Belinda Lange, Mi Zhang, Sebastian Koenig, Phil Requejo, Noom Somboon, Alexander A. Sawchuk y Albert A. Rizzo, "Towards pervasive physical rehabilitation using Microsoft Kinect" [Online]. Disponible: https://ieeexplore.ieee.org/abstract/document/6240377 [Accedido: 7-enero-2020].

[20] "The emerging role of Microsoft Kinect in physiotherapy rehabilitation for stroke patients" [Online]. Disponible: https://www.physio-pedia.com/The_emerging_role_of_Microsoft_Kinect_in_physiotherapy_rehabilitation_for_stroke_patients [Accedido: 8-enero-2020].

[21] Dr. José Vicente Lozano, "¿Qué es un ictus?" [Online]. Disponible: https://rithmi.com/que-es-un-ictus/ [Accedido: 8-enero-2020].

[22] C. Claucich, L. C. Carrere y C. B. Tabernig, "Virtual Reality Interface Built Using Unity3D for Rehabilitation with BCI Systems Based on Motor Imagery", "2018 IEEE Biennial Congress of Argentina" (ARGENCON), San Miguel de Tucumán, junio 2018. Disponible: https://ieeexplore.ieee.org/document/8646096/authors#authors

[23] E. Vogiatzaki and A. Krukowski, “Serious Games for Stroke Rehabilitation Employing Immersive User Interfaces in 3D Virtual Environment”, 3ra “International Conference on Serious Games and Applications for Health” (SeGAH’2014), Rio de Janeiro, mayo 2014. Disponible: http://www.jhi-sbis.saude.ws/ojs-jhi/index.php/jhi-sbis/article/view/370 [

[24] J. Corden, “Farewell, dear sweet Kinect” [Online]. Disponible: https://www.windowscentral.com/ode-kinect [Accedido: 10-enero-2020].

[25] M. Ma, R. Proffitt y M. Skubic, “Validation of a Kinect V2 based rehabilitation game”, agosto 2018. Disponlibe: https://doi.org/10.1371/journal.pone.0202338 [Accedido: 10-enero-2020]

[26] E. Vogiatzaki, Y. Gravezas and A. Krukowski, “Rehabilitation System for Stroke Patients using Mixed-Reality and Immersive User Interfaces”, junio 2014.

[27] M. Matamala Gómez, “The use of immersive virtual Reality in neurorehabilitation and its impact in neuroplasticity”, mayo 2017. Disponible: https://www.tdx.cat/bitstream/handle/10803/666826/MMG_PhD_THESIS.pdf?sequence=1&isAllowed=y

[28] R. Adams, “Virtual Activities of Daily Living for Recovery of Upper Extremity Motor Function”, vol.26, no. 1, enero 2018. Disponible: https://www.ncbi.nlm.nih.gov/pubmed/29324411

[29] H. Suk Lee, Y. Junk Park y S. Wook Park, “The Effects of Virtual Reality Training on Function in Chronic Stroke Patients: A Systematic Review and Meta-Analysis”, vol. 2019, junio 2019. Disponible: https://doi.org/10.1155/2019/7595639

[30] Anónimo, “Historia de las Bases de Datos”, enero 2011. [Online]. Disponible: https://histinf.blogs.upv.es/2011/01/04/historia-de-las-bases-de-datos/ [Accedido:12-enero-2020].

[31] M. A. Maria Mercedes, “Reseña histórica”, junio 2000. [Online]. Disponible: http://www3.uji.es/~mmarques/proyecto/node4.html [Accedido: 12-enero-2020].

[32] E. F. Codd, “A Relational Model of Data for Large Shared Data Banks”, vol. 13, n.6, junio 1970. Disponible: https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf

[33] “XAMPP Apache + MariaDB + PHP + Perl”. [Online]. Disponible: https://www.apachefriends.org/es/index.html [Accedido: 12-enero-2020].

[34] “About”. [Online]. Disponible: https://www.phpmyadmin.net/ [Accedido: 12-enero-2020].

[35] "World's Most Popular Open Source Database". [Online]. Disponible: https://www.oracle.com/mysql/ [Accedido: 12-enero-2020].

[36] Gustavo B., "¿Qué es MySQL? Explicación detallada para principiantes", mayo 2019 [Online]. Disponible: https://www.hostinger.es/tutoriales/que-es-mysql/ [Accedido: 12-enero-2020].

[37] "Que es XAMPP y que necesita para ser instalado". [Online]. Disponible: https://sites.google.com/site/portafoliovicenciosr/poll [Accedido:12-enero-2020].

[38] A. Castán Salinas, "Material informático y contaminación medioambiental", Rev. 10, junio 2004. Disponible: https://www.grefa.org/grefa/containfor.pdf

[39] "Los ordenadores también emiten CO_2", abril 2016. [Online]. Disponible: https://www.ecoembes.com/es/planeta-recicla/blog/los-ordenadores-tambien-emiten-co2 [Accedido: 12-enero-2020].

[40] "Conozca sus productos electrónicos y sus sustancias químicas". [Online]. Disponible: https://chemicalsinourlife.echa.europa.eu/es/know-your-electronics [Accedido: 12-enero-2020].

[41] "¿Cuánto cuesta la electricidad que consume tu PC?". [Online]. Disponible: https://hardzone.es/2015/03/31/cuanto-cuesta-la-electricidad-que-consume-tu-pc/ [Accedido:12-enero-2020].

[42] "Calculadora de emisiones de CO". [Online]. Disponible: http://calcarbono.servicios4.aragon.es/index.html [Accedido: 12-enero-2020].

[43] J. Alberto Pizarro, "RAEE ii: Revisión de la Directiva de residuos de aparatos eléctricos y electrónicos", enero 2018. [Online]. Disponible: https://blogespanol.se.com/normativa-energetica/2018/01/15/revision-de-la-directiva-de-residuos-de-aparatos-electricos-y-electronicos/ [Accedido: 12-enero-2020].

[44] Stanford University, "Virtual reality could serve as powerful environmental education tool." ScienceDaily. ScienceDaily, 30 November 2018. Disponible: <www.sciencedaily.com/releases/2018/11/181130080349.htm>. [Accedido:13-enero-2020].

[45] The National Science Foundation, "Increasing ecological understanding with virtual worlds and augmented reality", abril 2015. [Online]. Disponible: https://www.nsf.gov/discoveries/disc_summ.jsp?cntn_id=134922&org=NSF&from=news [Accedido: 13-enero-2020].

[46] “Programa ADFO | UVic.” [Online]. Disponible: https://www.uvic.cat/uhub/voluntariat/adfo. [Accessed: 12-Dec-2019].

[47] “Tarifas de luz de hoy. Precio de la electricidad ahora.,” 2018. [Online]. Available: http://www.tarifadeluz.com/index.php. [Accessed: 14-Ene-2020]

Anexo A: Manual de uso para acceder a ejercicios de desequilibrio

Antes de iniciar

Este trabajo es la continuación de un trabajo de grado anterior en donde ya se habían planteado unas instrucciones de uso **[3]**. Este será un manual que servirá para acceder a nuestra parte del proyecto, no obstante, el inicio de la aplicación se hace de la misma manera, es por esto que se utilizaran sus instrucciones de uso para este apartado y se buscará complementarlo con nuestros aportes.

A.1. Puesta en marcha e inicio de la aplicación

A.1.1. Inicio de la base de datos

En primer lugar, es importante dar inicio a la base de datos ya que, si este paso no es realizado, no podrá ser guardado registro alguno y como consecuencia de esto no se podrá iniciar la sesión para empezar la terapia.

Para poder dar inicio a la base de datos se seguirán los siguientes pasos:

1. Abrir el software XAMPP
2. Dar marcha a los módulos Apache y MySQL clicando sobre el botón ”Start”. (Figura A. 1**,** recuadro verde)
3. Si se inició correctamente la base de datos los módulos iniciados deben aparecer subrayados en verde ((Figura A. 1, recuadro rojo)

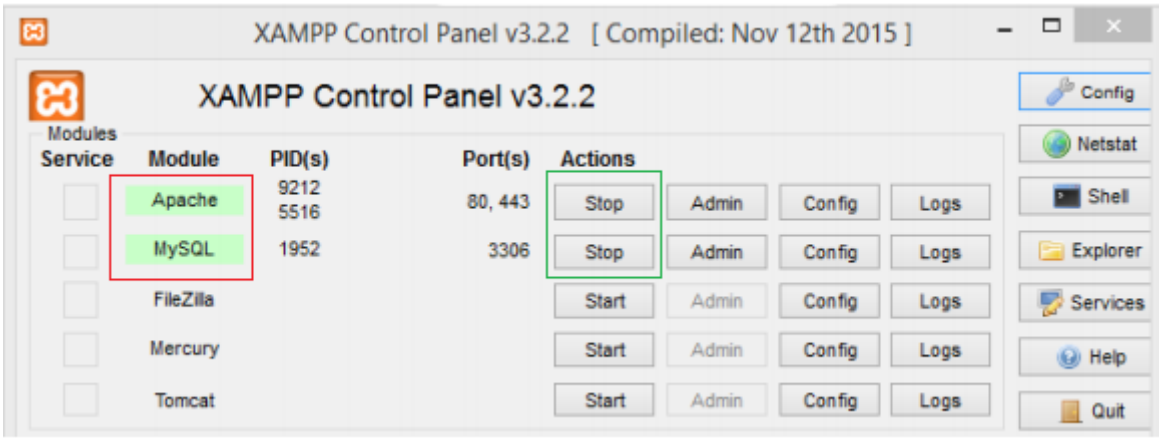


Figura A.1. Panel de control de software XAMPP

A.1.2. Coneccion del kinect

La conexión del Kinect al ordenador se hace por medio del adaptador Kinect/Windows, este a su vez va conectado a la toma de corriente (Figura A. 2).

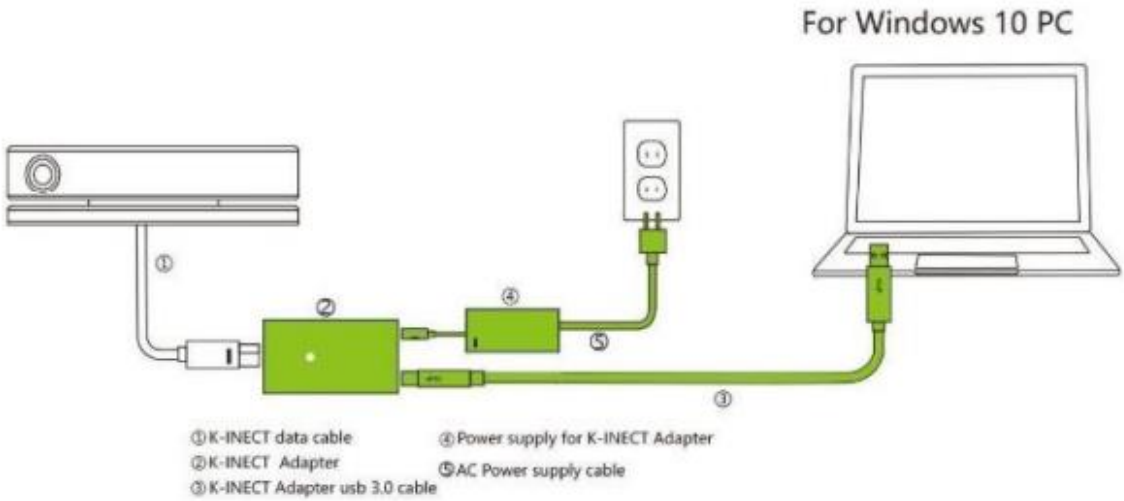


Figura A. 2. Montaje y conexión de la kinect con el ordenador []

Una vez realizada la conexión tal y como se muestra en la figura A. 2, se procede a abrir el software Microsoft Kinect for Windows SDK y dar click sobre el botón que se indica en la Figura A.3 para iniciar el sensor Kinect.

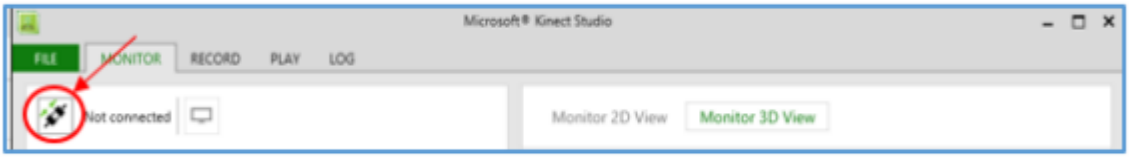


Figura A.3. Puesta en marcha del sensor Kinect

A.1.3. Instalación del material

Para el montaje y la instalación de las gafas HTC Vive se tienen que seguir las instrucciones de uso de estas. en cuanto a la conexión con la aplicación esta se hará a través del software “SteamVR”, este se abrirá de forma automática y estará listo para el uso cuando todos los recuadros se hayan puesto verdes (figura A.3)

Figura A.4. Software Steam VR listo para el uso

A.2. Acceder a la aplicación

Para acceder a la aplicación, se buscará en el escritorio del computador el Icono de Unity con el nombre de la aplicación "TFG_FINAL" al dar clic sobre este se abrira la aplicacion en la primera escena en donde se realizarán los inicios de sesión ya sea del fisioterapeuta o del paciente (figura A.5)

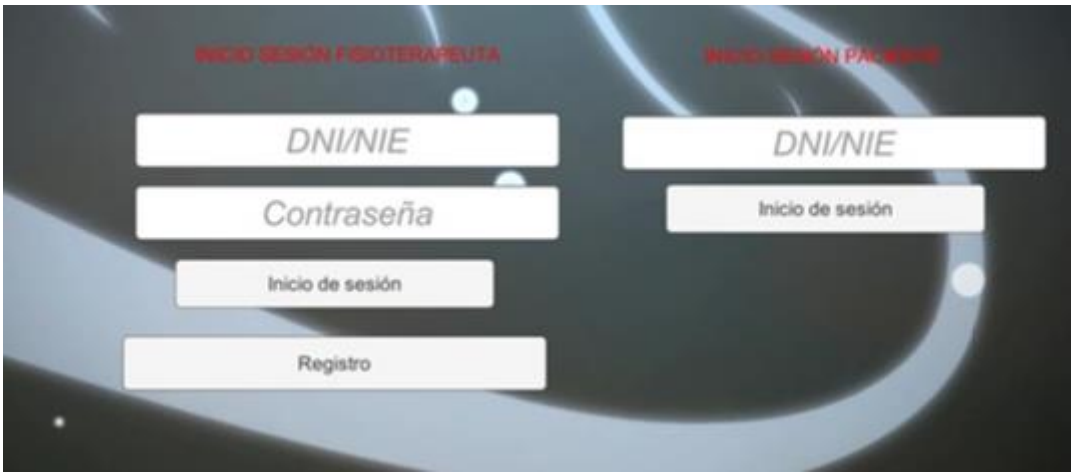


Figura A.5. Inicio de sesión

Para lograr un inicio de sesión correcto es necesario haberse registrado anteriormente completando los recuadros que aparecen al dar clic sobre el botón "Registró", una vez el terapeuta tenga una sesion iniciada podrá eliminar o añadir nuevos pacientes ingresando el DNI correspondiente a estos.

Al iniciar sesión ya sea como terapeuta y escoger el paciente o como paciente se podrán observar las opciones de ejercicios a realizar, en este caso el ejercicio a elegir estará ubicado en la parte de juegos, dando clic sobre el ejercicio de desequilibrio. Figura A.6

Figura A.6. Selección del ejercicio de desequilibrio.

Una vez escogido el ejercicio de desequilibrio, se abrirá una ventana en donde se podrá escoger el nivel de realización de la terapia, este nivel lo escogerá el fisioterapeuta teniendo en cuenta la opinión del paciente y el nivel de dificultad que se maneja en cada nivel.(figura A. 7)

Figura A. 7. Selección del ejercicio de desequilibrio.

Como adicional tanto en esta escena como en cada escena de juego es posible modificar la dificultad a cada nivel, regulando la velocidad y el control de movimiento, por otro lado también es posible leer las instrucciones del juego, modificar la calidad de la imagen y regular el volumen de la música de fondo como se ve en la figura A. 8.

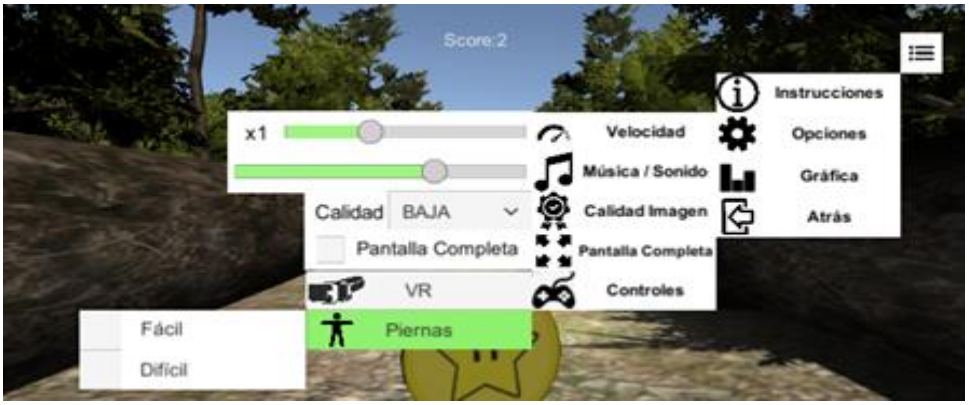


Figura A. 8. Botón de información del juego.

Una vez escogido el nivel el objetivo del juego es el mismo, recoger la mayor cantidad de monedas mientras se esquivan los obstáculos realizando inclinaciones laterales como es indicado en la animación de la figura A. 6.

Para comenzar a jugar el paciente deberá estar ubicado en un radio de 1 metro, totalmente de frente al Kinect en posición de equilibrio, al dar clic sobre cualquiera de estos se abrirá el respectivo nivel mostrando un conteo regresivo desde 5 para dar inicio al juego (figura A. 9).

Figura A. 9. Inicio del segundo nivel de juego.

Una vez iniciado el juego el paciente deberá recoger las monedas esquivando los obstáculos que va encontrando a lo largo de la escena hasta llegar a la línea meta. Mientras el paciente está realizando el ejercicio de rehabilitación el Kinect toma una variedad de datos que al final el juego/terapia, se guardarán en una carpeta del computador llamada seguimiento, esta carpeta contendrá los ficheros con los datos extraídos y cada uno de estos tendrá el DNI de la persona que realizó la terapia, el día y la hora en que fue realizada la terapia. (figura A. 10)

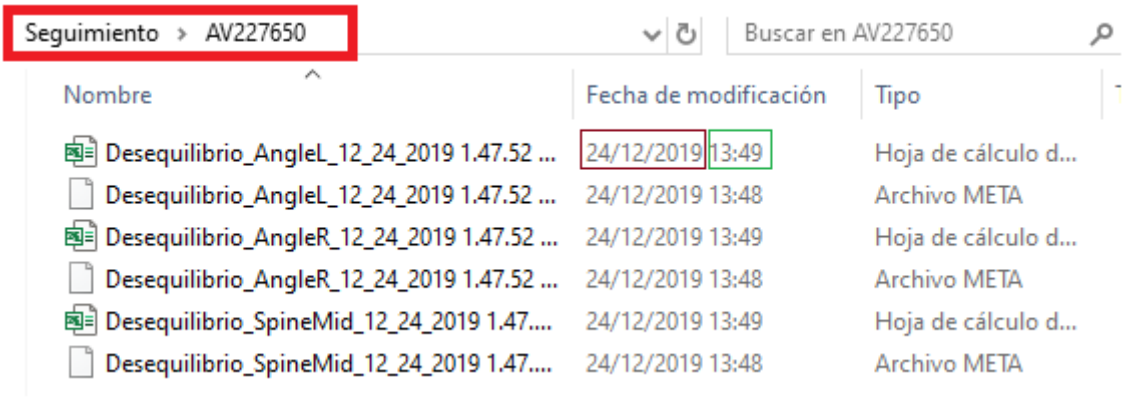


Figura A. 10. Carpeta de seguimiento a los pacientes.

Gracias a esto cada fisioterapeuta podrá tener acceso a las carpetas de sus pacientes y mediante estas podrá evaluar la evolución y progreso de este mediante datos cuantitativos.

Anexo B: *Scripts C# y PHP*

B.1. *BodySourceManager*

```
using UnityEngine;
using System.Collections;
using Windows.Kinect;

public class BodySourceManager : MonoBehaviour
{
    private KinectSensor _Sensor;
    private BodyFrameReader _Reader;
    private Body[] _Data = null;


    public Body[] GetData()
    {
        return _Data;
    }

    public Windows.Kinect.Vector4 FloorPlanes
    {
        get;
        private set;
    }

    void Start ()
    {
        _Sensor = KinectSensor.GetDefault();

        if (_Sensor != null)
        {
            _Reader = _Sensor.BodyFrameSource.OpenReader();

            if (!_Sensor.IsOpen)
            {
                _Sensor.Open();
            }
        }
    }

    void Update ()
    {
        if (_Reader != null)
        {
            var frame = _Reader.AcquireLatestFrame();
            if (frame != null)
            {
                if (_Data == null)
                {
                    _Data = new Body[_Sensor.BodyFrameSource.BodyCount];
                }

                frame.GetAndRefreshBodyData(_Data);

                frame.Dispose();
                frame = null;
            }
        }
    }

    void OnApplicationQuit()
    {
        if (_Reader != null)
        {
            _Reader.Dispose();
```

```
            _Reader = null;
        }

        if (_Sensor != null)
        {
            if (_Sensor.IsOpen)
            {
                _Sensor.Close();
            }

            _Sensor = null;
        }
    }
}
```

B.2. *BodySourceView*

```
using UnityEngine;
using System.Collections;
using Windows.Data;
using System.Collections.Generic;
using Kinect = Windows.Kinect;
using Windows.Kinect;
using System.Linq;
using System.IO;
using System;
using UnityEngine.UI;
using CodeMonkey.Utils;

public class BodySourceView : MonoBehaviour
{
    public Material BoneMaterial;// que pts
    public GameObject BodySourceManager;

    public GameObject _personaje1_Unity;
    public GameObject Master;
    public GameObject Reference;
    public GameObject Hips; // me interesa
    public GameObject LeftUpLeg;
    public GameObject LeftLeg;
    public GameObject RightUpLeg;
    public GameObject RightLeg;
    public GameObject Spine; // CG
    public GameObject Spine1;
    public GameObject Spine2;
    public GameObject LeftShoulder;
    public GameObject LeftArm;
    public GameObject LeftForeArm;
    public GameObject LeftHand;
    public GameObject Neck;
    public GameObject Head;
    public GameObject RightShoulder;
    public GameObject RightArm;
    public GameObject RightForeArm;
    public GameObject RightHand;
    public GameObject LeftFoot;
    public GameObject RightFoot;
    public GameObject RightAnkle;
    public GameObject LeftAnkle;

    public GameObject[] DNI;
    public string nombre;
    public string dataDia;

    public float maxR = 0;
    public float maxL = 0;
    public float MAXR = 0;
    public Text MAXRT =null;
    public float MAXL = 0;
    public Text MAXLT =null;
    public float Px0;
    public float Py0;
    public float Pz0;
    public float Magnitud;
    public List<int> angulocaidaLList;
    public int angulocaidaLInt;
    public List<int> angulocaidaRList;
    public int angulocaidaRInt;
    public float anguloinicialL;
    public float anguloinicialR;
    public float anguloCPL;
    public float anguloCPR;

    private int ii = 0; // contador para punto inicial (equilibrio)
```

```
    public float anguloDeseqL = 90;
    public float anguloDeseqR = 90;
    private Dictionary<ulong, GameObject> _Bodies = new Dictionary<ulong, GameObject>();
    private BodySourceManager _BodyManager;
    public bool DeseqPos = false;
    public bool Desequilibrio = false;

    private Dictionary<Kinect.JointType, Kinect.JointType> _BoneMap = new Dictionary<Kinect.JointType,
Kinect.JointType>()
    {
        { Kinect.JointType.FootLeft, Kinect.JointType.AnkleLeft },
        { Kinect.JointType.AnkleLeft, Kinect.JointType.KneeLeft },
        { Kinect.JointType.KneeLeft, Kinect.JointType.HipLeft },
        { Kinect.JointType.HipLeft, Kinect.JointType.SpineBase },

        { Kinect.JointType.FootRight, Kinect.JointType.AnkleRight },
        { Kinect.JointType.AnkleRight, Kinect.JointType.KneeRight },
        { Kinect.JointType.KneeRight, Kinect.JointType.HipRight },
        { Kinect.JointType.HipRight, Kinect.JointType.SpineBase },

        { Kinect.JointType.HandTipLeft, Kinect.JointType.HandLeft },
        { Kinect.JointType.ThumbLeft, Kinect.JointType.HandLeft },
        { Kinect.JointType.HandLeft, Kinect.JointType.WristLeft },
        { Kinect.JointType.WristLeft, Kinect.JointType.ElbowLeft },
        { Kinect.JointType.ElbowLeft, Kinect.JointType.ShoulderLeft },
        { Kinect.JointType.ShoulderLeft, Kinect.JointType.SpineShoulder },

        { Kinect.JointType.HandTipRight, Kinect.JointType.HandRight },
        { Kinect.JointType.ThumbRight, Kinect.JointType.HandRight },
        { Kinect.JointType.HandRight, Kinect.JointType.WristRight },
        { Kinect.JointType.WristRight, Kinect.JointType.ElbowRight },
        { Kinect.JointType.ElbowRight, Kinect.JointType.ShoulderRight },
        { Kinect.JointType.ShoulderRight, Kinect.JointType.SpineShoulder },

        { Kinect.JointType.SpineBase, Kinect.JointType.SpineMid },
        { Kinect.JointType.SpineMid, Kinect.JointType.SpineShoulder },
        { Kinect.JointType.SpineShoulder, Kinect.JointType.Neck },
        { Kinect.JointType.Neck, Kinect.JointType.Head },

    };
public void Start()
    {

        _personaje1_Unity = GameObject.Find("personaje1_Unity");
        Master = GameObject.Find("master");
        Reference = GameObject.Find("Reference");
        Hips = GameObject.Find("Hips");
        LeftUpLeg = GameObject.Find("LeftUpLeg");
        LeftLeg = GameObject.Find("LeftLeg");
        RightUpLeg = GameObject.Find("RightUpLeg");
        RightLeg = GameObject.Find("RightLeg");
        Spine = GameObject.Find("Spine");
        Spine1 = GameObject.Find("Spine1");
        Spine2 = GameObject.Find("Spine2");
        LeftShoulder = GameObject.Find("LeftShoulder");
        LeftArm = GameObject.Find("LeftArm");
        LeftForeArm = GameObject.Find("LeftForeArm");
        LeftHand = GameObject.Find("LeftHand");
        Neck = GameObject.Find("Neck");
        Head = GameObject.Find("Head");
        RightShoulder = GameObject.Find("RightShoulder");
        RightArm = GameObject.Find("RightArm");
        RightForeArm = GameObject.Find("RightForeArm");
        RightHand = GameObject.Find("RightHand");
        RightAnkle = GameObject.Find("RightToeBase");
        LeftAnkle = GameObject.Find("LeftToeBase");
        RightFoot = GameObject.Find("RightFoot");
        LeftFoot = GameObject.Find("LeftFoot");
```

```
        #region Nuevo directorio paciente (angulos)
        DNI = GameObject.FindGameObjectsWithTag("DNI");
        nombre = DNI[1].name;

        dataDia = System.DateTime.Now.ToString();
        dataDia = dataDia.Replace('/', '_');
        dataDia = dataDia.Replace(':', '.');

        if (!Directory.Exists(Application.dataPath + "/Seguimiento/" + nombre))
        {
            Directory.CreateDirectory(Application.dataPath + "/Seguimiento/" + nombre);
        }

        if (!Directory.Exists(Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia))
        {
            Directory.CreateDirectory((Application.dataPath + "/Seguimiento/" + nombre + "/" +
dataDia));
        }
        #endregion

    }
#region Temporizador Desequilibrio
    IEnumerator TemporizadorDesequilibrio(float angulocaidaR, float angulocaidaL)
    {

        DeseqPos = false;
        yield return new WaitForSeconds(0.5f);//Mira angulos FLex Cada segundo
        MAXDES(angulocaidaR, angulocaidaL);
        DeseqPos = true;

    }
    #endregion

    public void MAXDES(float angulocaidaR, float angulocaidaL)
    {

        if (angulocaidaR > maxR)
        {
            maxR = angulocaidaR;


        }
        if (angulocaidaL > maxL)
        {
            maxL = angulocaidaL;
        }

        MAXR = maxR;
        Debug.Log("Angulo maximo R:" + MAXR);
        MAXRT.text = ("" + MAXR);
        MAXL = maxL;
        Debug.Log("Angulo maximo L:" + MAXL);
        MAXLT.text = ("" + MAXL);

    }

    void Update()
    {

        if (BodySourceManager == null)
        {
            return;
        }

        _BodyManager = BodySourceManager.GetComponent<BodySourceManager>();
        if (_BodyManager == null)
        {
            return;
```

```
        }

        Kinect.Body[] data = _BodyManager.GetData();
        if (data == null)
        {
            return;
        }

        var floorPlane = _BodyManager.FloorPlanes;
        var newrotation = Quaternion.FromToRotation(new Vector3(floorPlane.X, floorPlane.Y,
floorPlane.Z), Vector3.up);


        List<ulong> trackedIds = new List<ulong>();
        foreach (var body in data)
        {
            if (body == null)
            {
                continue;
            }

            if (body.IsTracked)
            {
                trackedIds.Add(body.TrackingId);
            }
        }

        List<ulong> knownIds = new List<ulong>(_Bodies.Keys);

        // First delete untracked bodies
        foreach (ulong trackingId in knownIds)
        {
            if (!trackedIds.Contains(trackingId))
            {

                Destroy(_Bodies[trackingId]);
                _Bodies.Remove(trackingId);
            }
        }

        foreach (var body in data)
        {
            if (body == null)
            {
                continue;
            }

            if (body.IsTracked)
            {
                if (!_Bodies.ContainsKey(body.TrackingId))
                {
                    _Bodies[body.TrackingId] = CreateBodyObject(body.TrackingId);
                }

                RefreshBodyObject(body, _Bodies[body.TrackingId]);

                Quaternion SpineBase =
body.JointOrientations[JointType.SpineBase].Orientation.ChangeQuat(newrotation);
                Quaternion SpineMid =
body.JointOrientations[JointType.SpineMid].Orientation.ChangeQuat(newrotation);
                Quaternion SpineShoulder =
body.JointOrientations[JointType.SpineShoulder].Orientation.ChangeQuat(newrotation);
                Quaternion ShoulderLeft =
body.JointOrientations[JointType.ShoulderLeft].Orientation.ChangeQuat(newrotation);
                Quaternion ShoulderRight =
body.JointOrientations[JointType.ShoulderRight].Orientation.ChangeQuat(newrotation);
                Quaternion ElbowLeft =
body.JointOrientations[JointType.ElbowLeft].Orientation.ChangeQuat(newrotation);
                Quaternion WristLeft =
body.JointOrientations[JointType.WristLeft].Orientation.ChangeQuat(newrotation);
```

```
                Quaternion HandLeft =
body.JointOrientations[JointType.HandLeft].Orientation.ChangeQuat(newrotation);
                Quaternion ElbowRight =
body.JointOrientations[JointType.ElbowRight].Orientation.ChangeQuat(newrotation);
                Quaternion WristRight =
body.JointOrientations[JointType.WristRight].Orientation.ChangeQuat(newrotation);
                Quaternion HandRight =
body.JointOrientations[JointType.HandRight].Orientation.ChangeQuat(newrotation);
                Quaternion KneeLeft =
body.JointOrientations[JointType.KneeLeft].Orientation.ChangeQuat(newrotation);
                Quaternion AnkleLeft =
body.JointOrientations[JointType.AnkleLeft].Orientation.ChangeQuat(newrotation);
                Quaternion KneeRight =
body.JointOrientations[JointType.KneeRight].Orientation.ChangeQuat(newrotation);
                Quaternion AnkleRight =
body.JointOrientations[JointType.AnkleRight].Orientation.ChangeQuat(newrotation);
                Quaternion HipRight =
body.JointOrientations[JointType.HipRight].Orientation.ChangeQuat(newrotation); // se cambiaron leg for
hip
                Quaternion HipLeft =
body.JointOrientations[JointType.HipLeft].Orientation.ChangeQuat(newrotation); //
                Quaternion FootRight =
body.JointOrientations[JointType.FootRight].Orientation.ChangeQuat(newrotation); //
                Quaternion FootLeft =
body.JointOrientations[JointType.FootLeft].Orientation.ChangeQuat(newrotation); //
FootRight.y = 1;
                FootRight.w = 0;
                FootLeft.y = 1;
                FootLeft.w = 0;
                Spine1.transform.rotation = SpineMid * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
                RightArm.transform.rotation = ElbowRight * Quaternion.AngleAxis(-90, new Vector3(0, 0,
1));
                RightForeArm.transform.rotation = WristRight * Quaternion.AngleAxis(-90, new Vector3(0,
0, 1));
                RightHand.transform.rotation = HandRight * Quaternion.AngleAxis(-90, new Vector3(0, 0,
1));
                LeftArm.transform.rotation = ElbowLeft * Quaternion.AngleAxis(180, new Vector3(0, 1, 0))
*
                    Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
                LeftForeArm.transform.rotation = WristLeft * Quaternion.AngleAxis(180, new Vector3(0, 1,
0)) *
                    Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
                RightFoot.transform.rotation = FootRight * Quaternion.AngleAxis(0, new Vector3(0, 0, 1))
* Quaternion.AngleAxis(-90, new Vector3(0, 1, 0)); // ojo -90

                LeftFoot.transform.rotation = FootLeft * Quaternion.AngleAxis(180, new Vector3(1, 0, 0))
* Quaternion.AngleAxis(-90, new Vector3(0, 1, 0));//
                LeftHand.transform.rotation = HandLeft * Quaternion.AngleAxis(180, new Vector3(0, 1, 0))
* Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
                RightUpLeg.transform.rotation = KneeRight * Quaternion.AngleAxis(90, new Vector3(0, 1,
0)) * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
                RightLeg.transform.rotation = AnkleRight * Quaternion.AngleAxis(90, new Vector3(0, 1,
0)) * Quaternion.AngleAxis(-90, new Vector3(0, 0, 1));
                LeftUpLeg.transform.rotation = KneeLeft * Quaternion.AngleAxis(90, new Vector3(0, 1, 0))
* Quaternion.AngleAxis(90, new Vector3(0, 0, 1));
                LeftLeg.transform.rotation = AnkleLeft * Quaternion.AngleAxis(90, new Vector3(0, 1, 0))
* Quaternion.AngleAxis(90, new Vector3(0, 0, 1));

                var moveposition = body.Joints[JointType.SpineMid].Position;
                Master.transform.position = new Vector3(moveposition.X, moveposition.Y, -
moveposition.Z);



                ////////////////superior

                // ::::::::::::::::::::   HORIZONTAL ARMS EXERCISE   :::::::::::::::::::: //

                if (Input.GetKeyDown(KeyCode.Q))
                {
```

```
                    Debug.Log("Horizontal Exercise selected");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", " ::::::   HORIZONTAL
EXERCISE   :::::: ");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", Environment.NewLine);

                }

                // ::::::::::::::::::::   TOUCH SHOULDERS EXERCISE   :::::::::::::::::::: //

                if (Input.GetKeyDown(KeyCode.W))
                {
                    Debug.Log("Touch Shoulders Exercise selected");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", " ::::::   TOUCH SHOULDERS
EXERCISE   :::::: ");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", Environment.NewLine);

                }

                // ::::::::::::::::::::   ARMS UP EXERCISE   :::::::::::::::::::: //

                if (Input.GetKeyDown(KeyCode.E))
                {
                    Debug.Log("Arms Up Exercise Selected");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", " ::::::   ARMS UP
EXERCISE   :::::: ");
                    File.AppendAllText(@"C:\Users\Aleix\Desktop\Results.txt", Environment.NewLine);
                }


                #region Desequilibrio Cálculos
                // ::::::::::::::::::::   DESEQPOS  edito esto  :::::::::::::::::::: //
                if (DeseqPos == true)
                {
                    //variables
                    float x1 = body.Joints[JointType.SpineMid].Position.X;
                    float y1 = body.Joints[JointType.SpineMid].Position.Y;
                    float z1 = body.Joints[JointType.SpineMid].Position.Z;
                    float x0 = body.Joints[JointType.AnkleRight].Position.X;
                    float y0 = body.Joints[JointType.AnkleRight].Position.Y;
                    float z0 = body.Joints[JointType.AnkleRight].Position.Z;
                    float x2 = body.Joints[JointType.AnkleLeft].Position.X;
                    float y2 = body.Joints[JointType.AnkleLeft].Position.Y;
                    float z2 = body.Joints[JointType.AnkleLeft].Position.Z;



                    //planos
                    Vector3 vector1 = new Vector3(x1 - x0, y1 - y0, z1 - z0);
                    Vector3 vector2 = new Vector3(x0 - x0, y1 - y0, z1 - z0);
                    float angulocaidaR = Vector3.Angle(vector2, vector1);
                    Debug.Log("Angulo derecha" + angulocaidaR);
                    anguloCPR = angulocaidaR;
                    Vector3 vector3 = new Vector3(x1 - x2, y1 - y2, z1 - z2); // hipo
                    Vector3 vector4 = new Vector3(x2 - x2, y1 - y2, z1 - z2);
                    float angulocaidaL = Vector3.Angle(vector4, vector3);
                    Debug.Log("Angulo izquierda" + angulocaidaL);
                    anguloCPL = angulocaidaL;

                    StartCoroutine(TemporizadorDesequilibrio(angulocaidaR, angulocaidaL));
                    // Puntos de interes
                    if (ii == 0)
                    {
                        Debug.Log("Proyección SpineMid Inicial = (" + x1 + ";" + y1 + ";" + z0 + ")");
                        Px0 = x1;
                        Py0 = y1;
                        Pz0 = z0;
                        ii = ii + 1;
                    }
                    Debug.Log("Proyección SpineMid = (" + x1 + ";" + y1 + ";" + z0 + ")");
                    Vector3 VectorDesviacion = new Vector3(x1 - Px0, y1 - Py0, z0 - Pz0);
```

```
                    Magnitud = Vector3.Magnitude(VectorDesviacion);
                    Debug.Log("Desviacion = " + Magnitud);

                    #region Creación .xls

                    string path = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia +
"/Desequilibrio_AngleR" + ".xls"; //Creamos un documento de excel (.xls) para guardar los datos
                    string content = angulocaidaR + "\n";   //Contenido del .xls

                    if (!File.Exists(path))
                    {
                        File.AppendAllText(path, "Data Angulo Derecho\nLogin date: " + dataDia + "\n");
                    }
                    File.AppendAllText(path, content);
                    //////////

                    string path2 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia +
"/Desequilibrio_AngleL" + ".xls"; //Creamos un documento de excel (.xls) para guardar los datos
                    string content2 = angulocaidaL + "\n";   //Contenido del .xls

                    if (!File.Exists(path2))
                    {
                        File.WriteAllText(path2, "Data Angulo Izquierdo\nLogin date: " + dataDia +
"\n");
                    }
                    File.AppendAllText(path2, content2);

                    string path3 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia +
"/Desequilibrio_SpineMid" + ".xls"; //Creamos un documento de excel (.xls) para guardar los datos
                    string content3 = x1 + "\t" + y1 + "\t" + z0 + "\n";   //Contenido del .xls

                    if (!File.Exists(path3))
                    {
                        File.AppendAllText(path3, "Data SpineMid\nLogin date: " + dataDia + "\nEje
X\tEje Y\tEje Z\n");
                    }
                    File.AppendAllText(path3, content3);

                    string path4 = Application.dataPath + "/Seguimiento/" + nombre + "/" + dataDia +
"/Desequilibrio_Desviación" + ".xls"; //Creamos un documento de excel (.xls) para guardar los datos
                    string content4 = Magnitud + "\n";   //Contenido del .xls

                    if (!File.Exists(path4))
                    {
                        File.AppendAllText(path4, "Data Desviación\nLogin date: " + dataDia + "\n");
                    }
                    File.AppendAllText(path4, content4);
                    #endregion
                }

                #endregion
            }
        }
    }

    private GameObject CreateBodyObject(ulong id)
    {
        GameObject body = new GameObject("Body:" + id);

        for (Kinect.JointType jt = Kinect.JointType.SpineBase; jt <= Kinect.JointType.ThumbRight; jt++)
        {
            GameObject jointObj = GameObject.CreatePrimitive(PrimitiveType.Cube);

            LineRenderer lr = jointObj.AddComponent<LineRenderer>();
            lr.SetVertexCount(2);
            lr.material = BoneMaterial;
            lr.SetWidth(0.0f, 0.0f);

            jointObj.transform.localScale = new Vector3(0.0f, 0.0f, 0.0f);
```

```
            jointObj.name = jt.ToString();
            jointObj.transform.parent = body.transform;
        }

        return body;


    }

    private void RefreshBodyObject(Kinect.Body body, GameObject bodyObject)
    {
        for (Kinect.JointType jt = Kinect.JointType.SpineBase; jt <= Kinect.JointType.ThumbRight; jt++)
        {
            Kinect.Joint sourceJoint = body.Joints[jt];
            Kinect.Joint? targetJoint = null;

            if (_BoneMap.ContainsKey(jt))
            {
                targetJoint = body.Joints[_BoneMap[jt]];
            }

            Transform jointObj = bodyObject.transform.Find(jt.ToString());
            jointObj.localPosition = GetVector3FromJoint(sourceJoint);

            LineRenderer lr = jointObj.GetComponent<LineRenderer>();
            if (targetJoint.HasValue)
            {
                lr.SetPosition(0, jointObj.localPosition);
                lr.SetPosition(1, GetVector3FromJoint(targetJoint.Value));
                lr.SetColors(GetColorForState(sourceJoint.TrackingState),
GetColorForState(targetJoint.Value.TrackingState));
            }
            else
            {
                lr.enabled = false;
            }
        }
    }

    private static Color GetColorForState(Kinect.TrackingState state)
    {
        switch (state)
        {
            case Kinect.TrackingState.Tracked:
                return Color.green;

            case Kinect.TrackingState.Inferred:
                return Color.red;

            default:
                return Color.black;
        }
    }

    private static Vector3 GetVector3FromJoint(Kinect.Joint joint)
    {

        return new Vector3(joint.Position.X * 10, joint.Position.Y * 10, joint.Position.Z * 10);
    }

}
```

B.3. *ControllerPiernas*

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

public class ControllerPiernas : MonoBehaviour
{
    public BodySourceView bSV;
    public VRLook vr;
    public Flym Fylm;

    public Transform player;
    public bool active = false;
    public bool facil;
    public bool dificil;
    public float angulo0L; //angulo natural del paciente para pierna Izquierda
    public float angulo0R; //angulo natural del paciente para pierna Derecha
    public float anguloR;
    public float anguloL;

    private void Start()
    {
        angulo0L = 11.0f;//bSV.anguloinicialL; //angulo natural del paciente para pierna Izquierda
        angulo0R = 11.0f;//bSV.anguloinicialR;   //angulo natural del paciente para pierna Izquierda
    }

    public void ActivarPiernas()
    {
        active = true;
        Fylm.active = false;
    }
    public void CambioNivelDificil()
    {
        facil = false;
        dificil = true;
    }
    public void CambioNivelFacil()
    {
        facil = true;
        dificil = false;
    }
    public void Facil()
    {
        if (vr.currentTime <= 0 && active == true && facil == true)
        {
            anguloL = bSV.anguloCPL;
            anguloR = bSV.anguloCPR;

            if (0.0f <= anguloL && anguloL < angulo0L && anguloR > angulo0R + 4) //mirar si es necesario
insertar otra condicion para eliminar Bugs
            {
                if (anguloL < 3.0f)
                {
                    anguloL = 3.0f;
                }

                player.transform.position = new Vector3(transform.position.x - 0.05f,
transform.position.y, transform.position.z);
            }

            if (0.0f <= anguloR && anguloR < angulo0R && anguloL > angulo0L + 4) //mirar si es necesario
insertar otra condicion para eliminar Bugs
            {

                if (anguloR < 3.0f)
                {
                    anguloR = 3.0f;
```

```
                }

                player.transform.position = new Vector3(transform.position.x + 0.05f,
transform.position.y, transform.position.z);
            }

            if (angulo0L + 4 >= anguloL && anguloL > angulo0L && angulo0R + 4 >= anguloR && anguloR >
angulo0R)
            {
                player.transform.position = new Vector3(transform.position.x, transform.position.y,
transform.position.z);
            }
        }
    }
    public void Dificil()
    {
        if (vr.currentTime <= 0 && active == true && dificil == true)
        {
            anguloL = bSV.anguloCPL;
            anguloR = bSV.anguloCPR;

            if (0.0f <= anguloL && anguloL < angulo0L && anguloR > angulo0R + 4) //mirar si es necesario
insertar otra condicion para eliminar Bugs
            {
                // al ser el nivel facil vamos a hacer una ecuación directamente proporcional con un
potenciador de 2:1

                if (anguloL < 3.0f)
                {
                    anguloL = 3.0f;
                }

                player.transform.position = new Vector3(player.transform.position.x - (0.05f + 1 /
(anguloL * anguloL)), transform.position.y, transform.position.z);
            }

            if (0.0f <= anguloR && anguloR < angulo0R && anguloL > angulo0L + 4) //mirar si es necesario
insertar otra condicion para eliminar Bugs
            {
                // al ser el nivel facil vamos a hacer una ecuación directamente proporcional con un
potenciador de 2:1

                if (anguloR < 3.0f)
                {
                    anguloR = 3.0f;
                }

                player.transform.position = new Vector3(player.transform.position.x + (0.05f + 1 /
(anguloR * anguloR)), transform.position.y, transform.position.z);
            }

            if (angulo0L + 4 >= anguloL && anguloL >= angulo0L && angulo0R + 4 >= anguloR && anguloR >=
angulo0R)
            {
                player.transform.position = new Vector3(player.transform.position.x,
player.transform.position.y, player.transform.position.z);
            }

        }
    }

    void Update()
    {
        Facil();
        Dificil();
    }

}
```

B.4. *Destroybythecontat*

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

public class Destroybythecontat : MonoBehaviour {

    public GameObject explosion;
    private AudioSource _audioSource;
    void Awake()
    {
        _audioSource = gameObject.GetComponent<AudioSource>();
    }

    // Use this for initialization
    void OnTriggerEnter (Collider other) {
        Instantiate(explosion, transform.position, transform.rotation);
        _audioSource.enabled = true;

    }


}
```

B.5. *EnvioanguloLowerbody*

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

public class EnvioanguloLowerbody : MonoBehaviour
{
    public GameObject[] DNI;
    public string nombre;


    public BodySourceView vS;
    public Trigger_Final tF;
    //public VRLook VR;
    public Text mytext = null;
    public string fecha;
    public float hora;
    public float minutos;
    public string Hora;
    public bool onetime = false;
    public string Movimiento;
    public string Nivel;
    public string MAXLT;
    public string MAXRT;
    public string Desviacion;
    public string Magnitud;
    public string Media;

    // Update is called once per frame
    string AngleInsertDesURL = "localhost/users/AngleInsertDes.php";//Abre el script AngleInsertDes.php

    public void Update()
    {
        if (onetime == false && tF.final == true)
```

```
        {
            //fecha
            System.DateTime dateTime = System.DateTime.Now;
            fecha = dateTime.ToString("dd-MM-yyyy");
            //Debug.Log("HOLA");
            //hora
            hora = System.DateTime.Now.Hour;
            minutos = System.DateTime.Now.Minute;
            Hora = hora + ":" + minutos;
            //DNI

            DNI = GameObject.FindGameObjectsWithTag("DNI");
            nombre = DNI[1].name;
            Debug.Log(nombre);
            Movimiento = GameObject.FindGameObjectWithTag("MovimientoText").ToString();
            Nivel = GameObject.FindGameObjectWithTag("NivelText").ToString();
            MAXLT = vS.MAXLT.text.ToString();
            MAXRT = vS.MAXRT.text.ToString();
            Magnitud = vS.Magnitud.ToString();
            Desviacion = Magnitud;
            StartCoroutine(AngleInsertDes(nombre, fecha, Hora, Movimiento, Nivel, MAXLT, MAXRT,
Desviacion));
            onetime = true;
        }
    }

    IEnumerator AngleInsertDes(string nombre, string fecha, string Hora, string Movimiento, string
Nivel, string MAXRT, string MAXLT, string Desviacion)
    {

        WWWForm form = new WWWForm();

        form.AddField("DNIPost", nombre);
        form.AddField("FechaPost", fecha);
        form.AddField("HoraPost", Hora);
        form.AddField("MovimientoPost", Movimiento);
        form.AddField("MAXRTPost", MAXRT);
        form.AddField("MAXLTPost", MAXLT);
        form.AddField("DesviacionPost", Desviacion);
        form.AddField("NivelPost", Nivel);


        WWW www = new WWW(AngleInsertDesURL, form);
        yield return www;

        Debug.Log(www.text);
        //Mensaje.text = www.text;
    }
}
```

B.6. *FEfect*

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

[ExecuteInEditMode, ImageEffectAllowedInSceneView]
public class FEfect : MonoBehaviour {
    public Material _mat;
    public Color _fogColor;
    public float _depthStart;
    public float _depthDistance;
    // Use this for initialization
    void Start () {
        GetComponent<Camera>().depthTextureMode = DepthTextureMode.Depth;
         }

         // Update is called once per frame
         void Update () {
        _mat.SetColor("_FogColor", _fogColor);
        _mat.SetFloat("_DepthStart", _depthStart);
        _mat.SetFloat("_DepthDistance", _depthDistance);

    }
    private void OnRenderImage(RenderTexture source, RenderTexture destination)
    {
        Graphics.Blit(source, destination, _mat);
    }
}
```

B.7. *Flym*

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

public class Flym : MonoBehaviour {
    public ControllerPiernas cP;
    private float vrROLL;
    public bool active = true;

    public void ActivarVR()
    {
        active = true;
        cP.active = false;
    }
         void Update () {
        float vrROLL = Camera.main.transform.eulerAngles.z;
        if(active == true)
        {
            if (vrROLL >= (10) && vrROLL <= (80))
            {
                transform.Translate(-0.1f, 0f, 0f);
                //checkBOUNDS();
                Debug.Log("leaving left");
            }
            if (vrROLL <= (340) && vrROLL >= (280))
            {
                transform.Translate(0.1f, 0f, 0f);
                //checkBOUNDS();
                Debug.Log("leaving right");
            }
        }
    }

}
```

B.8. *FogShader.shader*

```
Shader "PeerPlay/FogEffect"
{
	Properties
	{
		_MainTex ("Texture", 2D) = "white" {}
		_FogColor("Fog Color", Color) = (1,1,1,1)
		_DepthStart("Depth Start", float) = 1
		_DepthDistance("Depth Distance", float) = 1



	}
		SubShader
	{
		// No culling or depth
		Cull Off ZWrite Off ZTest Always

		Pass
		{
			CGPROGRAM
			#pragma vertex vert
			#pragma fragment frag

			#include "UnityCG.cginc"
			sampler2D _CameraDepthTexture;
			fixed4 _FogColor;
			float _DepthStart,_DepthDistance;


			struct appdata
			{
				float4 vertex : POSITION;
				float2 uv : TEXCOORD0;
			};

			struct v2f
			{
				float2 uv : TEXCOORD0;
				float4 vertex : SV_POSITION;
				float4 scrPos : TEXCOORD1;
			};

			v2f vert (appdata v)
			{
				v2f o;
				o.vertex = UnityObjectToClipPos(v.vertex);
				o.scrPos = ComputeScreenPos(o.vertex);
				o.uv = v.uv;
				return o;
			}

			sampler2D _MainTex;

			fixed4 frag (v2f i) : COLOR
			{
				float depthValue = Linear01Depth(tex2Dproj(_CameraDepthTexture,
UNITY_PROJ_COORD(i.scrPos)).r)*_ProjectionParams.z;
				depthValue = saturate((depthValue - _DepthStart) / _DepthDistance);
				fixed4 fogColor = _FogColor * depthValue;
				fixed4 col = tex2Dproj(_MainTex, i.scrPos);
				return lerp(col,fogColor,depthValue);
			}
			ENDCG
		}
	}
}
```

B.9. *ll*

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

public class ll : MonoBehaviour
{
    public static int PickUpCount;

    void Awake()
    {
        PickUpCount = 0;
    }


    void OnGUI()
    {
        GUI.Label(new Rect((Screen.width / 2.0f), 25, 200, 35), string.Format("Score:{0}",
PickUpCount));
    }
}
```

B.10. *MusicaSlider*

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


public class MusicaSlider : MonoBehaviour
{
    public Text value =null;
    public Slider sL;

    private void Update()
    {
        value.text = sL.value.ToString();
    }


}
```

B.11. *noiseSimplex*

```
#ifndef NOISE_SIMPLEX_FUNC
#define NOISE_SIMPLEX_FUNC
/*

Description:
        Array- and textureless CgFx/HLSL 2D, 3D and 4D simplex noise functions.
        a.k.a. simplified and optimized Perlin noise.

        The functions have very good performance
        and no dependencies on external data.

        2D - Very fast, very compact code.
        3D - Fast, compact code.
        4D - Reasonably fast, reasonably compact code.

------------------------------------------------------------------

Ported by:
        Lex-DRL
        I've ported the code from GLSL to CgFx/HLSL for Unity,
        added a couple more optimisations (to speed it up even further)
        and slightly reformatted the code to make it more readable.

Original GLSL functions:
        https://github.com/ashima/webgl-noise
        Credits from original glsl file are at the end of this cginc.

------------------------------------------------------------------

Usage:

        float ns = snoise(v);
        // v is any of: float2, float3, float4

        Return type is float.
        To generate 2 or more components of noise (colorful noise),
        call these functions several times with different
        constant offsets for the arguments.
        E.g.:

        float3 colorNs = float3(
                snoise(v),
                snoise(v + 17.0),
                snoise(v - 43.0),
        );


Remark about those offsets from the original author:

        People have different opinions on whether these offsets should be integers
        for the classic noise functions to match the spacing of the zeroes,
        so we have left that for you to decide for yourself.
        For most applications, the exact offsets don't really matter as long
        as they are not too small or too close to the noise lattice period
        (289 in this implementation).

*/

// 1 / 289
#define NOISE_SIMPLEX_1_DIV_289 0.00346020761245674740484429065744f

float mod289(float x) {
        return x - floor(x * NOISE_SIMPLEX_1_DIV_289) * 289.0;
}

float2 mod289(float2 x) {
        return x - floor(x * NOISE_SIMPLEX_1_DIV_289) * 289.0;
```

```
}

float3 mod289(float3 x) {
	return x - floor(x * NOISE_SIMPLEX_1_DIV_289) * 289.0;
}

float4 mod289(float4 x) {
	return x - floor(x * NOISE_SIMPLEX_1_DIV_289) * 289.0;
}


// ( x*34.0 + 1.0 )*x =
// x*x*34.0 + x
float permute(float x) {
	return mod289(
		x*x*34.0 + x
	);
}

float3 permute(float3 x) {
	return mod289(
		x*x*34.0 + x
	);
}

float4 permute(float4 x) {
	return mod289(
		x*x*34.0 + x
	);
}



float taylorInvSqrt(float r) {
	return 1.79284291400159 - 0.85373472095314 * r;
}

float4 taylorInvSqrt(float4 r) {
	return 1.79284291400159 - 0.85373472095314 * r;
}



float4 grad4(float j, float4 ip)
{
	const float4 ones = float4(1.0, 1.0, 1.0, -1.0);
	float4 p, s;
	p.xyz = floor( frac(j * ip.xyz) * 7.0) * ip.z - 1.0;
	p.w = 1.5 - dot( abs(p.xyz), ones.xyz );

	// GLSL: lessThan(x, y) = x < y
	// HLSL: 1 - step(y, x) = x < y
	s = float4(
		1 - step(0.0, p)
	);
	p.xyz = p.xyz + (s.xyz * 2 - 1) * s.www;

	return p;
}



// ----------------------------------- 2D -------------------------------------

float snoise(float2 v)
{
	const float4 C = float4(
		0.211324865405187, // (3.0-sqrt(3.0))/6.0
		0.366025403784439, // 0.5*(sqrt(3.0)-1.0)
	 -0.577350269189626, // -1.0 + 2.0 * C.x
```

```
                0.024390243902439  // 1.0 / 41.0
        );

// First corner
        float2 i = floor( v + dot(v, C.yy) );
        float2 x0 = v - i + dot(i, C.xx);

// Other corners
        // float2 i1 = (x0.x > x0.y) ? float2(1.0, 0.0) : float2(0.0, 1.0);
        // Lex-DRL: afaik, step() in GPU is faster than if(), so:
        // step(x, y) = x <= y
        int xLessEqual = step(x0.x, x0.y); // x <= y ?
        int2 i1 =
                int2(1, 0) * (1 - xLessEqual) // x > y
                + int2(0, 1) * xLessEqual // x <= y
        ;
        float4 x12 = x0.xyxy + C.xxzz;
        x12.xy -= i1;

// Permutations
        i = mod289(i); // Avoid truncation effects in permutation
        float3 p = permute(
                permute(
                                i.y + float3(0.0, i1.y, 1.0 )
                ) + i.x + float3(0.0, i1.x, 1.0 )
        );

        float3 m = max(
                0.5 - float3(
                        dot(x0, x0),
                        dot(x12.xy, x12.xy),
                        dot(x12.zw, x12.zw)
                ),
                0.0
        );
        m = m*m ;
        m = m*m ;

// Gradients: 41 points uniformly over a line, mapped onto a diamond.
// The ring size 17*17 = 289 is close to a multiple of 41 (41*7 = 287)

        float3 x = 2.0 * frac(p * C.www) - 1.0;
        float3 h = abs(x) - 0.5;
        float3 ox = floor(x + 0.5);
        float3 a0 = x - ox;

// Normalise gradients implicitly by scaling m
// Approximation of: m *= inversesqrt( a0*a0 + h*h );
        m *= 1.79284291400159 - 0.85373472095314 * ( a0*a0 + h*h );

// Compute final noise value at P
        float3 g;
        g.x = a0.x * x0.x + h.x * x0.y;
        g.yz = a0.yz * x12.xz + h.yz * x12.yw;
        return 130.0 * dot(m, g);
}

// ----------------------------------- 3D -------------------------------------

float snoise(float3 v)
{
        const float2 C = float2(
                0.166666666666666667, // 1/6
                0.333333333333333333  // 1/3
        );
        const float4 D = float4(0.0, 0.5, 1.0, 2.0);

// First corner
        float3 i = floor( v + dot(v, C.yyy) );
        float3 x0 = v - i + dot(i, C.xxx);
```

```
// Other corners
        float3 g = step(x0.yzx, x0.xyz);
        float3 l = 1 - g;
        float3 i1 = min(g.xyz, l.zxy);
        float3 i2 = max(g.xyz, l.zxy);

        float3 x1 = x0 - i1 + C.xxx;
        float3 x2 = x0 - i2 + C.yyy; // 2.0*C.x = 1/3 = C.y
        float3 x3 = x0 - D.yyy;      // -1.0+3.0*C.x = -0.5 = -D.y

// Permutations
        i = mod289(i);
        float4 p = permute(
                permute(
                        permute(
                                        i.z + float4(0.0, i1.z, i2.z, 1.0 )
                        ) + i.y + float4(0.0, i1.y, i2.y, 1.0 )
                )       + i.x + float4(0.0, i1.x, i2.x, 1.0 )
        );

// Gradients: 7x7 points over a square, mapped onto an octahedron.
// The ring size 17*17 = 289 is close to a multiple of 49 (49*6 = 294)
        float n_ = 0.142857142857; // 1/7
        float3 ns = n_ * D.wyz - D.xzx;

        float4 j = p - 49.0 * floor(p * ns.z * ns.z); // mod(p,7*7)

        float4 x_ = floor(j * ns.z);
        float4 y_ = floor(j - 7.0 * x_ ); // mod(j,N)

        float4 x = x_ *ns.x + ns.yyyy;
        float4 y = y_ *ns.x + ns.yyyy;
        float4 h = 1.0 - abs(x) - abs(y);

        float4 b0 = float4( x.xy, y.xy );
        float4 b1 = float4( x.zw, y.zw );

        //float4 s0 = float4(lessThan(b0,0.0))*2.0 - 1.0;
        //float4 s1 = float4(lessThan(b1,0.0))*2.0 - 1.0;
        float4 s0 = floor(b0)*2.0 + 1.0;
        float4 s1 = floor(b1)*2.0 + 1.0;
        float4 sh = -step(h, 0.0);

        float4 a0 = b0.xzyw + s0.xzyw*sh.xxyy ;
        float4 a1 = b1.xzyw + s1.xzyw*sh.zzww ;

        float3 p0 = float3(a0.xy,h.x);
        float3 p1 = float3(a0.zw,h.y);
        float3 p2 = float3(a1.xy,h.z);
        float3 p3 = float3(a1.zw,h.w);

//Normalise gradients
        float4 norm = taylorInvSqrt(float4(
                dot(p0, p0),
                dot(p1, p1),
                dot(p2, p2),
                dot(p3, p3)
        ));
        p0 *= norm.x;
        p1 *= norm.y;
        p2 *= norm.z;
        p3 *= norm.w;

// Mix final noise value
        float4 m = max(
                0.6 - float4(
                        dot(x0, x0),
                        dot(x1, x1),
                        dot(x2, x2),
```

```
                        dot(x3, x3)
                ),
                0.0
        );
        m = m * m;
        return 42.0 * dot(
                m*m,
                float4(
                        dot(p0, x0),
                        dot(p1, x1),
                        dot(p2, x2),
                        dot(p3, x3)
                )
        );
}

// ---------------------------------- 4D -----------------------------------

float snoise(float4 v)
{
        const float4 C = float4(
                0.138196601125011, // (5 - sqrt(5))/20 G4
                0.276393202250021, // 2 * G4
                0.414589803375032, // 3 * G4
         -0.447213595499958  // -1 + 4 * G4
        );

// First corner
        float4 i = floor(
                v +
                dot(
                        v,
                        0.309016994374947451 // (sqrt(5) - 1) / 4
                )
        );
        float4 x0 = v - i + dot(i, C.xxxx);

// Other corners

// Rank sorting originally contributed by Bill Licea-Kane, AMD (formerly ATI)
        float4 i0;
        float3 isX = step( x0.yzw, x0.xxx );
        float3 isYZ = step( x0.zww, x0.yyz );
        i0.x = isX.x + isX.y + isX.z;
        i0.yzw = 1.0 - isX;
        i0.y += isYZ.x + isYZ.y;
        i0.zw += 1.0 - isYZ.xy;
        i0.z += isYZ.z;
        i0.w += 1.0 - isYZ.z;

        // i0 now contains the unique values 0,1,2,3 in each channel
        float4 i3 = saturate(i0);
        float4 i2 = saturate(i0-1.0);
        float4 i1 = saturate(i0-2.0);

        //      x0 = x0 - 0.0 + 0.0 * C.xxxx
        //      x1 = x0 - i1  + 1.0 * C.xxxx
        //      x2 = x0 - i2  + 2.0 * C.xxxx
        //      x3 = x0 - i3  + 3.0 * C.xxxx
        //      x4 = x0 - 1.0 + 4.0 * C.xxxx
        float4 x1 = x0 - i1 + C.xxxx;
        float4 x2 = x0 - i2 + C.yyyy;
        float4 x3 = x0 - i3 + C.zzzz;
        float4 x4 = x0 + C.wwww;

// Permutations
        i = mod289(i);
        float j0 = permute(
                permute(
                        permute(
```

```
                                permute(i.w) + i.z
                        ) + i.y
                ) + i.x
        );
        float4 j1 = permute(
                permute(
                        permute(
                                permute (
                                        i.w + float4(i1.w, i2.w, i3.w, 1.0 )
                                ) + i.z + float4(i1.z, i2.z, i3.z, 1.0 )
                        ) + i.y + float4(i1.y, i2.y, i3.y, 1.0 )
                ) + i.x + float4(i1.x, i2.x, i3.x, 1.0 )
        );

// Gradients: 7x7x6 points over a cube, mapped onto a 4-cross polytope
// 7*7*6 = 294, which is close to the ring size 17*17 = 289.
        const float4 ip = float4(
                0.00340136054421768707 5, // 1/294
                0.020408163265306122449, // 1/49
                0.142857142857142857143, // 1/7
                0.0
        );

        float4 p0 = grad4(j0, ip);
        float4 p1 = grad4(j1.x, ip);
        float4 p2 = grad4(j1.y, ip);
        float4 p3 = grad4(j1.z, ip);
        float4 p4 = grad4(j1.w, ip);

// Normalise gradients
        float4 norm = taylorInvSqrt(float4(
                dot(p0, p0),
                dot(p1, p1),
                dot(p2, p2),
                dot(p3, p3)
        ));
        p0 *= norm.x;
        p1 *= norm.y;
        p2 *= norm.z;
        p3 *= norm.w;
        p4 *= taylorInvSqrt( dot(p4, p4) );

// Mix contributions from the five corners
        float3 m0 = max(
                0.6 - float3(
                        dot(x0, x0),
                        dot(x1, x1),
                        dot(x2, x2)
                ),
                0.0
        );
        float2 m1 = max(
                0.6 - float2(
                        dot(x3, x3),
                        dot(x4, x4)
                ),
                0.0
        );
        m0 = m0 * m0;
        m1 = m1 * m1;

        return 49.0 * (
                dot(
                        m0*m0,
                        float3(
                                dot(p0, x0),
                                dot(p1, x1),
                                dot(p2, x2)
                        )
                ) + dot(
```

```
                        m1*m1,
                        float2(
                                dot(p3, x3),
                                dot(p4, x4)
                        )
                )
        );

}
```

B.12. *recoger*

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

public class recoger : MonoBehaviour {

    public int CoinValue;
    public GameObject pickupparticleefect;

    private bool _triggered;
    private AudioSource _audioSource;

    void Awake()
    {
        _audioSource = gameObject.GetComponent<AudioSource>();
    }

    // Update is called once per frame
    void Update()
    {
        if (_triggered && !_audioSource.isPlaying)
            Destroy(gameObject);
    }

    void OnTriggerEnter()
    {
        _triggered = true;
        _audioSource.enabled = true;

        ll.PickUpCount += CoinValue;
        gameObject.GetComponent<MeshRenderer>().enabled = false;
    }
}
```

B.13. *rotarcoin*

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

public class rotarcoin : MonoBehaviour
{
    public float spinForce;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        transform.Rotate(0, 0, spinForce * Time.deltaTime);
    }

}
```

B.14. *Settings*

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

public class Settings : MonoBehaviour
{
    public AudioMixer audioMixer;

    public void SetVolume (float volume)
    {
        audioMixer.SetFloat("Volumen", volume);
    }

    public void SetQuality(int qualityIndex)
    {
        QualitySettings.SetQualityLevel(qualityIndex);
    }

    public void FullScreen (bool isFullScreen)
    {
        Screen.fullScreen= isFullScreen;
    }
}
```

B.15. *SonidoAro*

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

public class SonidoAro : MonoBehaviour {
    private AudioSource _audioSource;
    private void Awake()
    {
        _audioSource = gameObject.GetComponent<AudioSource>();
    }
    void OnTriggerEnter(Collider coll)
    {
        Debug.Log("Entro al Trigger");
        _audioSource.enabled = true;
    }

    }
```

B.16. *Timer*

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

public class Timer : MonoBehaviour {
    public Text Contador_1;
    public float Tiempo = 10f;
    private bool started = false;
        // Use this for initialization
        void Start () {
        Contador_1.text = "" + Tiempo;
        }

        // Update is called once per frame
        void Update () {
        if(started && Tiempo > 0) {
            Tiempo -= Time.deltaTime;
            Contador_1.text = "" + Tiempo.ToString("f0");
        }

        //Debug.Log("Timer: update");
        }

    public void ButtonClick()
    {
        StartCoroutine(waitAndCount());

    }

    IEnumerator waitAndCount()
    {
        yield return new WaitForSeconds(7);
        started = true;

    }
}
```

B.17. *Timer*

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

public class Timer : MonoBehaviour {
    public Text Contador_1;
    public float Tiempo = 10f;
    private bool started = false;
        // Use this for initialization
        void Start () {
        Contador_1.text = "" + Tiempo;
        }

        // Update is called once per frame
        void Update () {
        if(started && Tiempo > 0) {
            Tiempo -= Time.deltaTime;
            Contador_1.text = "" + Tiempo.ToString("f0");
        }

        //Debug.Log("Timer: update");
        }

    public void ButtonClick()
    {
        StartCoroutine(waitAndCount());

    }

    IEnumerator waitAndCount()
    {
        yield return new WaitForSeconds(7);
        started = true;

    }
}
```

B.18. *Trigger_Final*

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

public class Trigger_Final : MonoBehaviour {


    public bool final = false;

    void OnTriggerEnter()  //Collider other
    {
        {
            final = true;
            if (final == true)
            {
                //Debug.Log(gameObject.name + " was triggered by " + other.gameObject.name);
                Debug.Log("Has terminado");
            }
        }
    }
}
```

B.19. *UnderWaterEfect*

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

[ExecuteInEditMode, ImageEffectAllowedInSceneView]
public class UnderWaterEfect : MonoBehaviour {

    public Material _mat;

    [Range(0.001f,0.1f)]
    public float _pixelOffset;
    [Range(0.1f,20f)]
    public float _noiseScale;
    [Range(0.1f, 20f)]
    public float _noiseFrequency;
    [Range(0.1f, 30f)]
    public float _noiseSpeed;
    public float _depthStart;
    public float _depthDistance;

    // Use this for initialization
    void Start () {

        }

        // Update is called once per frame
        void Update () {
        _mat.SetFloat(" _NoiseFrequency", _noiseFrequency);
        _mat.SetFloat(" _NoiseSpeed", _noiseSpeed);
        _mat.SetFloat(" _NoiseScale", _noiseScale);
        _mat.SetFloat(" _PixelOffset", _pixelOffset);
        _mat.SetFloat("_DepthStart", _depthStart);
        _mat.SetFloat("_DepthDistance", _depthDistance);
    }
    private void OnRenderImage(RenderTexture source, RenderTexture destination)
    {
        Graphics.Blit(source, destination, _mat);
    }
}
```

B.20. *UnderWaterEfect.shader*

```
Shader "PeerPlay/NewImageEffectShader"
{
	Properties
	{
		_MainTex ("Texture", 2D) = "white" {}
		_NoiseScale(" Noise Scale", float) = 1
		_NoiseFrequency(" Noise Frequency", float) = 1
		_NoiseSpeed(" Noise Speed", float) = 1
		_PixelOffset("Pixel Offset", float) =0.005
		_DepthStart("Depth Start", float) = 1
		_DepthDistance("Depth Distance", float) = 1
	}
		SubShader
		{
			// No culling or depth
			Cull Off ZWrite Off ZTest Always

			Pass
			{
				CGPROGRAM
				#pragma vertex vert
				#pragma fragment frag

				#include "UnityCG.cginc"
				#include "noiseSimplex.cginc"
				#define M_PI 3.14159265358979323846264338327795

				uniform float _NoiseFrequency, _NoiseScale, _NoiseSpeed, _PixelOffset;
				float _DepthStart, _DepthDistance;
				sampler2D _CameraDepthTexture;

			struct appdata
			{
				float4 vertex : POSITION;
				float2 uv : TEXCOORD0;
			};

			struct v2f
			{
				float2 uv : TEXCOORD0;
				float4 vertex : SV_POSITION;
				float4 scrPos : TEXCOORD1;
			};

			v2f vert (appdata v)
			{
				v2f o;
				o.vertex = UnityObjectToClipPos(v.vertex);
				o.scrPos = ComputeScreenPos(o.vertex);
				o.uv = v.uv;
				return o;
			}

			sampler2D _MainTex;

			fixed4 frag (v2f i) : COLOR
			{
				float depthValue = Linear01Depth(tex2Dproj(_CameraDepthTexture,
UNITY_PROJ_COORD(i.scrPos)).r)*_ProjectionParams.z;
				depthValue = 1- saturate((depthValue - _DepthStart) / _DepthDistance);

				float3 spos = float3(i.scrPos.x, i.scrPos.y,0)* _NoiseFrequency;
				spos.z += _Time.x * _NoiseSpeed;
				float noise = _NoiseScale * ((snoise(spos) + 1) / 2);
				float4 noiseToDirection = float4(cos(noise* M_PI * 2), sin(noise*M_PI
* 2), 0, 0);
```

```
                                fixed4 col = tex2Dproj(_MainTex, i.scrPos +
(normalize(noiseToDirection)* _PixelOffset* depthValue));
                                return col;
                            }
                            ENDCG
                        }
                    }
}
```

B.21. *VelocidadSlider*

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


public class VelocidadSlider : MonoBehaviour
{
    public Text value;
    public Slider sL;

    private void Update()
    {
        value.text = "x"+sL.value.ToString();
    }



}
```

B.22. *VRLook*

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

public class VRLook : MonoBehaviour
{

    public Transform vrCamera;
    public float toggleAngle = 30.0f;
    public float speed = 3.0f;
    public bool moveForward;
    public VelocidadSlider multspeed;

    public float currentTime = 0f;
    float startingTime = 5.4f;
    [SerializeField] Text Countdown;


    private CharacterController cc;

    // Use this for initialization
    void Start()
    {
        currentTime = startingTime;

        cc = GetComponent<CharacterController>();
    }

    // Update is called once per frame
    void Update()
    {
        currentTime -= 1 * Time.deltaTime;
        Countdown.text = currentTime.ToString("0");

        if (currentTime <= 0)
        {
            currentTime = 0;
        }


        if (currentTime <= 0)
        {
            Destroy(Countdown);
        }

        if (vrCamera.eulerAngles.x >= toggleAngle && vrCamera.eulerAngles.x < 90.0f && currentTime <= 0)
        {
            moveForward = true;


        }
        else
        {
            moveForward = false;
        }

        if (moveForward)
        {
            Vector3 forward = vrCamera.TransformDirection(Vector3.forward);

            cc.SimpleMove(forward * speed * multspeed.sL.value);
        }
    }
}
```

B.23. *Window_Graph*

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

public class Window_Graph : MonoBehaviour {

    private static Window_Graph instance;
    public BodySourceView vS;

    [SerializeField] private Sprite dotSprite;
    private RectTransform graphContainer;
    private RectTransform labelTemplateX;
    private RectTransform labelTemplateY;
    private RectTransform dashTemplateX;
    private RectTransform dashTemplateY;
    private List<GameObject> gameObjectList;
    private List<IGraphVisualObject> graphVisualObjectList;
    private GameObject TagGameObject;
    private List<RectTransform> yLabelList;

    // Cached values
    private List<int> valueList;
    private IGraphVisual graphVisual;
    private int maxVisibleValueAmount;
    private Func<int, string> getAxisLabelX;
    private Func<float, string> getAxisLabelY;
    private float xSize;
    private bool startYScaleAtZero;


    private void Awake() {
        instance = this;
        // Grab base objects references
        graphContainer = transform.Find("graphContainer").GetComponent<RectTransform>();
        labelTemplateX = graphContainer.Find("labelTemplateX").GetComponent<RectTransform>();
        labelTemplateY = graphContainer.Find("labelTemplateY").GetComponent<RectTransform>();
        //graphContainer = graphContainer.Find("graphContainer").GetComponent<RectTransform>();
        dashTemplateX = graphContainer.Find("dashTemplateX").GetComponent<RectTransform>();
        dashTemplateY = graphContainer.Find("dashTemplateY").GetComponent<RectTransform>();
        TagGameObject = graphContainer.Find("Tag").gameObject;

        startYScaleAtZero = true;
        gameObjectList = new List<GameObject>();
        yLabelList = new List<RectTransform>();
        graphVisualObjectList = new List<IGraphVisualObject>();

        IGraphVisual lineGraphVisual = new LineGraphVisual(graphContainer, dotSprite, Color.green, new
Color(1, 1, 1, .5f));
        IGraphVisual barChartVisual = new BarChartVisual(graphContainer, Color.white, .8f);

        // Set up buttons
        transform.Find("BarChartButton").GetComponent<Button_UI>().ClickFunc = () => {
            SetGraphVisual(barChartVisual);
        };
        transform.Find("LineChartButton").GetComponent<Button_UI>().ClickFunc = () => {
            SetGraphVisual(lineGraphVisual);
        };

        transform.Find("Decrease").GetComponent<Button_UI>().ClickFunc = () => {
            DecreaseVisibleAmount();
        };
        transform.Find("Increase").GetComponent<Button_UI>().ClickFunc = () => {
            IncreaseVisibleAmount();
        };
```

```
        HideTag();

        List<int> valueList = new List<int> { 18, 13, 12, 15, 12, 13, 13, 13, 12, 12, 12, 12, 12, 12,
12, 12, 12, 12, 12, 12, 13, 25, 27, 17, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 5,
0, 5, 13, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 10, 10, 10, 12, 11, 11, 11, 12,
12, 12, 12, 12, 12, 12, 12, 12, 12, 11, 18, 25, 18, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12,
12, 12, 12, 11, 16, 24, 16, 10, 11, 11, 12, 12, 12, 12, 12, 12, 11, 11, 11, 12, 12, 12, 12, 12, 11,
8, 3, 4, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 10, 11, 11, 11, 11, 11, 9, 1, 1, 7, 12, 11, 11, 11,
11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 11, 16, 24, 21, 12, 12, 11, 12, 12, 12, 12, 12, 12, 12, 12,
12, 11, 11, 12, 12, 12, 12, 12, 12, 12, 11, 11, 11, 11, 11, 12, 12, 12, 12 };
        ShowGraph(valueList, barChartVisual, -1, (int _i) => "Sessión " + (_i + 1), (float _f) =>
Mathf.RoundToInt(_f) + "º");
public void Update()
    {
        valueList.Add(vS.angulocaidaLInt);
        foreach (int element in valueList)
        {
            Debug.Log(element);
        }
    }

    public static void ShowTag_Static(string TagText, Vector2 anchoredPosition) {
        instance.ShowTag(TagText, anchoredPosition);
    }

    private void ShowTag(string TagText, Vector2 anchoredPosition) {
        // Show Tooltip GameObject
        TagGameObject.SetActive(true);

        TagGameObject.GetComponent<RectTransform>().anchoredPosition = anchoredPosition;

        Text tooltipUIText = TagGameObject.transform.Find("Text").GetComponent<Text>();
        tooltipUIText.text = TagText;

        float textPaddingSize = 4f;
        Vector2 backgroundSize = new Vector2(
            tooltipUIText.preferredWidth + textPaddingSize * 2f,
            tooltipUIText.preferredHeight + textPaddingSize * 2f
        );

        TagGameObject.transform.Find("background").GetComponent<RectTransform>().sizeDelta =
backgroundSize;

        // UI Visibility Sorting based on Hierarchy, SetAsLastSibling in order to show up on top
        TagGameObject.transform.SetAsLastSibling();
    }

    public static void HideTag_Static() {
        instance.HideTag();
    }

    private void HideTag() {
        TagGameObject.SetActive(false);
    }

    private void SetGetAxisLabelX(Func<int, string> getAxisLabelX) {
        ShowGraph(this.valueList, this.graphVisual, this.maxVisibleValueAmount, getAxisLabelX,
this.getAxisLabelY);
    }

    private void SetGetAxisLabelY(Func<float, string> getAxisLabelY) {
        ShowGraph(this.valueList, this.graphVisual, this.maxVisibleValueAmount, this.getAxisLabelX,
getAxisLabelY);
    }

    private void IncreaseVisibleAmount() {
        ShowGraph(this.valueList, this.graphVisual, this.maxVisibleValueAmount + 1, this.getAxisLabelX,
this.getAxisLabelY);
    }
```

```
    private void DecreaseVisibleAmount() {
        ShowGraph(this.valueList, this.graphVisual, this.maxVisibleValueAmount - 1, this.getAxisLabelX,
this.getAxisLabelY);
    }

    private void SetGraphVisual(IGraphVisual graphVisual) {
        ShowGraph(this.valueList, graphVisual, this.maxVisibleValueAmount, this.getAxisLabelX,
this.getAxisLabelY);
    }

    private void ShowGraph(List<int> valueList, IGraphVisual graphVisual, int maxVisibleValueAmount = -
1, Func<int, string> getAxisLabelX = null, Func<float, string> getAxisLabelY = null) {
        this.valueList = valueList;
        this.graphVisual = graphVisual;
        this.getAxisLabelX = getAxisLabelX;
        this.getAxisLabelY = getAxisLabelY;

        if (maxVisibleValueAmount <= 0) {
            // Show all if no amount specified
            maxVisibleValueAmount = valueList.Count;
        }
        if (maxVisibleValueAmount > valueList.Count) {
            // Validate the amount to show the maximum
            maxVisibleValueAmount = valueList.Count;
        }

        this.maxVisibleValueAmount = maxVisibleValueAmount;

        // Test for label defaults
        if (getAxisLabelX == null) {
            getAxisLabelX = delegate (int _i) { return _i.ToString(); };
        }
        if (getAxisLabelY == null) {
            getAxisLabelY = delegate (float _f) { return Mathf.RoundToInt(_f).ToString(); };
        }

        // Clean up previous graph
        foreach (GameObject gameObject in gameObjectList) {
            Destroy(gameObject);
        }
        gameObjectList.Clear();
        yLabelList.Clear();

        foreach (IGraphVisualObject graphVisualObject in graphVisualObjectList) {
            graphVisualObject.CleanUp();
        }
        graphVisualObjectList.Clear();

        graphVisual.CleanUp();

        // Grab the width and height from the container
        float graphWidth = graphContainer.sizeDelta.x;
        float graphHeight = graphContainer.sizeDelta.y;

        float yMinimum, yMaximum;
        CalculateYScale(out yMinimum, out yMaximum);

        // Set the distance between each point on the graph
        xSize = graphWidth / (maxVisibleValueAmount + 1);

        // Cycle through all visible data points
        int xIndex = 0;
        for (int i = Mathf.Max(valueList.Count - maxVisibleValueAmount, 0); i < valueList.Count; i++) {
            float xPosition = xSize + xIndex * xSize;
            float yPosition = ((valueList[i] - yMinimum) / (yMaximum - yMinimum)) * graphHeight;
// Add data point visual
            string TagText = getAxisLabelY(valueList[i]);
            IGraphVisualObject graphVisualObject = graphVisual.CreateGraphVisualObject(new
Vector2(xPosition, yPosition), xSize, TagText);
            graphVisualObjectList.Add(graphVisualObject);
```

```
            // Duplicate the x label template
            RectTransform labelX = Instantiate(labelTemplateX);
            labelX.SetParent(graphContainer, false);
            labelX.gameObject.SetActive(true);
            labelX.anchoredPosition = new Vector2(xPosition, -40f);
            labelX.GetComponent<Text>().text = getAxisLabelX(i);
            gameObjectList.Add(labelX.gameObject);

            // Duplicate the x dash template
            RectTransform dashX = Instantiate(dashTemplateX);
            dashX.SetParent(graphContainer, false);
            dashX.gameObject.SetActive(true);
            dashX.anchoredPosition = new Vector2(xPosition, -3f);
            gameObjectList.Add(dashX.gameObject);

            xIndex++;
        }

        // Set up separators on the y axis
        int separatorCount = 10;
        for (int i = 0; i <= separatorCount; i++) {
            // Duplicate the label template
            RectTransform labelY = Instantiate(labelTemplateY);
            labelY.SetParent(graphContainer, false);
            labelY.gameObject.SetActive(true);
            float normalizedValue = i * 1f / separatorCount;
            labelY.anchoredPosition = new Vector2(-14f, normalizedValue * graphHeight);
            labelY.GetComponent<Text>().text = getAxisLabelY(yMinimum + (normalizedValue * (yMaximum -
yMinimum)));
            yLabelList.Add(labelY);
            gameObjectList.Add(labelY.gameObject);

            // Duplicate the dash template
            RectTransform dashY = Instantiate(dashTemplateY);
            dashY.SetParent(graphContainer, false);
            dashY.gameObject.SetActive(true);
            dashY.anchoredPosition = new Vector2(-4f, normalizedValue * graphHeight);
            gameObjectList.Add(dashY.gameObject);
        }
    }

    private void UpdateValue(int index, int value) {
        float yMinimumBefore, yMaximumBefore;
        CalculateYScale(out yMinimumBefore, out yMaximumBefore);

        valueList[index] = value;

        float graphWidth = graphContainer.sizeDelta.x;
        float graphHeight = graphContainer.sizeDelta.y;

        float yMinimum, yMaximum;
        CalculateYScale(out yMinimum, out yMaximum);

        bool yScaleChanged = yMinimumBefore != yMinimum || yMaximumBefore != yMaximum;

        if (!yScaleChanged) {
            // Y Scale did not change, update only this value
            float xPosition = xSize + index * xSize;
            float yPosition = ((value - yMinimum) / (yMaximum - yMinimum)) * graphHeight;

            // Add data point visual
            string TagText = getAxisLabelY(value);
            graphVisualObjectList[index].SetGraphVisualObjectInfo(new Vector2(xPosition, yPosition),
xSize, TagText);
        } else {
            // Y scale changed, update whole graph and y axis labels
            // Cycle through all visible data points
            int xIndex = 0;
```

```
            for (int i = Mathf.Max(valueList.Count - maxVisibleValueAmount, 0); i < valueList.Count;
i++) {
                float xPosition = xSize + xIndex * xSize;
                float yPosition = ((valueList[i] - yMinimum) / (yMaximum - yMinimum)) * graphHeight;

                // Add data point visual
                string TagText = getAxisLabelY(valueList[i]);
                graphVisualObjectList[xIndex].SetGraphVisualObjectInfo(new Vector2(xPosition,
yPosition), xSize, TagText);

                xIndex++;
            }

            for (int i = 0; i < yLabelList.Count; i++) {
                float normalizedValue = i * 1f / yLabelList.Count;
                yLabelList[i].GetComponent<Text>().text = getAxisLabelY(yMinimum + (normalizedValue *
(yMaximum - yMinimum)));
            }
        }
    }

    private void CalculateYScale(out float yMinimum, out float yMaximum) {
        // Identify y Min and Max values
        yMaximum = valueList[0];
        yMinimum = valueList[0];

        for (int i = Mathf.Max(valueList.Count - maxVisibleValueAmount, 0); i < valueList.Count; i++) {
            int value = valueList[i];
            if (value > yMaximum) {
                yMaximum = value;
            }
            if (value < yMinimum) {
                yMinimum = value;
            }
        }

        float yDifference = yMaximum - yMinimum;
        if (yDifference <= 0) {
            yDifference = 5f;
        }
        yMaximum = yMaximum + (yDifference * 0.2f);
        yMinimum = yMinimum - (yDifference * 0.2f);

    }
private interface IGraphVisual {

        IGraphVisualObject CreateGraphVisualObject(Vector2 graphPosition, float graphPositionWidth,
string TagText);
        void CleanUp();

    }

    /*
     * Represents a single Visual Object in the graph
     * */
    private interface IGraphVisualObject {

        void SetGraphVisualObjectInfo(Vector2 graphPosition, float graphPositionWidth, string TagText);
        void CleanUp();

    }


    /*
     * Displays data points as a Bar Chart
     * */
    private class BarChartVisual : IGraphVisual {

        private RectTransform graphContainer;
        private Color barColor;
```

```
        private float barWidthMultiplier;

        public BarChartVisual(RectTransform graphContainer, Color barColor, float barWidthMultiplier) {
            this.graphContainer = graphContainer;
            this.barColor = barColor;
            this.barWidthMultiplier = barWidthMultiplier;
        }

        public void CleanUp() {
        }

        public IGraphVisualObject CreateGraphVisualObject(Vector2 graphPosition, float
graphPositionWidth, string TagText) {
            GameObject barGameObject = CreateBar(graphPosition, graphPositionWidth);

            BarChartVisualObject barChartVisualObject = new BarChartVisualObject(barGameObject,
barWidthMultiplier);
            barChartVisualObject.SetGraphVisualObjectInfo(graphPosition, graphPositionWidth, TagText);

            return barChartVisualObject;
        }

        private GameObject CreateBar(Vector2 graphPosition, float barWidth) {
            GameObject gameObject = new GameObject("bar", typeof(Image));
            gameObject.transform.SetParent(graphContainer, false);
            gameObject.GetComponent<Image>().color = barColor;
            RectTransform rectTransform = gameObject.GetComponent<RectTransform>();
            rectTransform.anchoredPosition = new Vector2(graphPosition.x, 0f);
            rectTransform.sizeDelta = new Vector2(barWidth * barWidthMultiplier, graphPosition.y);
            rectTransform.anchorMin = new Vector2(0, 0);
            rectTransform.anchorMax = new Vector2(0, 0);
            rectTransform.pivot = new Vector2(.5f, 0f);

            // Add Button_UI Component which captures UI Mouse Events
            Button_UI barButtonUI = gameObject.AddComponent<Button_UI>();

            return gameObject;
        }


        public class BarChartVisualObject : IGraphVisualObject {

            private GameObject barGameObject;
            private float barWidthMultiplier;

            public BarChartVisualObject(GameObject barGameObject, float barWidthMultiplier) {
                this.barGameObject = barGameObject;
                this.barWidthMultiplier = barWidthMultiplier;
            }

            public void SetGraphVisualObjectInfo(Vector2 graphPosition, float graphPositionWidth, string
TagText) {
                RectTransform rectTransform = barGameObject.GetComponent<RectTransform>();
                rectTransform.anchoredPosition = new Vector2(graphPosition.x, 0f);
                rectTransform.sizeDelta = new Vector2(graphPositionWidth * barWidthMultiplier,
graphPosition.y);


                Button_UI barButtonUI = barGameObject.GetComponent<Button_UI>();
// Show Tooltip on Mouse Over
                barButtonUI.MouseOverOnceFunc = () => {
                    ShowTag_Static(TagText, graphPosition);
                };

                // Hide Tooltip on Mouse Out
                barButtonUI.MouseOutOnceFunc = () => {
                    HideTag_Static();
                };
            }
```

```
            public void CleanUp() {
                Destroy(barGameObject);
            }


        }

    }


    /*
     * Displays data points as a Line Graph
     * */
    private class LineGraphVisual : IGraphVisual {

        private RectTransform graphContainer;
        private Sprite dotSprite;
        private LineGraphVisualObject lastLineGraphVisualObject;
        private Color dotColor;
        private Color dotConnectionColor;

        public LineGraphVisual(RectTransform graphContainer, Sprite dotSprite, Color dotColor, Color
dotConnectionColor) {
            this.graphContainer = graphContainer;
            this.dotSprite = dotSprite;
            this.dotColor = dotColor;
            this.dotConnectionColor = dotConnectionColor;
            lastLineGraphVisualObject = null;
        }

        public void CleanUp() {
            lastLineGraphVisualObject = null;
        }


        public IGraphVisualObject CreateGraphVisualObject(Vector2 graphPosition, float
graphPositionWidth, string TagText) {
            GameObject dotGameObject = CreateDot(graphPosition);


            GameObject dotConnectionGameObject = null;
            if (lastLineGraphVisualObject != null) {
                dotConnectionGameObject =
CreateDotConnection(lastLineGraphVisualObject.GetGraphPosition(),
dotGameObject.GetComponent<RectTransform>().anchoredPosition);
            }

            LineGraphVisualObject lineGraphVisualObject = new LineGraphVisualObject(dotGameObject,
dotConnectionGameObject, lastLineGraphVisualObject);
            lineGraphVisualObject.SetGraphVisualObjectInfo(graphPosition, graphPositionWidth, TagText);

            lastLineGraphVisualObject = lineGraphVisualObject;

            return lineGraphVisualObject;
        }

        private GameObject CreateDot(Vector2 anchoredPosition) {
            GameObject gameObject = new GameObject("dot", typeof(Image));
            gameObject.transform.SetParent(graphContainer, false);
            gameObject.GetComponent<Image>().sprite = dotSprite;
            gameObject.GetComponent<Image>().color = dotColor;
            RectTransform rectTransform = gameObject.GetComponent<RectTransform>();
            rectTransform.anchoredPosition = anchoredPosition;
            rectTransform.sizeDelta = new Vector2(11, 11);
            rectTransform.anchorMin = new Vector2(0, 0);
            rectTransform.anchorMax = new Vector2(0, 0);

            // Add Button_UI Component which captures UI Mouse Events
            Button_UI dotButtonUI = gameObject.AddComponent<Button_UI>();
```

```
            return gameObject;

        }
private GameObject CreateDotConnection(Vector2 dotPositionA, Vector2 dotPositionB) {
            GameObject gameObject = new GameObject("dotConnection", typeof(Image));
            gameObject.transform.SetParent(graphContainer, false);
            gameObject.GetComponent<Image>().color = dotConnectionColor;
            gameObject.GetComponent<Image>().raycastTarget = false;
            RectTransform rectTransform = gameObject.GetComponent<RectTransform>();
            Vector2 dir = (dotPositionB - dotPositionA).normalized;
            float distance = Vector2.Distance(dotPositionA, dotPositionB);
            rectTransform.anchorMin = new Vector2(0, 0);
            rectTransform.anchorMax = new Vector2(0, 0);
            rectTransform.sizeDelta = new Vector2(distance, 3f);
            rectTransform.anchoredPosition = dotPositionA + dir * distance * .5f;
            rectTransform.localEulerAngles = new Vector3(0, 0, UtilsClass.GetAngleFromVectorFloat(dir));
            return gameObject;
        }


        public class LineGraphVisualObject : IGraphVisualObject {

            public event EventHandler OnChangedGraphVisualObjectInfo;

            private GameObject dotGameObject;
            private GameObject dotConnectionGameObject;
            private LineGraphVisualObject lastVisualObject;

            public LineGraphVisualObject(GameObject dotGameObject, GameObject dotConnectionGameObject,
LineGraphVisualObject lastVisualObject) {
                this.dotGameObject = dotGameObject;
                this.dotConnectionGameObject = dotConnectionGameObject;
                this.lastVisualObject = lastVisualObject;

                if (lastVisualObject != null) {
                    lastVisualObject.OnChangedGraphVisualObjectInfo +=
LastVisualObject_OnChangedGraphVisualObjectInfo;
                }
            }

            private void LastVisualObject_OnChangedGraphVisualObjectInfo(object sender, EventArgs e) {
                UpdateDotConnection();
            }

            public void SetGraphVisualObjectInfo(Vector2 graphPosition, float graphPositionWidth, string
TagText) {
                RectTransform rectTransform = dotGameObject.GetComponent<RectTransform>();
                rectTransform.anchoredPosition = graphPosition;

                UpdateDotConnection();

                Button_UI dotButtonUI = dotGameObject.GetComponent<Button_UI>();

                // Show Tooltip on Mouse Over
                dotButtonUI.MouseOverOnceFunc = () => {
                    ShowTag_Static(TagText, graphPosition);
                };

                // Hide Tooltip on Mouse Out
                dotButtonUI.MouseOutOnceFunc = () => {
                    HideTag_Static();
                };

                if (OnChangedGraphVisualObjectInfo != null) OnChangedGraphVisualObjectInfo(this,
EventArgs.Empty);
            }

            public void CleanUp() {
```

```
                Destroy(dotGameObject);
                Destroy(dotConnectionGameObject);
            }

            public Vector2 GetGraphPosition() {
                RectTransform rectTransform = dotGameObject.GetComponent<RectTransform>();
                return rectTransform.anchoredPosition;
            }

            private void UpdateDotConnection() {
                if (dotConnectionGameObject != null) {
                    RectTransform dotConnectionRectTransform =
dotConnectionGameObject.GetComponent<RectTransform>();
                    Vector2 dir = (lastVisualObject.GetGraphPosition() - GetGraphPosition()).normalized;
                    float distance = Vector2.Distance(GetGraphPosition(),
lastVisualObject.GetGraphPosition());
                    dotConnectionRectTransform.sizeDelta = new Vector2(distance, 3f);
                    dotConnectionRectTransform.anchoredPosition = GetGraphPosition() + dir * distance *
.5f;
                    dotConnectionRectTransform.localEulerAngles = new Vector3(0, 0,
UtilsClass.GetAngleFromVectorFloat(dir));
                }
            }

        }

    }

}
```

B.24. *AngleInsertDes*

```
<?php
//Variables for the connection
        $servername = "localhost";
        $server_username =  "root";
        $server_password = "";
        $dbName = "Users";

//Variable from the patient
  $Nombre = $_POST["DNIPost"];
  $MAXRT=$_POST["MAXRTPost"];
  $MAXLT=$_POST["MAXLTPost"];
  $Fecha= $_POST["FechaPost"];
  $Hora=$_POST["HoraPost"];
  $Movimiento=$_POST["MovimientoPost"];
  $Desviacion=$_POST["DesviacionPost"];
  $Nivel=$_POST["NivelPost"];
  $Media=$_POST["MediaPost"];


        //Make Connection
        $conn = new mysqli($servername, $server_username, $server_password, $dbName);
        //Check Connection
        if(!$conn){
                die("Connection Failed. ". mysqli_connect_error());
        }

        $sqc= "SELECT PatientName FROM pacients WHERE pacients.DNI='".$Nombre."'";

    $sql = "INSERT INTO lowerbody (DNI_Paciente, AnguloMaxIzq, AnguloMaxDer, Nombre, Dia,
Hora,Ejercicio, Desviacion, Media, Nivel) SELECT pacients.DNI, '".$MAXLT."','".$MAXRT."',
pacients.PatientName,
'".$Fecha."','".$Hora."','".$Movimiento."','".$Desviacion."','".$Media."','".$Nivel."' FROM
pacients WHERE pacients.DNI='".$Nombre."'";

        $result = mysqli_query($conn ,$sql);

        if(!result) echo "there was an error";
        else echo "Everything ok.";


?>
```

Anexo C

C.1. Plantilla encuesta realizada

ENCUESTA:

Identificador:

APLICACIÓN DE LA REALIDAD VIRTUAL A LA FISIOTERAPIA

Antes de hacer la prueba

1. Género

2. Edad

3. ¿Alguna vez has realizado una terapia de rehabilitación?

    a. Del 1(mínimo) al 5 (máximo) ¿qué tan aburridas considera que son las terapias de rehabilitación?

Después de hacer la prueba

4. Después de realizar la terapia del el 1(mínimo) al 5(máximo) ¿Que tanto te costó realizarla?

5. Después de realizar la terapia del 1(mínimo) al 5(máximo) ¿qué puntuación le darías a esa experiencia?

6. ¿Cree que la aplicación de la VR mejoraría las experiencias en las sesiones de rehabilitación?

7. ¿Prefiere un juego estático o dinámico?

8. ¿Prefieres un control mediante las piernas o VR?

9. ¿En algún momento se ha sentido mareado/a o desubicada con las gafas puestas?

Resultados de la prueba

10. Máximo (Objetivo a alcanzar)

11. Gráfica

Reputación que tiene la rehabilitación

12. ¿Conoces a alguna persona que haya hecho rehabilitación?

13. Si a pregunta es afirmativa y conoces su experiencia. Del 1 (mínimo) al 5 (máximo) ¿Qué experiencia tuvo?

14. ¿Porqué?

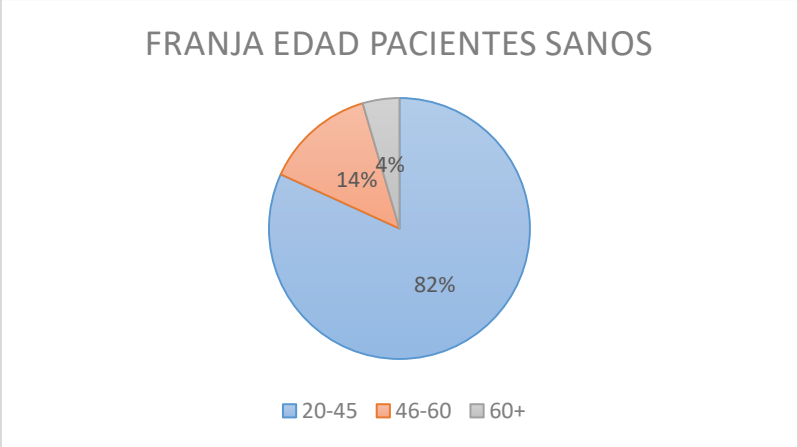
C.2. Resultados encuestas

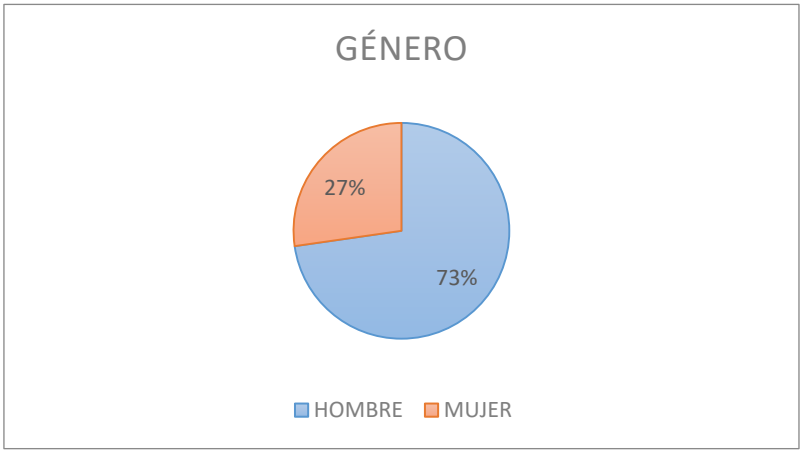
BLOQUE NÚMERO 1					
EDAD MUESTRA			**GÉNERO**	**EXPERIENCIA PREVIA**	**CUALIFICACIÓN**
20-45	**46-60**	**60+**	**H / M**	**SI / NO**	**1 - 5**
Paciente 1			H	NO	-
Paciente 3			H	SI	4
Paciente 4			H	NO	-
Paciente 5			H	NO	-
Paciente 6			H	NO	-
Paciente 7			H	SI	5
Paciente 8			H	NO	-
Paciente 13			M	SI	4
	Paciente 14		M	NO	-
	Paciente 11		H	NO	-
	Paciente 16		M	SI	5
Paciente 15			M	NO	-
Paciente 17			H	NO	-
Paciente 18			H	SI	4
Paciente 19			H	NO	-
Paciente 20			H	NO	-
Paciente 21			H	NO	-
Paciente 22			H	NO	-
Paciente 9			H	NO	-
		Paciente 12	M	NO	-
Paciente 10			M	NO	-
Paciente 2			H	SI	4

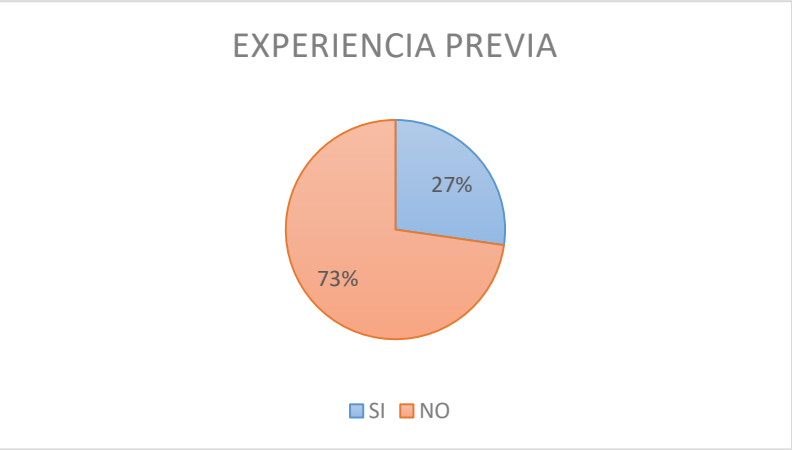
BLOQUE NÚMERO 2						
DIFICULTAD VR	**EXPERIENCIA VIVIDA**	**UTILIDAD**	**DINÁMICO O ESTÁTICO**	**MAREOS**	**CONTROL**	**OBSERVACIONES**
1 - 5	**1 - 5**	**SI / NO**	**D / E**	**SI / NO**	**VR / PIERNAS**	**-**
2	5	SI	D	NO	-	N/A
2	4	SI	D	NO	VR	Podría llevar confusión en pacientes de tercera edad
1	4	SI	D	NO	PIERNAS	Haría las sesiones más divertidas
2	4	SI	D	NO	-	Ameniza la experiencia y ayuda a darle un objetivo más a corto plazo a la recuperación
1	4	SI	D	NO	-	Crec que per a la gent que ha de fer moviments de rehabilitació en estàtic sería més fàcil i entretingut fer-ho si aquests venen acompanyats d'altres estímuls sensorials
2	4	SI	D	SI	-	N/A
2	5	SI	D	NO	-	N/A
2	5	SI	D	SI	-	N/A
2	5	SI	D	NO	-	N/A
2	4	SI	D	SI	-	N/A
2	5	SI	D	NO	-	Més distret, treballes sense adonar-te'n
1	4	SI	D	SI	-	N/A
2	4	SI	D	NO	-	N/A
2	5	SI	D	NO	-	En alguns moments per aconseguir les monedes, mantenir l'equilibri pot ser complicat // Suposaria que les persones estiguéssin fent una activitat que els agradés mentre estan fent els exercicis per a la seva recuperació.
4	5	SI	D	NO	PIERNAS	Las sesiones se harían más amenas
2	4	NO	D	SI	VR	N/A
3	5	SI	D	NO	-	N/A
1	5	SI	D	SI	-	N/A
2	4	SI	D	NO	VR	N/A
1	5	SI	D	NO	PIERNAS	Es más ameno
4	4	SI	D	NO	PIERNAS	N/A
2	4	SI	D	SI	PIERNAS	Marea al empezar

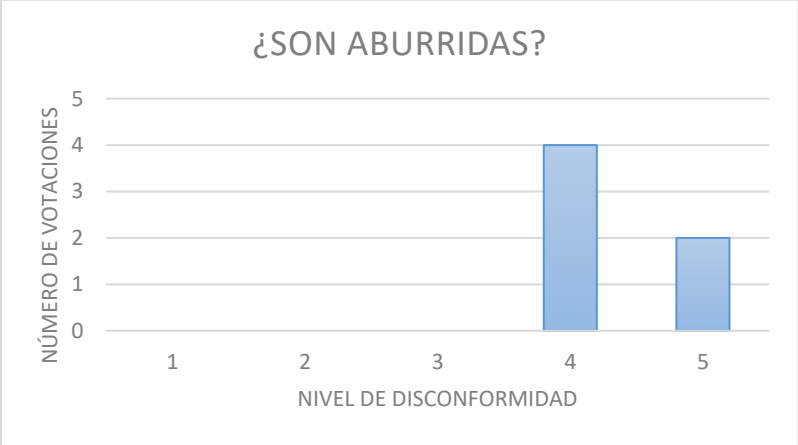
BLOQUE NÚMERO 3		
CONOCIDOS	**EXPERIENCIA DEL CONOCIDO**	**RAZÓN**
SI / NO	**1 - 5**	-
SI	3	Secuencia de movimiento monótona
SI	2	La rehabilitación duro 3 años, el objetivo era que ralentizar los efectos de la enfermedad (esclerosis múltiple), el proceso se convirtió en algo muy repetitivo para la paciente.
SI	2	Se cansaba mucho anímicamente al poco tiempo y las sesiones se le hacían eternas
SI	1	Lo encontraba aburrido, no le gustaban los ejercicios y al no ver mejoras a corto plazo se frustraba mucho
SI	N/A	N/A
NO	-	N/A
NO	-	N/A
SI	2	Mala experiencia
NO	-	N/A
SI	1	No le gustaban las sesiones y dejó de ir
SI	1	Negatiu, no pels exercicis en si, sinó pels nuls resultats després de la teràpia
NO	-	N/A
NO	-	N/A
SI	4	Els exercicis que li van proposar van servir-li per millorar-se
SI	N/A	N/A
NO	-	N/A
NO	-	N/A
SI	3	La fisioterapeuta hizo un buen trabajo
NO	-	N/A
SI	3	Sabía que sería doloroso y que tenía que pasar por la terapia de rehabilitación
SI	N/A	N/A
SI	2	Larga y dolorosa

EDAD MUESTRA				
20-45	**46-60**	**60+**		
18	3	1		
GÉNERO				
HOMBRE	**MUJER**	**N/A**		
16	6			
EXP. PREVIA REHAB.				
SI	**NO**			
6	16			
CUALIFICACIONES				
1	**2**	**3**	**4**	**5**
0	0	0	4	2

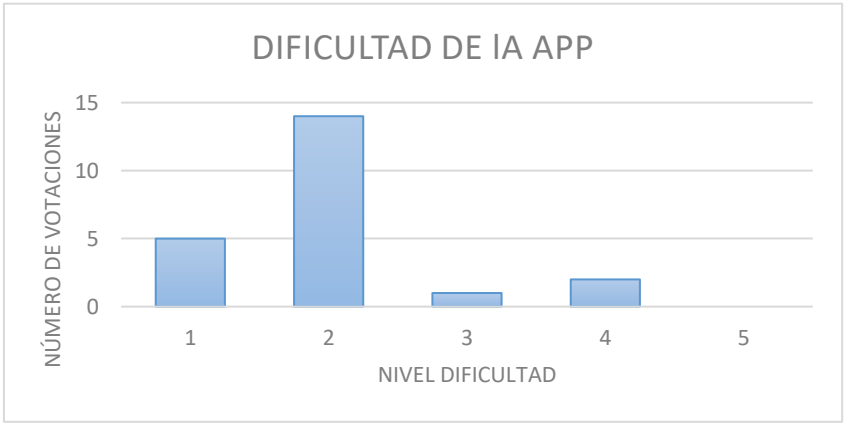




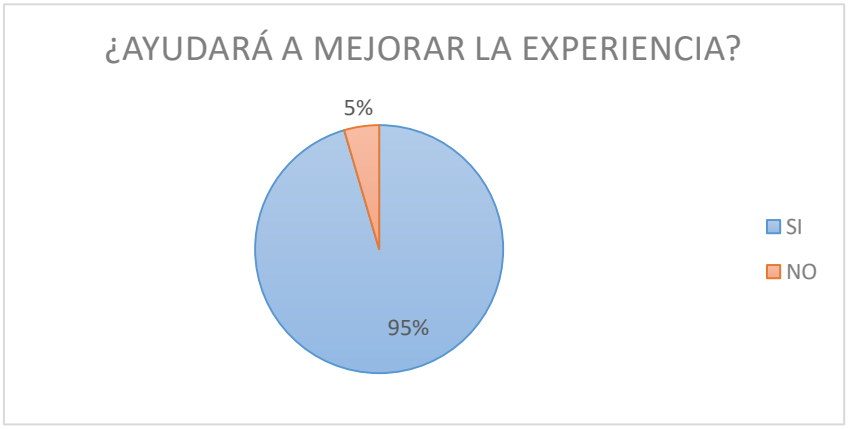

EXPERIENCIA PREVIA
27%
73%
SI
NO

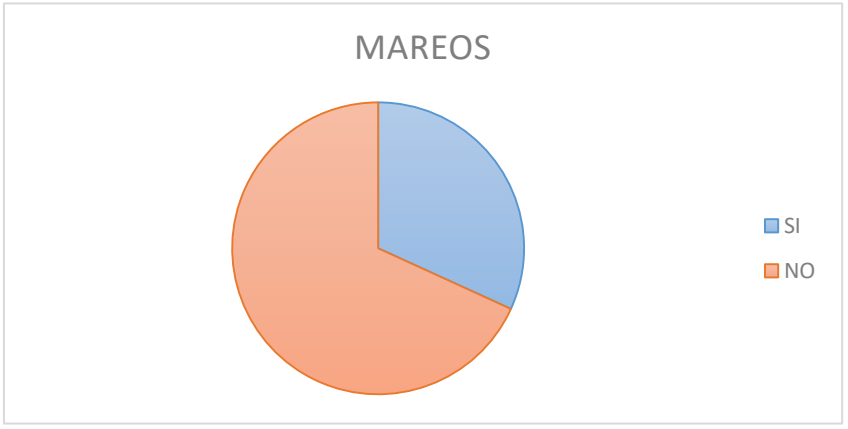

¿SON ABURRIDAS?
NÚMERO DE VOTACIONES
0
1
2
3
4
5
1
2
3
4
5
NIVEL DE DISCONFORMIDAD

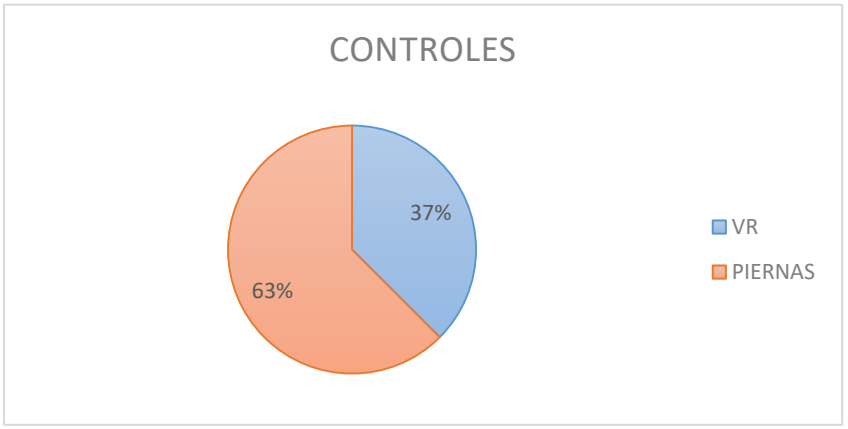
DIFICULTAD VR				
1	**2**	**3**	**4**	**5**
5	14	1	2	0
EXPERIENCIA				
1	**2**	**3**	**4**	**5**
0	0	0	12	10
¿AYUDARÁ A MEJORAR?				
SI	**NO**			
21	1			
MAREOS				
SI	**NO**			
7	15			
CONTROLES				
VR	**PIERNAS**			
3	5			



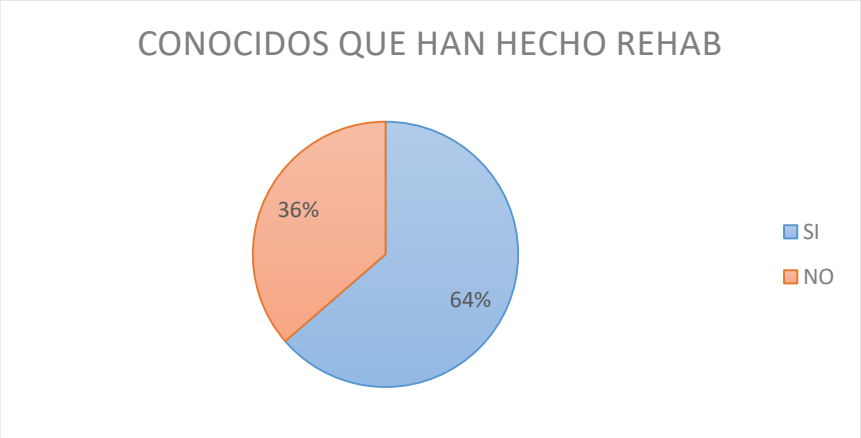

EXPERIENCIA VIVIDA
NÚMERO DE VOTACIONES
15
10
5
0
1
2
3
4
5
NIVEL DE DIVERSIÓN


¿AYUDARÁ A MEJORAR LA EXPERIENCIA?
5%
95%
SI
NO


MAREOS
SI
NO

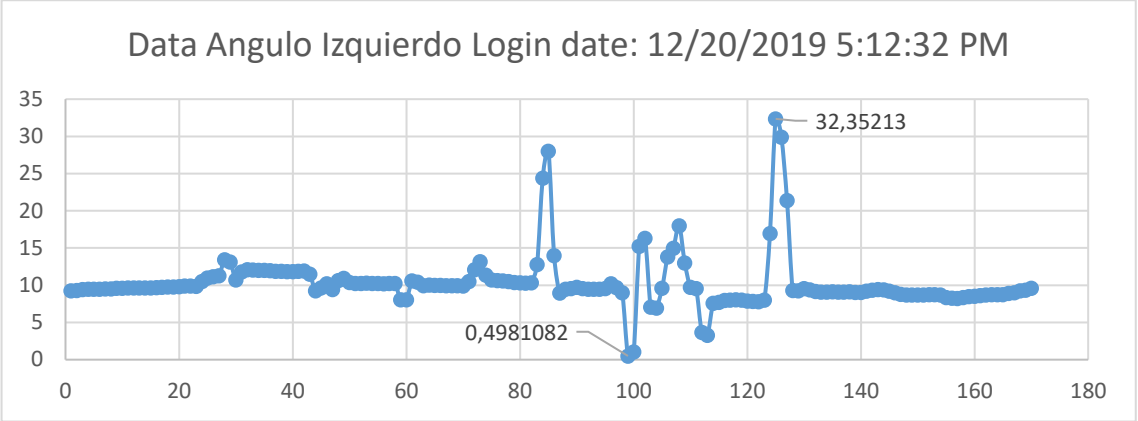

CONTROLES
37%
63%
VR
PIERNAS

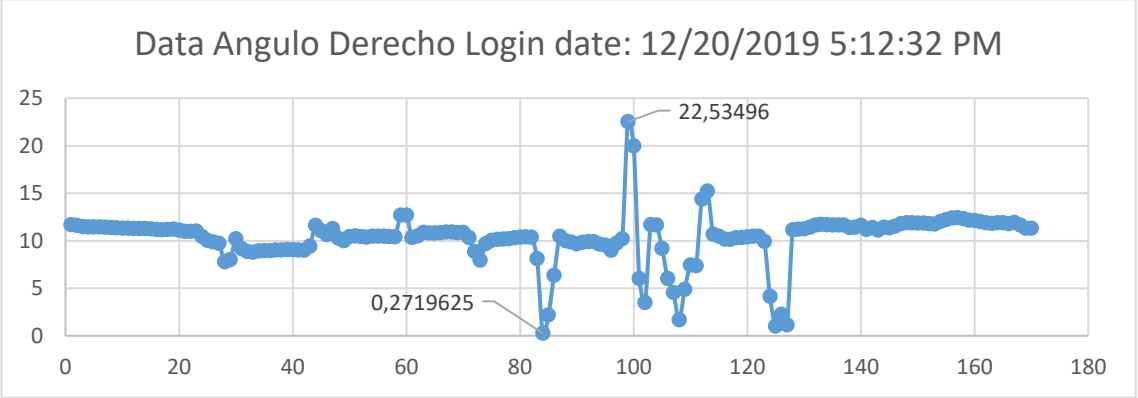
CONOCIDOS					
SI	**NO**				
14	8				
EXPERIENCIA					
1	**2**	**3**	**4**	**5**	**N/A**
3	4	3	1	0	3

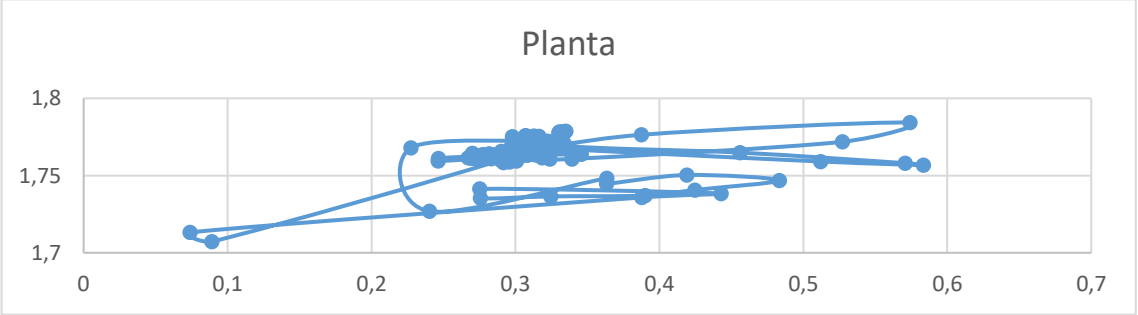


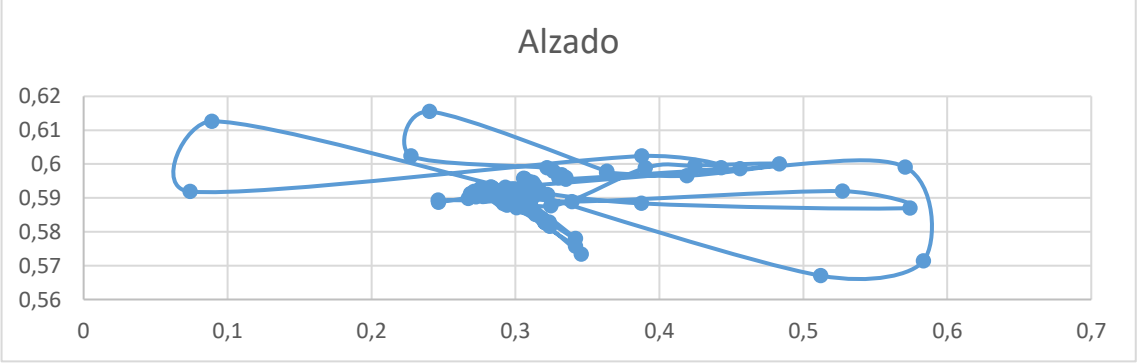
C.3 Resultados gráficos del ejercicio “Desequilibrio”

- Paciente 1 (Control VR)

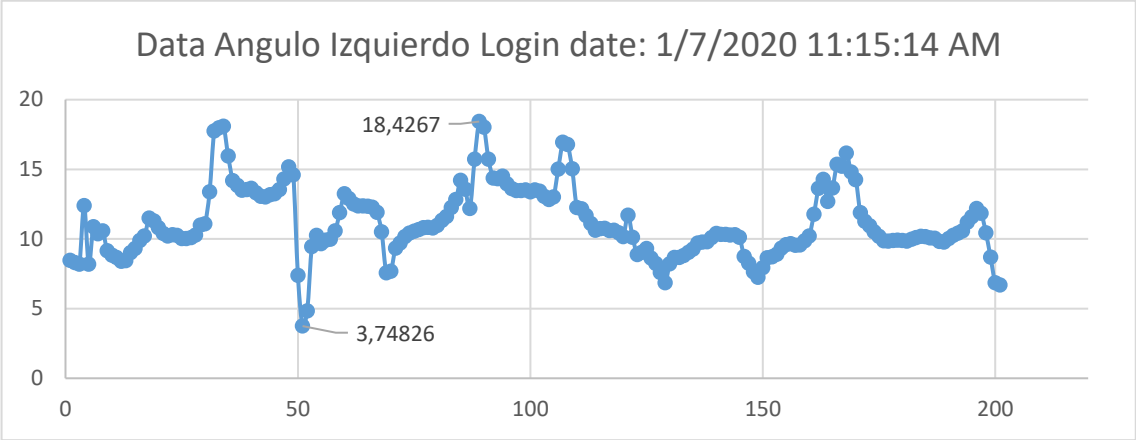


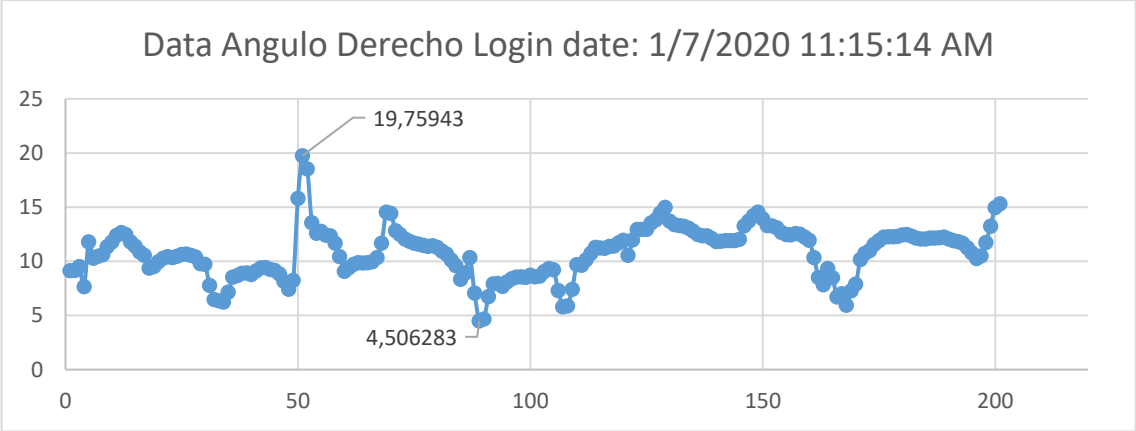


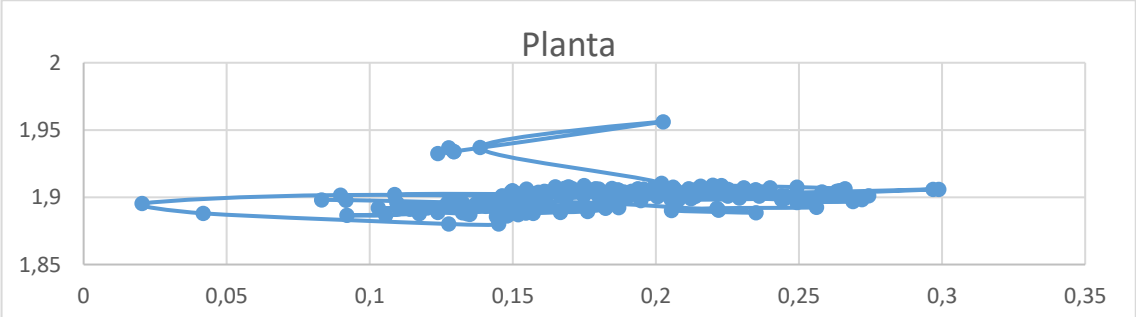


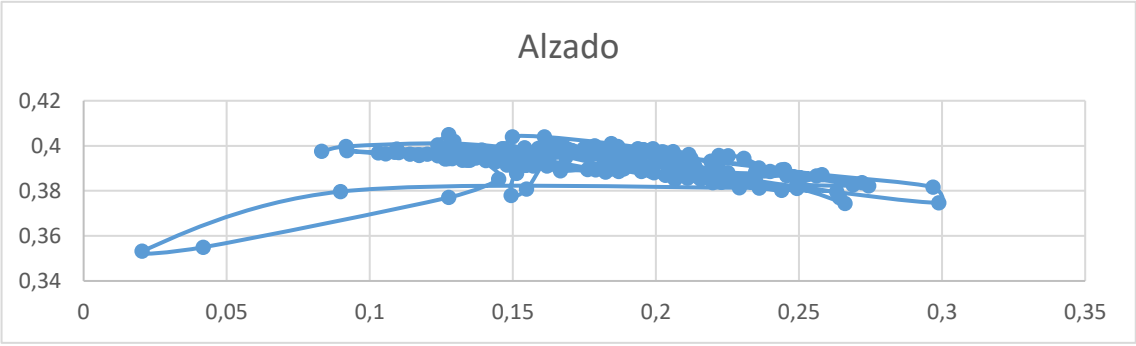


- Paciente 2 (Control VR)

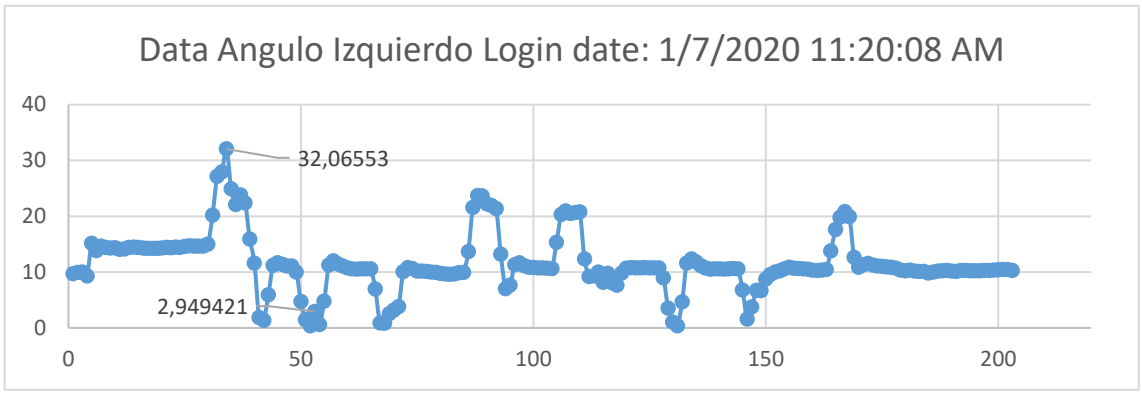


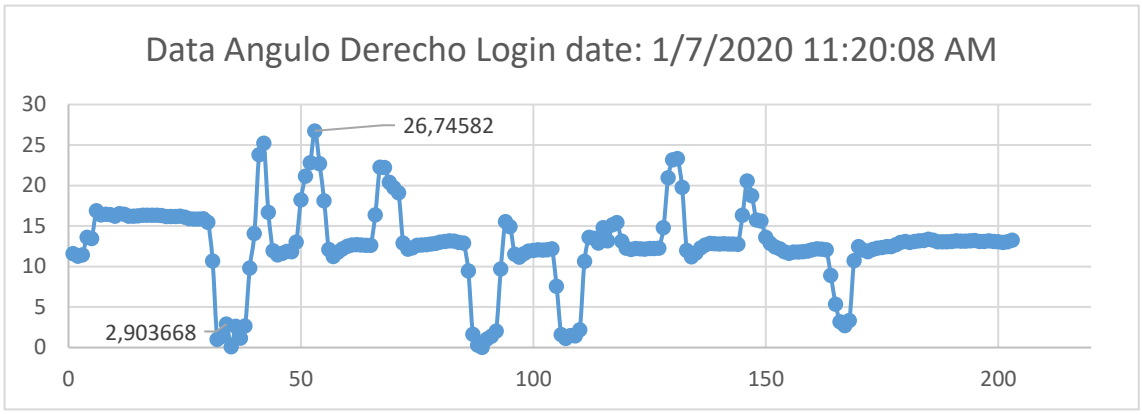


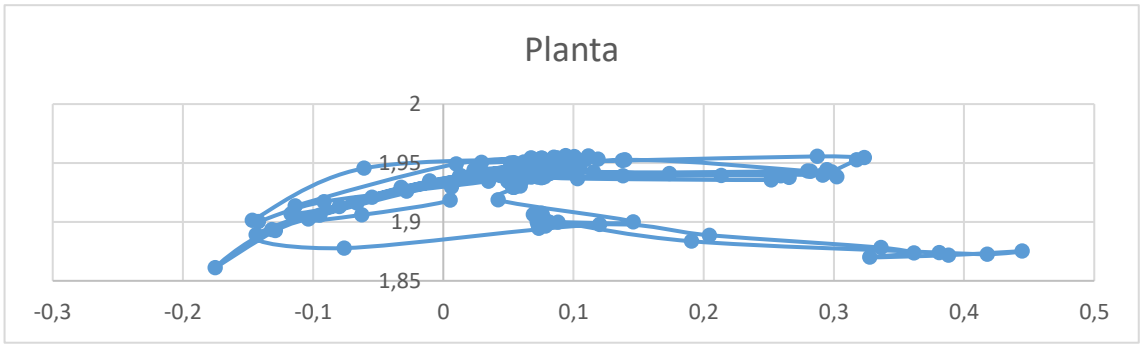


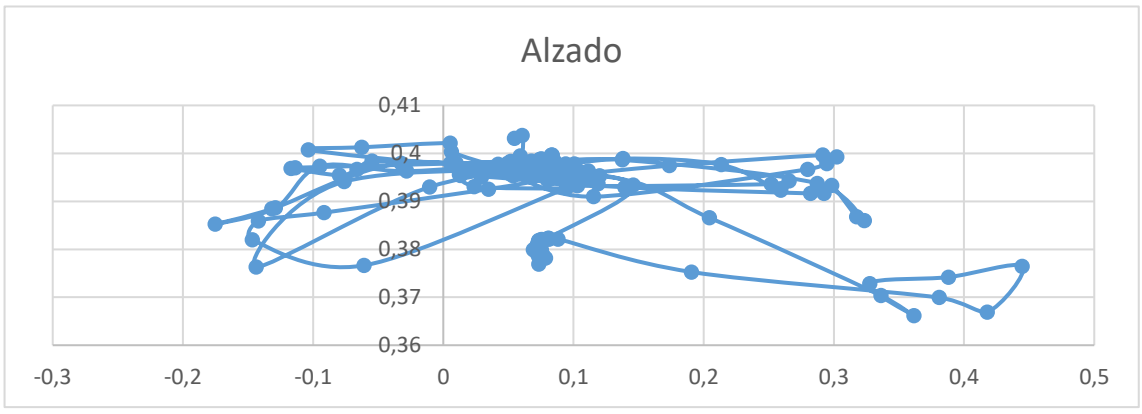


- Paciente 2 (Control Piernas)

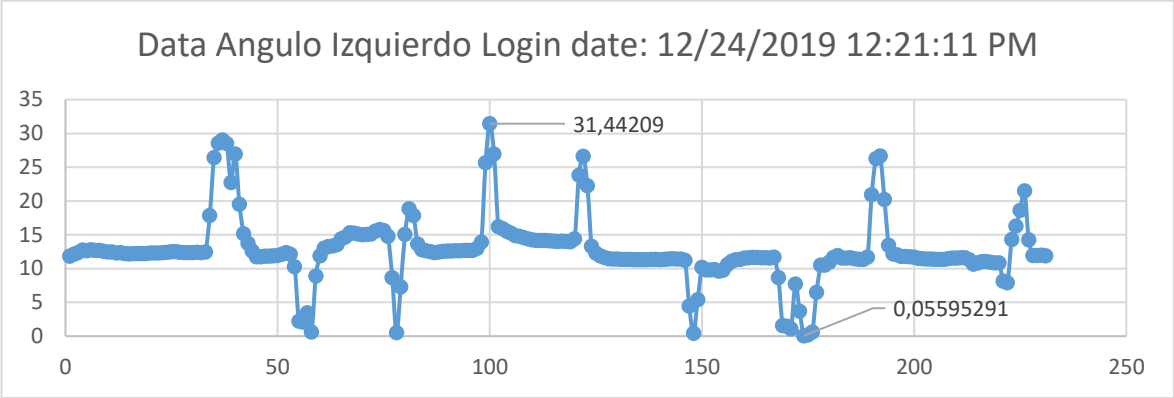


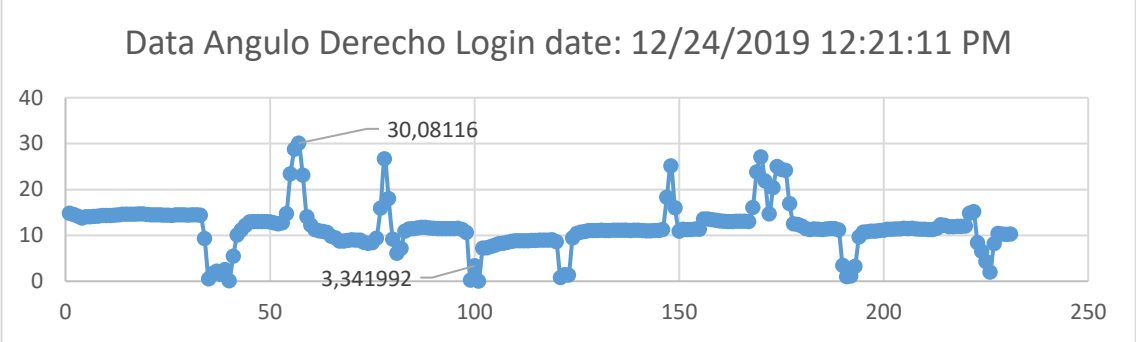


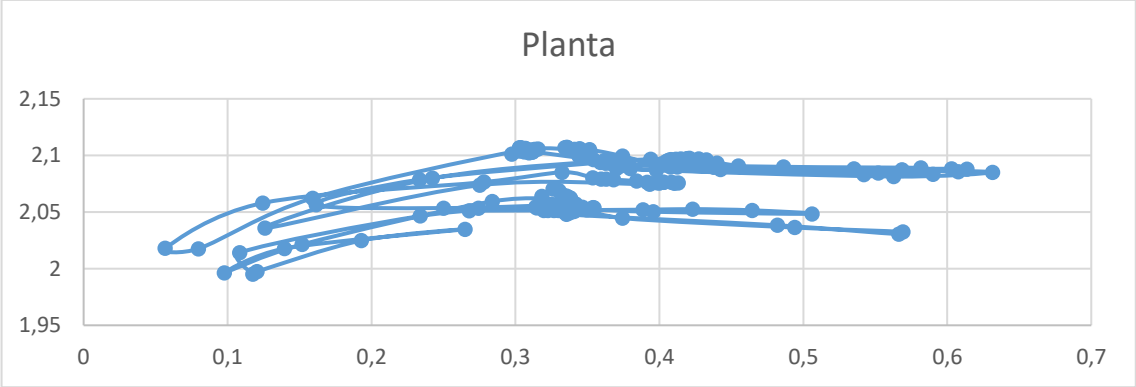


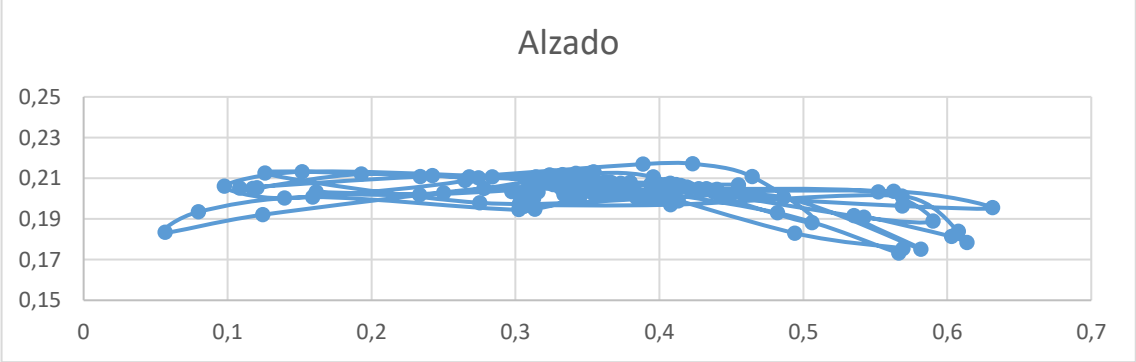


- Paciente 3 (Control VR)

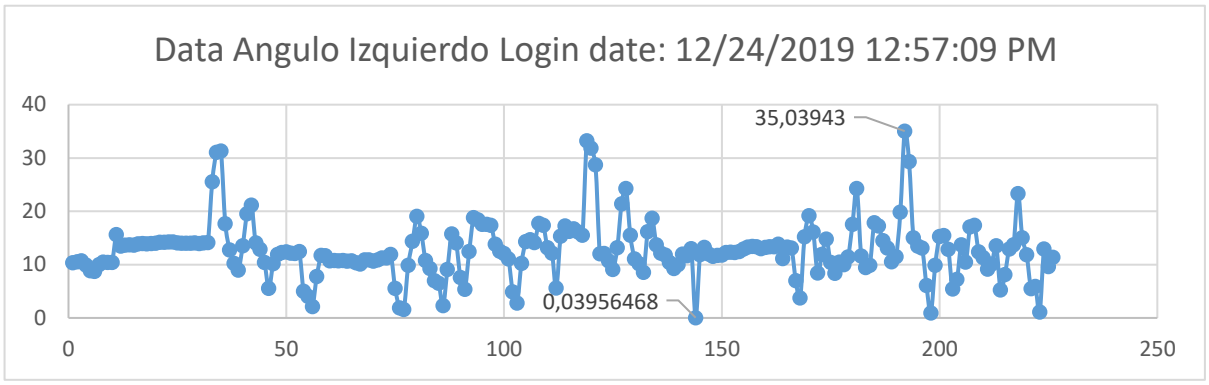


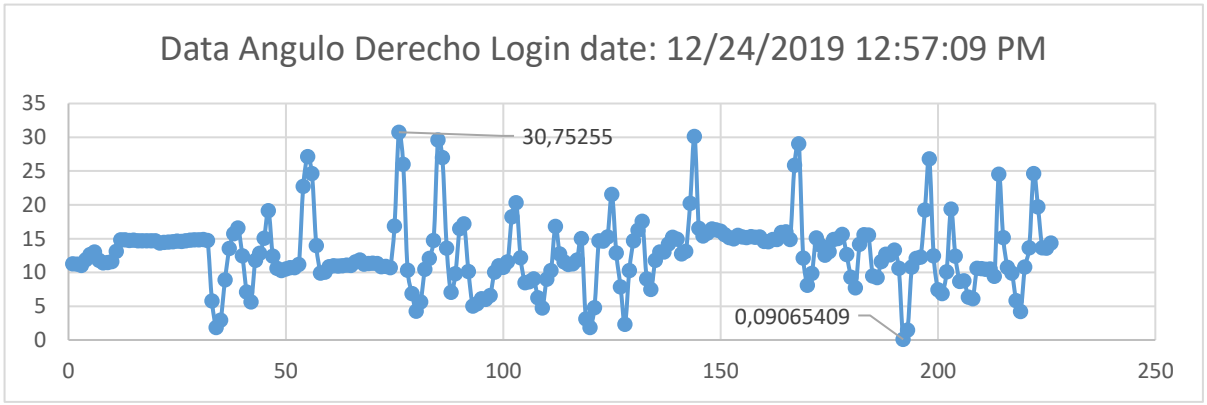


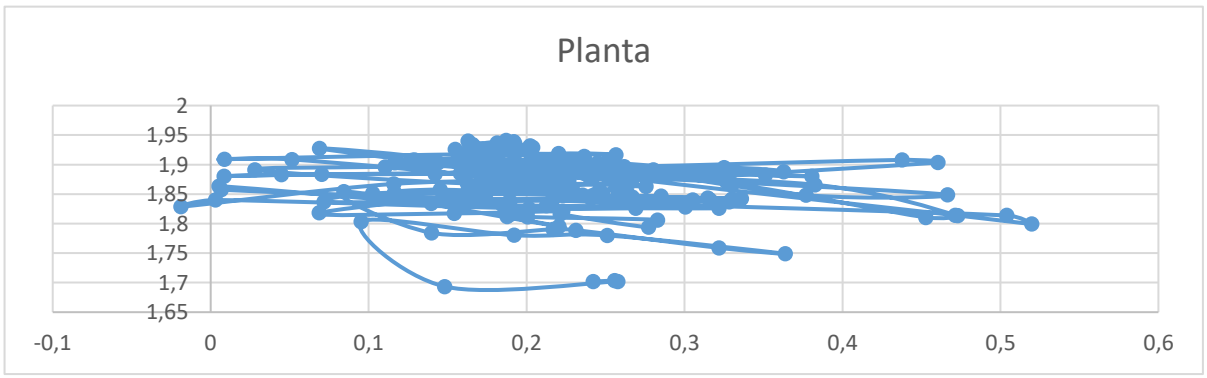


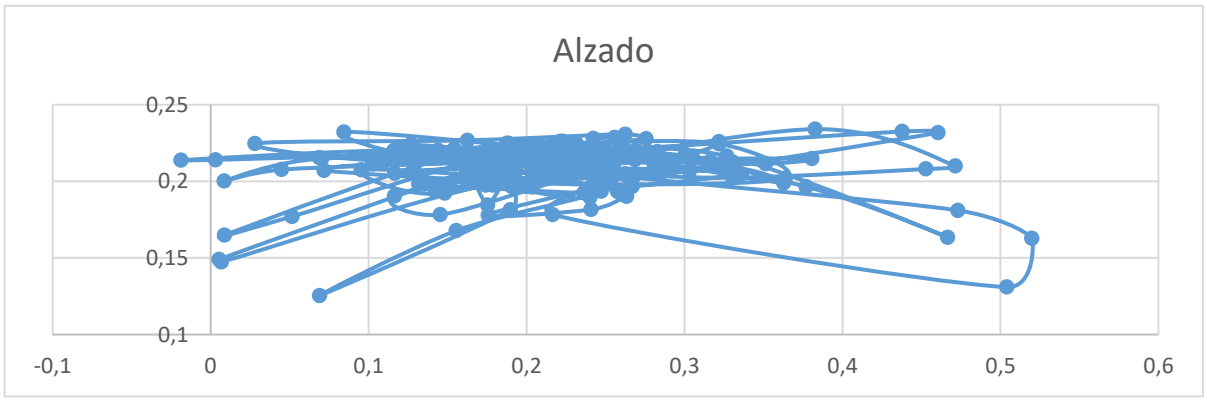


- Paciente 3 (Control Piernas)

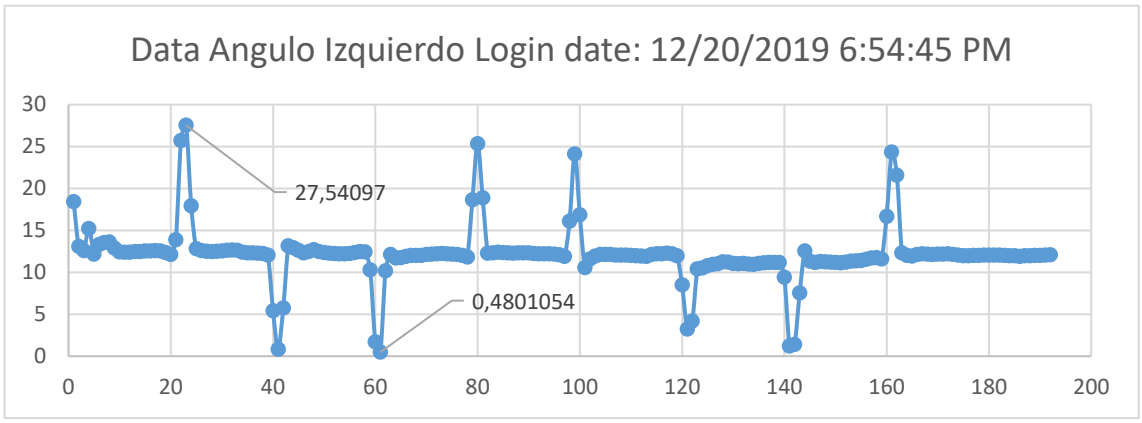


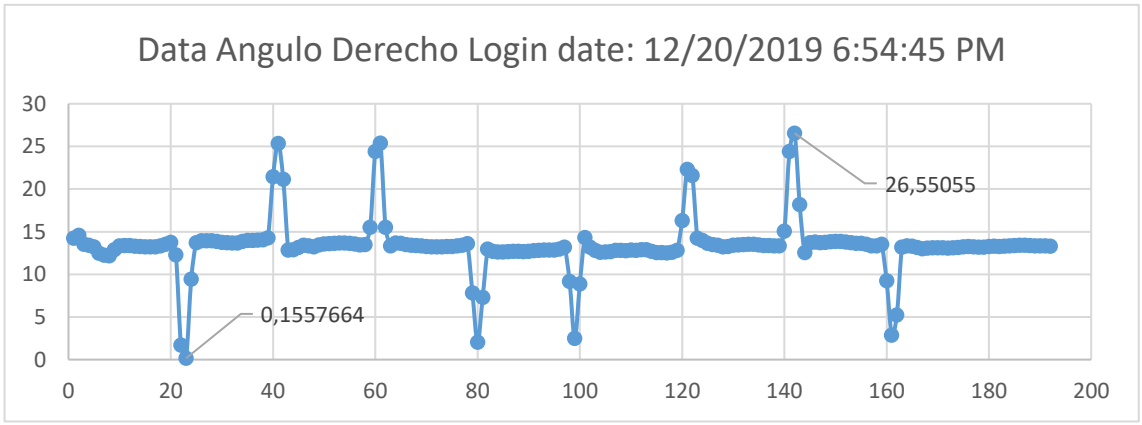


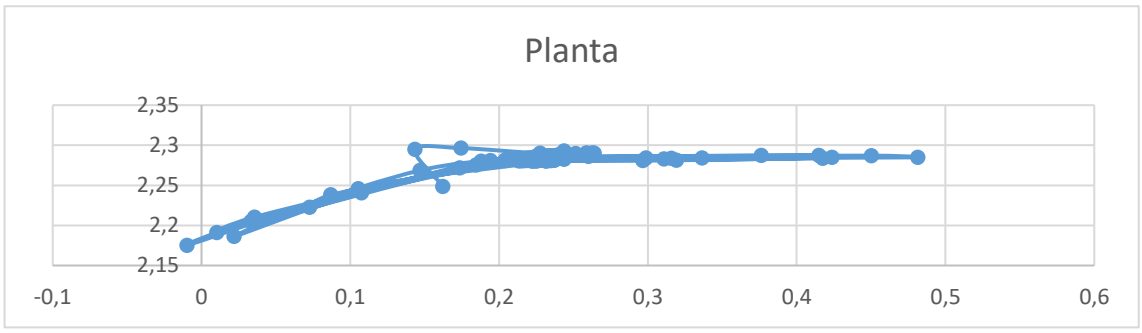


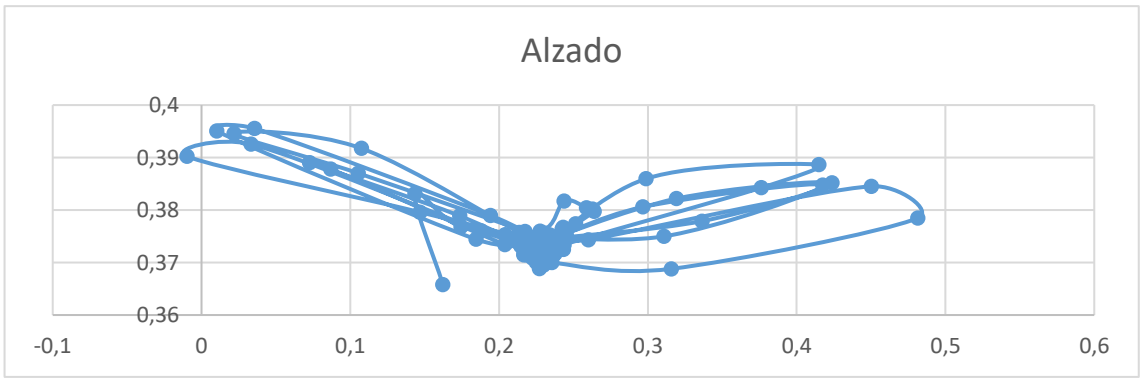


- Paciente 4 (Control VR)

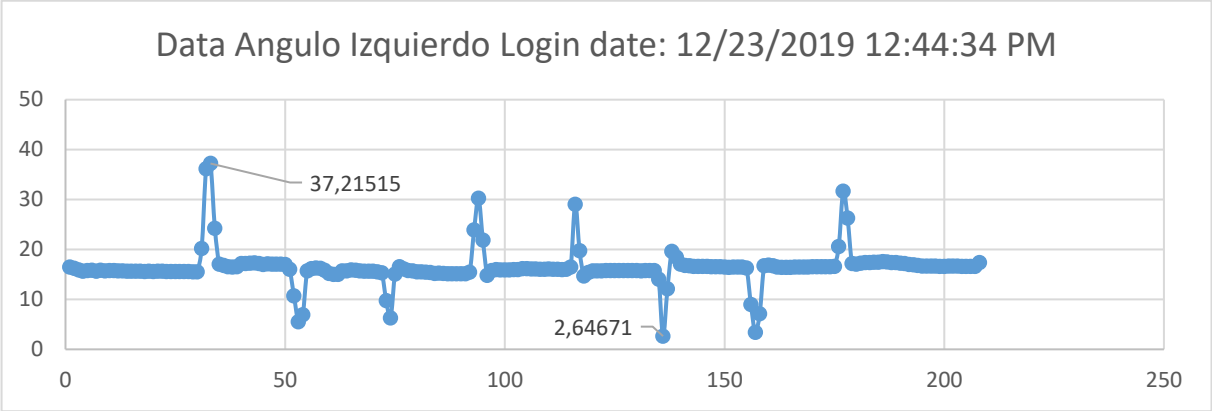


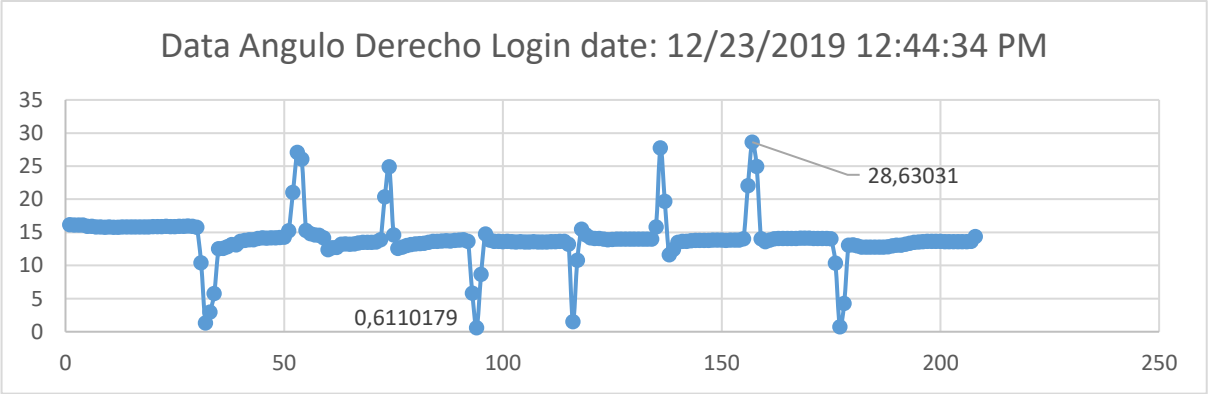


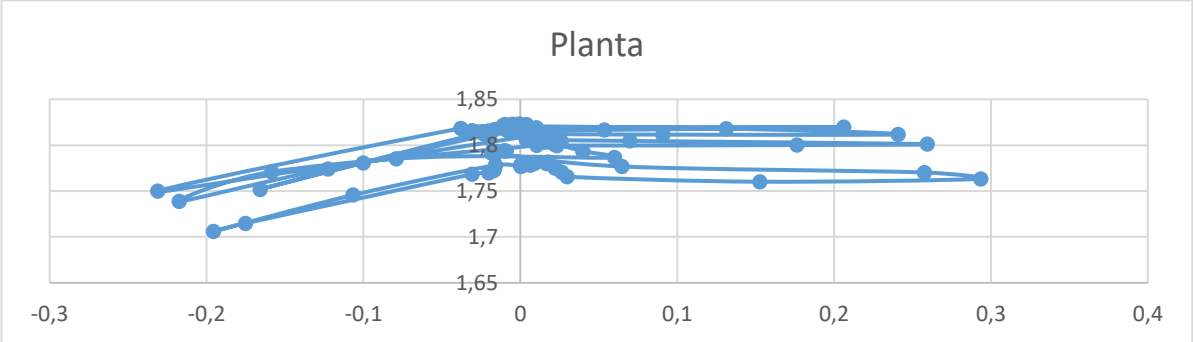


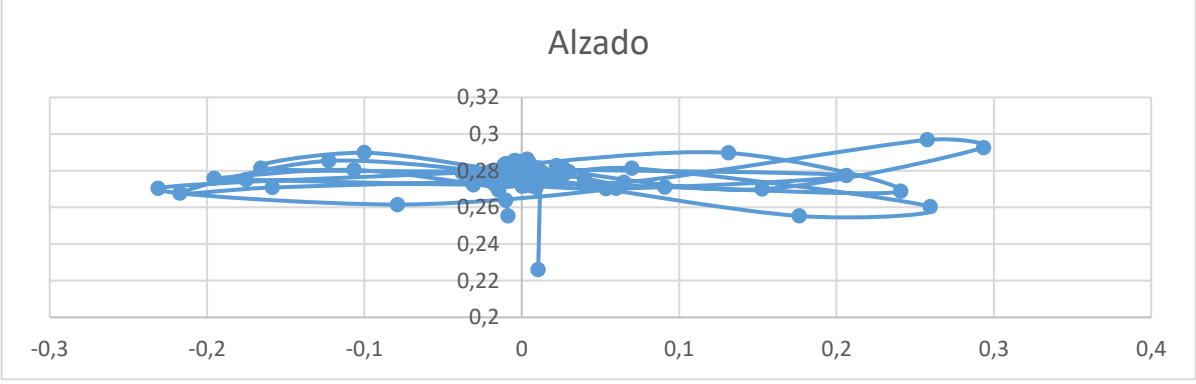


- Paciente 5 (Control VR)

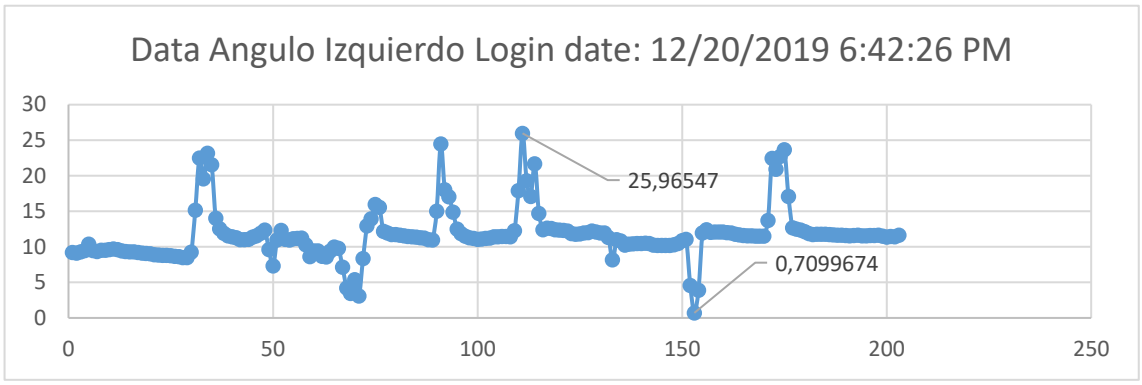


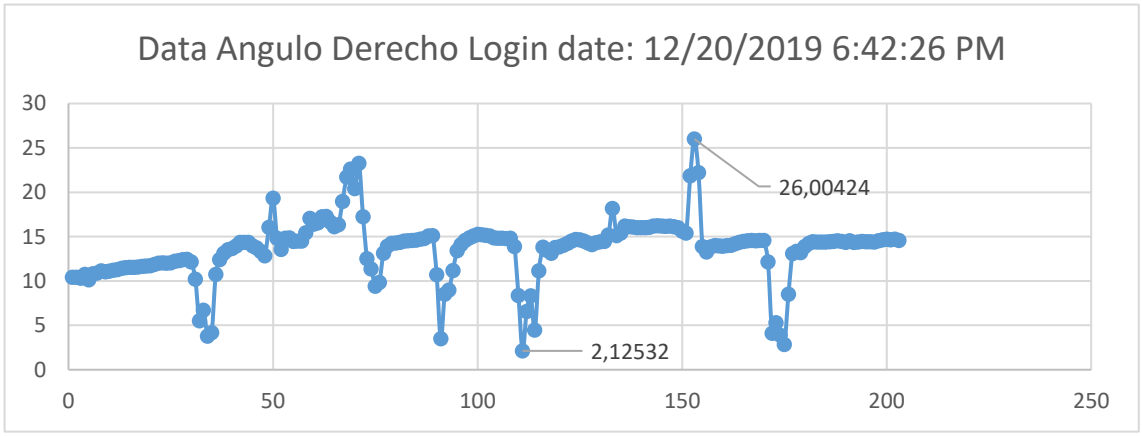


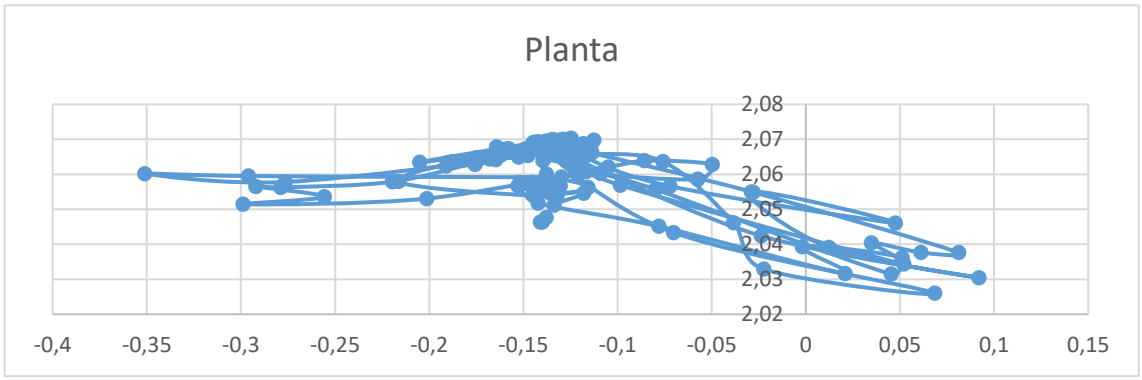


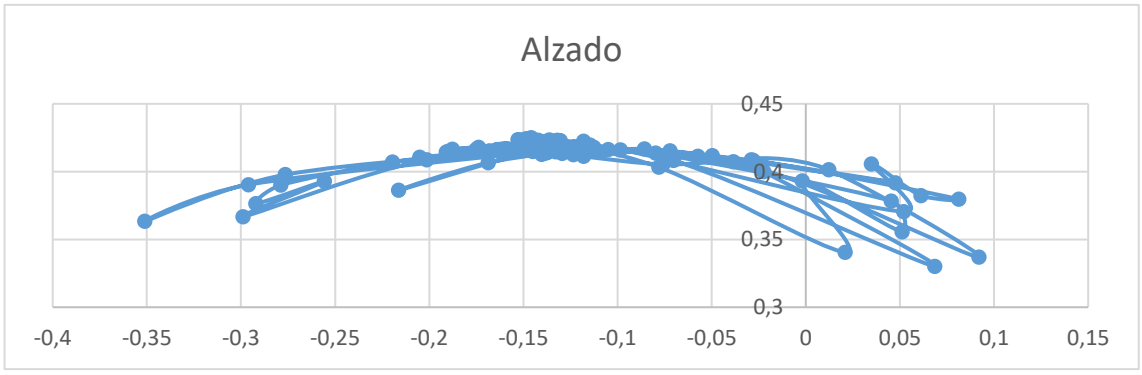


- Paciente 6 (Control VR)

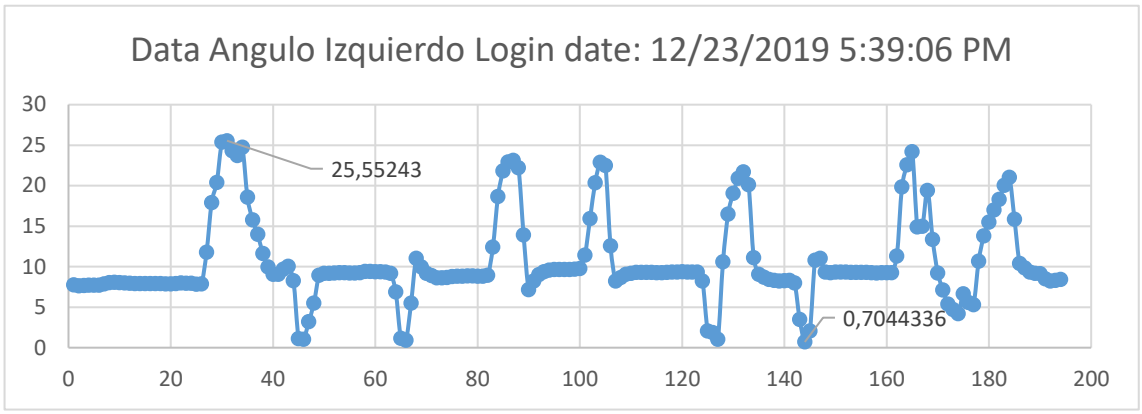


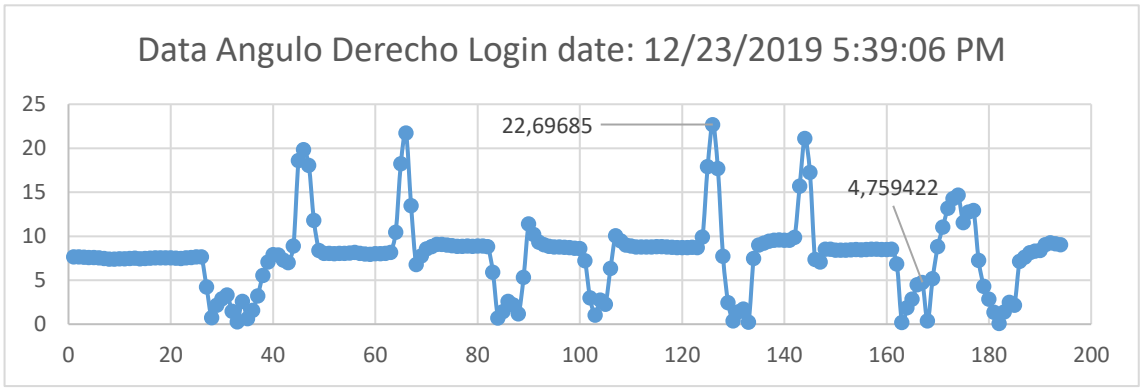


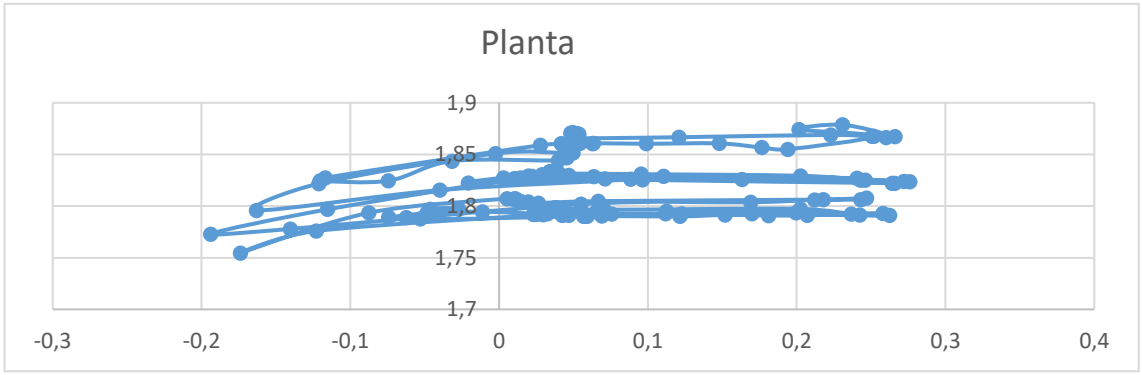


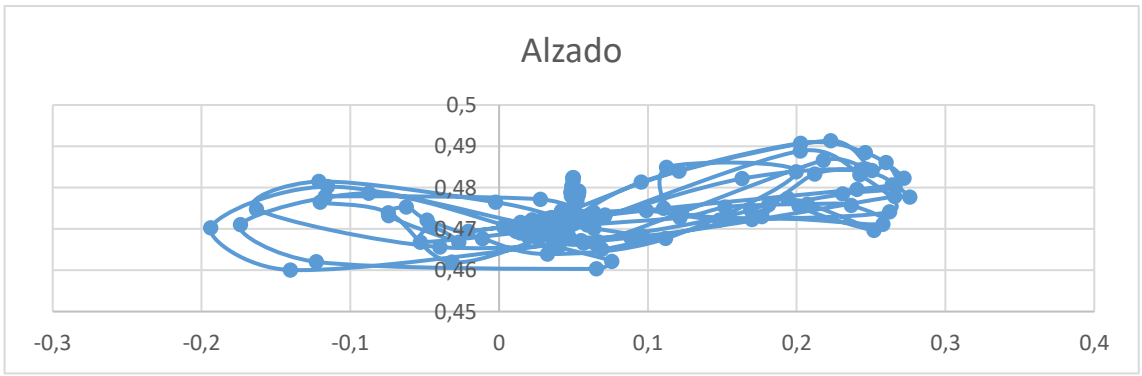


- Paciente 7 (Control VR)

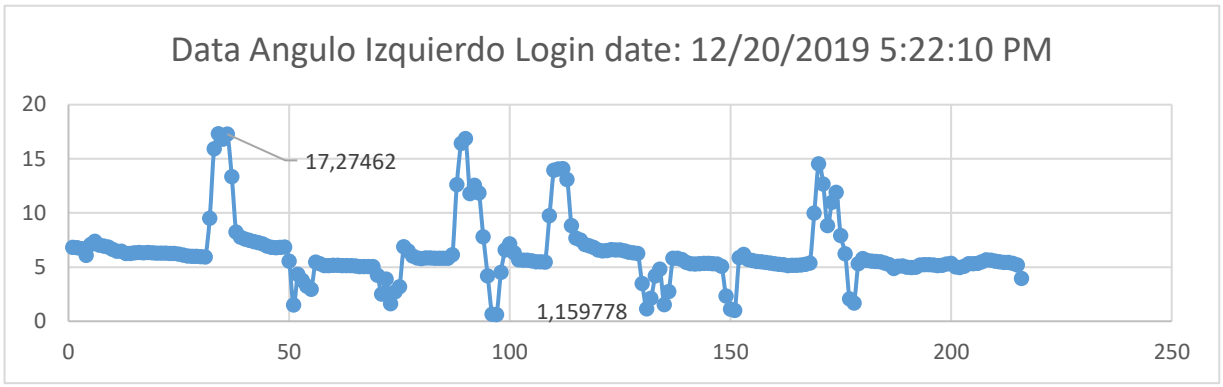


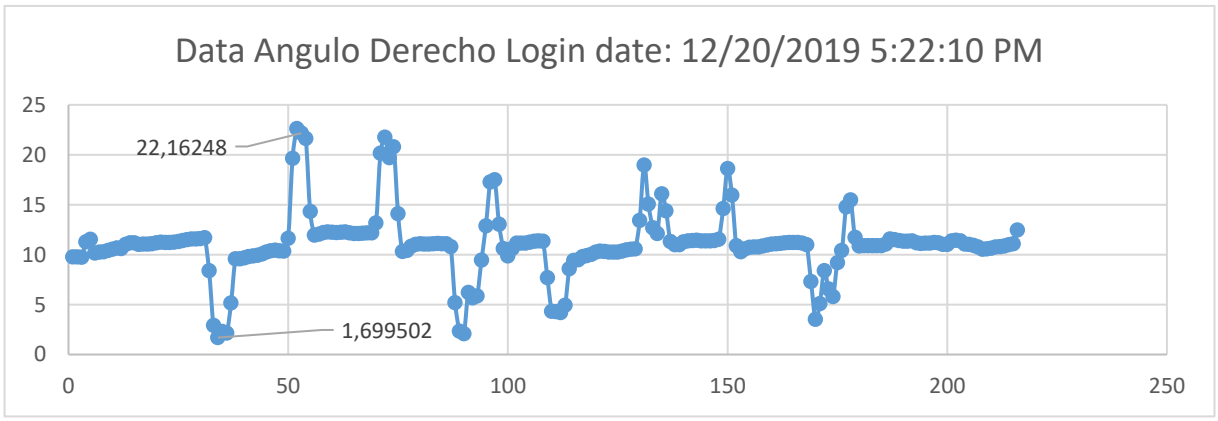


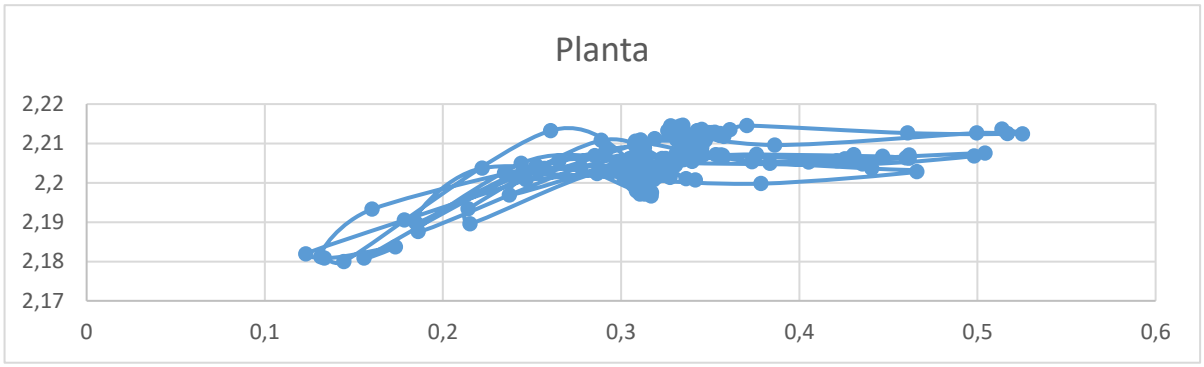


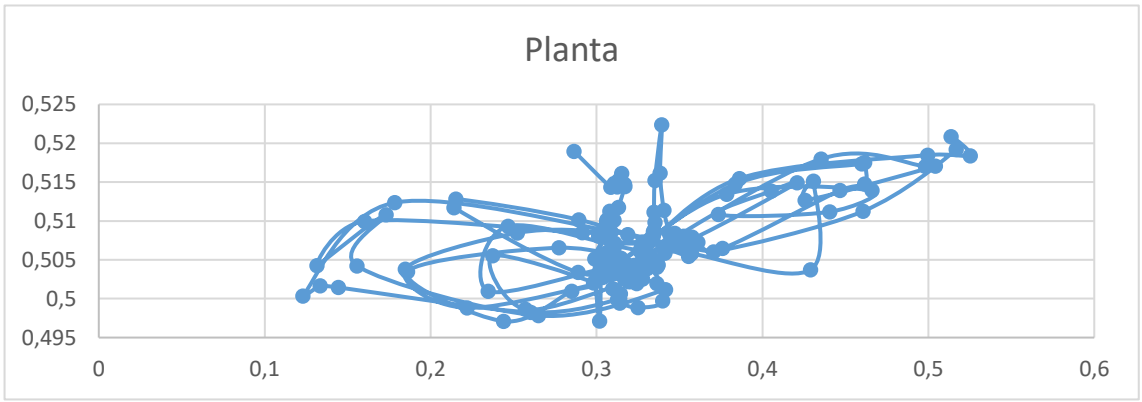


- Paciente 8 (Control VR)

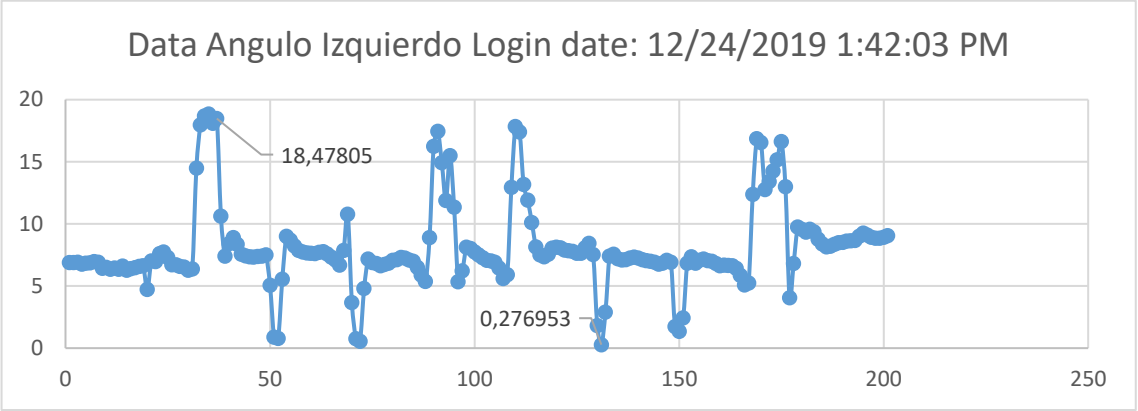


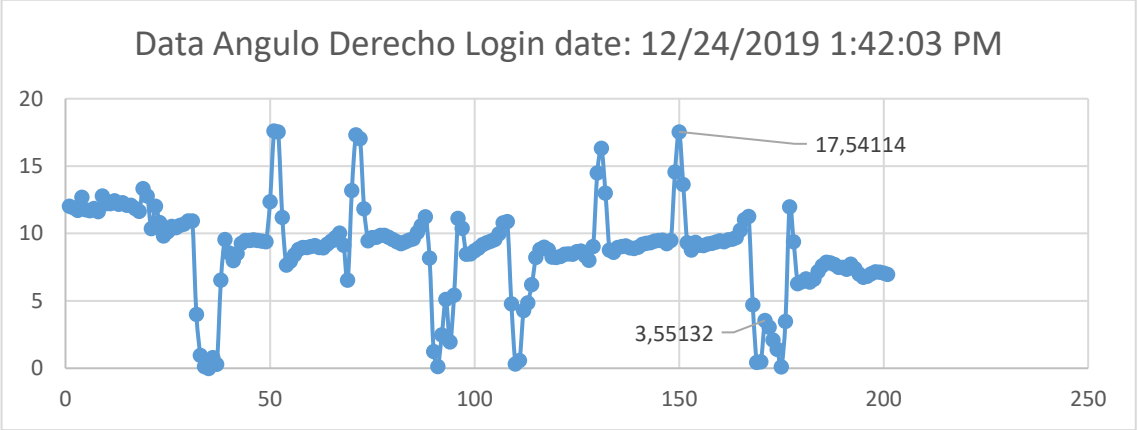


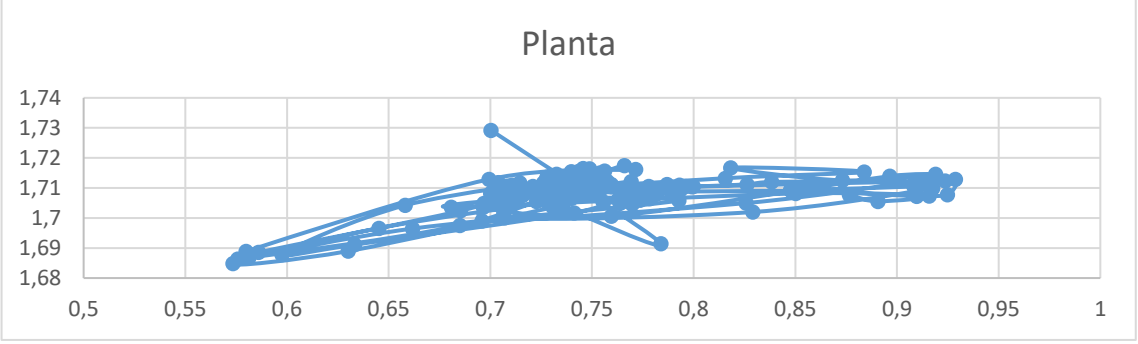


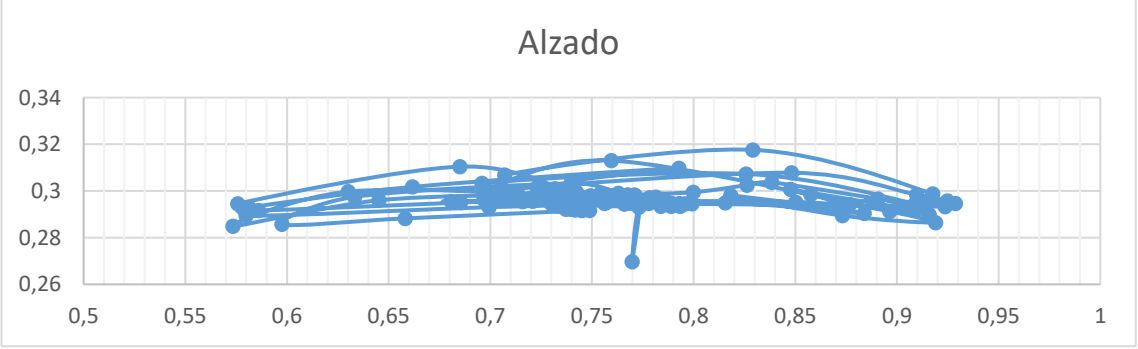


- Paciente 9 (Control VR)

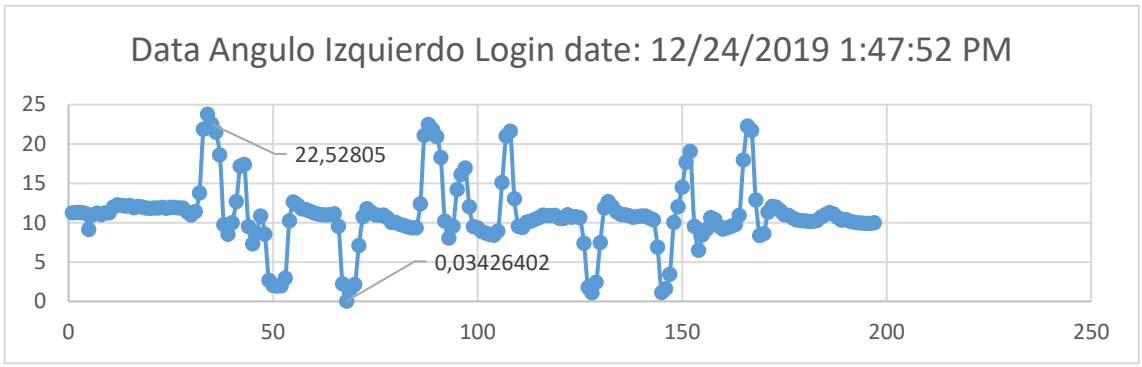


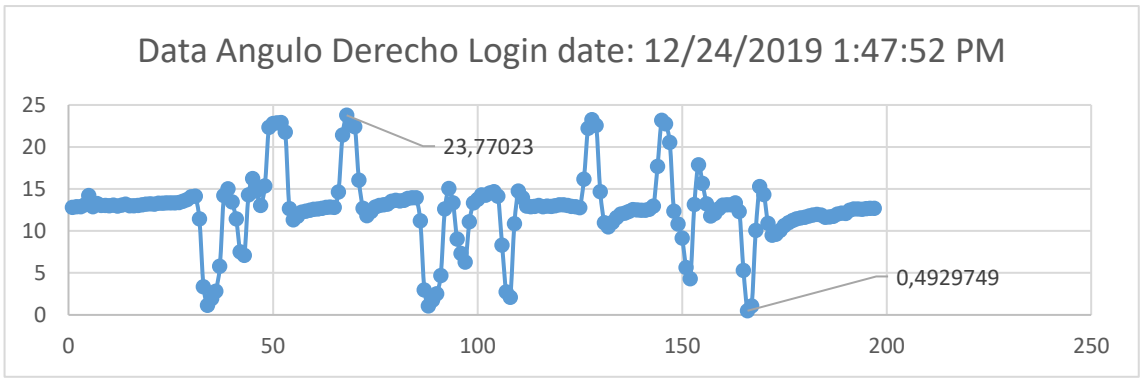


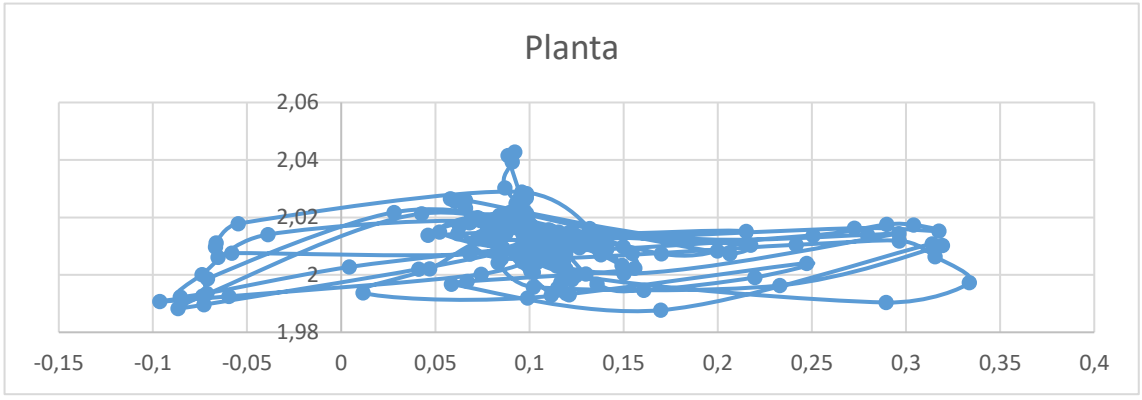


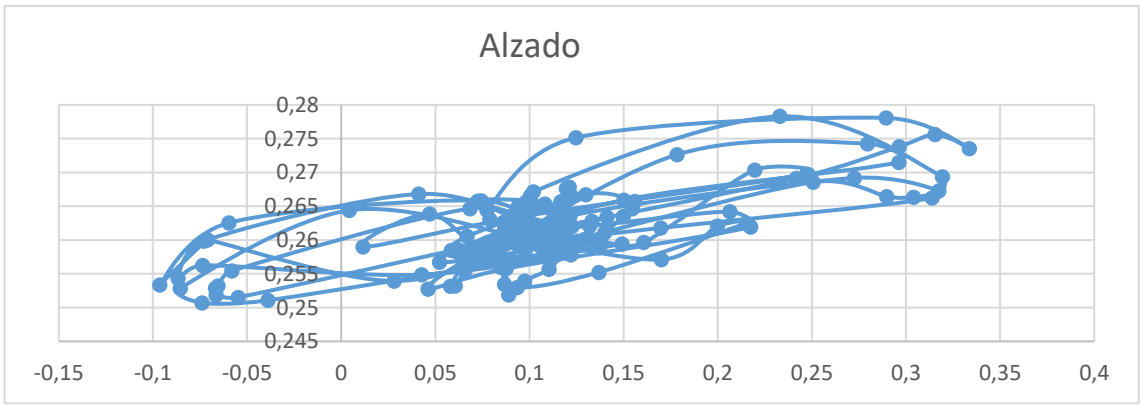


- Paciente 9 (Control Piernas)

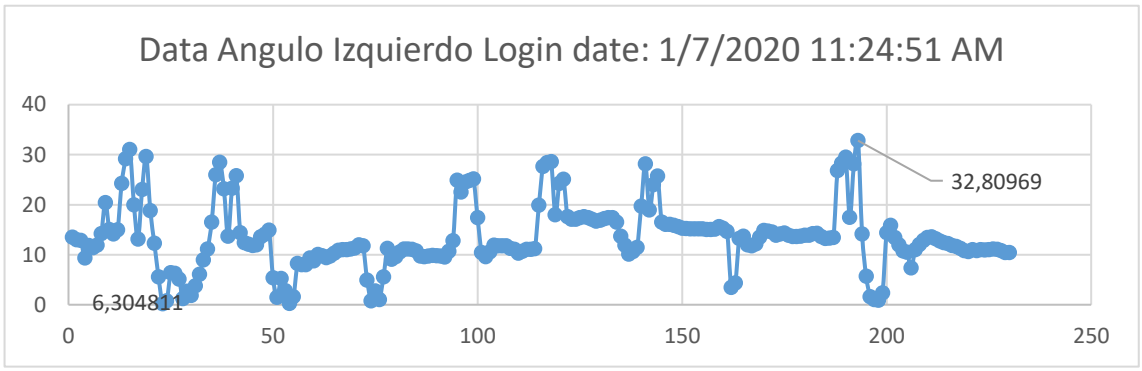


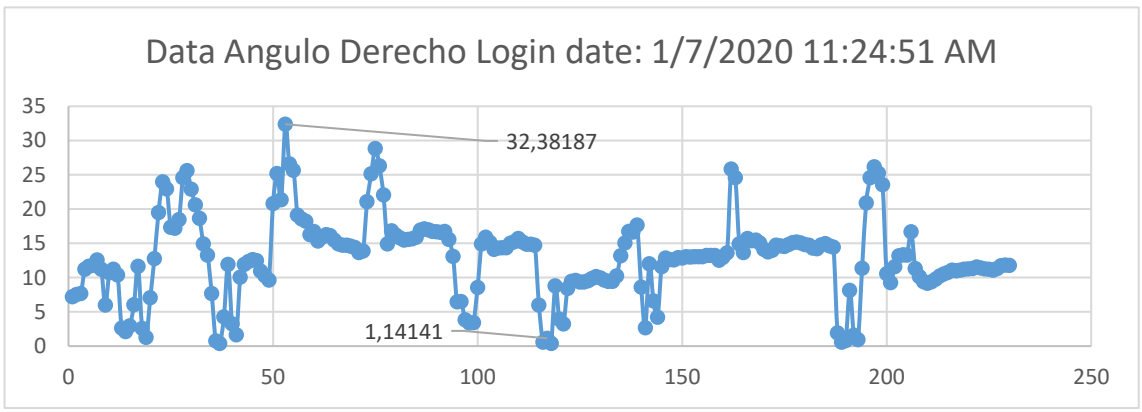


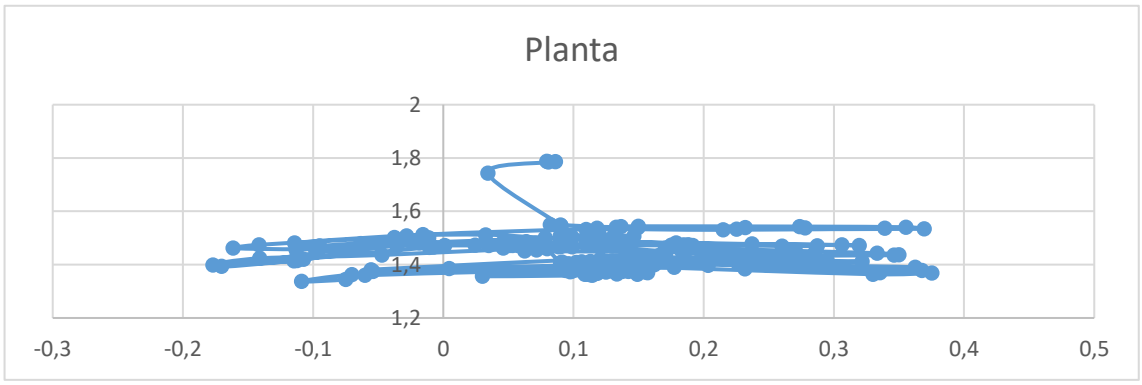


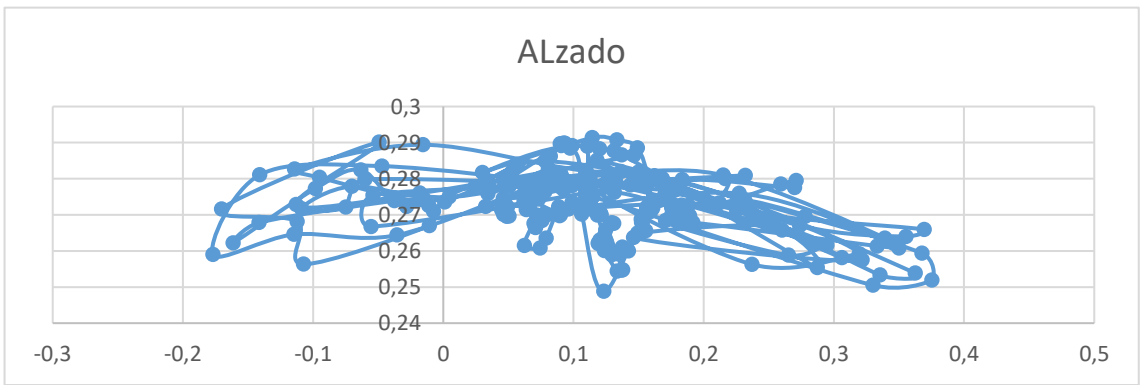


- Paciente 10 (Control VR)

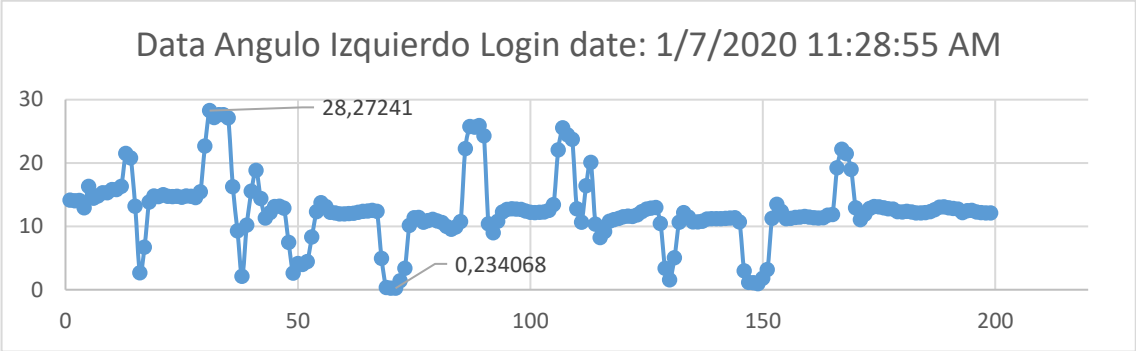


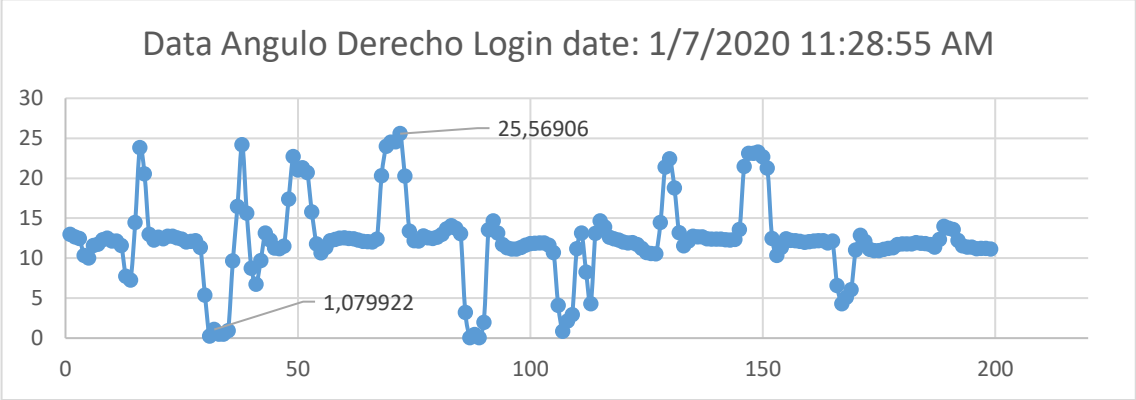


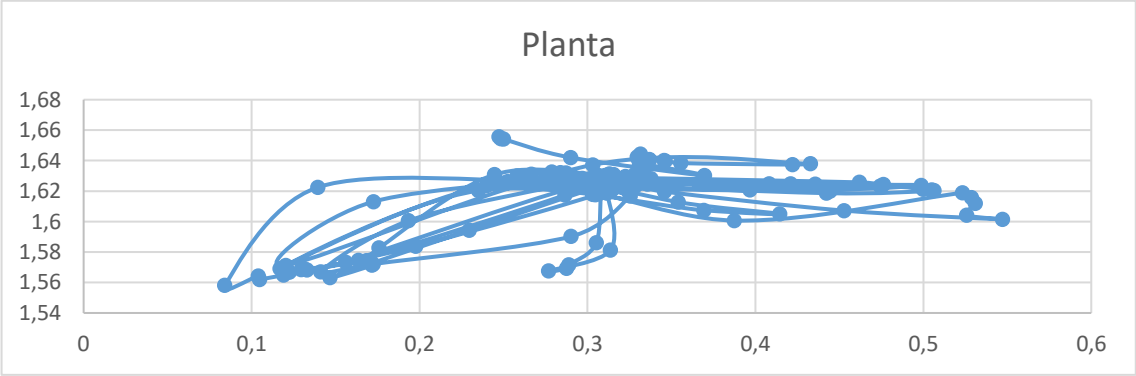


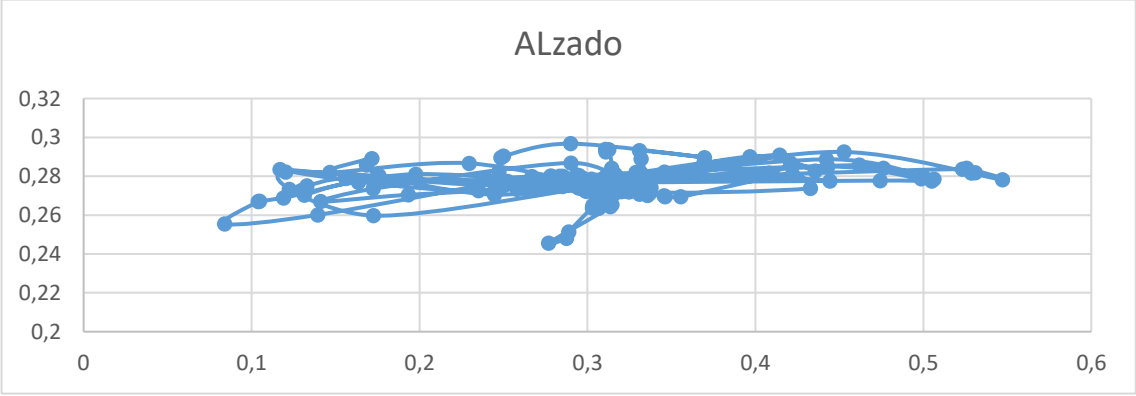


- Paciente 10 (Control Piernas)

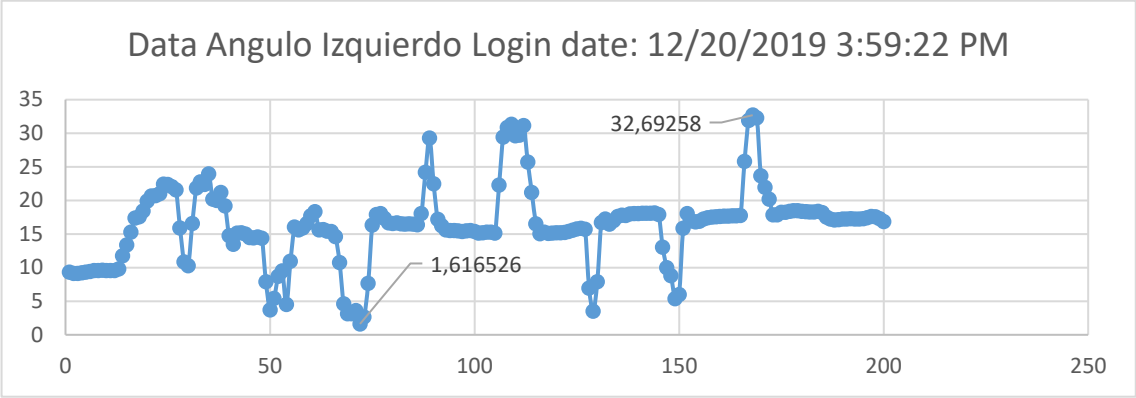


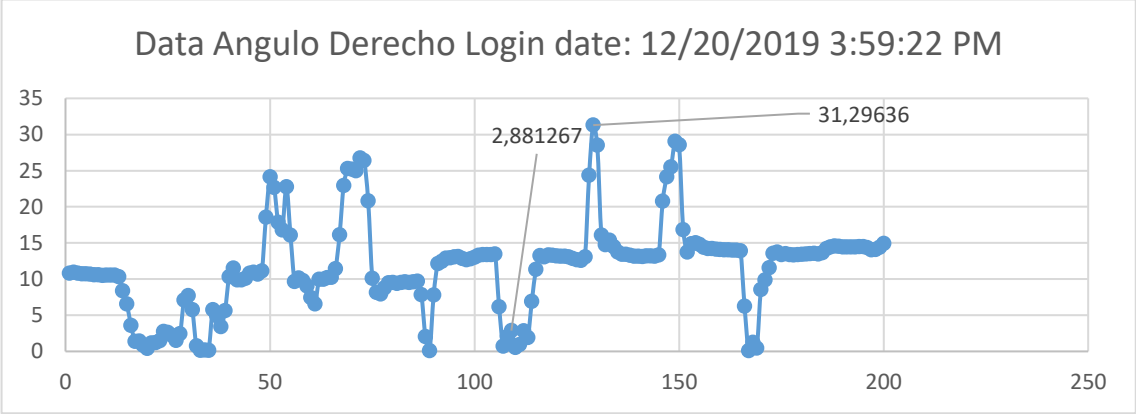


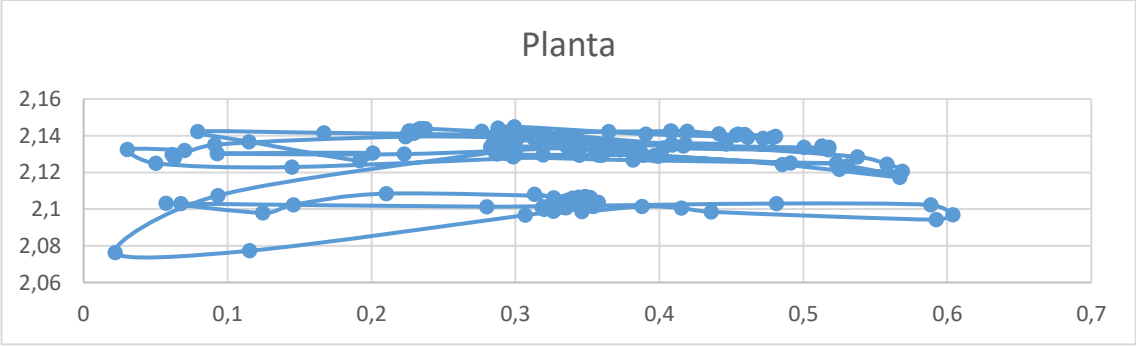


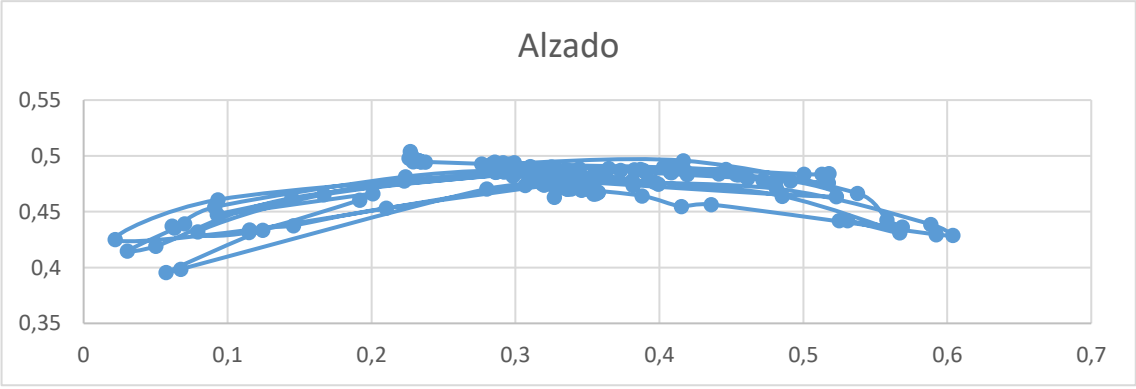


- Paciente 11 (Control VR)

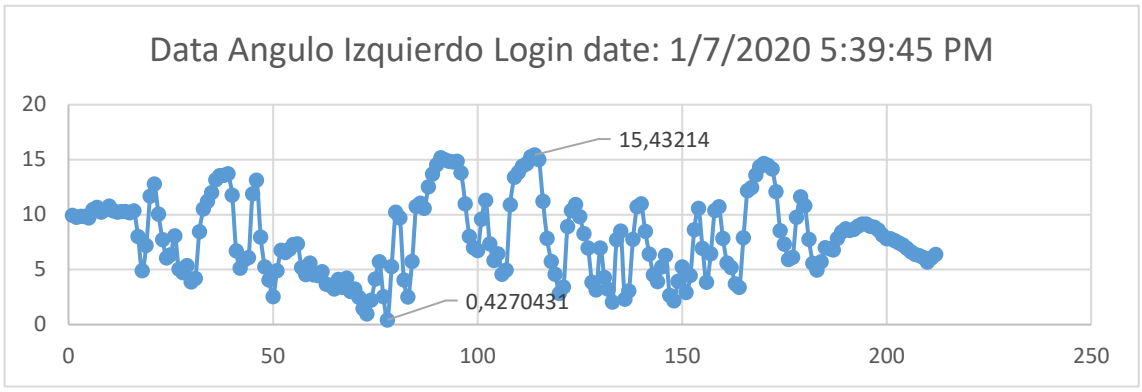


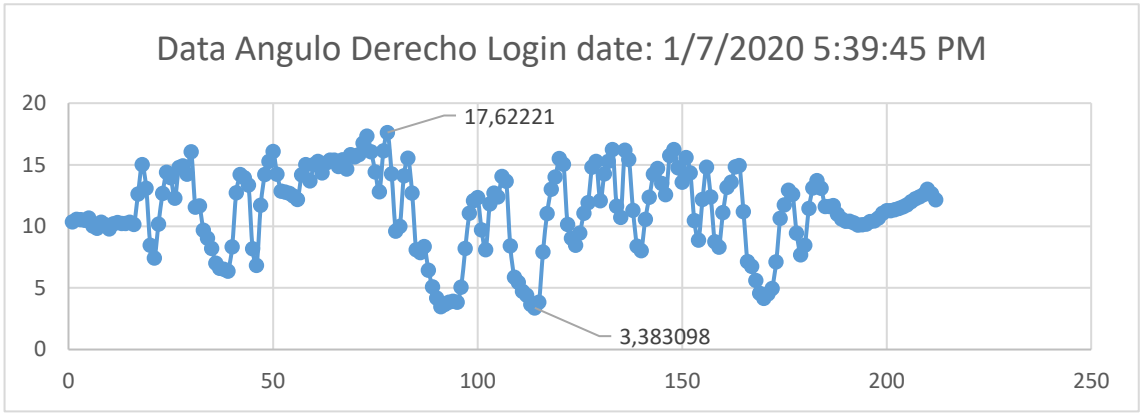


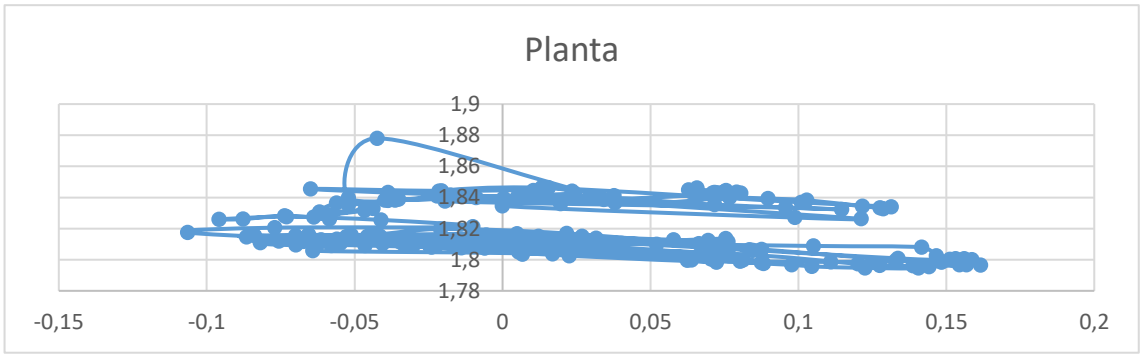


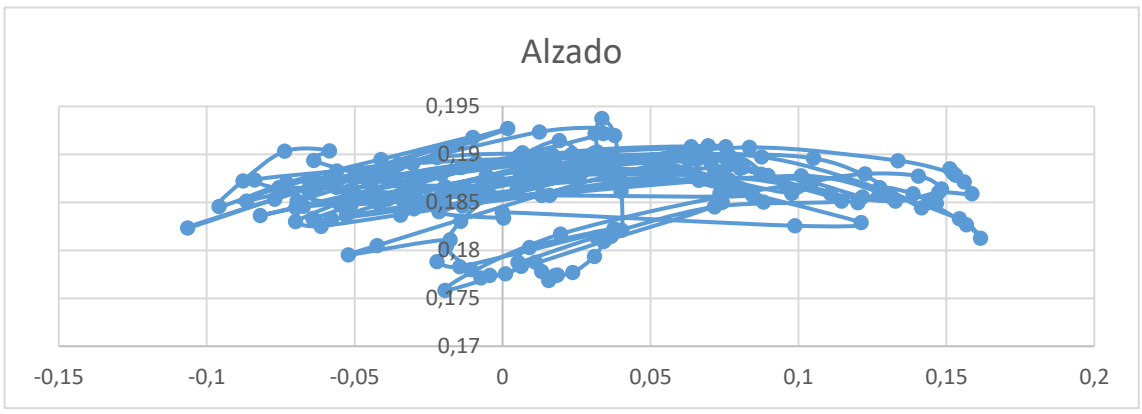


- Paciente 12 (Control VR)

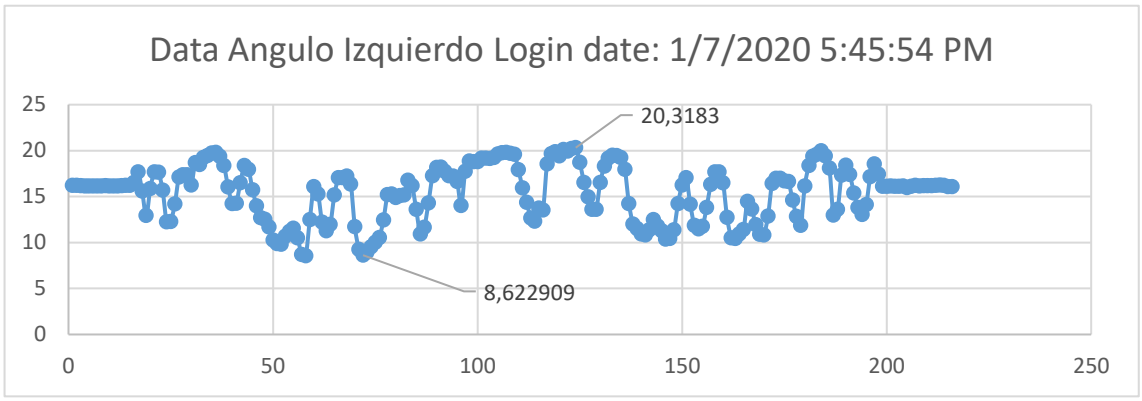


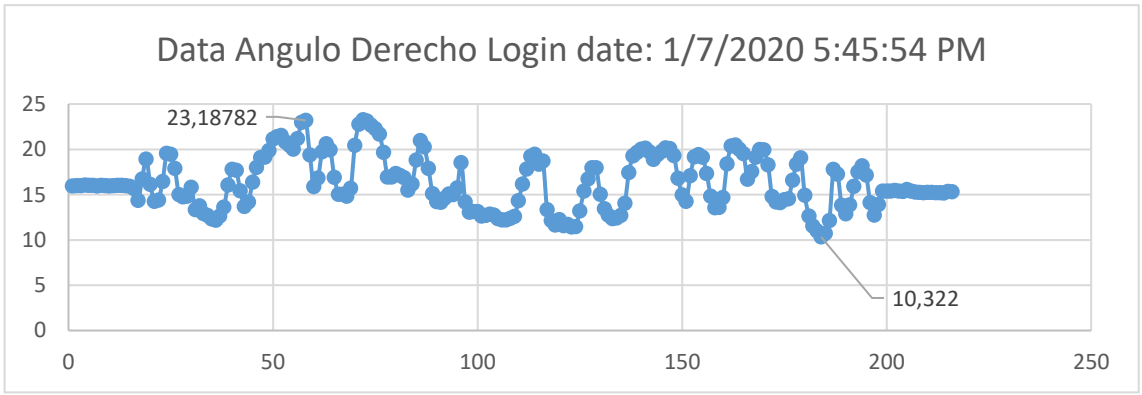


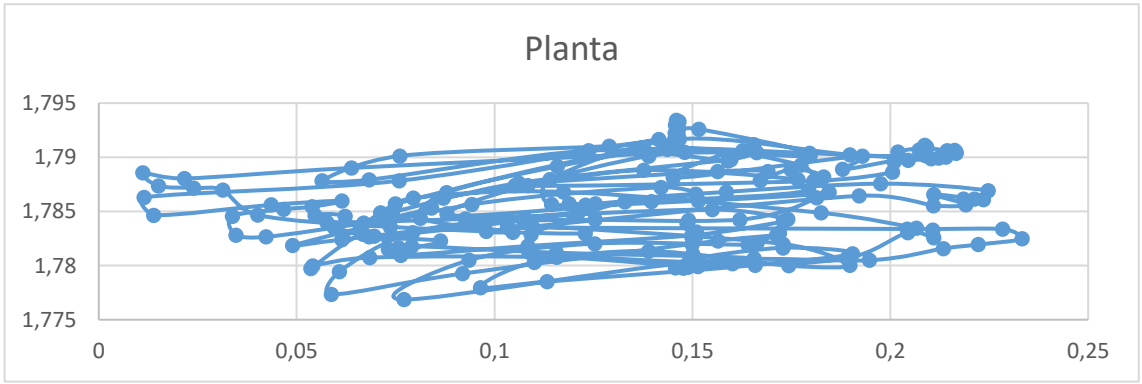


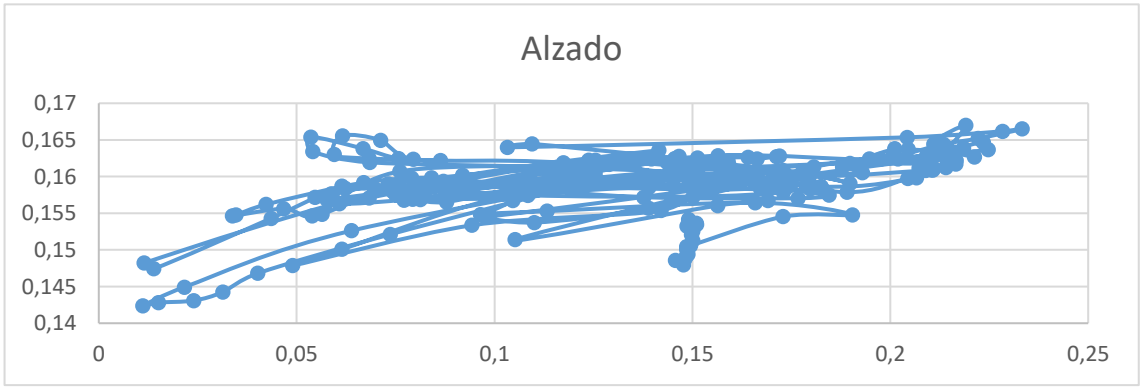


- Paciente 12 (Control Piernas)

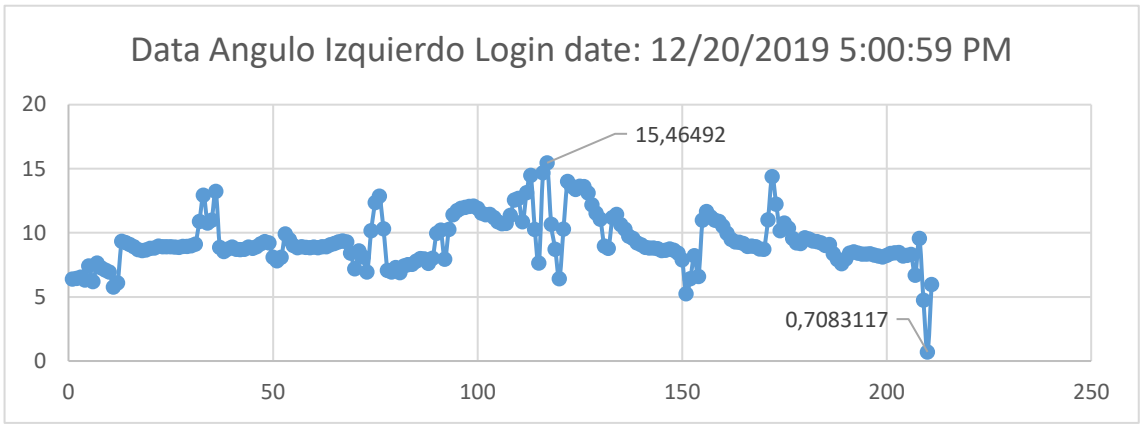


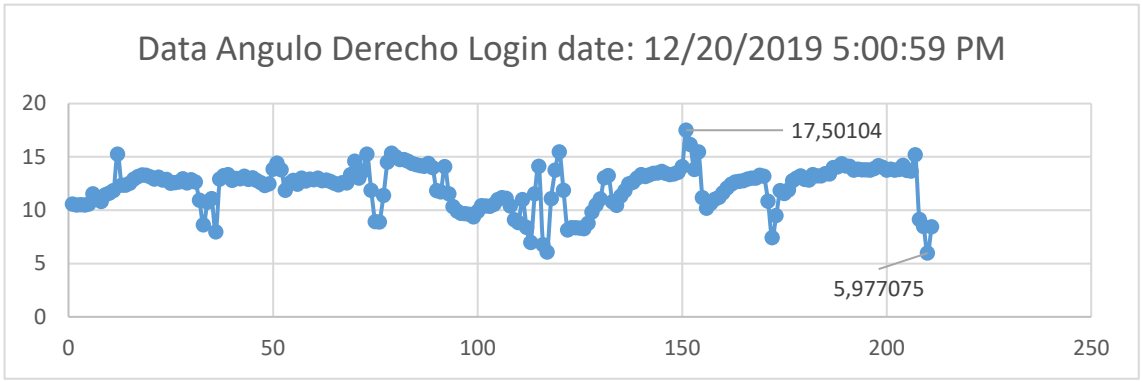


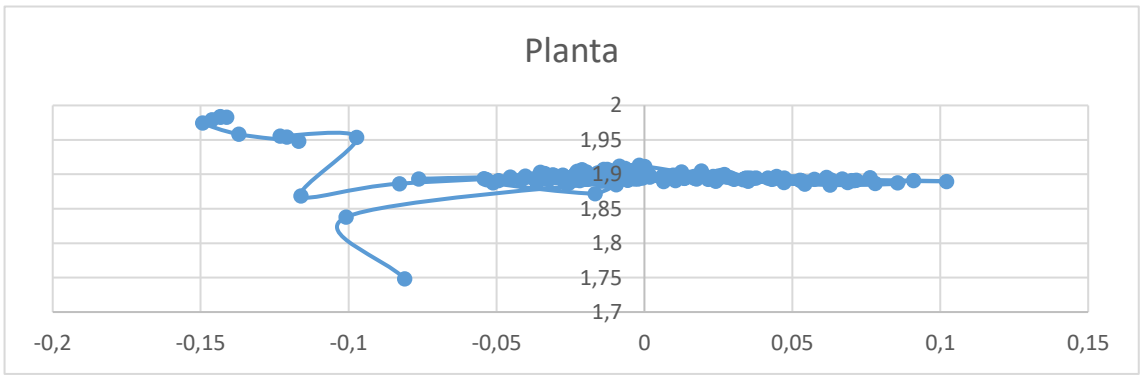


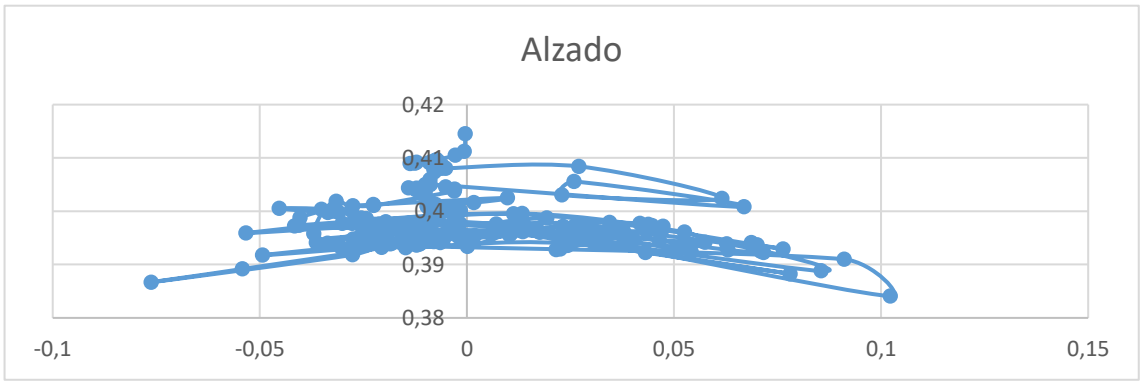


- Paciente 13 (Control VR)

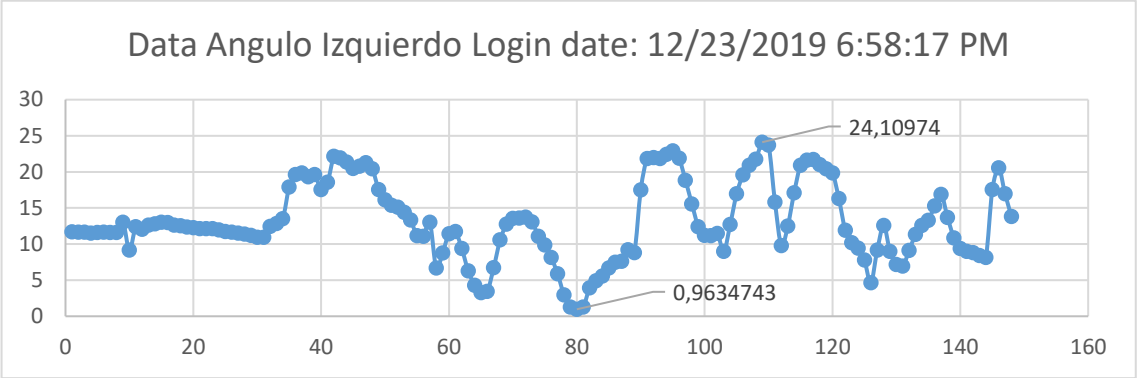


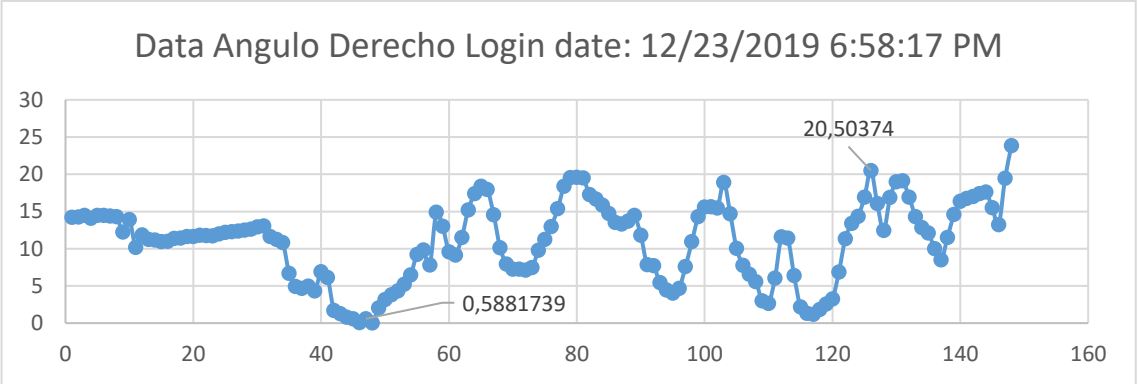


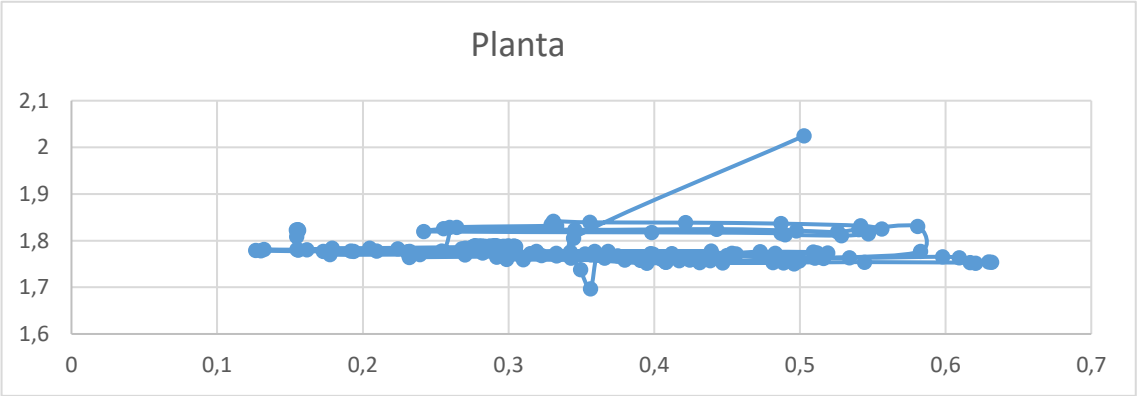


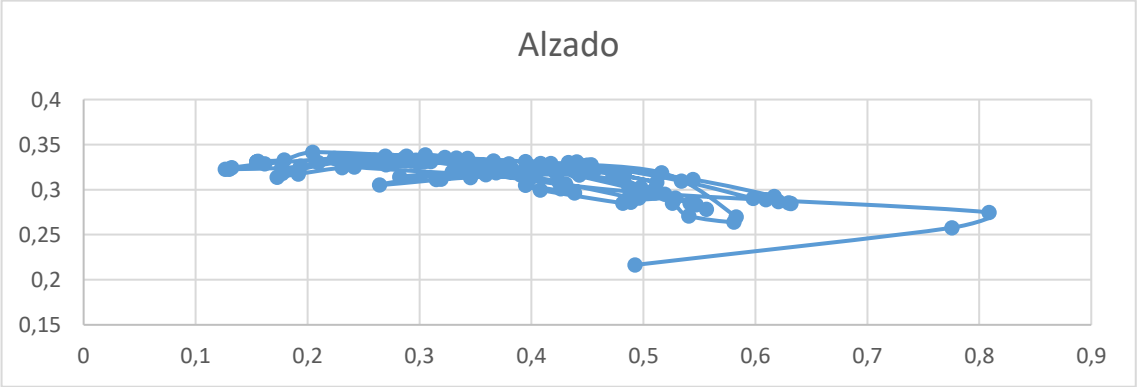


- Paciente 14 (Control VR)

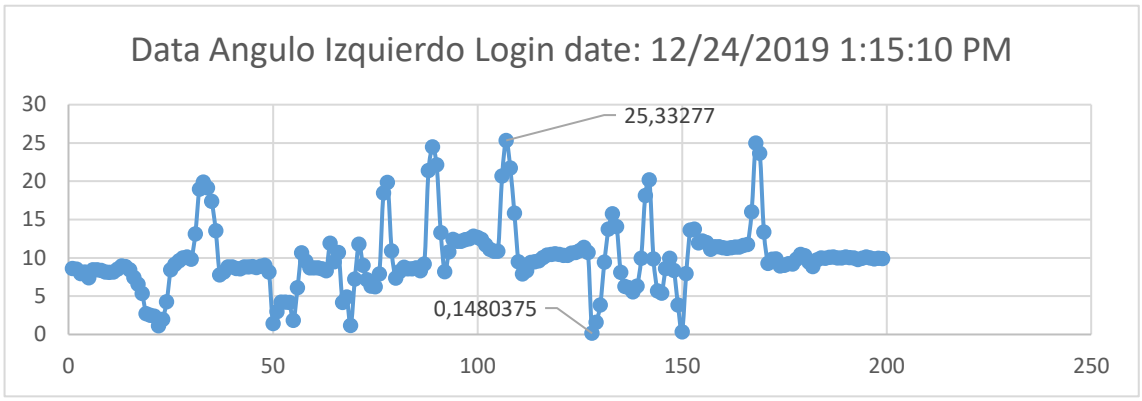


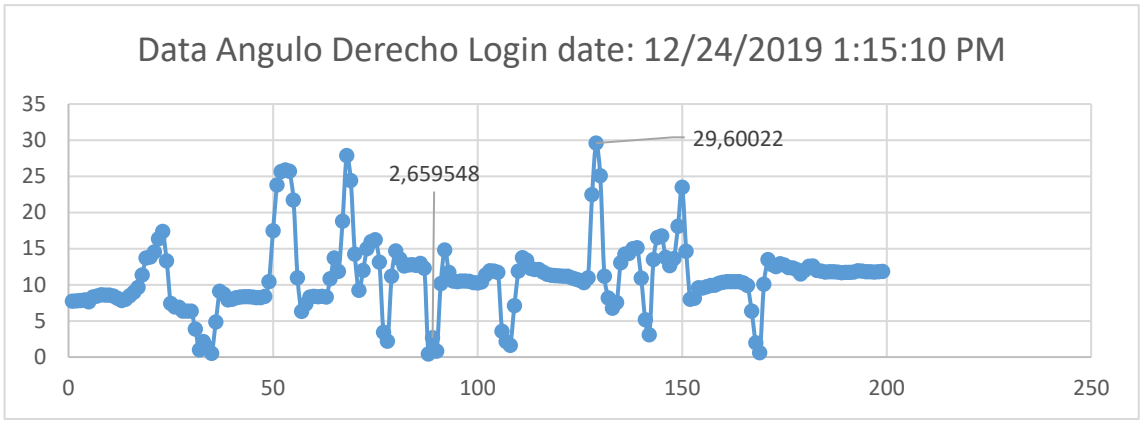


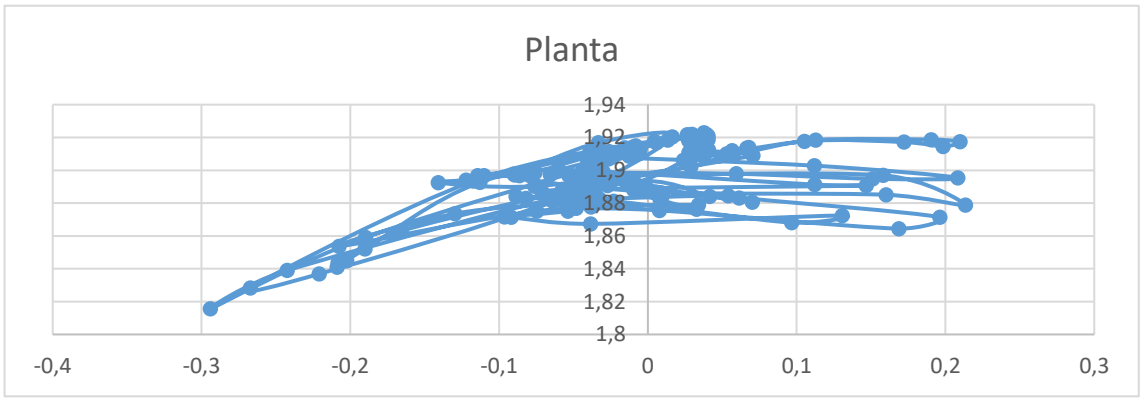


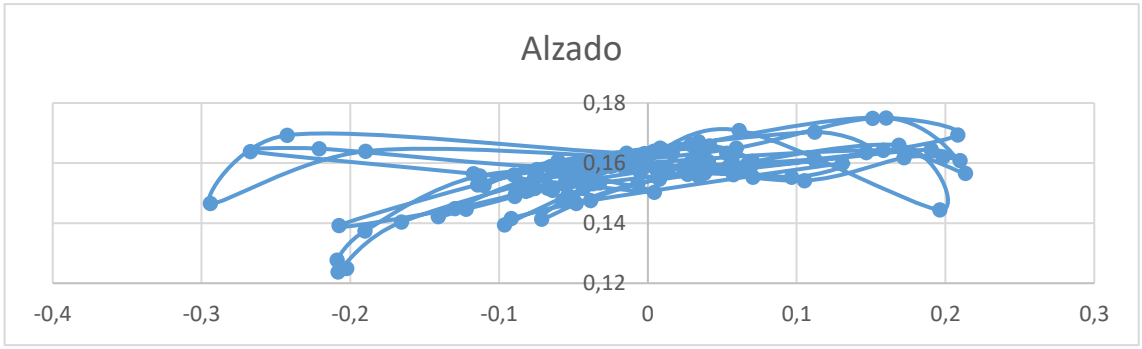


- Paciente 15 (Control VR)

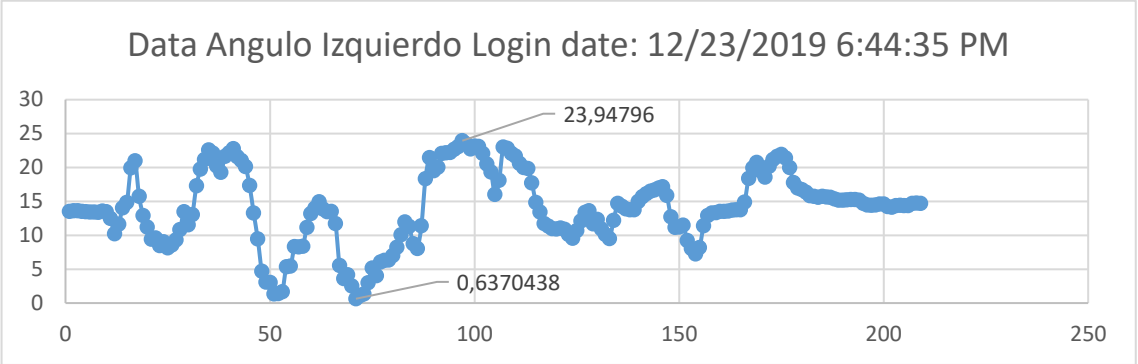


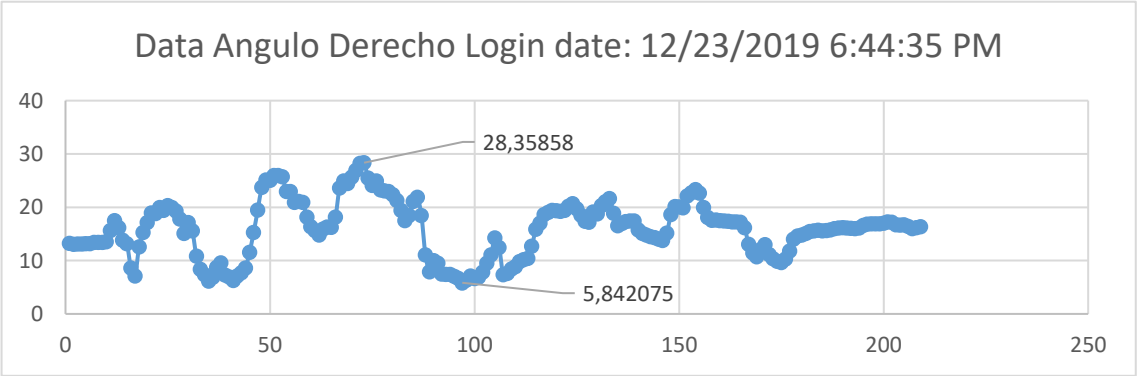


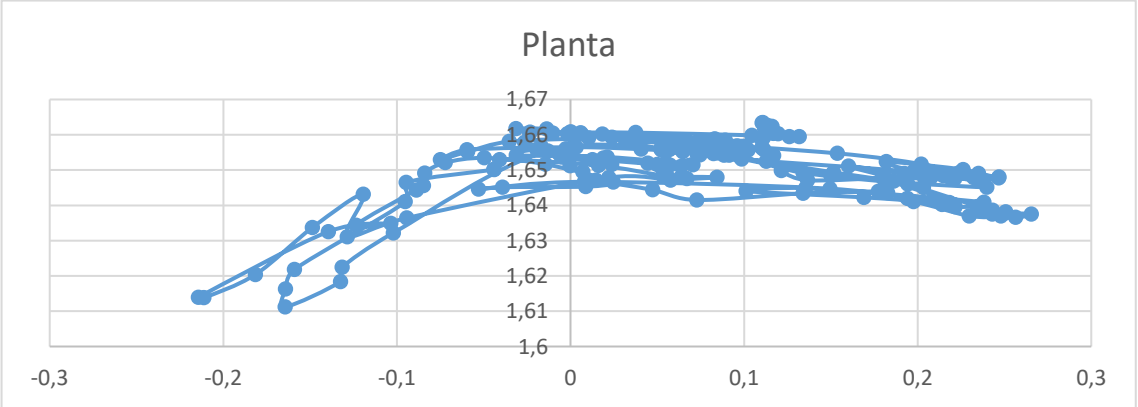


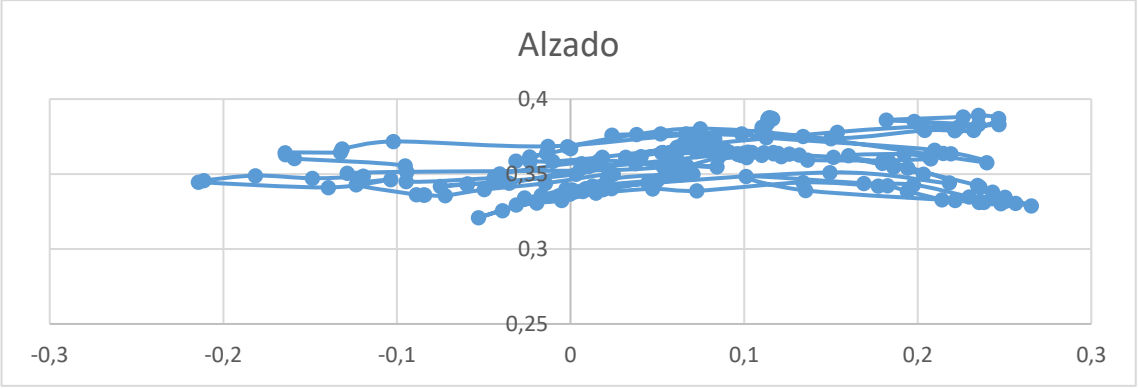


- Paciente 16 (Control VR)

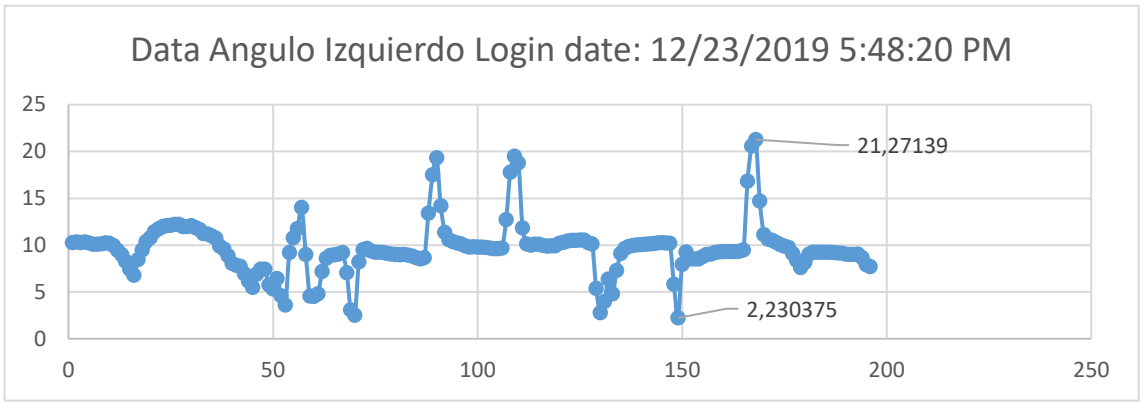


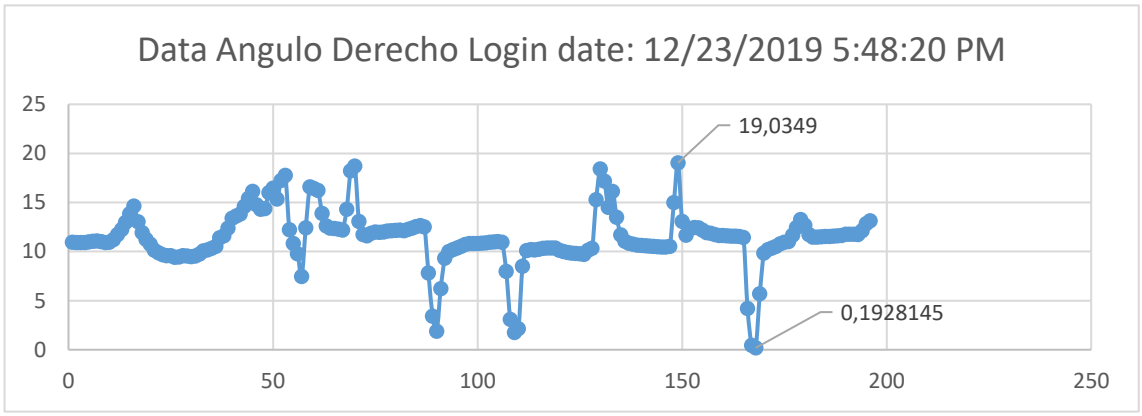


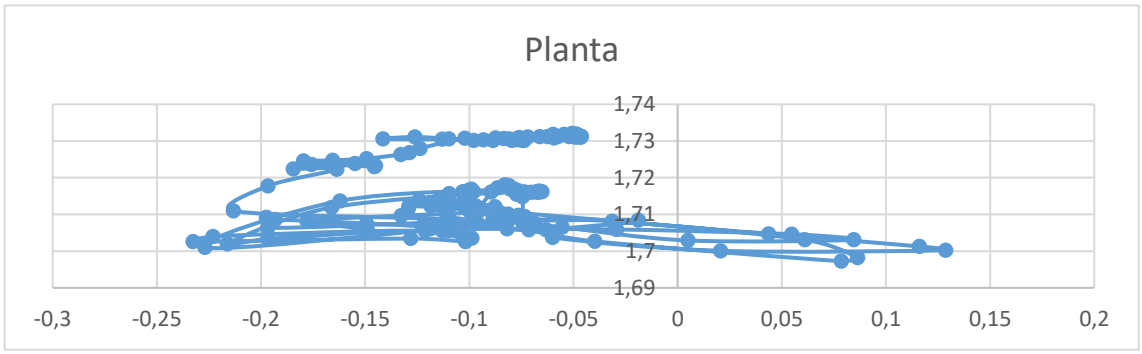


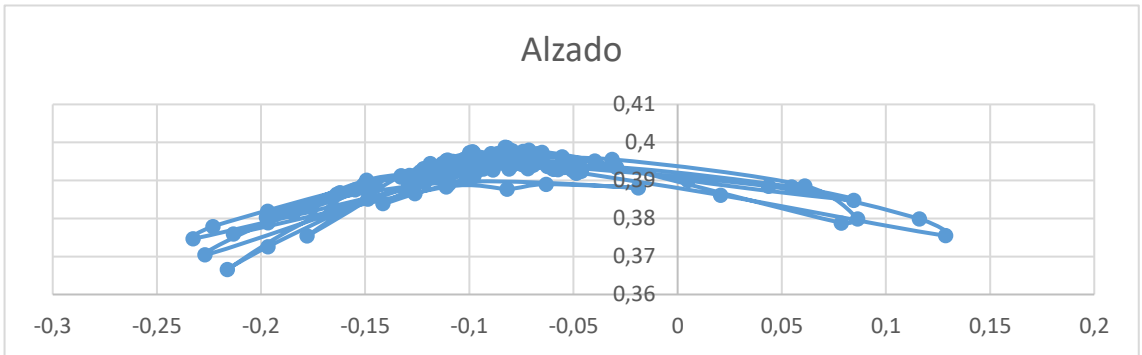


- Paciente 17 (Control VR)

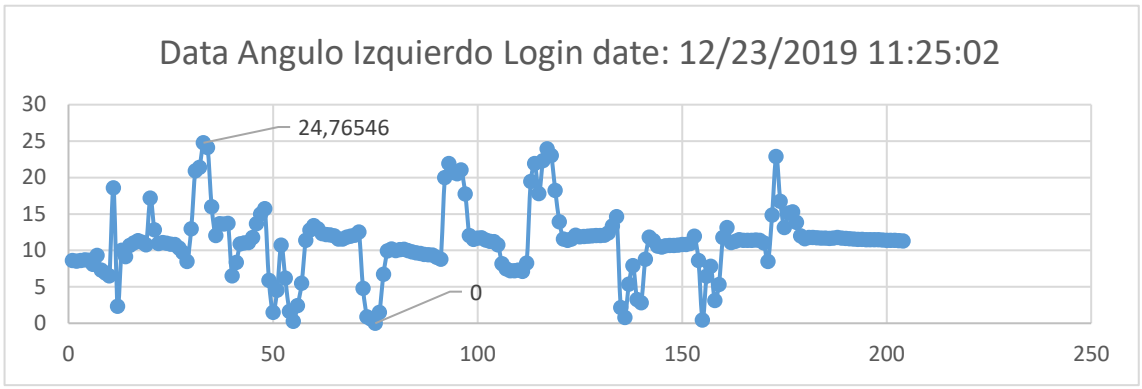


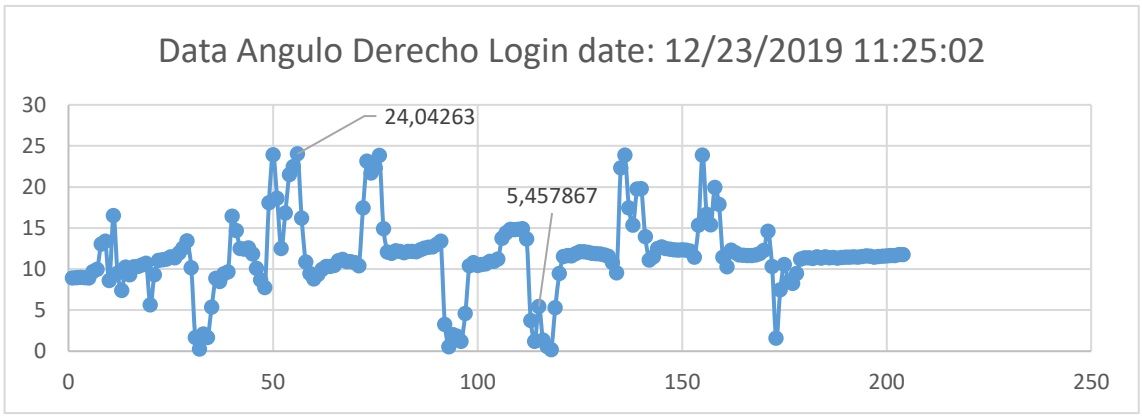


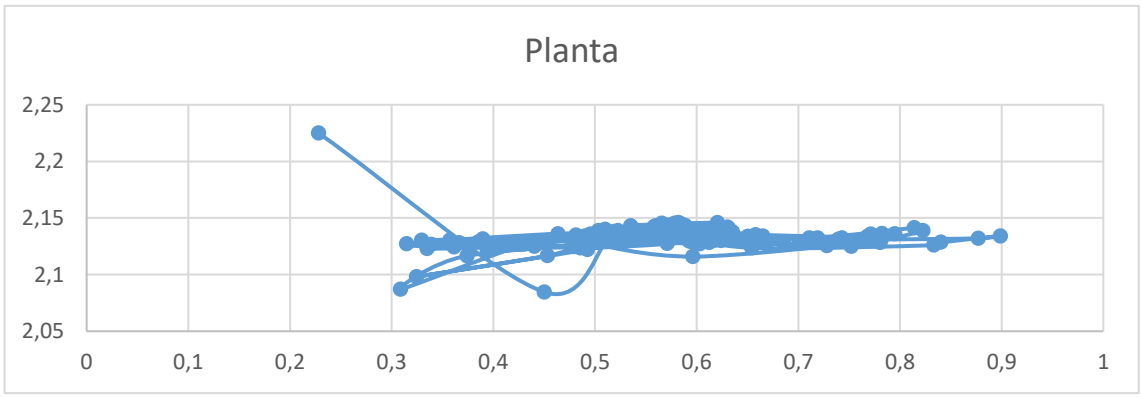


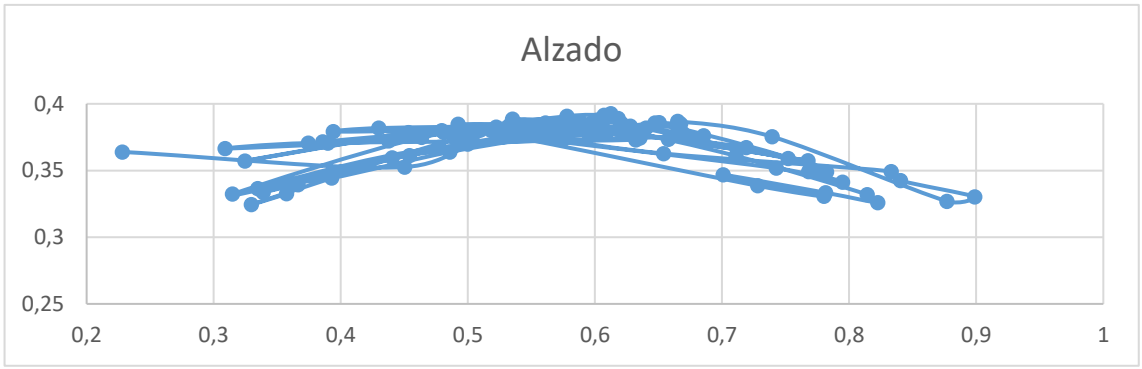


- Paciente 18 (Control VR)

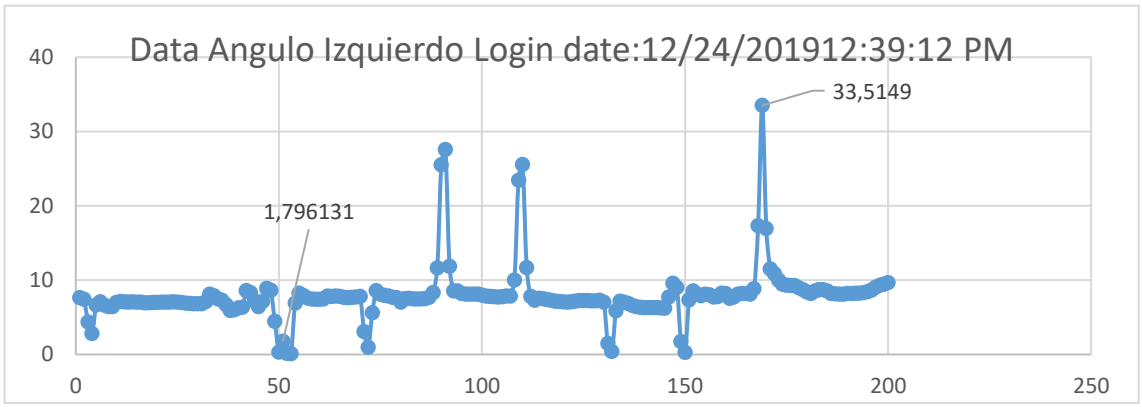


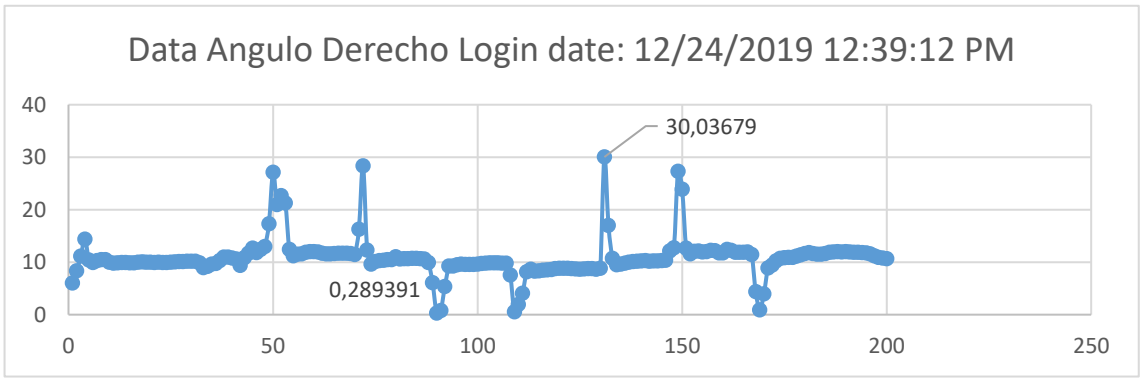


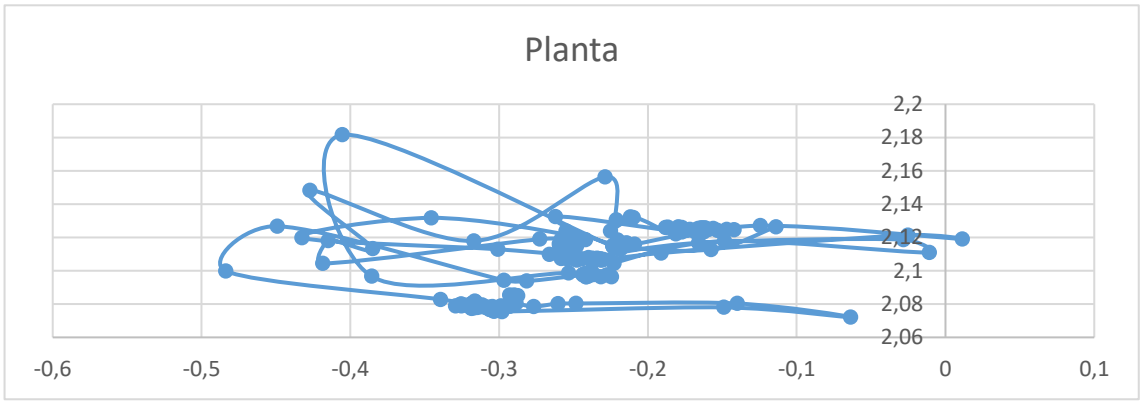


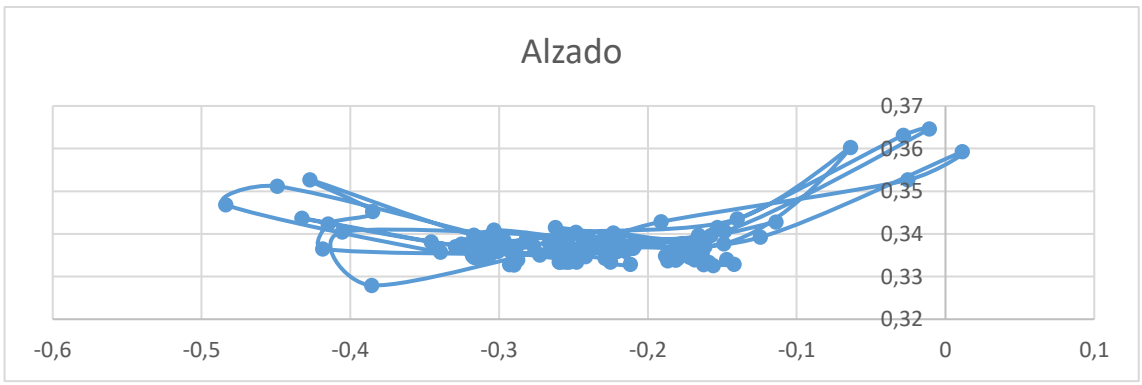


- Paciente 19 (Control VR)

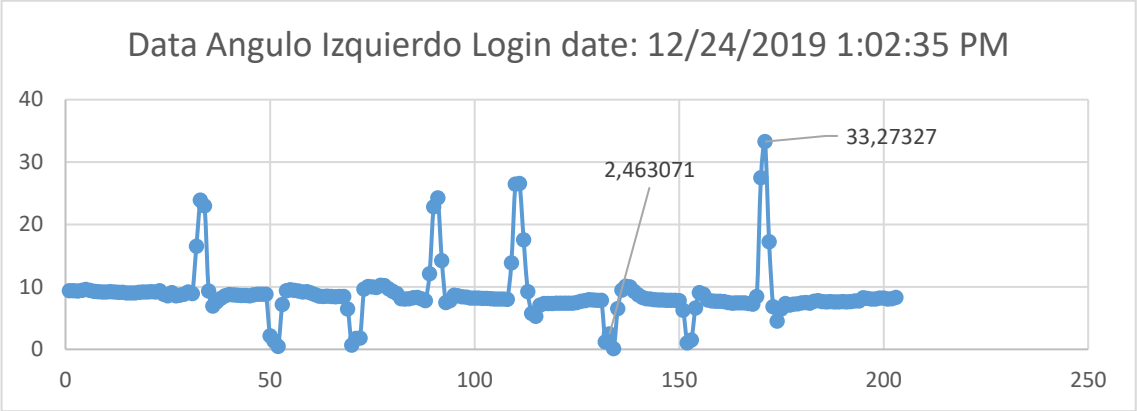


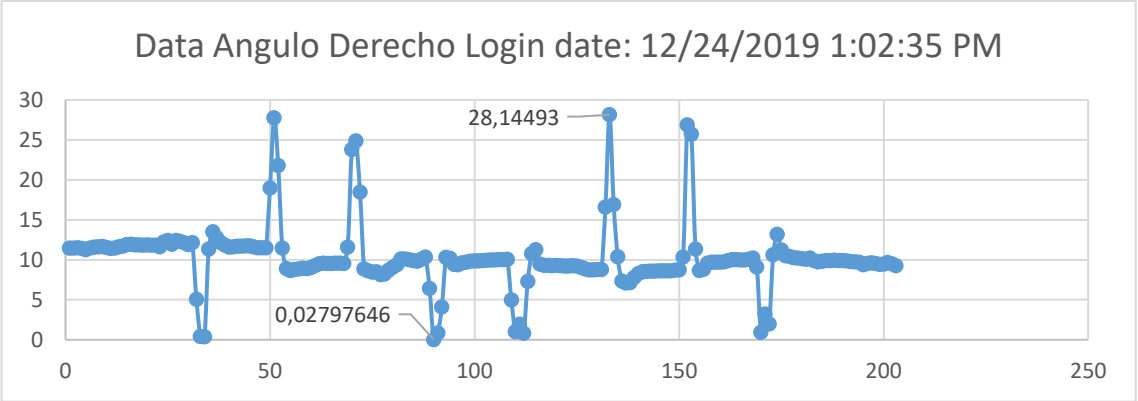


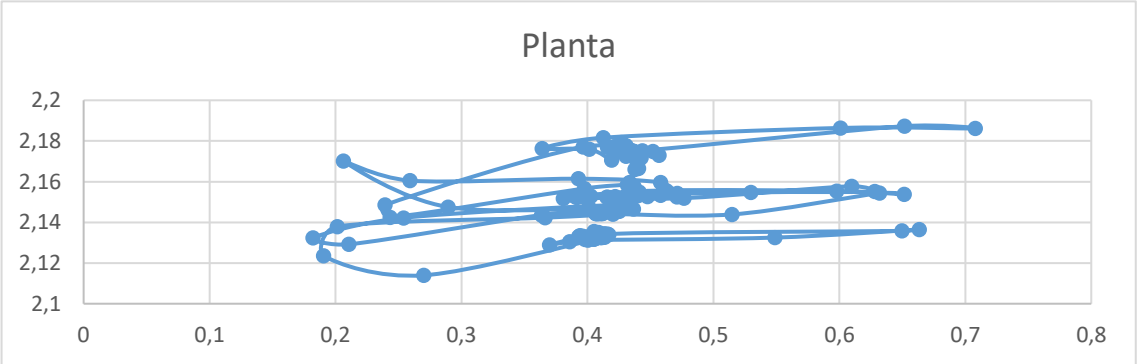


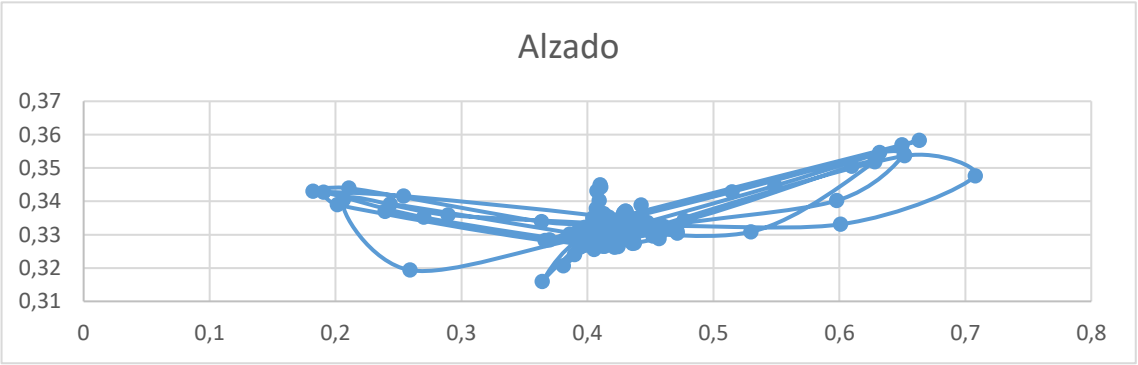


- Paciente 19 (Control Piernas)

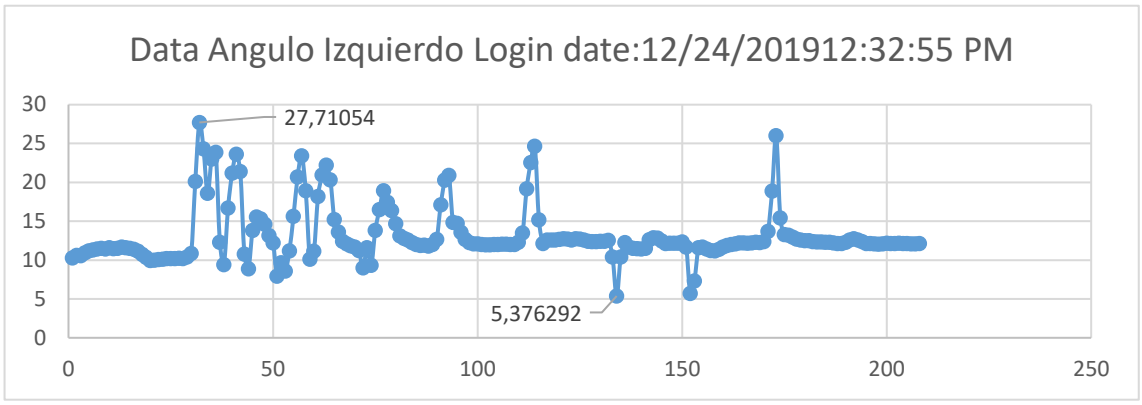


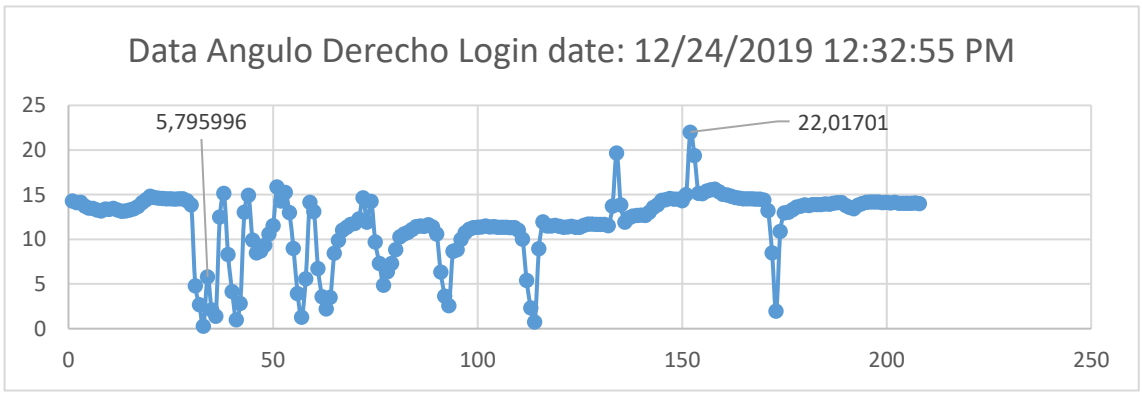


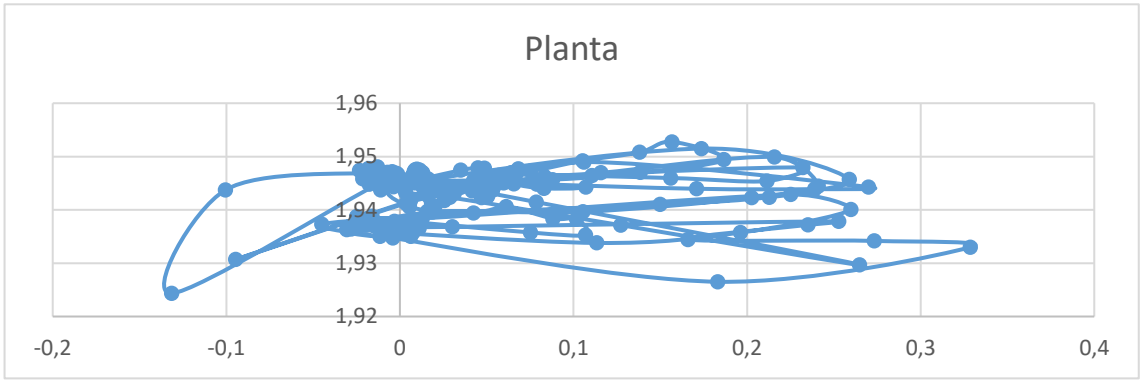


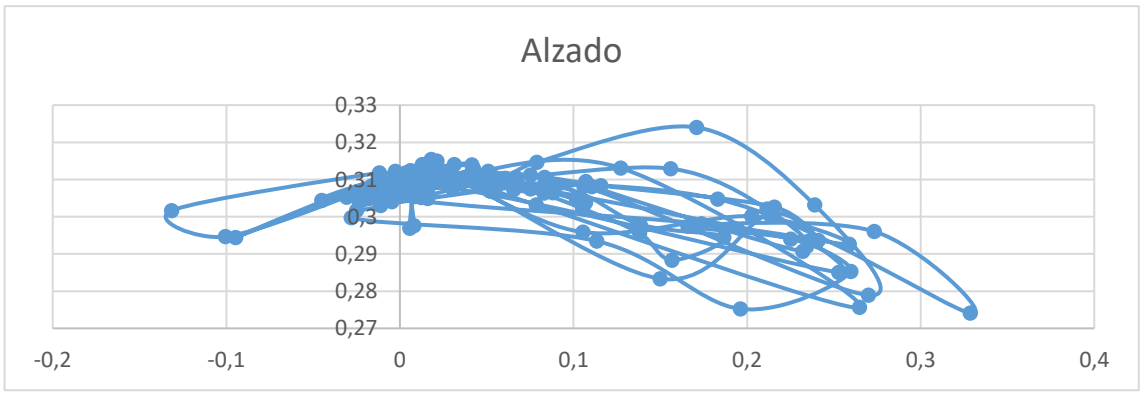


- Paciente 20 (Control VR)

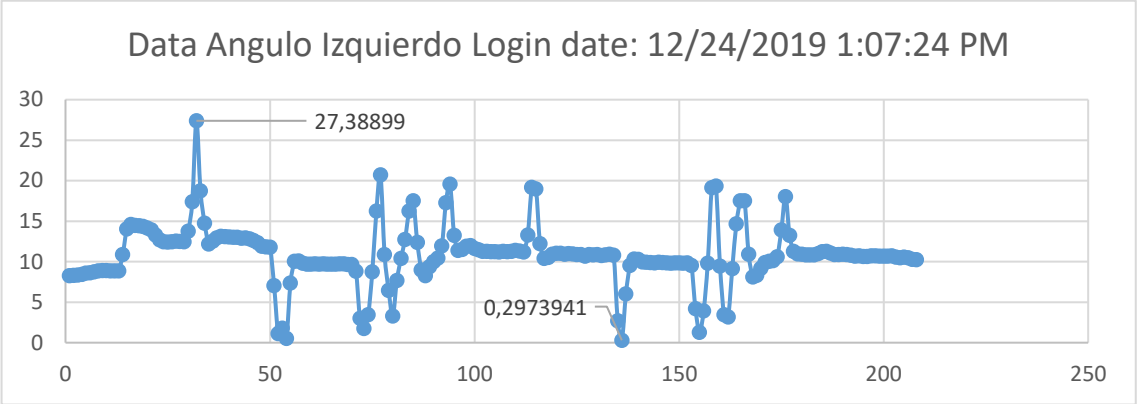


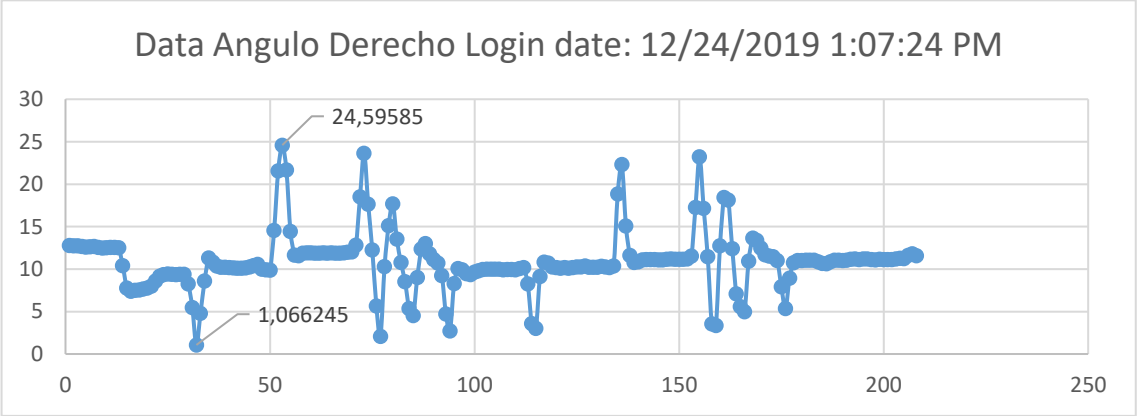


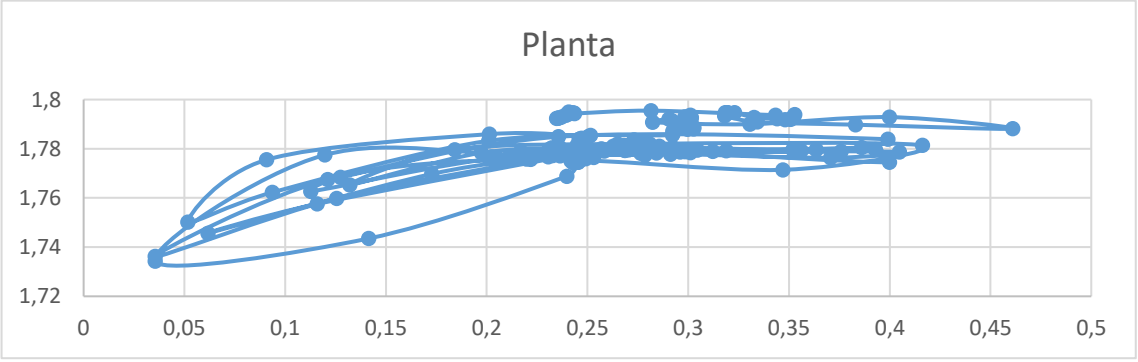


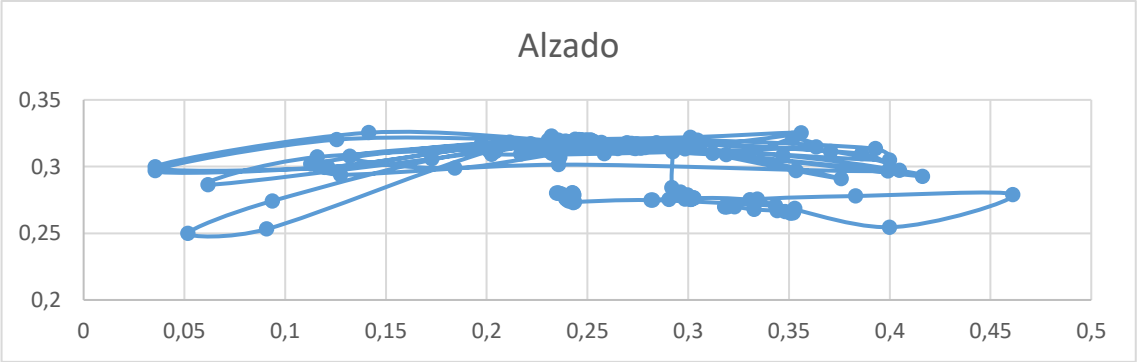


- Paciente 20 (Control Piernas)

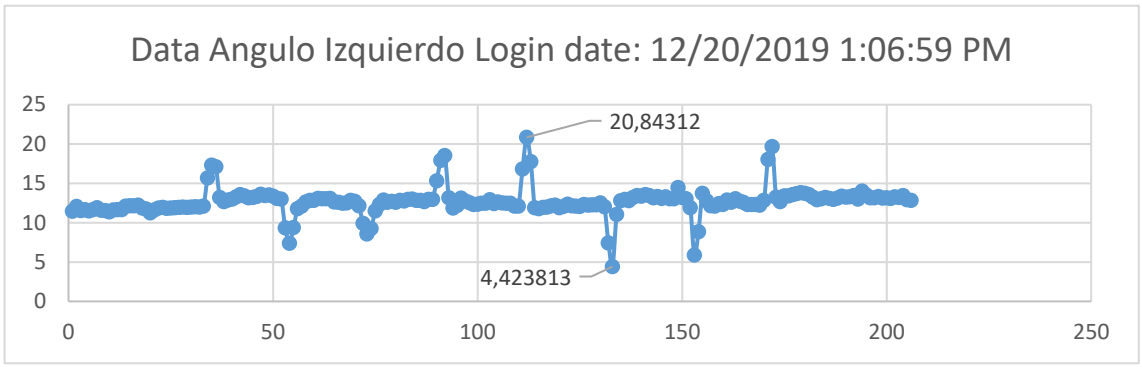


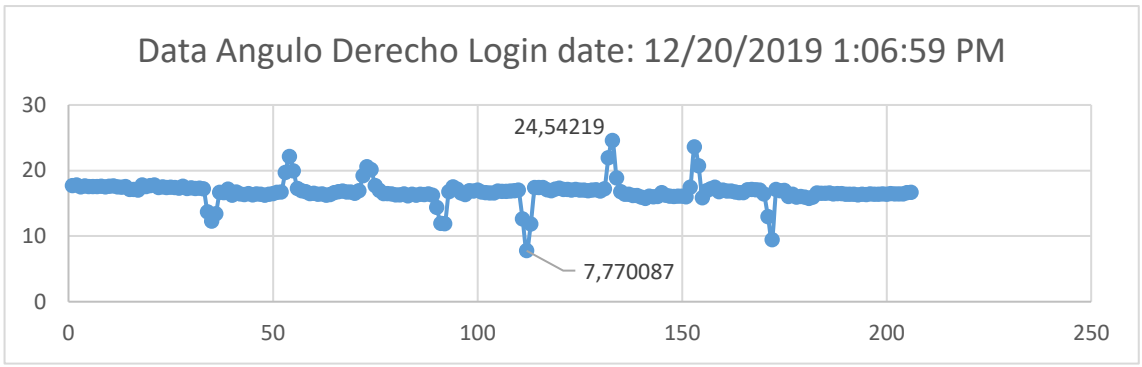


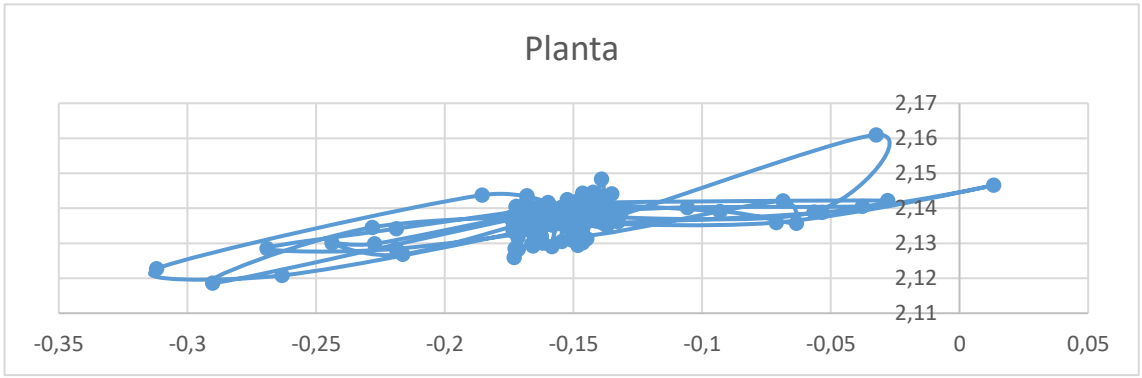


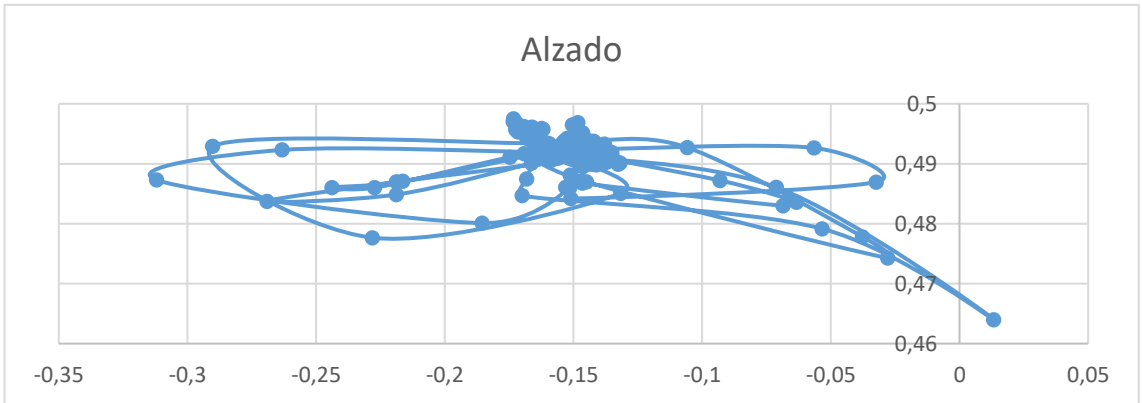


- Paciente 21 (Control VR)

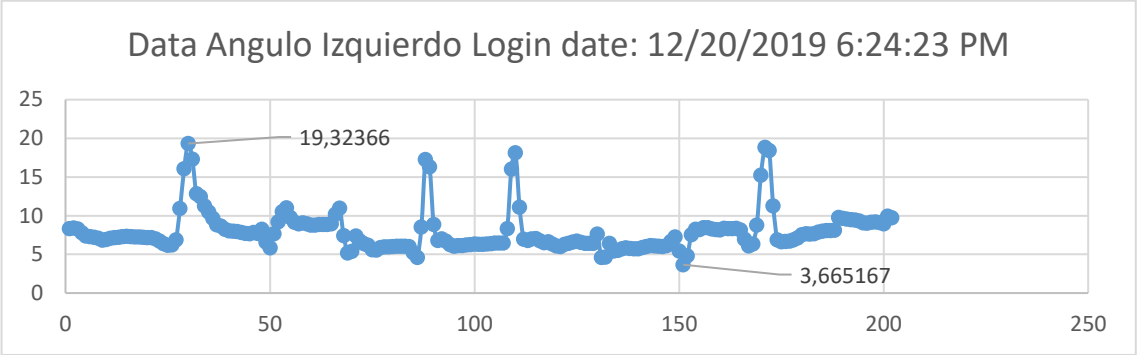


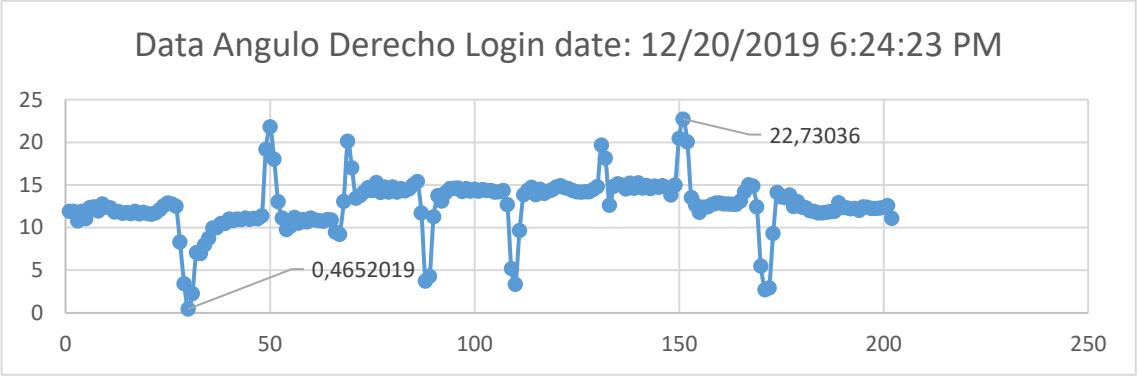


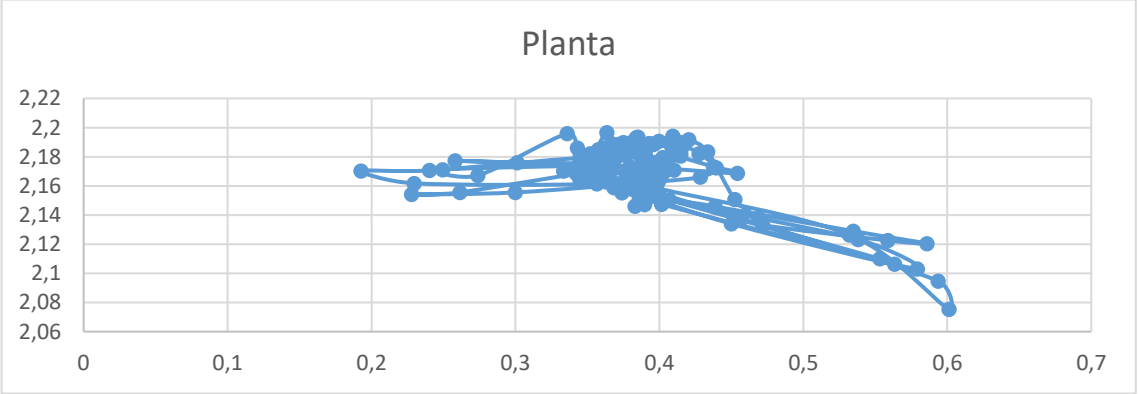


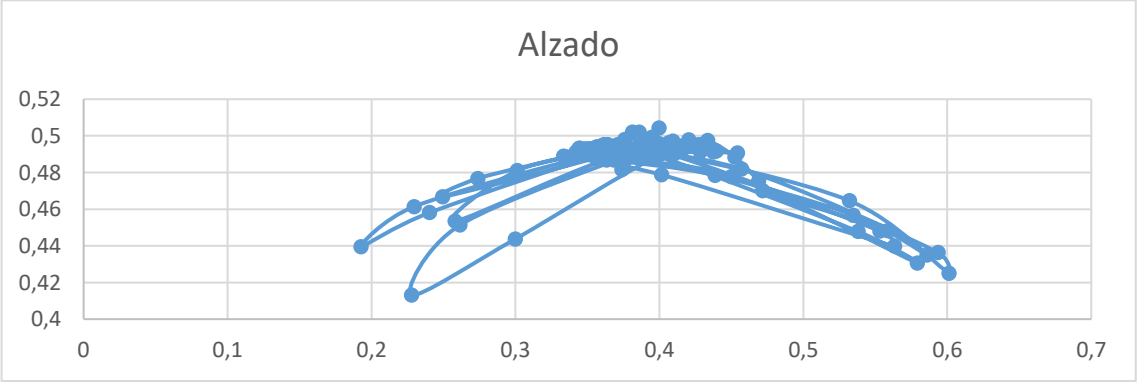


- Paciente 22 (Control VR)









Anexo C.4 y D

El anexo C.4 contiene los datos que forman las gráficas del Anexo C.3 en el programa Excel.

El anexo D contiene 4 vídeos que permiten la visualización de las personas voluntarias mientras prueban la aplicación realizada.

Al ser unos ficheros de gran tamaño se requiere de *Google Drive* para el almacenamiento. Desde esa plataforma se pueden ver, descargar y trabajar con los ficheros.

Enlace URL para visualización y descarga de los ficheros:

https://drive.google.com/drive/folders/1JJ_v2X-xZN31GegvF0NXlcAfg0alDlFD?usp=sharing