UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

GRADO EN INGENIERÍA EN INFORMÁTICA

SRES: Sistema de apoyo a la realización y seguimiento de ejercicios en casa para niños con dolencias en la columna lumbar

David García Silvestre

Julio, 2021

SRES: SISTEMA DE APOYO A LA REALIZACIÓN Y SEGUIMIENTO DE EJERCICIOS EN CASA PARA NIÑOS CON DOLENCIAS EN LA COLUMNA LUMBAR

UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA

Tecnologías y Sistemas de Información

GRADO EN INGENIERÍA EN INFORMÁTICA

INGENIERÍA DEL SOFTWARE

SRES: Sistema de apoyo a la realización y seguimiento de ejercicios en casa para niños con dolencias en la columna lumbar

Autor: David García Silvestre

Tutor académico: Dr. David Vallejo Fernández

Cotutor académico: D. Cristian Gómez Portes

Julio, 2021

David García Silvestre

Ciudad Real – Spain

E-mail: David.Garcia83@alu.uclm.es
Teléfono: 606 821 618

TRIBUNAL:

Presidente:

Vocal:

Secretario:

FECHA DE DEFENSA:

CALIFICACIÓN:

PRESIDENTE	VOCAL	SECRETARIO
Fdo.:	Fdo.:	Fdo.:

Resumen

En el contexto del tratamiento de afecciones de la espalda en niños y adolescentes, la realización de ejercicios en casa, recomendados por un profesional médico, resulta esencial para corregirlas y disminuir el foco del dolor. Esta situación plantea dos retos que requieren especial atención. En primer lugar, resulta fundamental motivar a los pacientes a realizar, de manera continuada, el programa de ejercicios asignado, puesto que, generalmente, un niño o adolescente realizará los ejercicios en casa, pero, sin la motivación necesaria, la constancia que requieren estos planes de rehabilitación se perderá. En segundo lugar, es necesario evaluar que los ejercicios son realizados de manera correcta.

En este contexto, este proyecto plantea el diseño y desarrollo de un sistema software aplicado al ámbito de la rehabilitación física de pacientes afectados por una enfermedad o lesión en la columna lumbar. Para ello, el sistema se apoyará en técnicas de visión por computador que favorezca la correcta realización de estos ejercicios de rehabilitación, especialmente dirigidos a niños y adolescentes que padecen este tipo de dolencias. El sistema motivará al paciente mediante técnicas de gamificación, proporcionando *feedback* visual en tiempo real. De esta forma se pretende fomentar que el usuario no abandone su rutina de ejercicios. Desde el punto de vista del rol que juega el terapeuta, el sistema permite definir nuevos ejercicios de rehabilitación gamificados.

Abstract

In the context of treating back conditions in children and adolescents, performing exercises at home, recommended by a medical professional, is essential to correct them and reduce the focus of pain. This situation proposes two challenges that require special attention. First, it is essential to motivate patients to continuously carry out the assigned exercise program. Generally, a child or adolescent will carry out the exercises at home but, without the necessary motivation, the perseverance that these rehab plans require will be lost. Second, it is necessary to evaluate that the exercises are performed correctly.

In this context, this project proposes the design and development of a software system applied to the field of physical rehabilitation of patients affected by a disease or injury in the lumbar spine. For it, the system will rely on computer vision techniques that favor the correct performance of these rehabilitation exercises, especially aimed at children and adolescents suffering from this type of diseases. The system will motivate the patient through gamification techniques, providing visual feedback in real time. Thus, it is intended to encourage the user not to abandon their exercise routine. From the point of view of the therapist's role, the system allows the definition of new gamified rehabilitation exercises.

Agradecimientos

Me gustaría comenzar dando las gracias a mis padres, por estar todos estos años soltando billetes a pesar de la inutilidad de su hijo, gracias a ellos tendré un futuro medio decente. A mi hermana Sandra, mi persona favorita sobre este mundo. A mi abuela Piedad y mi abuelo Salvador, quien finalmente no pudo cumplir su sueño de ver graduado a su nieto, del que me acuerdo todos los días de mi vida.

Por otro lado, agradecer a los todos los profesores de la ESI que me hayan dado clase alguna vez, en especial a mis tutores David Vallejo y Cristian Gómez, culpables de que todo esto haya salido adelante. Primero por la paciencia que han tenido conmigo al principio del proyecto, donde me pasaba meses enteros haciendo nada, y después al final de este, donde me surgieron las prisas por hacer las cosas y aun así estuvieron disponibles siempre para ayudarme. No merezco el trato que me han dado.

Por último, dar las gracias a aquellas personas que la vida ha puesto en mi camino y se han convertido en los mejores amigos. Los de toda la vida, pertenecientes al clan *Jose Autoescuela* formado por Cristian, Dani y Feigned, y los conocidos gracias a ellos recientemente pero que tiene pinta que van a estar conmigo siempre. Gracias también a los que conocí por el grupo *ComuBits <3 Teleporro*, sin su ayuda seguramente seguiría en segundo suspendiendo Lógica. Y por supuesto a los que me permitieron conocer dicho grupo (sí Porris, Uni y Céspedes, esto va por vosotros). También mencionar a Pegaso y Rubén, camareros de *El Traste*, encargados de servir felicidad a *ComuBits* en forma de minis de chupitos durante todos los jueves de estos últimos años. Finalmente, me queda dar las gracias a Álvaro, el único informático que ha estado a mi lado literalmente desde el primer día hasta el último. Lo hicimos, hermano. Lo hemos conseguido.

A todos, gracias por tanto, perdón por tan poco. Os quiero.

David

A todos aquellos que alguna vez confiaron en mí

Índice general

Índice de cuadros

Índice de figuras

Índice de listados

Listado de acrónimos

ESI	Escuela Superior de Informática
GPU	Graphics Processing Unit
IA	Inteligencia artificial
IMU	Inertial measurement unit
JSON	JavaScript Object Notation
MP	Megapíxel
OMS	Organización Mundial de la Salud
PIB	Producto Interior Bruto
RGB	Red-Green-Blue
SDK	Software Development Kit
TFG	Trabajo de Fin de Grado
TIC	Tecnologías de la información y la comunicación
UCLM	Universidad de Castilla-La Mancha
XML	eXtensible Markup Language

Capítulo 1

Introducción

ESTE primer capítulo tiene el objetivo servir de preámbulo del proyecto completo, analizando también el impacto socio-económico de este, y el contexto en el que está ubicado el trabajo. Así, se pretende que el lector tenga una visión general del mismo. Además, se incluye un resumen con la estructura del documento.

1.1 Contexto general

La **rehabilitación** o medicina física es una especialidad médica centrada en el tratamiento de personas que sufren alguna discapacidad como consecuencia de lesiones o enfermedades. Esta especialidad tiene como objetivos estudiar el diagnóstico para hacer una evaluación. De esta forma se busca prevenir, minimizar o revertir estas discapacidades, devolviendo el mayor grado de capacidad funcional e intentando adquirir la mayor independencia posible [FCFC14].

Actualmente, la rehabilitación médica es un ámbito de gran importancia en la medicina, puesto que, con el aumento de calidad de vida en los últimos años, la esperanza de vida también es mayor, propiciando un envejecimiento de la población [reh18]. Este hecho implica que aumenten las probabilidades de sufrir una lesión o enfermedad, que pueden ser físicas o neuronales, y afectar a diversas partes del cuerpo como los miembros superiores e inferiores, o la **columna lumbar**.

Si hacemos énfasis en esta última zona concreta del cuerpo, nos daremos cuenta de que es de vital importancia mantenerla en buenas condiciones, puesto que la columna lumbar se encarga de soportar el peso del torso, además de proteger la médula espinal. En la figura 1.1 podemos ver su ubicación. Está compuesta de cinco vertebras con una robustez mayor que el resto, y permite flexión lateral y un rango de rotación pequeño.

Existen distintas enfermedades que afectan a esta zona con múltiples orígenes, entre las que se incluye la lordosis. Este trastorno consiste en una curvatura excesiva de la columna lumbar, y puede aparecer en la niñez sin ningún motivo durante el crecimiento, aunque también puede ser provocada por otras causas, como adoptar malas posturas, obesidad o enfermedades como la osteoporosis [EML21]. Es de vital importancia su tratamiento, mediante

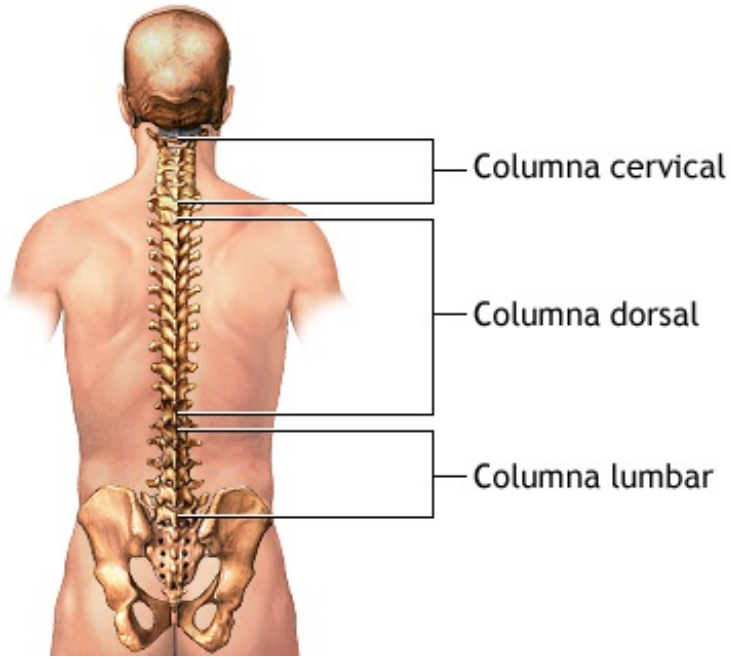

Figura 1.1: Estructura de la columna vertebral[1]

medicamentos o utilizando técnicas de rehabilitación.

Existen distintas formas mediante las cuales un paciente puede rehabilitarse de una lesión [abi14]. Este puede acudir a centros especializados donde realizar los ejercicios con la ayuda de un terapeuta y el empleo del instrumental disponible en dicho centro. Sin embargo, no siempre es necesario acudir a estos lugares, ya que el afectado puede realizar los ejercicios desde su casa. Al paciente también le puede resultar imposible acudir al centro de rehabilitación. Estos dos hechos favorecen la aparición de terapias de rehabilitación remota, de las que se hablará más detalladamente en la sección 3.1.1.

Existen diversos tipos de terapias remotas, en las que el paciente ha recibido instrucciones a seguir previamente de parte de un terapeuta con las que realizar los ejercicios de forma correcta, aunque en los últimos años han aparecido sistemas que permiten la realización de los ejercicios de rehabilitación mediante el uso de distintas tecnologías, como la realidad virtual y la realidad aumentada.

Sin embargo, los pacientes pueden encontrar que los ejercicios de rehabilitación resultan demasiado repetitivos y aburridos de realizar, por lo que es común que abandonen sus terapias de rehabilitación al haber perdido la motivación [dan19]. Para tratar esto, muchos trabajos han investigado como hacer más interesantes estos ejercicios, y para ello han utilizado técnicas de **gamificación** [Cho19] con el fin de convertir la rehabilitación en una actividad que, mediante la aplicación de juegos, una persona pueda realizar ejercicios rehabilitadores de forma amena.

[1]Fuente: `https://medlineplus.gov/spanish/ency/esp_imagepages/9766.htm`

Por todo lo expuesto hasta el momento, el presente Trabajo de Fin de Grado (TFG) trata de diseñar y desarrollar un sistema que proporcione una solución a los problemas mencionados anteriormente, proponiendo un proyecto con el que los niños afectados en la columna lumbar puedan realizar ejercicios de rehabilitación remota (principalmente desde casa) utilizando técnicas de gamificación para mantenerlos motivados.

1.2 Impacto socio-económico

Aunque se ha avanzado en los tratamientos de lesiones en la columna lumbar, sigue representando un problema importante desde el punto de vista social y económico. Una lesión en dicha zona del cuerpo puede derivar en diversas complicaciones secundarias. A pesar de que su tasa de mortalidad sea bajísima, un gran porcentaje de la población algún día se verá afectado por dolencias en esta parte de la espalda [abc15].

Existen múltiples motivos que pueden ocasionar lesiones en la espalda baja, como realización de poca actividad física, enfermedades, o simplemente la edad. En España, de la totalidad de las personas que han recibido tratamientos relacionados con la columna lumbar, un 50 % tienen entre 50 y 71 años, y un 25 % son mayores de 71 años [TJCR17]. Además, dependiendo del sexo, afecta en mayor medida a las mujeres, siendo estas un 63.9 % de la totalidad de personas tratadas frente al 36,1 % de los varones. En los últimos años, el porcentaje de niños que presentan dolor en la columna lumbar ha aumentado. Se espera que antes de llegar a la adolescencia, entre un 10 % y un 30 % de los niños haya experimentado este tipo de dolor [CSC20].

Sobre el impacto económico de las lesiones en la espalda baja, se estima que el coste anual en España es cerca de 9.000 millones de euros [AGSS20], representando un 0,68 % del Producto Interior Bruto (PIB) español. En estos costes están valorados diferentes aspectos, como pueden ser tratamientos de las lesiones, cuidados médicos, fisioterapias, enfermerías, o mantenimiento de los elementos necesarios, además de gastos derivados por haber un exceso de utilización de los servicios sanitarios.

1.3 Propuesta

El presente TFG propone un sistema que, mediante el uso de juegos, ayudará a que niños puedan hacer ejercicios de rehabilitación de lesiones en la columna lumbar. El sistema estará apoyado en tecnologías que usan técnicas de visión por computador, las cuales favorecerán la realización de estos ejercicios.

Este sistema está especialmente diseñado para niños y adolescentes que padezcan las dolencias mencionadas anteriormente. Esto es porque los tipos de ejercicio son más fáciles de llevar a cabo por un niño y las mecánicas de juego necesarias para completarlo le deberían resultar más atractivas que a una persona mayor. Sin embargo, cualquier adulto que lo necesitase también podría hace uso de este sistema debido a su capacidad para personalizar

ejercicios de rehabilitación.

De esta forma, el sistema tratará de que el paciente, al encontrar los ejercicios en forma de juegos, consiga facilitar sus tareas de rehabilitación al disponer de una herramienta amigable e interactiva. Esto pretende, además, aumentar la motivación del paciente para que no abandone la rutina que previamente un terapeuta le ha solicitado que realice.

1.4 Estructura del documento

La estructura de este TFG es la propuesta por la Escuela Superior de Informática (ESI) de la Universidad de Castilla-La Mancha (UCLM) según su normativa, y contiene los siguientes capítulos:

- **Capítulo 2: Objetivos**

  Este capítulo explica los objetivos que se pretenden alcanzar con este proyecto, primero de una forma general para después profundizar en ellos.

- **Capítulo 3: Estado del arte**

  Este capítulo muestra una visión general de diversos temas relacionados con la rehabilitación de lesiones, como capturar el movimiento de un cuerpo, el concepto de juego serio, y, las tecnologías necesarias para la creación de este proyecto.

- **Capítulo 4: Metodología**

  Este capítulo describe la metodología de trabajo elegida y su forma de empleo, además de la planificación del proyecto y los medios utilizados para el desarrollo de este.

- **Capítulo 5: Arquitectura**

  En este capítulo comienza mostrando al lector una visión general de la arquitectura del proyecto, para después profundizar en cada uno de los tres módulos contenidos en esta.

- **Capítulo 6: Resultados**

  Este capítulo muestra el resultado final obtenido tras el desarrollo del proyecto, donde se puede observar el sistema en ejecución, estadísticas del código desarrollado, los costes necesarios, y una prueba del funcionamiento del sistema.

- **Capítulo 7: Conclusiones**

  El último capítulo explica las conclusiones obtenidas, junto con los objetivos que se han alcanzado, las competencias adquiridas, y una serie de consideraciones en las que mejorar en el futuro.

Capítulo 2

Objetivos

ESTE capítulo describe de forma detallada los objetivos principales de este proyecto, planteados antes de su realización, centrándose en el objetivo general y enumerando los objetivos específicos que se deben lograr.

2.1 Objetivo general

El objetivo general de este proyecto es diseñar, desarrollar y validar un sistema software destinado a la rehabilitación física de pacientes afectados por una enfermedad o lesión en la columna lumbar.

Es importante que el sistema consiga motivar al paciente a que realice sus actividades de rehabilitación asignadas de manera continuada, por lo tanto, el sistema integrará un módulo basado en técnicas de gamificación, para que los pacientes sean incentivados a realizar los ejercicios.

Además, habrá un subsistema de progresión que servirá de punto de comunicación con el médico, para que este pueda comprobar la evolución del paciente y asignar nuevos ejercicios en consecuencia. Por otro lado, se usarán un conjunto de métricas, que aparte de medir el progreso de los pacientes, servirán de guía en la correcta ejecución de los ejercicios.

2.2 Objetivos específicos

Del objetivo general mencionado anteriormente se pueden extraer objetivos específicos más concretos, que son los siguientes:

- Elicitación de requisitos y elaboración de pruebas conceptuales para desarrollar un sistema enfocado a la rehabilitación de niños con dolencia lumbar. Se pretende crear una herramienta que consiga motivar a niños con sus tareas de rehabilitación, enseñándoles cómo hacer estas, y que, a su vez, cuente con un sistema que controle el progreso del paciente.
- Diseño y desarrollo de un módulo de gamificación. La utilidad de este módulo es motivar a los pacientes a que hagan sus ejercicios de rehabilitación, ya que es una tarea demasiado rutinaria para ser realizada por niños. Estos podrían no entender la

importancia de hacer ejercicios para recuperarse de sus lesiones, por lo que mostrar las actividades a realizar en forma de juegos debería estimular a los niños a seguir sus ejercicios de rehabilitación. Este módulo, además, se encargará de enseñar al paciente la forma en la que se hacen los ejercicios correctamente, indicando los movimientos que debe realizar y la parte del cuerpo implicada.

- Diseño y desarrollo de un módulo de definición de métricas. Este módulo tiene el propósito de establecer unas métricas con las que poder evaluar el rendimiento de los pacientes a la hora de realizar los ejercicios. De esta forma tanto el doctor como el usuario del sistema podrán controlar su evolución, al tener unas medidas en las que poder basarse a la hora de llegar a conclusiones sobre su estado.
- Diseño y desarrollo de una arquitectura escalable. El sistema debe de estar construido para integrar mejoras en el futuro. Por un lado, tiene que contar con la posibilidad de añadir nuevos ejercicios. Por otro lado, también debe contar con la posibilidad de añadir nuevas funcionalidades, por ejemplo, relacionadas con el uso de los micrófonos, ya que esto daría pie a una interacción natural mediante voz. Por lo tanto, hay que diseñar y desarrollar una arquitectura que permita la escalabilidad. Para ello, se hará uso de patrones de diseño para abordar este objetivo.
- Validación del sistema software. Para comprobar que el sistema funcione correctamente, se debe realizar algún ejercicio relacionado con la columna lumbar y verificar los resultados. Para ello, hay que pensar en afección concreta, después elegir el ejercicio que mejor se adapte a solucionarla, realizar tal ejercicio, y, por último, verificar que los resultados obtenidos son los esperados.

Capítulo 3

Estado del arte

El propósito principal de este capítulo es repasar los conceptos relacionados con la rehabilitación de lesiones de espalda de forma remota. Para ello, se ahondará brevemente en los primeros ejercicios de rehabilitación concebidos hace miles de años, concluyendo con los ejercicios actuales, en los que, gracias a los avances en la tecnología, un paciente puede realizar los ejercicios desde donde quiera de forma remota.

Para estos últimos ejercicios, se utilizan distintos dispositivos de captura de movimiento. En este capítulo se describirán los más destacados hasta el momento. Estos dispositivos capturan imágenes de los movimientos de un usuario, los cuales tienen que ser interpretadas de alguna forma, y ahí intervienen las bibliotecas de *Body Tracking*, las cuales permiten manipular la información de un esqueleto de forma sencilla. Este capítulo describirá las más usadas.

Además, este capítulo trata los juegos serios en el ámbito de la rehabilitación. Estos juegos aplican técnicas de gamificación con las que se pretende aumentar la motivación necesaria para hacer ejercicios de rehabilitación. Además, se mencionan las tecnologías que son necesarias para poder desarrollar estos juegos, entre las que destacan los motores de videojuegos.

3.1 Historia de la rehabilitación

La Organización Mundial de la Salud (OMS) ha declarado que el significado de la rehabilitación es «*un conjunto de intervenciones encaminadas a optimizar el funcionamiento y reducir la discapacidad en personas con afecciones de salud en la interacción con su entorno*» [WR220]. Para conocer la rehabilitación un poco mejor vamos a remontarnos a unos cuantos siglos antes.

En los tiempos de la antigua Grecia, ya se les daba importancia terapéutica a ejercicios corporales para mantener la salud. Esta tendencia se siguió durante el cristianismo, en la Roma Antigua. Entre los métodos que introdujeron los romanos se encuentran el hidromasaje, los estiramientos, y ejercicios realizados con la ayuda de poleas. Destaca principalmente Asclepíades, que rechaza el uso de medicamentos y propone realizar dietas, masajes, hidroterapia y ejercicios físicos [MM98].

Sin embargo, la rehabilitación, tal y como es conocida actualmente, tiene su origen en Estados Unidos, con el médico Dr. Frank Krusen. En 1934 estableció el Registro de Médicos Técnicos en Terapia Física [LV14]. Era un grupo de especialistas con habilidades relacionadas con el tratamiento de discapacidades. Años después publicó el libro de Medicina Física del Dr. Krusen, libro que hacía un amplio estudio sobre el uso de los procedimientos físicos en el cuidado de los enfermos.

También el comienzo de la Primera Guerra Mundial hizo que se aumentara el uso de ejercicios físicos para rehabilitación en los hospitales militares. Los primeros sitios parecidos a lo que hoy en día consideramos centro de rehabilitación eran demasiado simples comparados con los que tenemos en la actualidad. Aun así, ya había médicos especialistas con habilidades para el tratamiento de enfermedades patológicas y lesiones. Estos médicos utilizaban ya agentes físicos específicos para los tratamientos que permitían observar una gran mejoría en los pacientes tratados.

Hoy en día, un **centro de rehabilitación** es un lugar que cuenta con personal con conocimientos adecuados, que analizan las mejores formas de recuperación a través de las técnicas apropiadas, y con la tecnología que sea necesaria para ayudar en el tratamiento a personas que hayan sufrido lesiones a causa de accidentes o personas que sufran enfermedades que afecten a su calidad de vida [Abi21]. El objetivo de estos centros es conseguir que las personas no pierdan movilidad en las partes del cuerpo afectadas y puedan valerse por sí mismas.

Normalmente un centro de rehabilitación dispone de los aparatos necesarios para realizar ejercicios, como si se tratase de un gimnasio. Otros cuentan con tratamientos manuales por parte de los especialistas, en el que este ayuda a hacer los ejercicios necesarios o realiza masajes al paciente. Se puede contar con tratamientos de hidroterapia, en los que se proporcionan duchas, baños, baños de vapor o hidromasajes.

3.1.1 Rehabilitación remota

El campo de la rehabilitación ha ido evolucionando en los últimos años gracias a los avances en la medicina y a la aparición de nuevas tecnologías. Además, para realizar ejercicios de rehabilitación no es esencial estar en un lugar concreto, sino que normalmente se pueden hacer en cualquier lugar. Esto permite que se puedan investigar terapias de rehabilitación que se puedan hacer de manera remota.

De esta forma no siempre será necesario acudir a centros especializados para rehabilitar una zona del cuerpo afectada, ya que algunos ejercicios pueden hacerse desde casa. Esta ventaja se ha planteado en muchas investigaciones, como por ejemplo en [BDMM14], donde la rehabilitación remota permite proporcionar los servicios necesarios para realizar las terapias en el hogar del paciente o en otro lugar elegido. También es una alternativa a la rehabilitación domiciliaria, en la que el terapeuta debe viajar al hogar del paciente.

La rehabilitación remota no tiene por qué ser mejor que la rehabilitación hecha de forma presencial con un terapeuta, pero si cuenta con algunas ventajas destacables. Es especialmente útil cuando existen dificultades para ir al centro de rehabilitación, ya sea porque el paciente vive en entornos rurales o porque su salud no le permite viajar. También el hecho de poder realizar actividades de rehabilitación donde el paciente quiera puede suponer algo positivo para este, ya que podría motivar y facilitar su evolución. Este hecho acaba psicológicamente facilitando la recuperación del paciente, ya que este permanece en su hogar y cerca del núcleo familiar [BRP+19].

Otro beneficio de la rehabilitación remota es que se reducen los gastos, ya que el paciente se ahorra el viaje, y los terapeutas no necesitan mantener el espacio donde se realiza el ejercicio, los objetos ni las maquinas necesarias.

Se puede realizar rehabilitación remota cuando un especialista indica al paciente los ejercicios que deben hacer, y este los hace desde su hogar. Gracias a nuevas tecnologías es más fácil esto, porque se pueden incluir vídeos de demostración, páginas web, o programas y aplicaciones que guíen al paciente. Si es necesario un equipo, puede ser proporcionado por el centro o por el especialista. También se puede monitorizar el progreso del paciente para que el médico controle los avances desde su consulta.

Sin embargo, no todo son beneficios en la rehabilitación remota, ya que también cuenta con desventajas. Un ejemplo de estas es que los profesionales o los pacientes no terminen de aceptarlas [HRLRFB+19], porque prefieran seguir una forma tradicional, o, en el caso de personas mayores, no consigan entenderlas, debido al desconocimiento de la tecnología. Otra desventaja importante es el hecho de no poder realizar una comprobación, en primera mano, de los avances del paciente por parte del médico [65y20]. No obstante, estos inconvenientes tienen solución, por ejemplo, involucrando más al personal médico y pacientes en el diseño de las soluciones tecnológicas, al igual que desarrollar soluciones que no entorpezcan la terapia, sino que sean un método más para mejorar sus dolencias.

3.2 Seguimiento del movimiento

En esta sección se profundizará en cuál es el estado actual de los sistemas que permiten realizar el seguimiento de movimientos de una persona. Aunque más adelante se dará una descripción más especifica, de forma resumida, la correcta captura de los movimientos de un sujeto consta de una serie de aspectos. Por un lado, son necesarios una serie de dispositivos que permitan capturar el movimiento realizado. Por otro lado, es imprescindible contar con *software* capaz de interpretar los datos capturados por los dispositivos. Veamos a continuación el verdadero estado de estos sistemas.

3.2.1 Dispositivos con soporte para body tracking

Existen múltiples dispositivos que permiten la captura de movimientos de un cuerpo humano, cada uno con sus ventajas y desventajas, y con diferencias entorno al precio, facilidad de uso o si pertenecen a un sistema complejo o por el contrario van dirigidos a un público más casual.

Uno de los sistemas más usados para captura de movimientos, sobre todo en ámbitos como el cine o el desarrollo de videojuegos, es el de proporcionar a un actor un traje con el que detectar las acciones efectuadas. El actor realizará los movimientos que deben ser capturados con el traje puesto y serán trasladados a un modelo digital.

Esta técnica utiliza distintos tipos de trajes, de los cuales se van a destacar dos clases: ópticos y mecánicos [A+17]. La principal ventaja de los trajes ópticos respecto a los mecánicos es su comodidad, ya que al no necesitar un exoesqueleto el actor no debe cargar con un traje lleno de sensores que limitaría sus movimientos. Los trajes ópticos son un traje de neopreno negro con una serie de nodos colocados por todo el cuerpo en posiciones estratégicas. Estos nodos también pueden ser de dos tipos [EHM+18], pasivos, que su función es reflejar una luz infrarroja emitida por un dispositivo, y activos, que ellos mismos generarán la propia luz. En cualquier caso, el movimiento de estos nodos será recogido por una serie de cámaras. Esta técnica lleva años existiendo, pero destacó especialmente en 2001-2003 con la grabación de las películas de *El Señor de los Anillos*.

Figura 3.1: Traje de captura de movimiento utilizado en *El Señor de los Anillos*[1]

En cuanto a los trajes mecánicos, también se les denomina sistema de captura de movimiento inercial. Son una especie de exoesqueleto equipado sobre el actor que incorpora un conjunto de sensores, como acelerómetros y giroscopios encargados de registrar el movimiento.

Estos sistemas se han utilizado para otros tipos de fines más allá de la industria cinema-

[1] Fuente: `https://teseo.es/noticias/que-es-y-como-funciona-la-captura-de-movimiento/`

tográfica, como por ejemplo, en deporte y salud. Por ejemplo, una investigación relevante relacionada con ambos temas es la expuesta en [SMI14], la cual analiza el rendimiento de jugadoras de balonmano tiempo después de superar una lesión en el ligamento cruzado anterior mediante la reconstrucción de este. Colocando sensores inerciales se les medía el salto y se compararon los datos con jugadoras que nunca habían sufrido esa lesión. Los resultados demostraron que años después de recuperarse sigue habiendo diferencias en la biomecánica del salto.

Retomando los trajes de captura de movimiento inercial, una ventaja que tienen es su mayor precisión comparada con los ópticos, ya que están calibrados para obtener la máxima precisión. No obstante, su gran hándicap es que son poco ergonómicos y de difícil manejo, ya que son más incómodos e intrusivos para el actor porque sus movimientos pueden verse limitados por el propio traje. Por lo tanto, en temas de salud es preferible usar otros métodos de captura.

Un ejemplo de sistema de captura de movimiento inercial es **Xsens**[2]. Está compuesto por un conjunto de 17 sensores colocados en distintos puntos del cuerpo, se alimenta mediante baterías y es capaz de enviar los datos capturados de forma inalámbrica. En el artículo de la siguiente referencia [RLS09] se explica en detalle su funcionamiento.

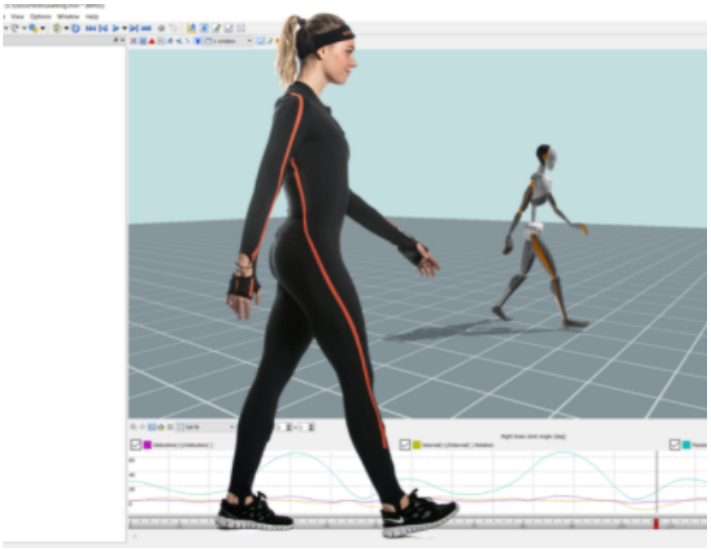

Figura 3.2: Sistema de captura de movimiento inercial Xsens[3]

Continuando con la idea del traje de captura mecánico, podemos simplificar el concepto y llegar hasta las gafas de realidad virtual, como por ejemplo **HTC VIVE**[4], como otro sistema de captura de movimientos. En este caso este sistema está más centrado en videojuegos.

Estas gafas capturan el movimiento de la cabeza también con acelerómetros y giroscopios,

[2] https://www.xsens.com/

[3] Fuente: https://www.xsens.com/products/mvn-analyze

[4] https://www.vive.com/eu/

y muestran en las lentes de su interior la imagen que el jugador necesita ver.Las gafas también incluyen dos controladores para las manos que se encargarán de capturar el movimiento de estas. De manera opcional se pueden añadir cinturones con sensores para capturar el movimiento de la cadera y correas para el movimiento de los pies.

Figura 3.3: Imagen de las gafas de realidad virtual HTC VIVE[5]

Seguramente sean los videojuegos quienes más han intentado aprovechar los sistemas de captura de movimientos, por lo que vamos a seguir centrándonos en ellos. Siempre se han intentado nuevas formas de interactuar con los videojuegos, y de esa búsqueda de nuevas formas de interactuar nació la consola **Wii**, de Nintendo, que salió al mercado en noviembre del 2006. Esta tenía una barra de infrarrojos que se colocaba encima de la televisión. Los dos mandos de la consola poseían una serie de acelerómetros. Estos eran capaces de calcular la distancia entre el usuario y la barra, y detectar patrones de movimiento realizados con los mandos.

Gracias a todas las novedosas funcionalidades que introdujo Wii, las consolas de Nintendo se han usado y se siguen usando en el campo sanitario. Un ejemplo de aplicación en este ámbito es el propuesto en [MMK+16]. La finalidad de este ensayo fue precisar la efectividad del del uso videojuego *Wii Fit* y su accesorio *Wii Balance Board* junto con tratamiento convencional para rehabilitar el equilibrio en personas que hayan sufrido la amputación unilateral de su miembro inferior. Para probar este estudio, los pacientes se dividieron en dos grupos. Ambos grupos tenían un conjunto de ejercicios de rehabilitación común. El primer grupo realizó 15 minutos de ejercicios aparte de los comunes, mientras que el tratamiento del segundo grupo consistió en jugar a tres de los juegos incluidos en Wii Fit sobre la Wii Balance Board durante 15 minutos. Podemos ver una imagen del tratamiento en la figura 3.4. Los resultados de este ensayo fueron una mejora en la carga de peso y mantenimiento del equilibrio en el grupo que utilizó Wii Fit.

La propuesta de Sony para pelear con Nintendo y su Wii fue lanzar al mercado **PlaySta-**

[5]Fuente: `https://es.digitaltrends.com/realidad-virtual/htc-vive-y-vive-pro/`
[6]Fuente: `https://tinyurl.com/5jdahsk7`

Figura 3.4: Paciente utilizando el dispositivo Wii Balance Board[6]

tion Move, que funcionaba de forma bastante parecida. Un mando dotado de un giroscopio y un acelerómetro de tres ejes. También poseía una bola luminosa que, junto con una cámara, servía para ayudar a localizar al jugador.

Figura 3.5: A la izquierda el mando de Wii, a la derecha el mando de PlayStation Move[7]

Microsoft también quiso participar en esta moda de videojuegos que capturan movimientos, y, para ello, en noviembre de 2010 lanzó **Kinect** para Xbox 360.

A diferencia de sus competidores, Microsoft decidió no utilizar mandos, sino que apostaron por un dispositivo independiente de ellos que llamaron Kinect y que permitía controlar la consola sin necesidad de contacto físico, mediante movimiento o comandos de voz. Este dispositivo contaba con una cámara RGB, una cámara de infrarrojos, sensores de profundidad y un micrófono. Este dispositivo y su versión para Xbox One también han sido usados en el ámbito sanitario [MSK15] junto con diversas aplicaciones y juegos, al igual que ocurre con Nintendo y sus consolas.

[7]Fuente: `https://tinyurl.com/pby9vje`
[8]Fuente: `https://es.wikipedia.org/wiki/Kinect`

Figura 3.6: Kinect original de Xbox 360[8]

Aunque Kinect para Xbox 360 actualmente se encuentra descatalogado, este dispositivo permitió a Microsoft aprender mucho sobre la tecnología de captura de movimientos, por lo que continuaron indagando esta vez en nuevas herramientas para Windows. Años después apareció una nueva versión de Kinect, conocida como **Azure Kinect** (figura 3.7), el cual está orientado al ámbito investigador. Este dispositivo se basa en la tecnología de la nube Microsoft Azure y la IA para mejorar su rendimiento. A diferencia de sus versiones anteriores, no está pensado para funcionar en Xbox.

Figura 3.7: Fotografía de un Azure Kinect

Azure Kinect está compuesto por distintos sensores, como una cámara de profundidad de 1 MP, una cámara RGB de 12 MP para obtener la secuencia de colores, una unidad IMU formada por un acelerómetro y un giroscopio para calcular la orientación del dispositivo, y por último siete micrófonos colocados en una matriz circular. Podemos ver todos estos componentes separados en la imagen 3.8.

Este dispositivo permite sincronizarse con varios Azure Kinects para obtener la información del entorno con mayor precisión.

Realmente, la innovación de este dispositivo es combinar los sensores de Azure Kinect con los servicios de inteligencia artificial de Azure para lograr resultados más precisos. Esto, junto con el uso de la nube de Azure, permite el procesamiento de imágenes de manera inteligente, y el uso de algoritmos para mejorar el escaneo.

[9]Fuente: `https://tinyurl.com/57wj4t2v`

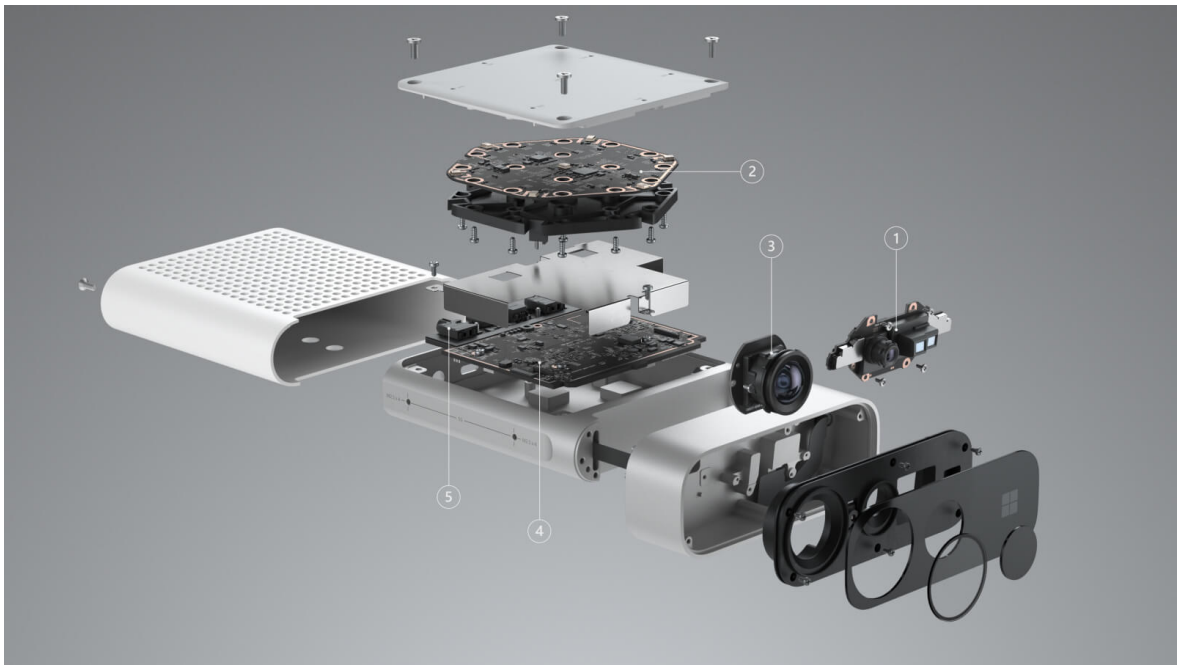

Figura 3.8: Imagen con los componentes internos de Azure Kinect[9]

Si tenemos en cuenta la idea que hay detrás de Kinect, y la reducimos un poco, tenemos dispositivos que capturan el movimiento de fracciones del cuerpo en lugar del cuerpo completo. Este es el caso de **Leap Motion**[10]. Este sensor del tamaño de un pendrive es capaz de detectar información tanto de los dedos como de la mano completa. Captura los datos relacionados con la posición de los dedos, sus medidas y los movimientos realizados. Funciona mediante óptica infrarroja y cámaras en lugar de sensores de profundidad, como en el caso de Kinect.

Figura 3.9: Imagen del sensor Leap Motion[11]

Para finalizar con esta sección hay que decir que ninguno de estos dispositivos es mejor o peor, puesto que cada uno realiza una función concreta y están pensados para diferentes ámbitos. Si nos centramos en el ámbito sanitario, elegir la herramienta correcta dependerá de

[10] `https://www.ultraleap.com/`
[11] Fuente: `https://tinyurl.com/hph354dy`

las funciones que sean necesarias. Por ejemplo, en el estudio propuesto en [MHK14] precisa que, para el seguimiento de la mano con dedos individuales, Leap Motion es la opción más adecuada mientras que cuando se requiere capturar el movimiento de un cuerpo completo es preferible utilizar Kinect. Además, en esta investigación se analiza una amplia gama de sistemas de rehabilitación que utilizaban Kinect.

3.2.2 Estado actual de bibliotecas para body tracking

Después de haber hablado del hardware utilizado para capturar movimiento, una parte no menos importante es el *software* que interpreta los datos. Es común que los fabricantes de cada dispositivo dispongan de su propio *software* para capturar movimientos, aunque existen otros Software Development Kit (SDK) independientes del dispositivo. Veamos algunos ejemplos de modelos de captura de movimiento comerciales.

En la actualidad uno de los SDK más utilizados es **Nuitrack**[12], ya que es multiplataforma y compatible con un gran número de dispositivos en sistemas Android, Windows y Linux. Ofrece distintas posibilidades como puede ser el seguimiento de cuerpo completo, creando un esqueleto con 19 articulaciones, como podemos comprobar en la figura 3.10, y hasta 6 esqueletos capturados a la vez.

Figura 3.10: Mapa de las articulaciones que utiliza Nuitrack[13]

Otra función es el reconocimiento de gestos. Estas acciones se expresan mediante un valor numérico que indica el progreso del movimiento realizado mediante un porcentaje. Los gestos disponibles en Nuitrack son agitar, deslizar hacia la izquierda, hacia la derecha, hacia arriba, hacia abajo y por último empujar.

[12] `https://nuitrack.com/`

[13] Fuente: `https://download.3divi.com/Nuitrack/doc/structtdv_1_1nuitrack_1_1Skeleton.html`

Una función interesante es la captura de nubes de puntos en 3D, que como podemos ver en la imagen 3.11, los puntos de la nube se colocan formando una figura en función de la distancia a la que se encuentren del sensor.

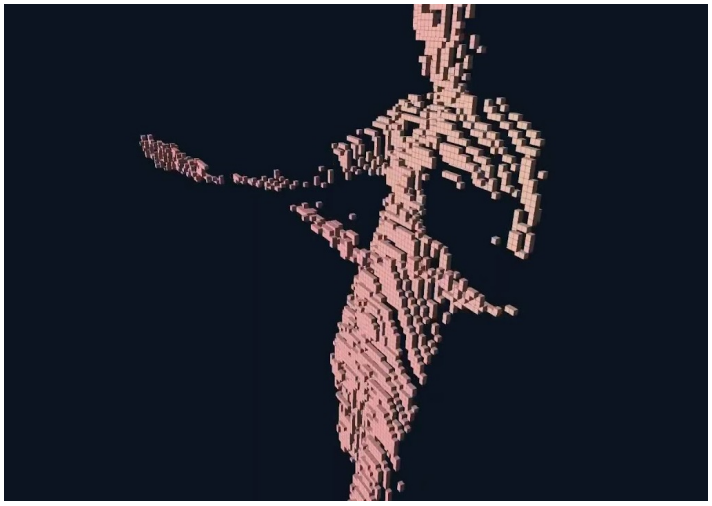

Figura 3.11: Captura de una nube de puntos en 3D generada por Nuitrack[14]

Similarmente, existe el SDK denominado **Skeleton Tracking SDK**[15], de **Cubemos**, que es utilizado principalmente por dispositivos Intel®RealSense™. Este SDK está diseñado para funcionar tanto en cámaras 2D como 3D. Es capaz de generar un esqueleto de 18 articulaciones y capturar un máximo de 5 personas a la vez. Disponible tanto para Windows como para Linux, y se puede desarrollar en diferentes lenguajes de programación, como C, C++, C# o Python. Una de las ventajas de este SDK que merece la pena destacar es que no necesita una GPU demasiado potente para ejecutar sus funciones.

Por último, cabe destacar el SDK del seguimiento de personas de **Azure Kinect DK**, *Body Tracking SDK*[16]. Este *software* está pensado única y exclusivamente disponible para el dispositivo Azure Kinect DK. Este SDK es multiplataforma y está disponible tanto para Linux como para Windows.

Este entorno de desarrollo contiene tres SDKs, el primero es el encargado de proporcionar el acceso a los sensores y a la configuración del dispositivo, otro responsable del *body tracking* (descrito aquí) mediante dispositivos que capturen en 3D, y por último un SDK encargado de utilizar el micrófono y los servicios de voz basados en la nube de Azure.

Algunas de las características del *Body Tracking SDK* de Azure Kinect DK son las siguientes:

- Acceso a la cámara de profundidad.

[14]Fuente: `https://tinyurl.com/4svrh4wx`

[15]`https://www.cubemos.com/skeleton-tracking-sdk`

[16]`https://azure.microsoft.com/es-es/services/kinect-dk/`

- Acceso a la cámara RGB.
- Permitir capturar movimiento del dispositivo mediante un giroscopio y un acelerómetro.
- Es posible ver los metadatos de los fotogramas capturados, entre los que se incluyen resolución de la imagen, marcas de tiempo, o temperatura.
- Se pueden comprobar los datos de calibración del dispositivo.

Este SDK permite capturar los cuerpos de forma segmentada si se da el caso de que aparezcan en la escena tan solo una parte en lugar del cuerpo completo. Permite capturar varios cuerpos a la vez y los esqueletos capturados contienen un total de 32 articulaciones.

3.3 Juegos serios

Este termino tiene su origen en el año 1970 en [Abt87], donde Clark Abt proporciona la siguiente definición para un juego serio: «*Un juego es un contexto con reglas entre adversarios que intentan conseguir objetivos. Nos interesan los juegos serios porque tienen un propósito educativo explícito y cuidadosamente planeado, y porque no están pensados para ser jugados únicamente por diversión*».

Por lo tanto, se entiende por juegos serios aquellos juegos en los que su objetivo principal no es el entretenimiento, sino que quieren transmitir conocimiento, o simular algún ambiente o actividad. En el contexto de este proyecto se estudiará el estado actual de los juego serios enfocados a la rehabilitación de pacientes

3.3.1 Juegos serios aplicados a la rehabilitación remota

En los últimos años, los avances en la tecnología han permitido la aparición de una serie de juegos en el mercado que ayudan a realizar los tratamientos de rehabilitación física remota con el propósito de mejorar la calidad de vida de los pacientes, además de tener otras utilidades orientadas al entrenamiento y *fitness*. Estos juegos son desarrollados con el propósito de rehabilitar una lesión concreta, ya que por lo general inciden en un déficit específico.

Los juegos serios tienen características distintas a las de los sistemas de rehabilitación tradicionales para poder realizar sus funciones. Por ejemplo, podemos hablar del uso de una interfaz, tanto en juegos de realidad virtual como en juegos de realidad aumentada, porque es una forma agradable de interactuar con los elementos virtuales.

El uso de un avatar virtual puede mejorar de forma positiva la conducta del paciente y las habilidades en la vida real. El avatar permite una vía fácil de interacción con objetos 3D en un escenario virtual en pacientes de diferentes géneros, edades o limitaciones físicas. También es importante tener retroalimentación durante la partida, para proporcionar información al paciente sobre su rendimiento. El juego debe llevar un registro de como se han ido realizando las actividades que el usuario ha hecho, y representarlo con algún sistema de puntuación que

muestre al jugador su desempeño.

Estos datos serán guardados en un soporte informático, ya que es mas fiable y consistente, y se usarán para la toma de decisiones por parte de los especialistas. Teniendo en cuenta estos datos, el terapeuta debería poder modificar los parámetros que el juego requiera para poder adaptarlo a las necesidades del paciente.

Ahora bien, si a un juego serio le añadimos la posibilidad de rehabilitar una lesión, surge un nuevo término que de ahora en adelante vamos a ver con bastante frecuencia a lo largo de este TFG, y este es el ***exergame***. Esta expresión proviene de la unión de las palabras *exercise* y *game*. Llamamos *exergame* a una actividad en la que el paciente se rehabilita indirectamente jugando. Es un término bastante utilizado en trabajos de investigación, como por ejemplo en [GM11]. Otra referencia a proyecto que use esta expresión es [BLLU14]. Este término se ha llevado desde el ámbito de la investigación al comercial. A continuación se presentan varios sistemas que hacen uso tanto del término juego serio como de la expresión *exergame*.

Un ejemplo de *exergame* es **Le Village aux Oiseaux** [MDNG12], que tiene el objetivo de ayudar a personas mayores con la enfermedad de Alzheimer. El propósito es ayudar a los pacientes a entrenar su atención para frenar la pérdida cognitiva provocada por esta enfermedad.

El juego consiste en que el jugador va a un pueblo y se convierte en un fotógrafo. El jugador debe hacer fotos a las aves que hay en dicho pueblo. Para jugar se requiere de un dispositivo similar al mando de una televisión, por lo que las personas mayores estarán familiarizadas con él, y se realizarán las fotos apuntando con el mando a la televisión.

Figura 3.12: Captura de pantalla de Le Village aux Oiseaux[17]

[17]Fuente: `https://smader.interaction-project.net/projects/le-village-aux-oiseaux.php`

Otra propuesta de juego serio aplicado en rehabilitación es la de la empresa californiana **Reflexion Health**, con su asistente virtual de ejercicios de rehabilitación (**VERA**) [VER18].

VERA es un sistema de fisioterapia que utiliza un avatar para ayudar a los pacientes a hacer sus ejercicios. La plataforma escanea al paciente con el objetivo de controlar su movimiento y analizar los progresos que vaya alcanzando el paciente, como podemos ver en la imagen 3.13. También se encarga de supervisar el cumplimiento de los ejercicios y de informar a los terapeutas de los resultados obtenidos.

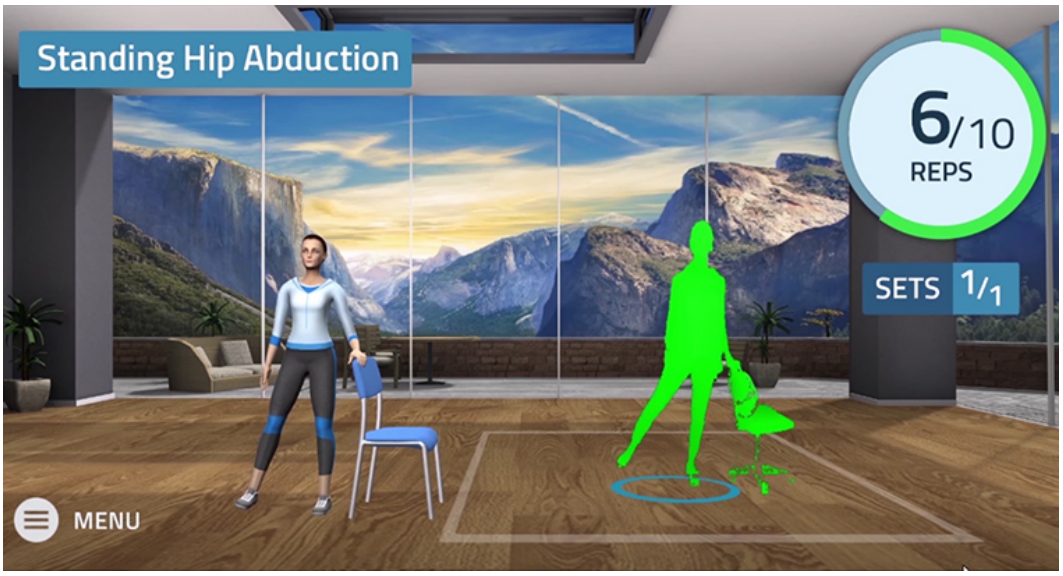

Figura 3.13: Ejemplo de ejercicio del sistema VERA[18]

El sistema funciona usando una cámara Microsoft Kinect. Para realizar un ejercicio primero el avatar mostrará al paciente como debe hacerlo, y después se le pedirá que realice el ejercicio, mientras la cámara captura los movimientos.

Por ultimo se informa al paciente como de bien ha realizado el ejercicio y como mejorar su técnica, y se generará un informe que será enviado a los médicos, los cuales realizan las comprobaciones necesarias para la revisión de su progreso y las correcciones si fuera necesario. También permite realizar videollamadas entre el paciente y el terapeuta.

Finalmente, otro ejemplo de juego serio, que también contiene características de la gamificación, es el de la empresa española **EvolvRehab**[19], con un sistema de ejercicios bastante similar a VERA. Contiene un conjunto de ejercicios para la rehabilitación de todas las extremidades del cuerpo y es usado tanto en consultas como en hogares.

Es de fácil uso ya que al igual que VERA, tan solo necesita una cámara para capturar el movimiento del usuario, y no tiene por que ser Kinect.

[18] Fuente: `https://tinyurl.com/3ykypr4n`
[19] `https://evolvrehab.com/`

EvolvRehab muestra ejemplos que explican como debes realizar sus actividades. Los ejercicios que se incluyen son lo que se indican en la imagen 3.14 y sirven para entrenar diversas funciones motoras de las extremidades superiores e inferiores, y se pueden personalizar para las necesidades de cada paciente en función de su nivel físico de capacidad. Todos los resultados se presentan de forma gráfica y se pueden exportar a un archivo para su uso posterior.

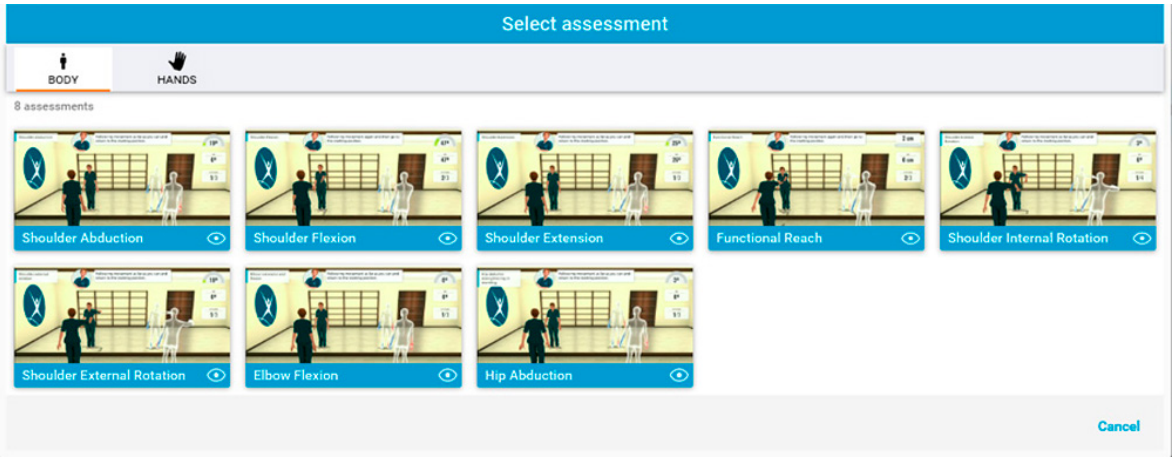

Figura 3.14: Ejercicios disponibles en EvolvRehab[20]

EvolvRehab también incluye *exergames* en los que en vez de repetir movimientos de un avatar, tienes que interactuar con objetos del escenario, como por ejemplo boxeando contra un rival (figura 3.15), o recogiendo y colocando objetos del escenario (figura 3.16), para el entrenamiento de la motricidad de las extremidades, y al igual que los ejercicios también se pueden personalizar.

Figura 3.15: Boxeo en EvolvRehab[21]

[20]Fuente: https://evolvrehab.com/evolvrehab/evolvrehab_body/
[21]Fuente: https://tinyurl.com/2fp95pej
[22]Fuente: https://tinyurl.com/2fp95pej

Figura 3.16: Ejemplo de juego de interactuar con objetos en EvolvRehab[22]

3.3.2 Gamificación aplicada a la rehabilitación remota

La gamificación es el empleo de las mecánicas y elementos divertidos de los juegos en ámbitos aburridos y rutinarios, o en aplicaciones no lúdicas. Esta trata de juntar las mecánicas del juego, las distintas acciones, técnicas y mecanismos y utilizarlos para convertir el juego en una actividad.

Este concepto se debe aplicar en aquellos entornos en los que la motivación es clave para el éxito. Esto se hace para conseguir potenciar la concentración, la motivación o el esfuerzo empleado en realizar la actividad gracias a la consecución de objetivos y con la finalidad de obtener reconocimientos positivos. Estos aspectos finalmente crean una experiencia atractiva para el usuario.

Una correcta implementación de técnicas de gamificación ayuda a que los usuarios del juego realicen sus ejercicios de una forma más proactiva, ya que normalmente estos ejercicios requieren de bastante esfuerzo de voluntad.

La idea de introducir la gamificación a las actividades más normales no es nueva, ya que desde siempre se ha intentado usar en contextos como la educación o en las empresas con el objetivo de hacer más entretenidas estas actividades.

Para convertir un ejercicio normal en una actividad gamificada, esta debe incluir algunos de los principales elementos básicos de un juego, que son:

- Reglas, mediante las cuales se va a orientar al usuario, porque el usuario necesita saber qué es lo que debe hacer y cómo hacerlo.
- Objetivos, que el usuario tendrá en mente para poder alcanzar la meta.
- Retroalimentación, ya que el usuario necesita conocer si lo que está haciendo es co-

rrecto o no.

- Competición, ya sea mediante la acumulación de puntos o la superación de niveles, el usuario quiere tener alguna motivación para jugar.
- Recompensa: El usuario tiene que ser estimulado mediante la obtención de beneficios para continuar jugando.

3.4 Tecnologías relevantes para el desarrollo de aplicaciones gráficas interactivas

A lo largo de la historia de los videojuegos hemos podido comprobar una evolución visual, sonora y jugable en los videojuegos, como por ejemplo animaciones faciales, el comportamiento de las luces y sombras, texturas, o físicas. También han ido apareciendo experiencias de juego de toda clase, desde aventuras gráficas a videojuegos que usan realidad virtual y realidad aumentada.

Antiguamente los videojuegos se desarrollaban desde cero, de principio a fin, como entidades únicas, y cada juego se desarrollaba para una plataforma especifica [GS18]. Entre estos se podría resaltar videojuegos como las primeras entregas de *Donkey Kong*, *Dragon Quest* (figura 3.17), o *Final Fantasy*. Estos dos últimos originalmente desarrollados para la plataforma *NES*. Sin embargo, con el paso del tiempo la aplicación de nuevas tecnologías y nuevos paradigmas de programación convirtieron el desarrollo de videojuegos en una tarea árida. Debido a esto surgieron herramientas para facilitar las actividades de desarrollo: **los motores de videojuegos**. De esta manera se permitía reutilizar código creado anteriormente, y consecuentemente, abaratar los costes del desarrollo. Los desarrolladores ya no tienen que comenzar proyectos desde la nada, teniendo que compilar nuevo código. Ahora parten de bases establecidas. Gracias a la creación de motores, su difusión y liberación de licencias, hemos vivido la evolución de los videojuegos.

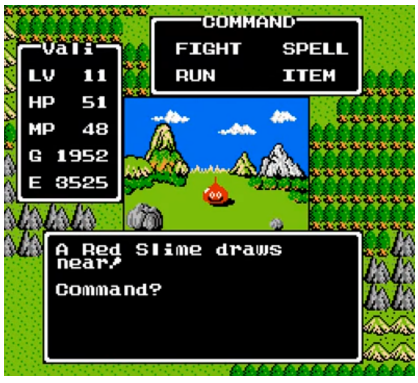

Figura 3.17: Captura de un gameplay de Dragon Quest (1986)

Los detalles y algoritmos de un videojuego no significan nada sin un sistema para diseñarlos, contenerlos y entregarlos. Un motor de videojuegos es el conjunto de rutinas de pro-

gramación que permiten el diseño, la creación y el funcionamiento de un videojuego. Este permite tratar aspectos como el renderizado 2D y 3D, gestionar la física, iluminación, detectar colisiones, utilizar música y audio, aplicar inteligencia artificial a los elementos del juego, uso de algoritmos, manejo de memoria, y, administrar redes (en juegos multijugador).

Casi todo lo relacionado con el desarrollo de un juego ocurre en un motor. Una empresa puede decidir si crear su propio motor o usar uno que ya exista. Para optar por una u otra opción habrá que tener en cuenta bastantes factores, aunque normalmente son las grandes empresas las que desarrollan sus propios motores. A continuación, se hace un repaso de algunos de los motores de videojuegos existentes hasta la fecha.

3.4.1 Unity

Seguramente se pueda afirmar que Unity[23] es el motor de videojuegos más popular actualmente. Por lo menos así lo demuestra Mario García Lázaro, programador líder en MercurySteam[24], en un estudio [GL20] que realizó sobre los motores usados para crear los mejores[25] juegos en los últimos años. En la figura 3.18 se puede apreciar la tendencia creciente a usar Unity.

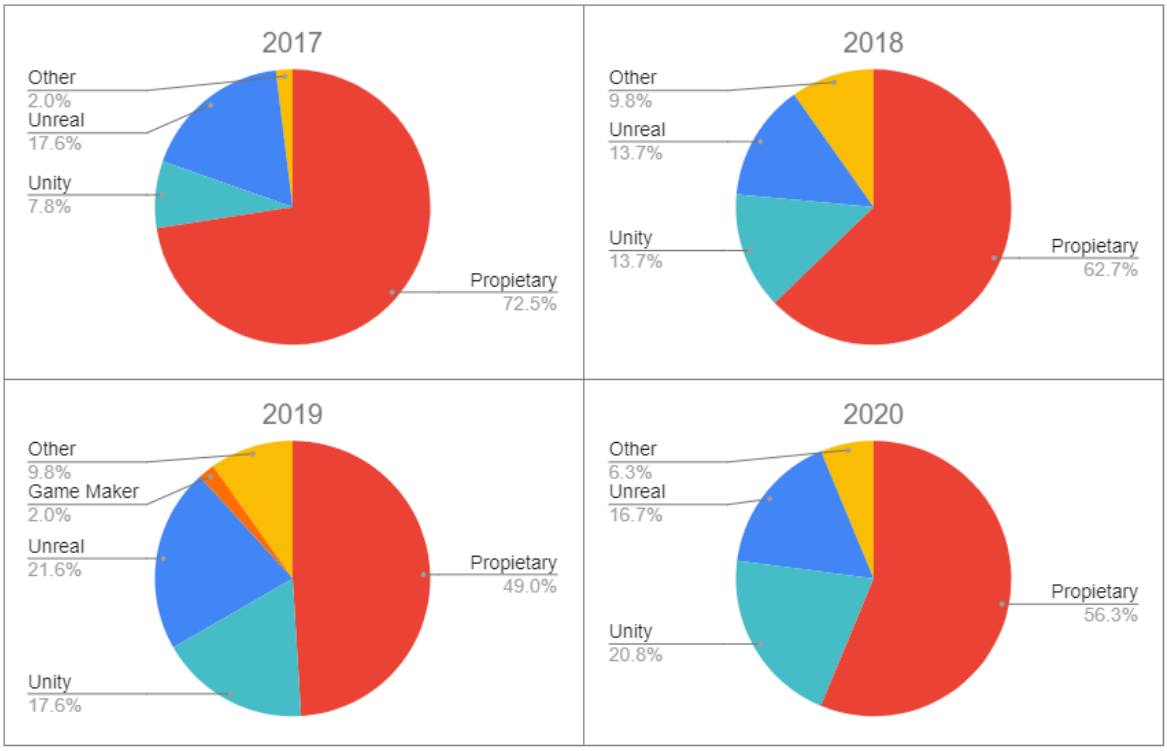

Figura 3.18: Motores usados para desarrollar los mejores juegos de los últimos 4 años[26]

[23] `https://unity.com/`
[24] `https://www.mercurysteam.com/`
[25] Los juegos fueron seleccionados buscando «*best games of the year 20XX*» en Google
[26] Fuente: `https://tinyurl.com/tksjgy5`

Este motor es perfecto para desarrollar juegos en 2D y 3D, y aplicaciones de realidad virtual y realidad aumentada. Además, es utilizado en otros sectores como películas y animación. Su popularidad se debe a su facilidad de uso, pero también por lo flexible que resulta, ya que permite desarrollar cualquier clase de proyectos por potentes que sean. Incluye una tienda muy variada de recursos utilizables para los juegos, muchos de ellos gratuitos.

Su característica mas destacable es lo barato que resulta utilizarlo, ya que la licencia personal es gratuita (siempre que los ingresos sean inferiores a 100 mil dólares en los últimos 12 meses). De esta forma, se encuentra accesible para todo aquel que desee utilizarlo, aunque también cuenta con planes de pago.

Unity es un motor de desarrollo de juegos multiplataforma, ya que permite desplegar el desarrollo a plataformas como iOS, Android, Windows, Linux, y consolas, o incluso en los principales navegadores.

Unity apareció en el año 2004, exclusivamente para desarrollar proyectos en Mac de Apple con dos versiones, *indie* y profesional. Pocos años más tarde, con el lanzamiento del iPhone y su compatibilidad con él se hizo todavía más conocido, y al final tuvo tanto éxito que se desmarcó como motor independiente. A mediados de la década sirvió para desarrollar juegos bastante emblemáticos como pueden ser *Hearthstone* y *Firewatch*. En los últimos años los juegos mas conocidos en los que se ha utilizado fueron *Hollow Knight*, un juego metroidvania bastante popular, o *Cuphead* (imagen 3.19), que destaca bastante por su aspecto gráfico.

Figura 3.19: Captura de un gameplay de Cuphead

3.4.2 Unreal Engine

Unreal Engine[27] es un motor creado por la compañía Epic Games en el año 1998. Inicialmente se dedicaba a desarrollar *shooters* en primera persona, aunque se utiliza con éxito en cualquier tipo de género, y prácticamente en cualquier plataforma, como PCs, consolas y smartphones. Esto es debido a su arquitectura abierta, por lo que puede adaptarse a las necesidades específicas de los desarrolladores para funcionar en diversas plataformas. También es usado en ámbitos como el cine, y para desarrollar aplicaciones de realidad virtual y realidad aumentada.

Actualmente es junto con Unity uno de los motores de juego más populares y usados del momento. Algunas de sus características es que cuenta con uno de los mejores motores de físicas que hay, o que tiene mucha potencia a nivel gráfico y estético [Eng20]. También es considerado uno de los motores con mejores visuales en el mercado, con un buen sistema de *ray casting*, tal y como podemos comprobar en [LLK+19]. Debido a esto, es bastante utilizado para el desarrollo de juegos de categoría AAA, y juegos de alto presupuesto.

Sin embargo, es un motor muy complejo y puede ser complicado para personas que cuentan con poca experiencia. Además, ha sido calificado de poco estable por algunos programadores en varios foros de la comunidad.

Unreal Engine cuenta con licencia gratuita, al igual que pasa con Unity, por lo que también permite desarrollar sin costes.Pero si se consiguen beneficios superiores a los 3000 dolares en un trimestre, Epic Games cobrará un 5 % de los beneficios.

Algunos ejemplos de videojuegos actuales desarrollados con este motor son *Bioshock* (ver figura 3.20), *Fornite*, *Street Fighter V*, *Sea of Thieves* y *Gears of War*.

Figura 3.20: Captura de un gameplay de Bioshock Infinite

[27] https://www.unrealengine.com/en-US/

3.4.3 CryEngine

CryEngine[28] es propiedad de la empresa desarrolladora de videojuegos Crytek, y fue lanzado al mercado en el año 2002. Este motor es el responsable de videojuegos tan conocidos como *Crysis* o *Far Cry*, título que dejó a todo el mundo impresionado con sus avances visuales en el año 2004. CryEngine está más centrado en el desarrollo de juegos en 3D.

Entre sus principales características, podemos destacar que se trata de un motor gráfico realmente potente, con un buen motor de físicas, sistema de iluminación y buena animación de modelos. Sin embargo, no está tan bien optimizado como los motores anteriores y cuenta con una gran curva de aprendizaje, por lo que será necesario dedicarle bastantes horas para usarlo con fluidez.

CryEngine permite ser empleado de forma gratuita, aunque siempre cobrará un 5 % de los beneficios obtenidos sin importar cuales sean estos, al contrario de lo que ocurría con Unity y Unreal, ya que estos necesitaban que el desarrollador alcanzara un mínimo de ganancias.

3.5 Conclusiones

Desde siempre ha existido la necesidad de tratar las lesiones, y uno de los remedios es realizar ejercicios de rehabilitación. Estos se han realizado de diferentes maneras a lo largo de la historia. En los últimos años estos se han dejado en manos de especialistas para ser hechos en los centros adecuados. Pero los avances en la tecnología han permitido la aparición de sistemas que permiten la realización de estos ejercicios de forma remota desde donde el paciente quiera hacerlos. Para poder utilizar estos sistemas se necesita capturar el movimiento de alguna forma.

Después de haber analizado las opciones disponibles en el mercado para capturar movimiento de personas, se ha llegado a la conclusión de que el dispositivo más adecuado para realizar este proyecto debe ser una cámara, ya que debe ser un dispositivo que no ponga demasiadas dificultades para realizar los ejercicios de rehabilitación, que ya de por sí son duros de hacer. Uno de los objetivos es motivar al paciente a que realice los ejercicios, por lo que no hay que poner problemas antes de empezar con ellos. De esta forma se ha concluido que debe ser un dispositivo ajeno al cuerpo del paciente, por lo que será una cámara la que capte el movimiento de este. La cámara elegida ha sido Azure Kinect DK, debido a las ventajas que posee como su facilidad de manejo, facilidad de transporte gracias a su tamaño, o el buen rendimiento que ofrece.

Por lo tanto, la biblioteca elegida para desarrollar el proyecto ha sido la que proporciona Microsoft para su Kinect, que en este caso es el SDK de *body tracking* de Azure Kinect DK. Este software es bastante eficiente y cumple su función realmente bien, por lo que no habrá ningún inconveniente en su uso.

[28] `https://www.cryengine.com/`

Una vez que se sabe el dispositivo a utilizar y la biblioteca necesaria para desarrollar, hay que pensar en qué es lo que se quiere desarrollar. Como el objetivo es motivar al paciente a que realice sus ejercicios, debe ser algo que le proporcione dicha motivación, y teniendo en cuenta eso, aparecieron los juegos serios. El propósito de estos juegos es ayudar al paciente con sus ejercicios de rehabilitación, ya sea mostrando ejemplos en pantalla, o con la aparición de un avatar que proporcione indicaciones. Para mantener motivado al paciente es importante que haya una retroalimentación del sistema con el usuario, para indicar cómo de bien o mal se está realizando el ejercicio. Utilizar un sistema de puntuación también es útil para saber el desempeño del usuario, o para motivarle y que intente superar sus mejores marcas.

Finalmente, una vez que ya se tiene la idea que se quiere desarrollar es necesario una tecnología para poder convertir esa idea en un producto. El motor elegido para tal propósito ha sido Unity. Hay diversos motivos de su elección, por ejemplo, su precio, su facilidad de uso, o su curva de aprendizaje respecto al resto de motores. Cabe destacar además su calidad gráfica y la comunidad detrás de este *software*.

Capítulo 4

Metodología

El presente capítulo se va a encargar de detallar la metodología de trabajo elegida, con la que se pretende desarrollar los objetivos propuestos al principio de trabajo (véase capítulo 2), y que mejor se adaptan a las características que necesita este proyecto. Para esto, se va a hablar de como se ha llevado a cabo la metodología de trabajo y la metodología seleccionada finalmente, junto a la planificación del proyecto. Para completar el capítulo se van a describir los medios software y hardware utilizados durante el desarrollo.

4.1 Metodología de trabajo

Teniendo en cuenta la naturaleza de este proyecto, ha sido necesaria una metodología que posibilite trabajar de forma paralela, sumando nuevas características a la vez que se corrigen los fallos que puedan aparecer, para después añadir todo ello al sistema final. Cada nueva funcionalidad se ha podido realizar de forma aislada para poder probar su funcionamiento sin interferencias en la versión del proyecto que existiera en ese instante. Una vez añadidas las nuevas funciones y corregidos los errores, se ha a procedido a realizar las pruebas de integración necesarias, con las que se ha a verificado que el sistema funciona correctamente.

El planteamiento de este proyecto ha consistido en realizar reuniones cada dos semanas, con el fin de comprobar el estado actual de desarrollo y acordar los objetivos que deben ser alcanzados en la reunión siguiente. Este método de trabajo también se ha basado en hitos que deben cumplirse de forma iterativa e incremental durante ese periodo de tiempo, acatando los requisitos de diseño del sistema. Cuando una parte está implementada por completo, se le han de realizar las pruebas de integración que sean necesarias para poder dar el hito correspondiente como finalizado y, a continuación, pasar al siguiente hito.

4.2 Desarrollo iterativo e incremental

Tras analizar las distintas metodologías disponibles, se ha llegado a la conclusión de que la mejor metodología usable para este proyecto es la de **desarrollo iterativo e incremental**. Es un tipo de metodología ágil en la cual el proyecto se divide en varias iteraciones de menor tamaño, las cuales duran un periodo corto de tiempo [Dis08].

Es una metodología bastante usada a la hora de desarrollar software. Debido a las grandes

dimensiones de la mayor parte de proyectos software, es bastante útil dividir el desarrollo de un proyecto en otras partes que sean proyectos más pequeños, para poder facilitar el desarrollo, con unas funcionalidades más específicas, pero que una vez integradas todas estas pequeñas partes, formen el sistema final que se había buscado.

Cada una de estas partes es una iteración, durante cada una de las cuales se repite un proceso de trabajo similar a la iteración anterior. Al estar todas las iteraciones separadas entre ellas, se puede trabajar en cada una de forma que no se interpongan entre otras iteraciones. En esta metodología se debe planificar al inicio del proyecto cuando se realiza cada iteración. Cada iteración se convierte en un incremento sobre el sistema final, aunque este incremento no siempre resulta ser aditivo.

En la figura 4.1 se puede comprobar el funcionamiento del desarrollo iterativo e incremental, de forma que cada iteración se divide en una serie de fases las cuales son la planificación, elicitación de requisitos, análisis y diseño, implementación, verificación y evaluación. Una vez acabada una iteración se pasa a la siguiente, repitiendo las mismas fases teniendo en cuenta los nuevos objetivos de la nueva iteración.

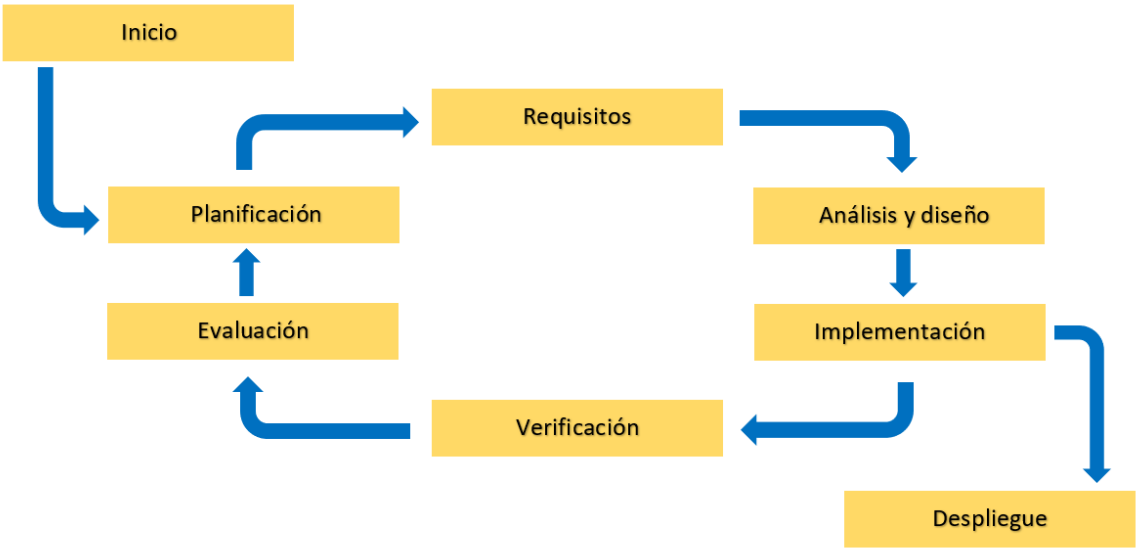

Figura 4.1: Esquema del desarrollo iterativo e incremental

4.3 Planificación

Una vez seleccionada la metodología, hay que planificar como desarrollar los objetivos planteados en el capítulo 2 de forma eficiente y completa. Para llevar a cabo esto, el proyecto fue dividido en hitos durante los cuales se debe realizar una tarea. La extensión de cada hito se mide mediante *sprints*, los cuales tienen una duración de dos semanas cada uno. En las siguientes líneas se explica con detalle que es lo que se realiza en cada hito, junto con la duración que fue necesaria para cada uno.

1. **Creación de ejercicios con elementos de gamificación**: Para iniciar el proyecto, fue necesario generar unos prototipos de ejercicios muy básicos, con elementos 3D que recreasen la trayectoria del ejercicio que el paciente debe realizar, junto a etiquetas en la interfaz para mostrar/incrementar el número de repeticiones y puntuación. Esta tarea se realizó entre el 16 de octubre de 2020 y el 13 de noviembre de 2020 con una duración de dos sprints y finalizó en el sprint 2.

2. **Definición de enfoque para automatizar la generación de ejercicios de rehabilitación**: Hubo que analizar cuál sería el mejor enfoque para acelerar la generación de *exergames*. Después de revisar varios artículos con distintos enfoques, como por ejemplo la especificación del lenguaje PEL para automatizar la generación de *exergames* [VGPA+20], finalmente se estableció que el mejor enfoque era un archivo JavaScript Object Notation (JSON) que almacenase toda la información relacionada con un ejercicio. Esta tarea fue realizada entre el 14 de noviembre de 2020 y el 18 de diciembre de 2020 durante dos sprints, se hizo de forma paralela con el *desarrollo de un prototipo de parser para generar simples exergames*, y fue terminada en el sprint 4.

3. **Desarrollo de un prototipo de parser para generar simples *exergames***: Como punto de partida, se tuvo que desarrollar un prototipo de parser que tenía que leer información, entre la que se encontraba límite de puntos por ejercicio, tiempo límite, además de repeticiones y un número máximo de repeticiones, para poder generar, automáticamente, un *exergame*. Esta tarea fue desarrollada entre el 14 de noviembre de 2020 y el 18 de diciembre de 2020 durante dos sprints, se realizó de forma paralela con la *definición de enfoque para automatizar la generación de ejercicios de rehabilitación*, y su final fue en el sprint 4.

4. **Creación de jerarquía de archivos para cargar niveles**: En esta tarea fue definida una nueva jerarquía entre los archivos que contienen la definición de un *exergame*. De esta forma, existe un archivo padre que contiene la información básica del ejercicio, y archivos hijos que heredan esa información relacionada con los distintos niveles de dificultad que puede tener un ejercicio. Esta tarea fue realizada desde el 19 de diciembre de 2020 hasta el 5 de febrero de 2021. Tuvo una duración de tres sprints y se hizo de forma paralela junto al *desarrollo de un parser que lea la estructura de un ejercicio y genere un exergame*. Esta tarea finalizó en el sprint 7.

5. **Desarrollo de un parser que lea la estructura de un ejercicio y genere un *exergame***: Esta tarea está bastante relacionada con la tarea vista anteriormente del *desarrollo de un prototipo de parser para generar simples exergames*, ya que consistió en establecer de manera definitiva todo lo planteado en el prototipo, corrigiendo los errores cometidos y añadiendo características que aún no habían sido pensadas para el prototipo. Esta tarea tuvo lugar entre el 19 de diciembre de 2020 y el 5 de febrero de 2021,

tuvo una duración de tres sprints, fue realizada de forma paralela con la *creación de jerarquía de archivos para cargar niveles*, y finalizó en el sprint 7.

6. **Integración de avatar virtual**: Para continuar el proyecto, hubo que añadir en la escena un avatar virtual y mapear el conjunto de articulaciones del avatar a los movimientos de un paciente. Fue necesario investigar otros proyectos de *body tracking*, como por ejemplo *Sample Unity Body Tracking Application*[1], del repositorio Azure-Kinect-Samples de Microsoft, para tener referencias de como funciona el mapeo de articulaciones. Esta tarea fue desarrollada entre el 5 de febrero de 2021 y el 2 de abril de 2021, se alargó cuatro sprints, y se acabó en el sprint 11.

7. **Agregación de escenarios virtuales**: El motivo de la realización de esta tarea fue mejorar el aspecto visual del proyecto. Por lo tanto, fue necesario añadir escenarios virtuales con el objetivo de que el sistema integre contenido variado y ayuda a motivar al usuario a realizar ejercicios. Los escenarios añadidos fueron una aldea, una cafetería, un apartamento y un desierto. Esta tarea se llevó a cabo entre el 2 de abril de 2021 y el 16 de abril de 2021 durante un sprint, y fue terminada en el sprint 12.

8. **Mejoras de interfaz**: El texto incluido en la interfaz era del prototipo inicial que se realizó en la primera tarea, el cual se mejoró cambiando su tipografía. También hubo más mejoras de interfaz como la adición de partículas cuando se realicen ciertas acciones para dar *feedback* al usuario de que se está realizando correctamente, o hacer que las esferas cambien de color cuando pase por ella la articulación correcta. Esta tarea se realizó entre el 17 de abril de 2021 y el 30 de abril de 2021 durante un sprint, y fue finalizada en el sprint 13.

9. **Integración del cambio entre niveles de dificultad**: Tarea relacionada con la de *creación de jerarquía de archivos para cargar niveles*, puesto que debe ser posible cambiar la dificultad de un ejercicio en tiempo de ejecución mediante la pulsación de teclas en el teclado. Hasta el momento tan solo se podía cambiar de dificultad reiniciando el ejercicio con la nueva dificultad elegida. Esto sirve de ayuda al terapeuta a cambiar la dificultad de un ejercicio en tiempo de ejecución si observa que el paciente no es capaz de realizar el ejercicio propuesto. Esta tarea tuvo lugar entre el 1 de mayo de 2021 y el 14 de mayo de 2021 en un sprint, y fue acabada en el sprint 14.

10. **Adición de nuevos ejercicios**: Una vez terminada la funcionalidad principal del proyecto era necesario añadir más ejercicios en la especificación JSON creada anteriormente, puesto que hasta entonces tan solo había un *exergame* que probaba que el sistema funcionaba correctamente. Esta tarea fue realizada desde el 14 de mayo de 2021 hasta el 25 de junio de 2021 durante tres sprints. Esto se realizó de forma paralela con el *desarrollo de nuevo estilo de ejercicios* y finalizó en el sprint 17.

[1] `https://tinyurl.com/z4n4z3uk`

11. **Desarrollo de nuevo estilo de ejercicios**: Esta tarea añadió una nueva funcionalidad al proyecto principal. La nueva funcionalidad consistió en crear un nuevo tipo de ejercicio en el que el paciente interactuase con objetos del escenario, por ejemplo, recogiéndolos o posicionándolos en un nuevo lugar. El objetivo de esto fue que el paciente vea el ejercicio como un juego en el cual se ejercita al mismo tiempo que se divierte. Esta tarea se llevó a cabo entre el 14 de mayo de 2021 y el 25 de junio de 2021, tuvo una duración de tres sprints, se hizo de forma paralela junto a la *adición de nuevos ejercicios*, y fue terminada en el sprint 17.

Toda la información acerca de la planificación completa del proyecto se encuentra resumida de manera gráfica en el diagrama de Gantt de la figura 4.2. Como podemos observar, en este TFG se han invertido ocho meses y medio, y ha sido dividido en 11 hitos realizados durante 17 sprints.

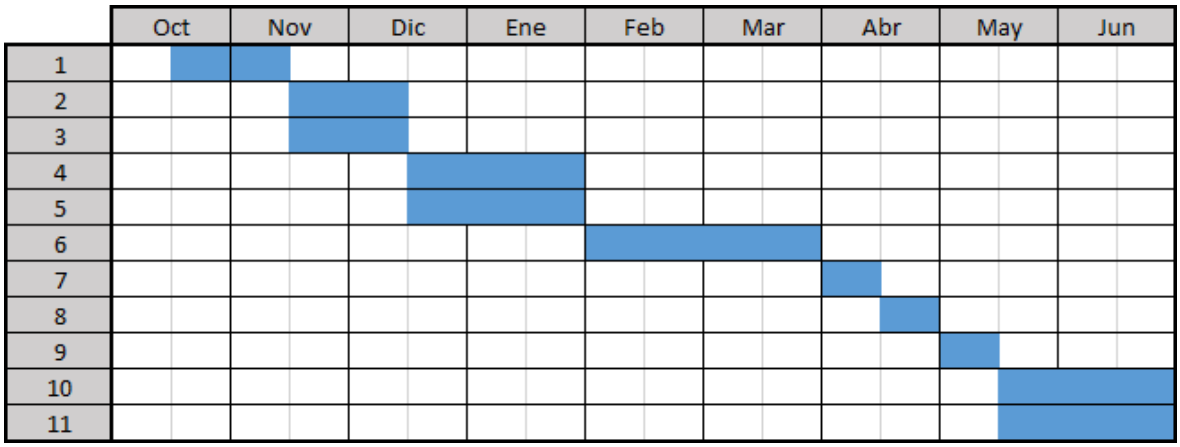

Figura 4.2: Diagrama de Gantt con la planificación del trabajo

4.4 Medios empleados

Para finalizar el presente capítulo, en este apartado se van a describir todos los medios utilizados para el desarrollo y la implementación de este proyecto. Primero se detallará el equipamiento hardware, para después pasar a los medios software, entre los que se incluyen sistemas operativos, herramientas de desarrollo, de edición gráfica, de control de versiones, lenguajes de programación y bibliotecas, entre otros muchos.

4.4.1 Medios hardware

En primer lugar, en esta subsección se van a especificar los medios hardware utilizados para el desarrollo y despliegue de la aplicación, que son los siguientes:

- **Azure Kinect**: Dispositivo hardware compuesto por una serie de cámaras y sensores. Se trata de un medio hardware bastante usado en el ámbito de la investigación. Proporcionado por la UCLM, será el dispositivo utilizado para la captura de movimientos.

Nos va a permitir obtener la información relacionada con las articulaciones del usuario, gracias a su precisión en el *tracking*.

- **PC**: Ordenador utilizado para el desarrollo y despliegue del proyecto. Es propiedad del alumno y cuenta con los requisitos necesarios para desarrollar aplicaciones con gran coste computacional. Sus características principales se muestran en el cuadro 4.1:

Procesador	AMD Ryzen 5 1400 Quad-Core Processor
Memoria Ram	16 GB
Tarjeta gráfica	NVIDIA GeForce GTX 1650
Almacenamiento	SSD de 500 GB y HDD de 2 TB

Cuadro 4.1: Especificaciones del pc utilizado

4.4.2 Medios software

En segundo lugar, toca precisar los medios software más relevantes que han sido requeridos para el desarrollo, que son los que se describen a continuación. Además, han sido clasificados por distintas categorías para poder agruparlos según su utilidad:

Sistemas operativos

- **Windows 10 Home**: Sistema operativo utilizado para desarrollar la aplicación. Es el más adecuado para este proyecto por motivos de compatibilidad con Kinect, ya que ambos pertenecen a Microsoft.

Herramientas de desarrollo

- **Unity**[2]: Motor de videojuegos multiplataforma utilizado para desarrollar software. Es compatible con diversos dispositivos, entre los que se incluye Azure Kinect. La versión seleccionada es la 2019.1.2f1.
- **Visual Studio 2019**[3]: Es el editor de código con el que se va a escribir las partes de código en lenguaje C#. Además también es de Microsoft y es el recomendado por Unity.

Lenguajes de programación

- **C#**: Lenguaje de programación orientada a objetos desarrollado por Microsoft. Su elección viene por ser el mejor lenguaje soportado por Unity para la elaboración de *scripts*.

Bibliotecas

- **Azure Kinect Sensor SDK**[4]: SDK utilizado por Azure Kinect. Esta biblioteca nos

[2] https://unity.com/

[3] https://visualstudio.microsoft.com/es/

[4] https://docs.microsoft.com/en-us/azure/kinect-dk/sensor-sdk-download

permite controlar todos los sensores del dispositivo, activar las cámaras y usar el array de micrófonos incluido.

- **Azure Kinect Body Tracking SDK**[5]: Biblioteca proporcionada por Azure Kinect. Es la encargada de interpretar la información relacionada con el movimiento de las articulaciones de los cuerpos capturados por el dispositivo.

Formato de texto

- **JSON**[6]: Es un formato de texto con el que se recopilan datos. Se utiliza para almacenar la información relacionada con un *exergame*.

Herramientas de edición gráfica

- **GIMP**[7]: Programa de edición de imágenes. Se ha utilizado para modificar las imágenes que necesita mostrar el proyecto.

Herramientas de planificación

- **Trello**[8]: Es un software utilizado para administrar y planificar proyectos. Consiste en un tablón en el que se colocan tarjetas virtuales por secciones. Las tarjetas representan una tarea del proyecto que debe completarse, e incluyen información sobre esta, como puede ser una descripción, persona encargada de la misma, o el momento en el cual debe estar terminada.

Herramientas de control de versiones

- **Git**[9]: Sistema elegido para controlar las distintas versiones del proyecto. El servicio de repositorio que se ha utilizado ha sido GitHub[10], y además, también se ha usado la herramienta GitHub Desktop[11] para actualizar las versiones en el repositorio GitHub.

Documentación

- **LaTeX**[12]: Es un sistema de construcción de texto de alta calidad empleado para escritura de libros y artículos científicos. Se ha utilizado para elaborar la documentación del proyecto. El editor de texto LaTeX seleccionado es el editor en línea Overleaf[13]

[5] https://docs.microsoft.com/en-us/azure/kinect-dk/body-sdk-download
[6] https://www.json.org/json-en.html
[7] https://www.gimp.org/
[8] https://trello.com/
[9] https://git-scm.com/
[10] https://github.com/
[11] https://desktop.github.com/
[12] https://www.latex-project.org/
[13] https://www.overleaf.com/

Capítulo 5

Arquitectura

En este capítulo se van a estudiar los distintos elementos que componen la arquitectura del sistema desde un punto de vista técnico, exponiendo los problemas encontrados durante el desarrollo y su solución. Pero, antes de entrar en detalle, se proporciona una visión general del sistema para, después, profundizar en esta en cada uno de los apartados de este capítulo.

5.1 Visión general

Para comenzar a hablar de la arquitectura, es recomendable mostrar una visión general que resuma el sistema completo para después dar los detalles más específicos de cada una de las partes. Este proyecto esta compuesto por tres módulos claramente diferenciados que se comunican entre ellos formando un conjunto, los cuales son: módulo de definición *exergames*, módulo encargado de gestionar los movimientos de un usuario (*body tracking*), y un tercero para manejar la gamificación del sistema. Podemos ver un resumen de esto en el diagrama de la figura 5.1.

El primero de estos módulos es el encargado de definir los *exergames*. Este creará una especificación con la que automatizar el desarrollo de los ejercicios. Dicha especificación esta formada por una clase padre encargada de guardar la información general de los ejercicios, y una clase hija que hereda de esta y aporta parámetros nuevos para definir un nivel de dificultad del *exergame*. Los ejercicios se guardarán en una estructura en formato JSON. Este módulo se comunica con el de gamificación, puesto que le envía los datos de los ejercicios en formato JSON.

El segundo módulo recibe datos sobre el movimiento que realiza un usuario para transformar el avatar del juego en consecuencia. No obstante, este módulo debe seguir primero una serie de pasos, entre los cuales destacan activar la cámara, capturar *frames* y recoger los datos del cuerpo del usuario, es decir, los datos de las articulaciones. Esta información será trasladada al avatar de la escena, mediante la cual se conseguirá replicar los movimientos del jugador en el avatar. Este movimiento también será enviado al módulo de gamificación, ya que los necesita para la dinámica del juego.

Por último, el tercer módulo gestiona los procesos de gamificación del sistema. Este mó-

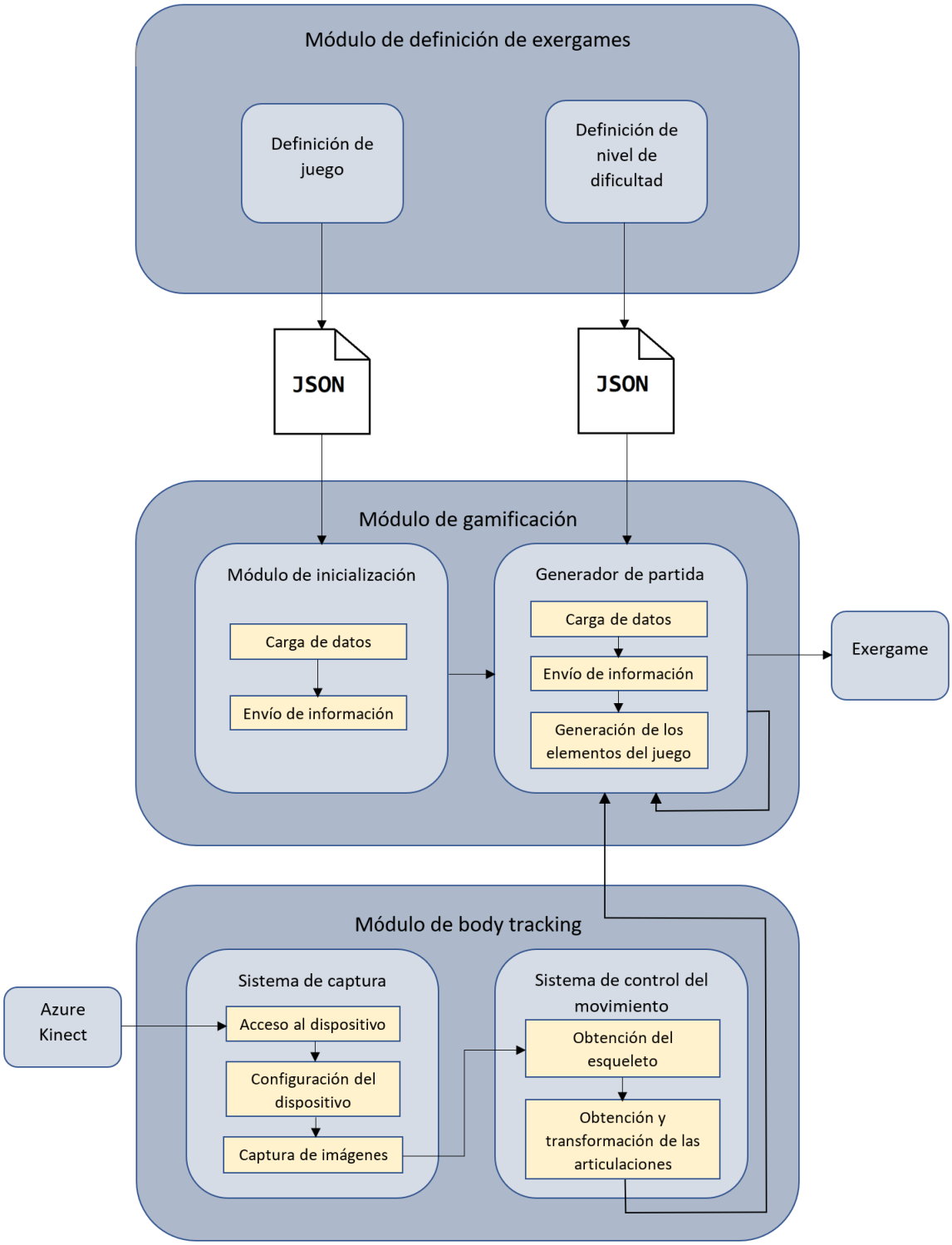

Figura 5.1: Diagrama general de la arquitectura

dulo hace uso de los datos recibidos tanto del primer módulo como del segundo. Por un lado, extrae los datos de un *exergame* para generar un escenario que tenga distintos objetos con los que el jugador tiene que interactuar. Además, para producir una partida de un *exergame*, también hace uso de distintos elementos de gamificación, como puede ser el uso de puntuación o

tiempo. Por otro lado, la información del movimiento recibida del módulo de *body tracking* sirve para replicar los movimientos del usuario en el modelo del avatar con el objetivo de poder interactuar con los elementos del *exergame*.

5.2 Módulo de definición de Exergames

En este proyecto un *exergame* esta dividido en dos elementos. El primero se encarga de definir los elementos generales de un ejercicio, como pueden ser el nombre, una descripción de este, la configuración de la cámara, el uso de puntuación en el juego, la cantidad de niveles de dificultad, o el tipo de ejercicio. El segundo elemento se refiere a especificaciones que se puedan añadir al *exergame* y, que en este caso, tratarán los aspectos relacionados con el nivel de dificultad del juego. De esta forma, se consigue mantener dichos niveles de dificultad relacionados entre sí, para tener un mismo juego que puede ser realizado de diversas maneras dependiendo de la dificultad seleccionada. Tanto el primer elemento del *exergame* como el segundo guardarán sus datos en archivos en formato JSON.

Existen dos tipos de *exergames* diferenciados entre ellos, y que serán denominados como «normal» o «especial». Un *exergame* **normal** consiste en un juego serio en el que el usuario debe mover la articulación correspondiente a ese juego por la trayectoria que le indique el sistema. Si pasa por todos los puntos de la trayectoria se completa una repetición, y debe hacer tantas como se le indique para poder dar por concluida la partida. El *exergame* **especial** es un juego en el que el jugador deber recoger un objeto del escenario y colocarlo encima de otro objeto especificado por el juego. Una vez que el objeto esté sobre el otro, se reinicia la posición de dicho objeto y se completa una repetición. Al igual que antes, la partida se acaba cuando se hayan completado todas las repeticiones.

5.2.1 Generación de exergames

Para poder organizar la generación de *exergames* se define una especificación que se debe seguir para que todos los ejercicios estén unificados bajo los mismos criterios, y además, acelerar la creación de los mismos. Para definir esta especificación se ha creado una clase llamada *Exergame* que contiene todos los atributos para generar objetos de esta clase. Podemos ver las clases necesarias para generar un *exergame*, sus atributos, y sus relaciones en la figura 5.2.

Este conjunto de clases relacionadas entre sí tiene la función de servir como definición para la creación de objetos de tipo *Exergame*, usando como base archivos en formato JSON con una estructura determinada de la que hablaremos más adelante. Estos JSON son los encargados de aportar la información de los atributos de los objetos *exergame*. Estos objetos tienen la función de guardar los datos generales de un ejercicio en el sistema para poder manipular sus atributos de una forma cómoda.

Un *exergame* está compuesto por los siguientes elementos:

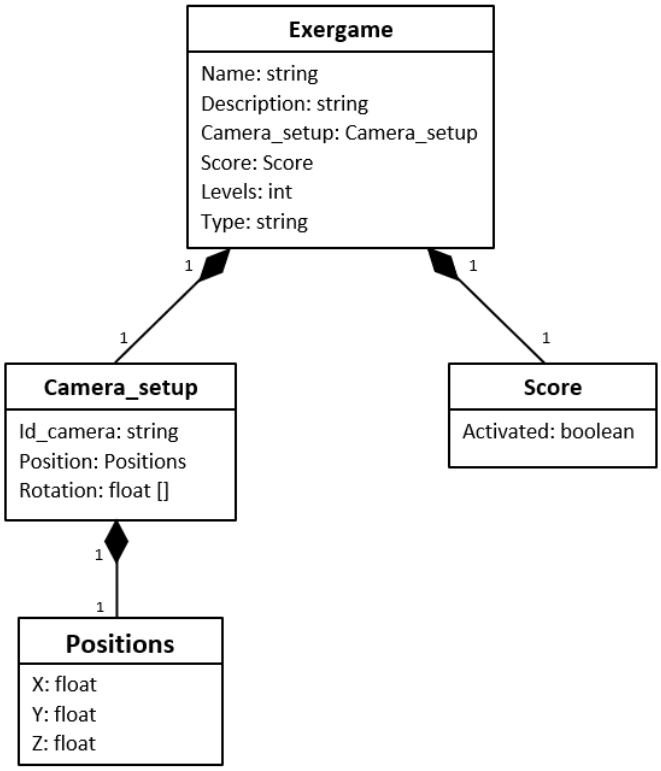

Figura 5.2: Clases para definir un exergame

- **Name**: Este atributo indica el nombre del *exergame* mediante un `string`.
- **Description**: muestra un `string` en el que se dan las instrucciones a seguir para poder completar el ejercicio.
- **Camera_setup**: Es un objeto de la clase también llamada *Camera_setup*. Se encarga de guardar la información relacionada con la cámara usada en el escenario de juego. Contiene los siguientes atributos:
  - **Id_camera**: Es un `string` que indica el identificador de la cámara. Es útil cuando en un mismo escenario tenemos varias cámaras disponibles.
  - **Position**: Mediante este objeto de tipo `Positions` se indica la posición de la cámara en el espacio de la escena, utilizando los tres atributos flotantes de la clase *Positions* llamados X, Y, Z con los que se indica una coordenada.
  - **Rotation**: Utilizando este *array* de flotantes se indica la rotación de la cámara en el espacio, también mediante coordenadas X, Y, Z.
- **Score**: Este objeto de la clase *Score* contiene la información de la puntuación.Esta clase contiene solo un elemento:
  - **Activated**: Mediante este atributo de tipo `boolean` se indica si el ejercicio necesita guardar o no la puntuación.

- **Levels**: Representa la cantidad de niveles de dificultad que tiene el *exergame* utilizando un número entero.
- **Type**: Es un `string` utilizado para indicar que tipo de ejercicio es. Debe tomar el valor «Normal» o el valor «Especial».

5.2.2 Sistema de niveles

Este conjunto de clases y sus relaciones tiene la función de ser una definición utilizada para la creación de objetos de tipo *ExergameLvl*, al igual que ocurría con el conjunto de clases cuya función era generar objetos *Exergame*. También toman como referencia archivos en formato JSON, de los que obtienen la información de los atributos de los objetos *exergameLvl*. Una vez que ha sido generado el objeto *exergameLvl*, el resto de módulos podrá manipular sus atributos de manera sencilla.

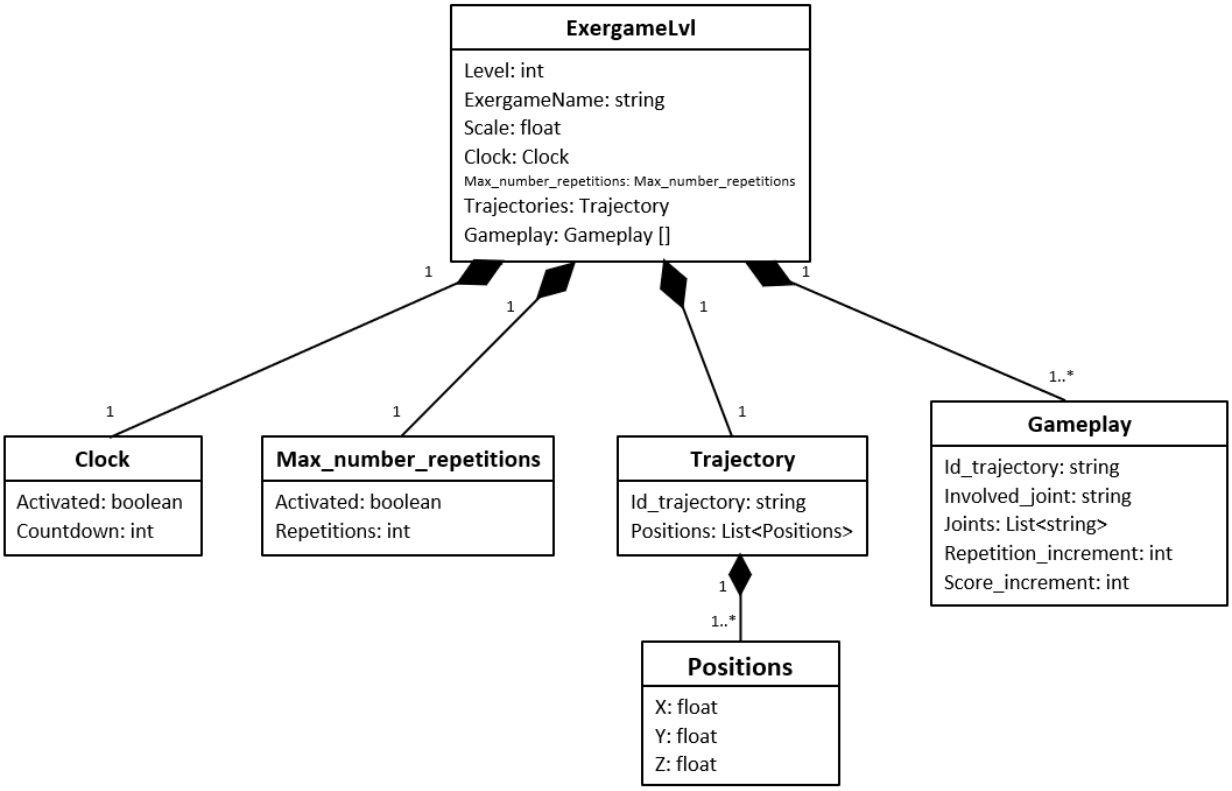

Figura 5.3: Clases necesarias en un nivel de dificultad

Un nivel de dificultad está compuesto por los siguientes elementos:

- **Level**: Este atributo indica cual es el nivel de dificultad utilizando un número entero. Se pueden definir tantos niveles de dificultad como sean necesarios.
- **ExergameName**: Mediante este atributo de tipo `string` se identifica el nombre del *exergame* padre del que está heredando este nivel de dificultad.

- **Scale**: Con este atributo de tipo flotante se cambia el tamaño de las esferas de colisión en el caso de ejercicios normales, o el tamaño de los objetos interactuables en el caso de ejercicios especiales. Cambiar el tamaño tiene como utilidad cambiar la dificultad en la realización de un ejercicio, puesto que a mayor tamaño de las esferas y los objetos mayor facilidad a la hora de conseguir que la articulación pase por ellos.
- **Clock**: Este elemento controla el tiempo de una partida. Es un objeto de la clase *Clock* y tiene las siguientes características:
  - **Activated**: Mediante este `boolean` se activa o desactiva la utilización del tiempo en el ejercicio.
  - **Countdown**: Es un número entero que indica el tiempo máximo que se tiene para hacer el ejercicio. Se mide en segundos y a mayor nivel de dificultad menor será el tiempo disponible para realizar el ejercicio.
- **Max_number_repetitions**: Este atributo es un objeto de la clase también llamada Max_number_repetitions y sirve para guardar los datos de las repeticiones para un ejercicio. Estos son sus elementos:
  - **Activated**: Es un `boolean` con el que se activa o desactiva el límite de repeticiones.
  - **Repetitions**: Con este entero se indica el numero de repeticiones que deben realizarse para completar un ejercicio. Cuanto mayor sea la dificultad del ejercicio más repeticiones serán necesarias para completarlo.
- **Trajectories**: Este elemento es un objeto de la clase *Trajectory*. Nos permite controlar la información del movimiento que debe seguir la articulación de un paciente para poder completar un ejercicio. Sus atributos son los siguientes:
  - **Id_trajectory**: Es un `string` que guarda un identificador con el que se identifica la trayectoria a seguir.
  - **Positions**: Es una lista de objetos `Positions`. Este tipo de objetos tienen tres atributos flotantes llamados X, Y, Z con los que se indica una coordenada en el espacio 3D para identificar una posición. Cabe recordar la existencia de dos tipos de ejercicios, «normal» y «especial», puesto que dependiendo del tipo, la información registrada hará referencia a distintos elementos de la escena de un juego. Si es normal, guarda la información de las posiciones en el escenario de las esferas de colisión. Se pueden añadir tantas posiciones como esferas sean necesarias para completar una repetición de un ejercicio. Cuanto más queramos aumentar la dificultad de un ejercicio, más posiciones serán necesarias. En cambio, si el ejercicio es especial, serán siempre dos posiciones de los dos objetos que intervienen en un ejercicio especial.

- **Gameplay**: El último atributo es una lista de objetos *Gameplay* que normalmente tendrá un solo objeto. Guarda información relacionada con una partida e incluye los siguientes elementos:
  - **Id_trajectory**: Mediante este `string` se relaciona el objeto *Gampeplay* con el objeto *Trajectories* mencionado antes al tener ambos el mismo identificador.
  - **Involved_joint**: Es un `string` que indica el nombre de la articulación que va a interactuar con los objetos creados en la escena.
  - **Joints**: En esta lista de `strings` que guarda el nombre de las articulaciones que queremos que el sistema de *tracking* capture. Para capturar el movimiento de todo el cuerpo completo será necesario incluir el nombre de todas las articulaciones.
  - **Repetition_increment**: Este entero indica cuántas repeticiones se quieren sumar al marcador cada vez que se interactue con todos los objetos de una partida. Por lo tanto, para añadir una repetición, este campo debe valer uno.
  - **Score_increment**: Este entero indica el valor que se sumará a la puntuación de la escena cada vez que la articulación toque de forma correcta los objetos que forman una trayectoria.

5.2.3 Estructura JSON

La estructura de un exergame se define mediante archivos JSON. Se ha elegido este formato para conservar los datos de un *exergame* debido a su sencillez, legibilidad y la facilidad para escribir que proporciona. Como el formato JSON es independiente de cualquier lenguaje de programación, ofrece la posibilidad de reutilizar los datos de los *exergames* en otros sistemas programados de distinta manera. Las ventajas de JSON con respecto a otros formatos, por ejemplo XML, es su simplicidad como formato, su alta velocidad de procesamiento, o que cuenta con menor complejidad a la hora de parsearlo.

Los *exergames* en formato JSON están basados en relaciones uno a uno entre claves y valores. En primer lugar, el campo clave proporciona el nombre del atributo de las clases *Exergame* y *ExergameLvl* vistas anteriormente en las secciones 5.2.1 y 5.2.2, para después aportarle la información de ese atributo rellenando el campo valor de la relación. De esta forma se puede personalizar cada *exergame*, y a través de un parser se generarán los ejercicios automáticamente extrayendo los datos.

Tenemos dos tipos de archivos JSON: El primero define la información general del ejercicio a realizar. Este guarda los datos relacionados con el nombre del *exergame*, su descripción. Además, almacena la configuración de la cámara con su identificador, posición y rotación. También se indica la puntuación a incrementar y si esta opción se muestra o no en el juego, el número de niveles de dificultad y el tipo de *exergame*. En el listado 5.1 podemos ver como ejemplo de *exergame* el de *Elevación de pierna extendida*.

```
{
    "Name": "Elevacion de pierna extendida",
    "Description": "Atraviesa con el pie derecho los objetivos desde abajo hacia arriba en la
        postura indicada.",
    "Camera_setup": {
        "Id_camera": "camera1",
        "Position": [0, 2.92, -5.5],
        "Rotation": [0, 0, 0]
    },
    "Score": {
        "Activated": true
    },
    "Levels": 3,
    "Type": "Normal"
}
```

Listado 5.1: Ejemplo de exergame en formato JSON

El segundo tipo de archivo JSON es el que contiene la información específica del nivel de dificultad del ejercicio seleccionado. Esta vez los datos guardados serán el nivel de dificultad, el nombre del *exergame* padre del que hereda, el tamaño de los objetos interactuables, la información del tiempo, las repeticiones necesarias, las posiciones de los objetos interactuables, y, por último, la información de la partida, con las articulaciones necesarias, los puntos que se obtienen, etc. En el listado 5.2 podemos ver como se codifica el nivel 1 del exergame *Elevación de pierna extendida* correspondiente al listado anterior (ver listado 5.1).

5.3 Módulo de Body Tracking

Este módulo está dividido en otros dos módulos bien diferenciados. El primero es el módulo de captura, y sus tareas son: acceder a la cámara de Azure Kinect, configurar sus parámetros e iniciar la captura de imágenes. Estas imágenes se actualizarán repetidas veces cada segundo y son enviadas al segundo módulo encargado de dar movimiento al avatar. Aquí es donde se extrae la información de un cuerpo. Esto permite obtener los datos relacionados con las articulaciones del cuerpo capturado. Finalmente, la información de las articulaciones se transfiere a las articulaciones del avatar del usuario que se muestra en la escena.

```
{
    "Level": 1,
    "ExergameName": "Elevacion de pierna extendida",
    "Scale": 0.55,
    "Clock": {
        "Activated": true,
        "Countdown": 120
```

```
    },
    "Max_number_repetitions": {
        "Activated": true,
        "Repetitions": 4
    },
    "Trajectories": {
            "Id_trajectory": "pie-derecho-arriba",
            "Positions": [
                {
                    "X": 1.8,
                    "Y": 1.6,
                    "Z": 0
                },
                {
                    "X": 1.7,
                    "Y": 2.6,
                    "Z": 0
                }
            ]
     },
    "Gameplay": [
        {
            "Id_trajectory": "pie-derecho-arriba",
            "Involved_joint": "foot.r",
            "Joints": [
                "chest",
                "cadera 1",
                "uperleg.l",
                "leg.l",
                "uperleg.r",
                "leg.r"
            ],
            "Repetition_increment": 1,
            "Score_increment": 50
        }
    ]
}
```

Listado 5.2: Ejemplo del nivel 1 de un exergame en formato JSON

5.3.1 Sistema de captura de movimiento

Para comenzar a obtener las imágenes capturadas por Azure Kinect es necesario acceder a la cámara. El dispositivo tiene dos cámaras: una RGB y otra de profundidad. Para el caso de este proyecto la cámara RGB ha sido obviada ya la cámara de profundidad proporciona la información que se requiere, es decir, la información relativa a los cuerpos de la escena. Para

ello, se inicializa el dispositivo utilizando la biblioteca Azure Kinect SDK. A continuación, se indica la configuración adecuada para nuestro caso de uso. En esta configuración se indica la velocidad necesaria de *frames* por segundo de los sensores de color y de profundidad, la resolución de la cámara RGB, que esta resolución debe ser desactivada puesto que no es necesaria, el modo de captura del sensor de profundidad, y finalmente, el modo de sincronización cuando están conectados dos o más dispositivos. En este caso el modo seleccionado será único. Todo este proceso de apertura del dispositivo y configuración se puede ver codificado en el listado 5.3.

```
using (Device device = Device.Open(id))
{
    device.StartCameras(new DeviceConfiguration()
    {
        CameraFPS = FPS.FPS30,
        ColorResolution = ColorResolution.Off,
        DepthMode = DepthMode.NFOV_Unbinned,
        WiredSyncMode = WiredSyncMode.Standalone,
    });
    ...
}
```

Listado 5.3: Apertura y configuración de Azure Kinect

Una vez que la cámara está activada, se continua con la siguiente fase del módulo de captura, donde se ejecuta un proceso en bucle durante todo el funcionamiento de la aplicación. Este proceso consiste en capturar las imágenes con el sensor de profundidad y almacenarlas en memoria. Los *frames* de la cámara se capturan a través de un hilo, para mejorar el rendimiento de la aplicación. La información se procesa dentro de ese hilo y se almacena en un objeto de tipo *BackgroundData*.

La información contenida en cada *BackgroundData* es una marca de tiempo, la altura y anchura de la imagen, su tamaño, el número de cuerpos capturados, una lista de objetos que describe la información de esos cuerpos (estos datos son un identificador, y las posiciones y rotaciones de sus articulaciones), y finalmente, la profundidad de la imagen. Todo esto lo podemos ver reflejado en el diagrama de clases de la figura 5.4

Finalmente, cada uno de estos objetos *BackgroundData* serán enviados al módulo que controla los movimientos del avatar, ya que estos *frames* tienen la información necesaria relacionada con los esqueletos capturados.

5.3.2 Control del movimiento

Veamos ahora qué pasos sigue el sistema cuando recibe un *frame* obtenido mediante el módulo de captura de movimiento (ver sección 5.3.1). Recordemos que el *frame* obtenido

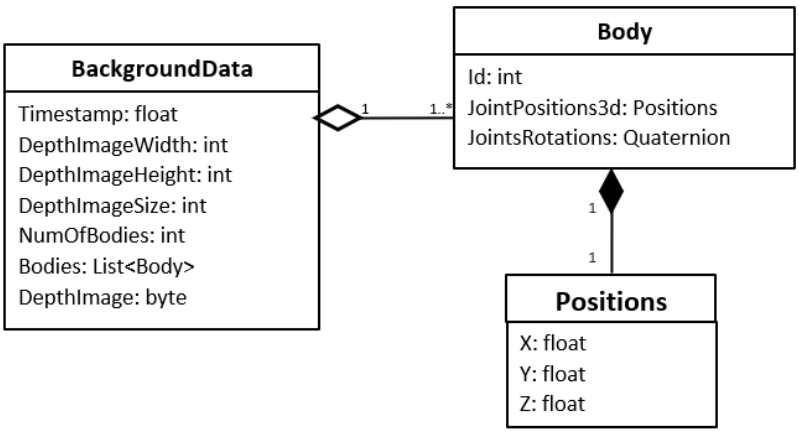

Figura 5.4: Clases necesarias para guardar la información de un frame

es un objeto de tipo *BackgroudData*, y que este objeto contiene diversa información de los *frames*, por ejemplo, el número de esqueletos capturados. Teniendo en cuenta esto, del *frame* capturado se extrae la lista de esqueletos reconocidos en un instante de tiempo. Esta lista analiza mediante la función del listado 5.4.

```
1  private int findClosestTrackedBody(BackgroundData trackerFrameData)
2  {
3      int closestBody = -1;
4      const float MAX_DISTANCE = 5000.0f;
5      float minDistanceFromKinect = MAX_DISTANCE;
6      for (int i = 0; i < (int)trackerFrameData.NumOfBodies; i++)
7      {
8          var pelvisPosition = trackerFrameData.Bodies[i].JointPositions3D[(int)JointId.Pelvis];
9          Vector3 pelvisPos = new Vector3((float)pelvisPosition.X, (float)pelvisPosition.Y, (float)
               pelvisPosition.Z);
10         if (pelvisPos.magnitude < minDistanceFromKinect)
11         {
12             closestBody = i;
13             minDistanceFromKinect = pelvisPos.magnitude;
14         }
15     }
16     return closestBody;
17 }
```

Listado 5.4: Algoritmo para extraer el cuerpo más cercano

Este algoritmo se encarga de retornar el cuerpo que se encuentre más cerca del sensor. Dicho cuerpo será el que contenga la información de las articulaciones capturadas y será el cuerpo utilizado para darle al avatar los movimientos necesarios para realizar el ejercicio. Finalmente, el esqueleto obtenido es usado para extraer la información de sus articulaciones,

las cuales son tratadas en una fase posterior del actual módulo.

La información de las articulaciones está contenido en el objeto *Body* perteneciente a la biblioteca de *body tracking*. El total de articulaciones que permite capturar esta biblioteca son las que aparecen en la figura 5.5.

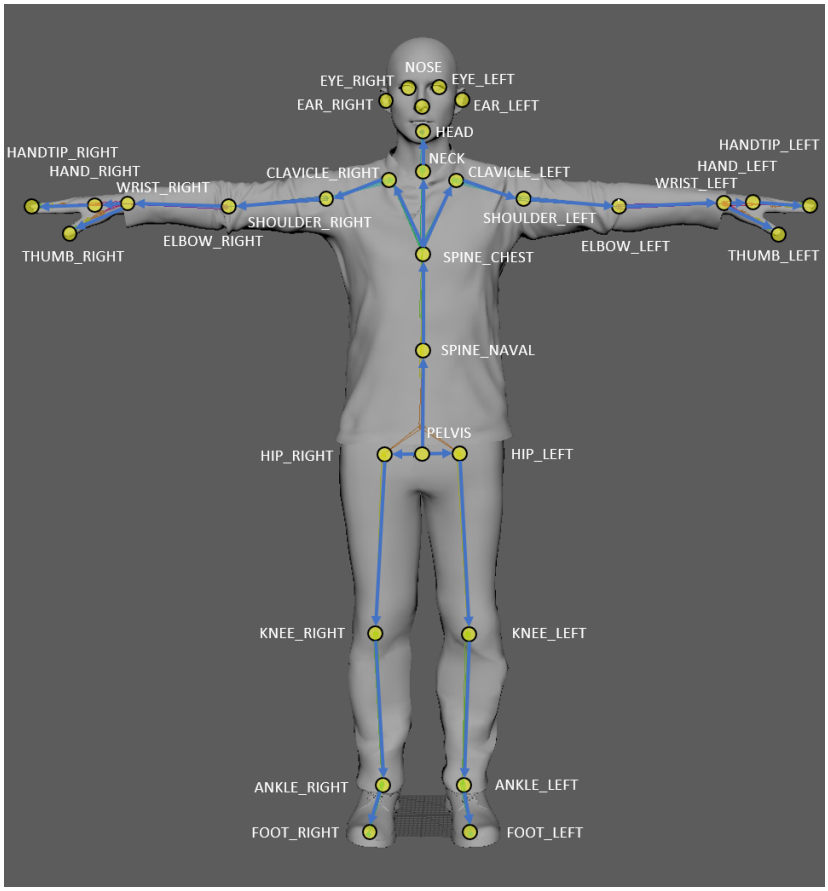

Figura 5.5: Mapa de las articulaciones que soporta Azure Kinect DK[1]

Para dar vida al avatar lo primero que se hizo fue realizar un mapeo entre las articulaciones recibidas por el SDK de *body tracking* y las articulaciones del avatar. Para ello se creó una lista compuesta por dos campos: un campo que mapea la articulación del avatar con el identificador de la articulación de la biblioteca de *body tracking*, y un valor que almacena la posición, rotación y escala de la articulación del modelo del avatar. Este mapeo se puede observar en 5.6.

Una vez realizado el mapeo, el siguiente paso del módulo es animar el modelo del avatar teniendo en cuenta las posiciones y rotaciones de las articulaciones del usuario en frente del sensor de captura de movimiento. Para ello se ha utilizado el algoritmo 5.5.

Esta algoritmo itera sobre cada elemento de la lista mencionada anteriormente para modificar la posición y rotación de las articulaciones del avatar. Esto es posible ya que los elemen-

[1]Fuente: `https://docs.microsoft.com/en-us/azure/Kinect-dk/body-joints`

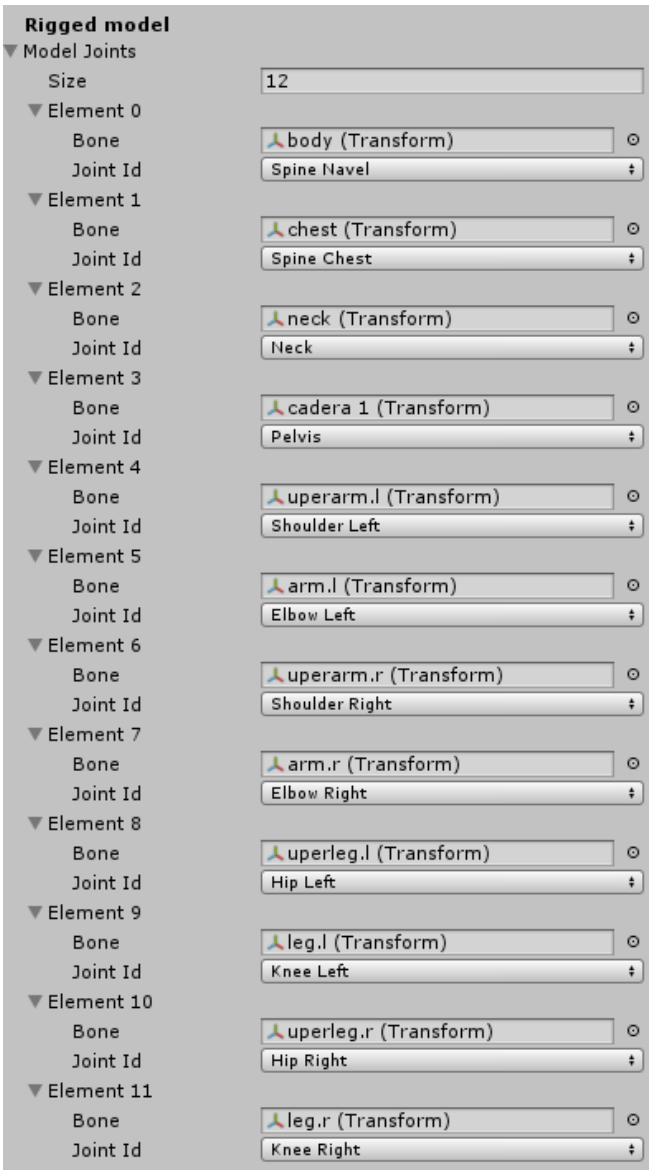

Figura 5.6: Mapeo de articulaciones de Unity

tos de la lista contienen el campo *transform* (datos relativos a posición, rotación y escalado) ligado a la articulación del avatar. Primero, se comprueba si una articulación está implicada en el ejercicio haciendo uso de una lista externa que es poblada en tiempo de ejecución (*isJointUsed*). Esta lista se consigue de un método que será comentado más adelante.

La rotación de las articulaciones necesarias se modifica ajustando el parámetro *rotation* con la información que contiene del objeto *Body*. Como se puede observar en el listado, se crea un cuaternión[2] para calcular la rotación del hueso del modelo, es decir, se toma la orientación de la articulación invertida (esto sirve para rotar el modelo 180º, es decir, posicionarlo de espaldas a nosotros) y se añade la rotación base del hueso del modelo, es decir, la inicial. Sin embargo, durante este proceso de codificación surgió un problema relacionado con los

[2] `https://es.wikipedia.org/wiki/Cuaterni%C3%B3n`

```
void ProcessSkeleton(Body skeleton)
{
    int jointId;
    bool isJointUsed = false;

    foreach (var riggedJoint in jointsRigged)
    {
        isJointUsed = jointsNeeded.Contains(riggedJoint.Value.bone.name);
        if (isJointUsed == true) {
            jointId = (int)riggedJoint.Key;
            ModelJoint modelJoint = riggedJoint.Value;
            Quaternion jointOrient = new Quaternion(skeleton.JointRotations[jointId].X, skeleton.
                JointRotations[jointId].Y, skeleton.JointRotations[jointId].Z, skeleton.
                JointRotations[jointId].W) * GetKinectTPoseOrientationInverse(riggedJoint.Key) *
                modelJoint.baseRotOffset;
            modelJoint.bone.rotation = jointOrient;
        }
    }
}
```

Listado 5.5: Algoritmo para animar las articulaciones del avatar en base a los movimientos de un usuario

ejes de coordenadas de *Azure Kinect Body Tracking SDK*[3], ya que su orientación es distinto al sistema de coordenadas de Unity[4]. Esto se consiguió solucionar adaptando la orientación del eje de coordenadas de cada articulación, en tiempo de ejecución, al sistema de coordenadas de Unity. En el listado 5.6 podemos ver un ejemplo de cómo se realiza esta adaptación

```
    case JointId.FootRight:
        return Quaternion.AngleAxis(180, Vector3.forward) * Quaternion.AngleAxis(-90, Vector3.up);
```

Listado 5.6: Modificación de las coordenadas de una articulación

Finalmente, gracias a este enfoque se ha podido hacer uso de los datos generados por la biblioteca de *body tracking* para replicar los movimientos de un usuario en el modelo del avatar.

5.4 Módulo de Gamificación

Esta es la parte de la arquitectura encargada de unificar la información recibida de los módulos anteriores. Es un punto común donde se transforme la información de la especificación de un *exergame* generado por el módulo de definición en un escenario gamificado, además de añadir a este escenario un modelo de avatar encargado de replicar los movimientos del usuario. Todo ello en conjunto se usa para representar un juego, donde el jugador puede realizar sus ejercicios de rehabilitación. Este módulo tiene dos elementos bien diferenciados:

[3] https://docs.microsoft.com/en-us/azure/Kinect-dk/body-joints

[4] https://docs.unity3d.com/es/2018.4/Manual/Transforms.html

uno obtiene los datos generales del *exergame*, mientras que el otro utiliza los datos del nivel de dificultad seleccionado para crear un juego donde el avatar puede interactuar con los elementos de este. Finalmente, también hay una sección que se comunica con la parte que carga el nivel dificultad, y que es la encargada de cambiar la dificultad en tiempo real durante la ejecución de una partida.

5.4.1 Inicialización del exergame

El primer módulo contenido dentro del módulo de gamificación extrae la información del *exergame* del archivo JSON que contiene la especificación del ejercicio correspondiente. Primero se valida la información contenida y, a continuación, se parsea para obtener un objeto de tipo *Exergame* visto en la sección 5.2.1.

Una vez recuperada la información del *exergame*, el módulo la envía a aquellos componentes del escenario de juego que la necesiten. Uno de los elementos es la cámara de la escena que a ajusta sus parámetros de posición y rotación. También se actualiza la imagen que indica la postura que debe adoptar el paciente para la correcta realización del ejercicio. Posteriormente, el sistema envía los datos que necesita la pantalla de preparación para el ejercicio, en la cual se muestra un mensaje de asistencia para la correcta realización del mismo, una imagen con la postura a adoptar para realizar el ejercicio, y una cuenta atrás para avisar al jugador de que la partida va a comenzar en breves instantes. Esto además sirve para que el usuario tenga el suficiente tiempo para adoptar la postura requerida. Podemos ver todo este proceso en el diagrama de secuencia de la figura 5.7.

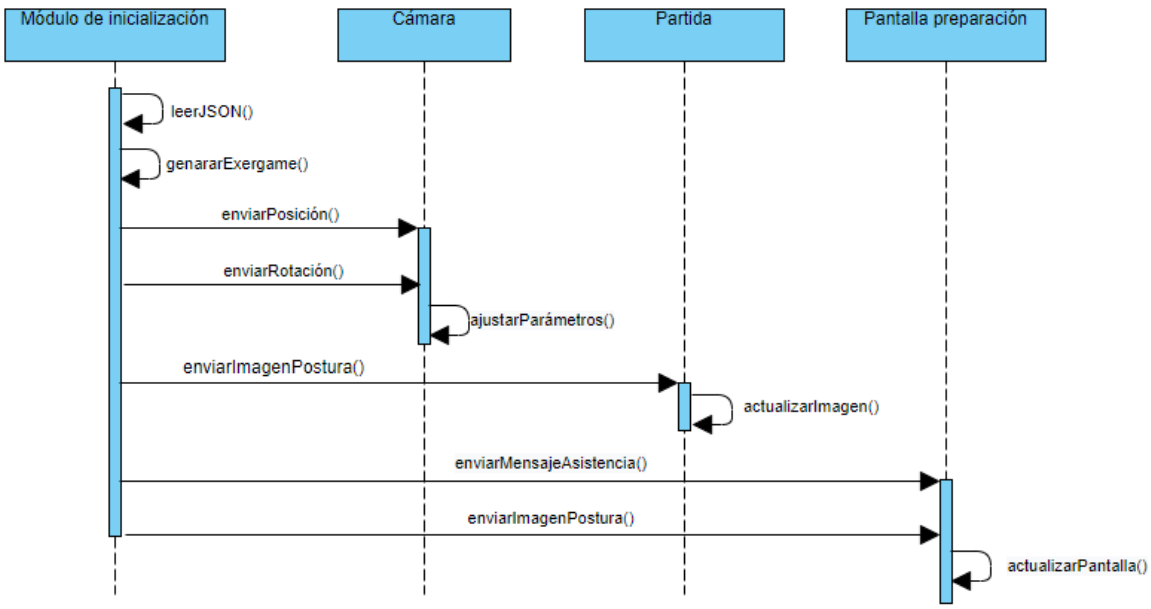

Figura 5.7: Diagrama de secuencia del módulo de inicialización

5.4.2 Generador de los elementos de la partida

A continuación, se inicia el siguiente módulo, que se encarga de obtener los datos relacionados con el nivel de dificultad de un ejercicio concreto. Al igual que ocurría con los datos generales del *exergame*, el módulo tiene que leer el archivo JSON relativo al nivel de dificultad seleccionado del ejercicio en cuestión validando la información y parseandola para guardarla en un objeto de tipo *exergameLvl*. Con los datos ya guardados el modulo se encarga de enviarlos a aquellos componentes que los necesiten para poder comenzar la partida.

Finalmente, antes de comenzar el juego es necesario generar los objetos con los que el paciente va a interactuar. En el caso de que sea un *exergame* «normal» se generan tantas esferas de colisión como sean necesarias para poder definir la trayectoria a seguir por parte del usuario. Si el *exergame* es «especial», se generan dos objetos: el primero, será un objeto interactuable que el usuario podrá recoger y mover. En cuanto al segundo objeto generado, este debe ponerse a la espera de que el primer objeto colisione con él.

5.4.3 Ejecución de partida normal

Cada ejercicio dispone de unas esferas que marcan la trayectoria del movimiento. Para que el paciente sepa que está realizando bien el ejercicio las esferas son de color amarillo cuando están a la espera de ser atravesadas, y cambian de color a verde cuando la articulación implicada en la realización del ejercicio atraviesa la esfera. Para que cambie de color amarillo a verde no basta con tocar una esfera cualquiera, si no que debe ser la esfera que corresponda según el orden determinado. Cuando la esfera es de color verde, su detector de colisiones es desactivado, por lo que al atravesarla no ocurrirá nada hasta que sea reactivado de nuevo.

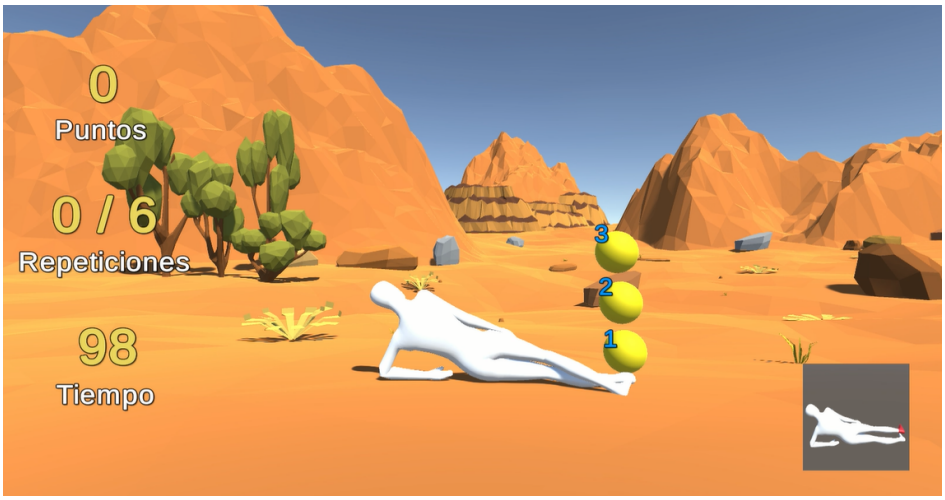

Figura 5.8: Imagen del sistema ejecutando una partida normal

Esta sección del módulo encargada de controlar el color de las esferas durante la ejecución de un juego se ha diseñado usando un enfoque basado en eventos para mantener el sistema desacoplado y ser lo más escalable posible. Para ello se ha usado el patrón de diseño *ob-*

server. De esta forma, cuando cambia el color de una esfera de amarillo a verde, se envía una notificación a los objetos dependientes de este cambio, que en este caso son el sistema de partículas, el marcador de la puntuación total, el marcador de puntuación temporal, y el generador de sonido.

Con el fin de dar *feedback* al paciente, cada vez que se toque una esfera de forma correcta se disparan los siguientes eventos:

- Desde el interior de la esfera atravesada salen unas partículas que simulan una explosión.
- Cambio del color de la esfera atravesada de amarillo a verde.
- Aumento de la puntuación total que lleva el usuario sumando el incremento indicado para el ejercicio.
- Los puntos obtenidos al tocar la esfera se muestran temporalmente encima del marcador de la puntuación.
- Generación de un sonido que verifica que la esfera ha sido atravesada.

Una vez que todas las esferas de la escena hayan sido atravesadas y sean de color verde, se habrá completado una repetición del ejercicio, por lo que el marcador de repeticiones se actualiza añadiendo una nueva repetición a la cantidad previa y las esferas vuelven a ser de color amarillo y su detector de colisiones es reactivado de nuevo, reiniciándose de esta forma la repetición.

Durante toda la ejecución del juego hay una cuenta atrás que indica el tiempo restante para hacer el ejercicio. La partida se da por finalizada cuando se han completado todas las repeticiones requeridas para el ejercicio, o cuando se ha agotado el tiempo. En ambos casos se muestra por pantalla un mensaje confirmando la finalización del ejercicio.

5.4.4 Ejecución de partida especial

En una partida especial, el juego crea dos objetos que van a interactuar entre ellos en la escena. El objetivo del juego es colocar el primer objeto sobre el segundo.

El primero de ellos se trata de un objeto móvil que el jugador debe recoger. Para ello debe pasar la articulación necesaria (normalmente la mano) a través de ese objeto móvil. Este objeto puede adoptar dos estados: «agarrado» y «soltado», por lo tanto, esta sección del modulo se ha desarrollado siguiendo un enfoque basado en estados mediante el patrón de diseño *state*. De esta forma el sistema se vuelve más escalable. Para controlar los dos estados que puede tener el objeto móvil, se ha creado una máquina de estados. En la figura 5.9 se puede comprobar su funcionamiento. Cuando el objeto móvil es atravesado por la articulación implicada, pasa a tener el estado «agarrado» y estará siguiendo a la articulación en todo momento.

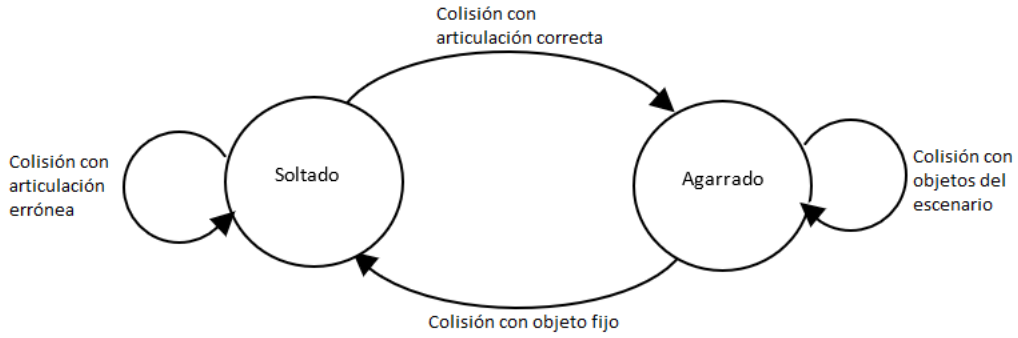

Figura 5.9: Máquina de estados de una partida especial

El segundo objeto, es un objeto que está fijo e integrado en el escenario como si fuera parte de él. Este objeto fijo se encuentra esperando a colisionar con el objeto móvil. Como el objetivo de la partida es dejar el objeto móvil sobre el fijo, el paciente tiene que seguir una trayectoria que va desde donde estaba originalmente el primer objeto hasta la zona donde está el segundo. Una vez que el paciente hace que el objeto agarrado colisione con el objeto fijo, se considera que el usuario lo ha colocado donde se le indicaba, por lo que el objeto móvil pasa a estar en el estado «soltado».

Una vez que el objeto móvil es colocado sobre el fijo se considera que se ha completado una repetición del ejercicio, por lo que se suma la puntuación correspondiente, además de añadir una repetición al marcador. Finalmente, el objeto móvil se despega de la articulación que estuviera siguiendo y reaparece en la posición original para que el usuario pueda agarrarlo de nuevo y realizar una nueva repetición.

De esta forma, se mantiene la idea de los ejercicios normales, puesto que el usuario también sigue trayectorias, que van desde la posición inicial del objeto móvil hasta la posición del objeto fijo, solo que ahora al haber un objeto con el que el paciente puede interactuar, crea una sensación lo bastante diferente como para entretener al usuario de una manera distinta.

5.4.5 Cambio de dificultad

Los niveles de dificultad han sido pensados para poder ser cambiados en cualquier momento durante la ejecución del ejercicio. De esta forma, si el terapeuta observa que el paciente presenta demasiadas complicaciones para realizar un ejercicio, podrá pulsar la flecha hacia abajo del teclado para bajar un nivel de dificultad. Si, por el contrario, considera que puede hacer el ejercicio sin inconvenientes, puede pulsar la flecha hacia arriba para aumentar el

nivel de dificultad.

Para cambiar de dificultad, el sistema vuelve a llamar al módulo encargado de cargar los elementos de la partida. Se eliminan los componentes que se estaban usando en la partida y son reemplazados por los nuevos del nivel de dificultad que corresponda. Los elementos de la partida que son reemplazados para cambiar el nivel de dificultad son el número de objetos en escena, el número de repeticiones necesarias para completar el ejercicio, el tamaño de los objetos, ya que cambiar el tamaño hace que sea más fácil o difícil atravesarlos, la posición de los objetos, los puntos que obtienes al colisionar con ellos, y por último, el tiempo limite para completar el ejercicio.

Capítulo 6

Resultados

A lo largo de este capítulo se van a describir los resultados obtenidos una vez que ha sido completado el desarrollo del proyecto. Para ello, se informará de unas indicaciones previas que deben ser tenidas en cuenta antes de comenzar un *exergame*. Después, se detallará cual ha sido el aspecto final resultante de una partida en ejecución. A continuación, se especificarán las estadísticas obtenidas del código desarrollado para, finalmente, mostrar los datos de una prueba del sistema realizada con un niño.

Todo el **código fuente del proyecto** se encuentra ubicado en el repositorio de GitHub del siguiente enlace: `https://github.com/davidgarsilves/TFG-David-Garcia`. En el propio repositorio, también hay disponible un **vídeo** que ofrece una visión general del sistema desarrollado. El enlace a este se encuentra en el archivo README.md del repositorio.

6.1 Consideraciones previas

Primero, un terapeuta se encarga de analizar el caso de un paciente. Este ha sufrido una o varias lesiones en la columna lumbar, y necesita una serie de ejercicios de rehabilitación para recuperarse. El terapeuta acaba asignándole una rutina de ejercicios a seguir para tratar sus dolencias. En esta rutina se incluirán los *exergames* en forma de archivos ejecutables.

Una vez que el paciente cuenta con sus juegos, debe preparar Azure Kinect para el correcto *tracking* de su cuerpo. Para ello debe colocar el dispositivo a una altura de aproximadamente un metro sobre el suelo, tal y como vemos en la figura 6.1. Tras ello, el paciente debe situarse a una distancia de unos tres metros mirando de frente al dispositivo. Aunque Azure Kinect se puede colocar a diferentes alturas y orientaciones, es recomendable hacerlo de la forma indicada, ya que tras las pruebas realizadas, se ha comprobado que es la mejor posición para capturar los movimientos del cuerpo de forma correcta.

6.2 Aspecto y funcionalidad final del sistema

En primer lugar, es fundamental mostrar de forma aislada los distintos elementos que intervienen durante la realización de un *exergame*, para después, presenciar como quedan integrados de manera conjunta durante la ejecución de este.

El primer componente a exponer es el **avatar** encargado de replicar los movimientos del

Figura 6.1: Fotografía de Azure Kinect listo para capturar el exergame

paciente en la escena. Como podemos observar en la figura 6.2, se trata de un modelo de color blanco que imita la apariencia de un cuerpo humano sin género, con no demasiados detalles puesto que su utilidad es simplemente representar al usuario, es decir, mostrar los movimientos de este de forma fiel a la realidad.

Figura 6.2: Imagen del avatar utilizado para replicar los movimientos

El siguiente elemento a analizar se trata de las **esferas de colisión**. Son los objetos instalados en la escena encargados de indicar la trayectoria que tiene que seguir el paciente para realizar el ejercicio. Cuentan con un número que le permite al paciente saber el orden en el que deben ser atravesadas. Estos objetos se encuentran a la espera de ser atravesados por la

articulación principal del ejercicio que esté realizando el paciente. Cuando esto ocurre, se originan unas partículas de explosión, de las que se va a hablar a continuación, y cambia el color de la esfera. Los colores que puede adoptar una esfera de colisión son el amarillo, que indica que la esfera está a la espera de ser atravesada por la articulación, y el verde, que indica que esto ya ha ocurrido. Podemos ver como es la esfera en la figura 6.3 con los dos estados que puede adoptar. El tamaño de estas esferas será variable, dependiendo de la dificultad del juego. Cuanto mas fácil sea este, el tamaño de las esferas será mayor, para de esta forma ayudar al paciente a que pueda interactuar con ellas.

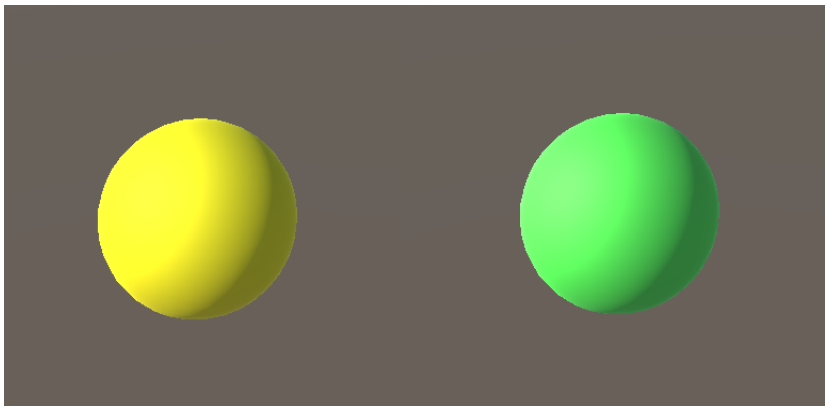

Figura 6.3: Esferas de colisión en sus dos estados. Amarillo indica un estado de espera, mientras que el color verde indica un estado de finalización

En tercer lugar, la siguiente parte del sistema existente son las **explosiones** de las esferas. Son originadas cuando la articulación correspondiente al ejercicio atraviesa una esfera de forma correcta. Se trata de un sistema de partículas, en el que cada una de estas aparecen en el interior de la esfera y se expanden hacia el exterior reduciendo su tamaño poco a poco hasta desaparecer. El aspecto individual de cada partícula es un cuadrado amarillo con lo bordes naranjas. En la figura 6.4 se ha separado el sistema de partículas de su correspondiente esfera para poder apreciar sus detalles.

Por último, el componente a tratar es el caso de los **escenarios** donde el paciente realiza los *exergames*. Los escenarios elegidos para el proyecto han sido una aldea (figura 6.5), un apartamento (figura 6.6) y un desierto (figura 6.7), tal y como podemos ver en las figuras. Estos escenarios no tienen ninguna función más allá de mejorar la apariencia estética del sistema. Se ha utilizado el escenario del apartamento para realizar ejercicios que tan solo usen los brazos, donde la cámara estará más cercana al avatar, y los escenarios de la aldea y el desierto, al ser mas abiertos, se han empleado para que el paciente realice ejercicios con las piernas.

Una vez visto cada uno de los elementos de forma individual, a continuación, se va a exponer como funciona el sistema de forma completa. Al iniciar la ejecución de un *exergame* se muestra durante unos segundos una pantalla estática (figura 6.8) que tiene la finalidad

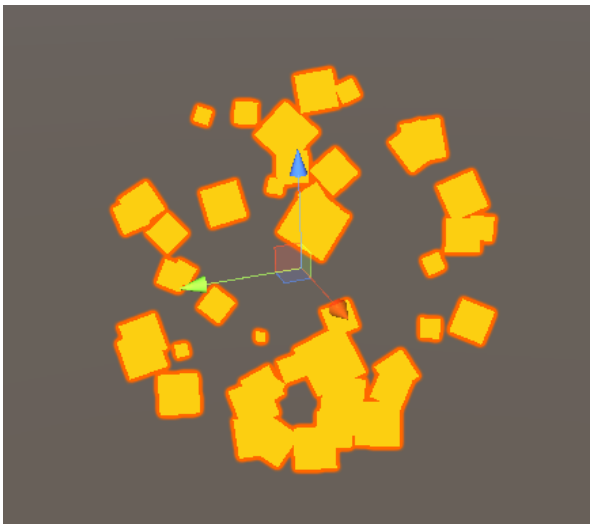

Figura 6.4: Sistema de partículas para simular una explosión

Figura 6.5: Imagen de la aldea usada como escenario

de dar las indicaciones a seguir para la correcta realización del ejercicio. Encontramos un mensaje encargado de explicar los objetivos a seguir para poder completar el *exergame*, y una imagen donde se muestra la postura que debe adaptar el usuario para llevar a cabo el ejercicio de rehabilitación. Además, en la parte inferior aparece una cuenta atrás para indicar cuando da comienzo el juego. De esta forma, el paciente tiene el suficiente tiempo para adoptar la postura necesaria, y de esta forma no desperdiciar el tiempo que el ejercicio tenía asignado.

Una vez que desaparece la pantalla de inicio, da comienzo el juego. Durante la partida se observa como los elementos mencionados anteriormente están integrados en la escena. En la

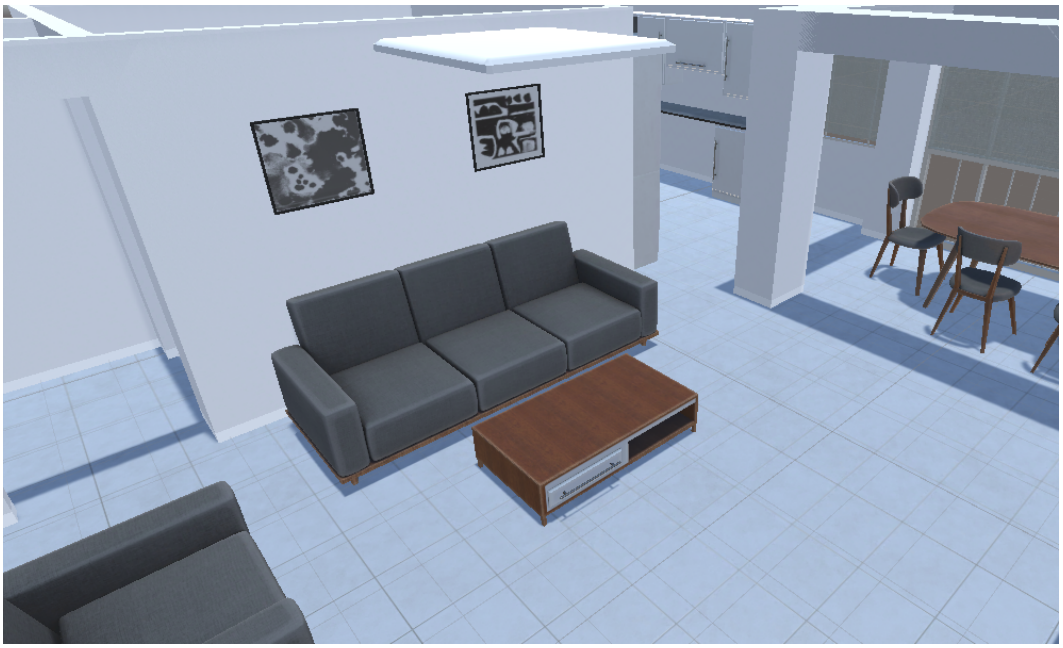

Figura 6.6: Figura del apartamento utilizado como escenario

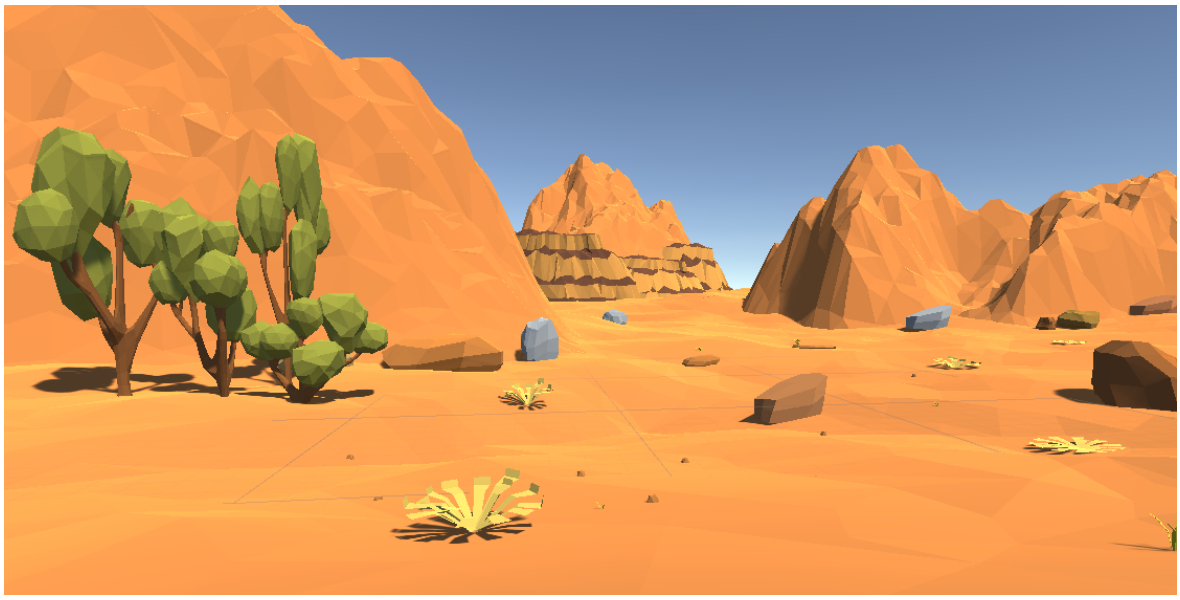

Figura 6.7: Imagen del desierto empleado como escenario

escena se carga por defecto el nivel intermedio de dificultad. El jugador debe atravesar las esferas de colisión con la articulación marcada en rojo en la imagen que indica la postura a adoptar. Estas esferas marcan la trayectoria del movimiento a seguir y el paciente debe atravesar en un orden determinado por los números que aparecen cerca de cada esfera. Podemos ver un ejemplo de ejecución de un *exergame* en la figura 6.9, donde se observa el instante en el que el jugador atraviesa la esfera correctamente. Las esferas originalmente son de color amarillo, y cambian a verde cuando la articulación cruza la esfera correcta, tal y como se

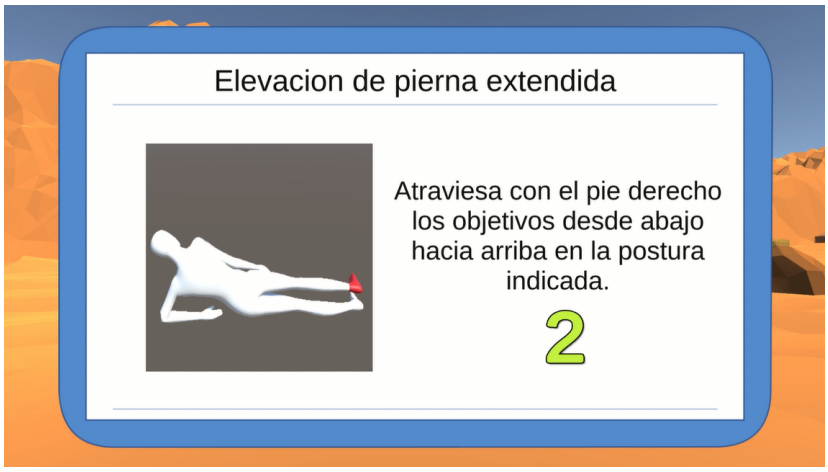

Figura 6.8: Pantalla de comienzo del exergame

había dicho anteriormente. A la vez que ocurre este cambio de color, también se producirá un sonido que indica que la esfera ha sido atravesada, y el sistema de partículas que simula una explosión será activado una vez desde la posición de la esfera, con tal de dar *feedback* al paciente. Además de todo eso, también se actualizará la puntuación de la partida, mostrando temporalmente encima del marcador cuanto ha aumentado esta puntuación.

Figura 6.9: Imagen de un *exergame* en ejecución

Una vez que todas las esferas de la escena hayan sido atravesadas y sean de color verde, se habrá completado una repetición del ejercicio, por lo que el marcador de repeticiones se actualizará añadiendo una nueva repetición a la cantidad previa y las esferas volverán a ser de color amarillo, comenzando una nueva repetición.

Durante toda la ejecución del juego hay una cuenta atrás que indica el tiempo restante para realizar el ejercicio. La partida se da por finalizada cuando se hayan completado todas las repeticiones requeridas para el ejercicio, o cuando se haya agotado el tiempo. En ambos casos se muestra por pantalla un mensaje confirmando la finalización del ejercicio.

Si durante la ejecución del ejercicio, el terapeuta se percata de que al paciente le cuesta demasiado realizar sus tareas de rehabilitación, o si por el contrario se da cuenta de que le están resultando demasiado fáciles de llevar a cabo, el terapeuta deberá modificar la dificultad del ejercicio que esté realizando. Para ello podrá pulsar las flechas hacia abajo o hacia arriba del teclado para subir o bajar el nivel de dificultad, respectivamente. Cuando se cambia el nivel de dificultad, se reinicia los contadores de repeticiones y puntuación a cero, y se restablece el tiempo restante para realizar el ejercicio. También se recargan los elementos de la partida con los que el usuario interactúa, y se eliminan los antiguos. De esta forma, el número de esferas de colisión en la escena aumenta o disminuye acorde a si aumenta o disminuye la dificultad, cambiando el tamaño de estas esferas para hacer más fácil o difícil el hecho de poder atravesarlas, actualizando también la posición de estas, y aumentando o reduciendo la trayectoria a realizar según el nivel de dificultad. Finalmente, también se actualiza la cantidad de puntos obtenidos al tocar una esfera y el limite de tiempo con el que completar el ejercicio. En la figura 6.10 se puede observar las diferencias entre los niveles de dificultad de un mismo juego, mostrando como la cantidad y el tamaño de las esferas ha cambiado, además de modificar ligeramente la trayectoria, y la puntuación obtenida al atravesar una esfera.

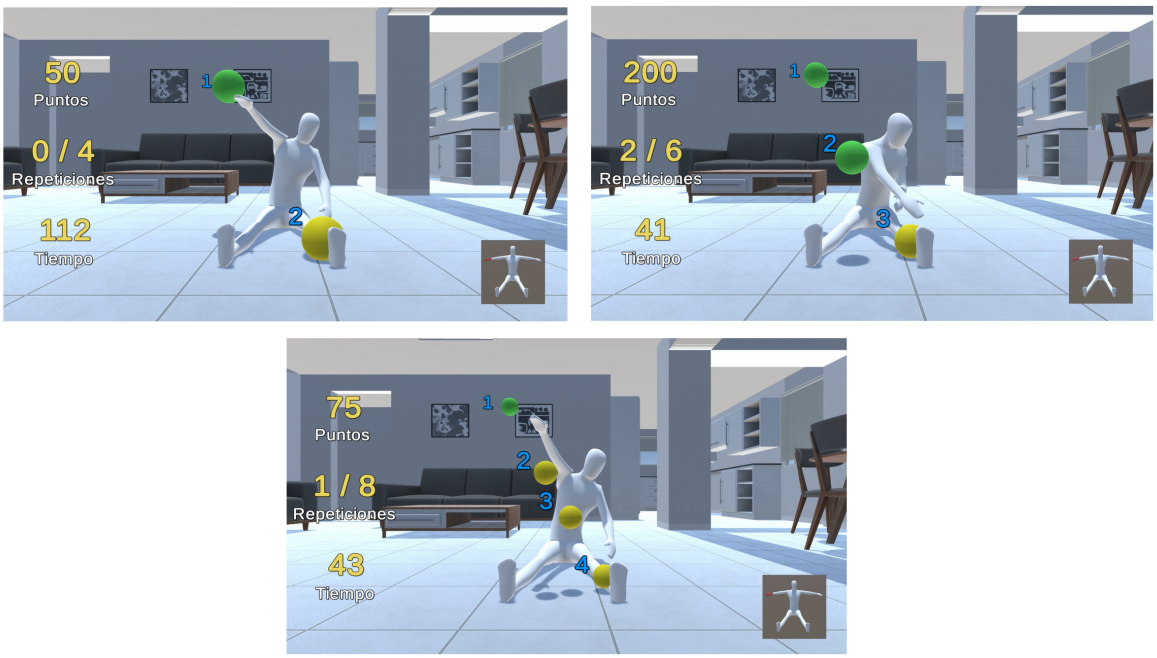

Figura 6.10: Comparación de los niveles de dificultad para un mismo exergame

Para concluir, en este proyecto se ha desarrollado un total de cuatro *exergames*, cada uno con sus respectivos niveles de dificultad. El primero de ellos se denomina *Elevación de pierna extendida*. El paciente debe colocarse tumbado lateralmente y elevar la pierna derecha unos 30 centímetros, atravesando con el pie las esferas de colisión y volviendo a la posición original. El segundo es conocido como *Elevación de brazo*. Para realizarlo correctamente, el paciente debe colocarse lateralmente con la cabeza orientada hacia la izquierda, apoyando el peso de su cuerpo en cuatro puntos que serán las manos y las rodillas. A continuación, procederá a elevar el brazo derecho atravesando las esferas con la mano. El tercer *exergame* son *Abdominales superiores de frente*. El paciente se debe colocar tumbado boca arriba y atravesará con la cabeza las esferas necesarias, mediante la elevación de su tronco. El ultimo ejercicio es denominado *Movimiento de brazo alterno*. El usuario se tiene que sentar sobre el suelo estirando las piernas. Con la mano derecha debe tocarse el pie izquierdo, realizando una trayectoria que le permitirá extender el brazo, y por lo tanto estirar la espalda. En la figura 6.11 vemos una captura de los cuatro *exergames*.

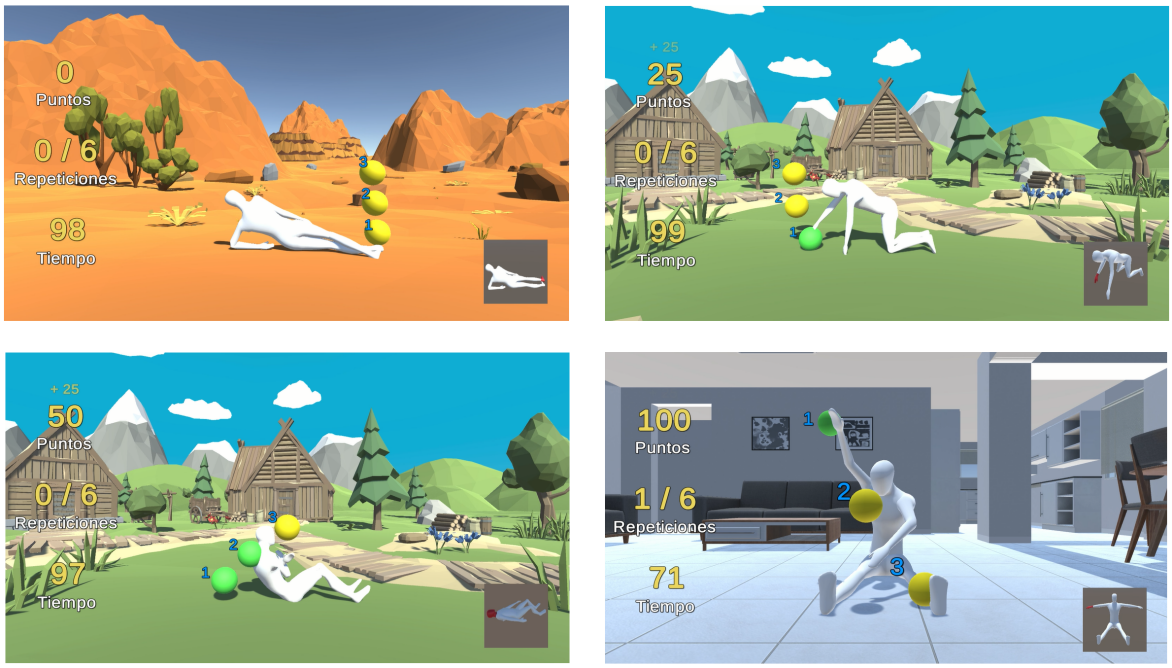

Figura 6.11: Imagen con los cuatro exergames desarrollados

6.3 Estadísticas del código desarrollado

A continuación, se van a analizar las estadísticas obtenidas del repositorio utilizado para el desarrollo del proyecto. En primer lugar, se exponen los datos relacionados con las líneas de código, reflejando también el número de archivos, espacios en blanco, y comentarios. Podemos ver todos estos datos en la tabla 6.1, donde los encontramos agrupados, por un lado en lenguaje C#, y por otro, en formato JSON. Para obtener todos estos datos se ha empleado

la aplicación *cloc*[1]. Sin embargo, se han omitido los archivos ajenos al código desarrollado, ya que pertenecen a herramientas externas y acabarían reflejando unas estadísticas sobre el código desarrollado que no serían reales. Para prescindir del recuento de esos archivos, *cloc* se ha aplicado sobre las carpetas *Assets/Scripts* y *Assets/Ejercicios*.

Lenguaje	Archivos	Espacios en blanco	Comentarios	Código
C#	16	179	52	1008
JSON	18	0	0	748
Total:	**34**	**179**	**52**	**1756**

Cuadro 6.1: Análisis del código desarrollado en el proyecto

En segundo lugar, se ha analizado la frecuencia con la que el repositorio GitHub que contiene el sistema ha sido actualizado. En la figura 6.12 se observa como la actividad está centrada en los meses de abril, mayo y junio, ya que, tras la inicialización de la aplicación, la frecuencia de actualización del repositorio decae, para posteriormente volver a subir.

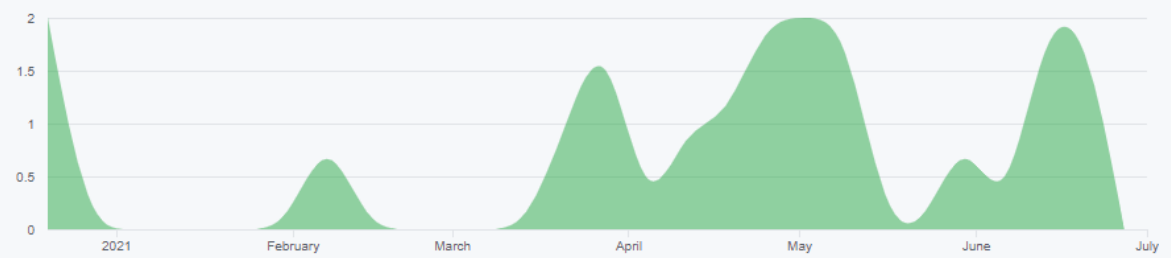

Figura 6.12: Actividad en el repositorio GitHub

6.4 Costes del desarrollo

El desarrollo del presente proyecto abarcó desde el día 16 de octubre de 2020 hasta el 25 de junio de 2021. Esto implica un tiempo de desarrollo de 35 semanas, con una dedicación de 4 horas diarias durante 5 días a la semana, lo que significa un total de 700 horas en el desarrollo del sistema. Sin embargo, no se han contabilizado las horas necesarias para elaborar la documentación del proyecto.

A continuación, se va a estimar un sueldo de 25€/hora, considerando que son los costes para una empresa que contrata un programador junior. Teniendo en cuenta que para este proyecto solo ha sido necesario un desarrollador, esto significa un coste de 17.000 €.

Además de los costes anteriores hay que añadir el hardware necesario para llevar a cabo este proyecto. Esto incluye un PC con un valor de 710 € y un dispositivo Azure Kinect de 336 €.

[1] `https://github.com/AlDanial/cloc`

A modo de resumen, se puede ver en el cuadro 6.2 un desglose de los gastos totales para el desarrollo del proyecto

Recurso	Cantidad	Coste
Sueldo del programador	1	17.000 €
PC	1	710 €
Azure Kinect	1	336 €
Total:		**18.546 €**

Cuadro 6.2: Costes totales del proyecto

6.5 Pruebas preliminares

Para terminar con este capítulo, se ha probado el sistema con un niño para verificar el correcto funcionamiento de este. El objetivo de realizar estas pruebas es extraer conclusiones relacionadas con cuestiones de usabilidad y toma de contacto del sistema por parte de alguien ajeno al mismo. Debido a que ha sido imposible encontrar un niño con dolencias de columna lumbar, se ha utilizado un niño sano para realizar las pruebas. El niño es familiar del desarrollador del sistema, y ha realizado las pruebas preliminares bajo la supervisión de este.

El participante realizó los *exergames Elevación de pierna extendida* y *Elevación de brazo*, cada uno con sus tres niveles de dificultad, y repitiendo cada nivel de dificultad tres veces. De estas pruebas se extrajeron los datos disponibles en el cuadro 6.3, donde las columnas indican el porcentaje de ejercicio completado y el tiempo sobrante. Las filas indican si es el ejercicio de brazo o de pierna y la tanda de repeticiones realizada. El sujeto tan solo no fue capaz de completar el mayor nivel de dificultad del primer *exergame* la primera vez que lo realizó. Viendo los tiempos, podemos llegar a la conclusión que una vez que el paciente se acostumbra al sistema, le resulta mucho más fácil realizar los ejercicios y por lo tanto necesita menos tiempo.

	Fácil		**Intermedio**		**Dificil**	
	%	**Tiempo**	**%**	**Tiempo**	**%**	**Tiempo**
P1	100 %	60	100 %	41	87.5 %	0
P2	100 %	73	100 %	49	100 %	4
P3	100 %	84	100 %	56	100 %	6
B1	100 %	72	100 %	46	100 %	32
B2	100 %	85	100 %	61	100 %	38
B3	100 %	89	100 %	63	100 %	41

Cuadro 6.3: Desempeño de los participantes en los exergames realizados

Capítulo 7

Conclusiones

ESTE capítulo final está dividido en cuatro secciones. La primera de ellas expone una conclusión general del sistema desarrollado. La segunda sección analiza si los objetivos planteados en el capítulo 2 han sido logrados. La tercera propone unas posibles mejoras que se pueden añadir en el proyecto. Finalmente, la última sección muestra las competencias adquiridas durante el desarrollo del mismo.

7.1 Conclusión general

Este proyecto ha tenido desde el principio dos claros objetivos que ha intentado satisfacer. Por un lado, muchos sistemas de rehabilitación tradicionales tienen el inconveniente de que el paciente tiene que desplazarse hasta al centro de rehabilitación indicado para hacer los ejercicios de su rutina necesarios, puesto que, sin el apoyo de un especialista, el afectado podría no saber cómo realizar sus ejercicios. El sistema desarrollado abre la puerta a realizar una rehabilitación remota, puesto que con el apoyo del presente proyecto, el paciente puede hacer sus actividades de rehabilitación de la columna lumbar desde la comodidad de sus hogares, puesto que recibirán por pantalla la información necesaria para tratar sus lesiones.

La otra cuestión que se pretendía resolver es la relacionada con la motivación de los pacientes. Realizar los ejercicios de rehabilitación asignados es una tarea pesada, lo que podría suponer que muchos niños se aburran y, consecuentemente, abandonen sus rutinas de rehabilitación. El sistema contiene distintas técnicas de gamificación que tratan de incentivar a un niño a que realice sus ejercicios. De esta forma, debería aumentar la motivación del paciente por seguir con sus rutinas.

Para llevar a cabo este proyecto, se ha desarrollado una aplicación, que, mediante el uso de Azure Kinect, implementa un sistema de captura de movimiento que permite representar al usuario en un escenario donde se le indica cómo realizar un ejercicio de rehabilitación a través de la traza en pantalla de la trayectoria a seguir por parte de la articulación afectada.

Además, esta aplicación incluye una especificación que permite a un terapeuta diseñar, con la ayuda de un profesional en TIC, ejercicios de rehabilitación gamificados basados en los movimientos que el paciente necesita realizar para recuperarse de sus lesiones en la columna lumbar.

7.2 Objetivos alcanzados

Durante la realización de este TFG se han alcanzado una serie de objetivos que habían sido planteados en el capítulo 2.

- **Elicitación de requisitos y elaboración de pruebas conceptuales para desarrollar un sistema enfocado a la rehabilitación de niños con dolencia lumbar**: Para desarrollar el proyecto final se han recogido todos los requisitos que debía cumplir el sistema, y se realizaron una serie de pruebas con proyectos de *body tracking* disponibles en la plataforma web, para extraer las conclusiones necesarias con las que elaborar el sistema final.
- **Diseño y desarrollo de un módulo de gamificación**: Otro objetivo logrado es el de desarrollar un módulo de gamificación, cuya utilidad es motivar a los pacientes a realizar sus ejercicios de rehabilitación. Para ello, las actividades han sido diseñadas de forma que el usuario debe realizar los movimientos del ejercicio necesarios siguiendo una trayectoria marcada por esferas, logrando conseguir puntos y recibiendo *feedback* visual cuando interactúa con estas de forma correcta.
- **Diseño y desarrollo de un módulo de definición de métricas**: Para satisfacer esto, se ha desarrollado el módulo de definición de *exergames*. Un terapeuta debe considerar una serie de métricas con las que cuantificar los parámetros necesarios para definir el *exergame*. Con estas métricas el terapeuta podrá evaluar la actuación del paciente cuando realice sus ejercicios.
- **Diseño y desarrollo de una arquitectura escalable**: El último objetivo logrado es el de desarrollar una arquitectura escalable, con el propósito de añadir nuevas mejoras en el futuro. El sistema permite añadir nuevos ejercicios, gracias al módulo de definición de *exergames*, mediante el uso de la estructura en formato JSON definida para ello. Además, gracias al uso de los patrones de diseño *observer* y *state* en el módulo de gamificación, se pueden añadir otras funcionalidades al sistema, ya que el uso de esos patrones en la arquitectura utilizada permite la escalabilidad.

7.3 Trabajo futuro

A continuación, se consideran una serie de aspectos que son mejorables para conseguir que el proyecto desarrollado sea mucho más completo. Las tareas a abordar en el futuro son:

- **Aumentar el número de exergames**: Originalmente, el sistema tenía diseñados más *exergames* con los que rehabilitar las lesiones, pero finalmente tan solo han sido desarrollados cuatro de ellos, los cuales se pueden ver en la parte final de la sección 6.2. Añadiendo nuevos *exergames*, aumentará la variedad de ejercicios que puede hacer un paciente, para de esta forma, conseguir nuevas rutinas de rehabilitación y aumentar el

numero de lesiones que un terapeuta puede tratar. Además, sería conveniente que estos nuevos *exergames* generados utilizaran como articulación principal alguna distinta a la mano o el pie, para que la diversidad de ejercicios sea todavia mayor. Para realizar esta tarea habría que crear los juegos siguiendo la misma estructura JSON comentada en este proyecto.

- **Añadir exergames especiales**: A pesar de haber desarrollado un módulo que permite definir exergames del tipo «especial», no se ha logrado integrar estos exergames de forma que tuviera algún tipo de relación con el sistema completo, ya que surgieron una serie de inconvenientes. Por un lado, las trayectorias definidas al coger un objeto y soltarlo encima de otro no se lograron diseñar lo bastante extensas como para considerar que ese movimiento permitiera rehabilitar alguna lesión. Por otro lado, el contexto de los escenarios planteados y los objetos móviles utilizados no eran lo bastante atractivos para un niño, sino que estaban pensados para un publico mas adulto, por lo que de esta forma no habría sido correcta su integración en un sistema que va dirigido a un público menos maduro.
- **Aumentar el número de articulaciones implicadas en un exergame**: En el sistema desarrollado, durante la ejecución de un *exergame* tan solo podemos seleccionar una articulación con la que se van a atravesar las esferas que marcan la trayectoria. De esta forma, tan solo es posible generar ejercicios que usen un solo miembro del cuerpo. Aumentar el número de articulaciones disponibles durante un *exergame* dará pie al desarrollo de más variedad de ejercicios, en los que por ejemplo el paciente tenga que realizar movimientos utilizando los dos brazos a la vez. Esta tarea se podría llevar a cabo cambiando el modo de generar las esferas de colisión, puesto que actualmente cada esfera generada es una copia de la anterior, por lo tanto todas estarán a la espera de ser atravesadas por la misma articulación.
- **Registrar posiciones de una articulación**: Actualmente, el sistema completa una repetición siempre que las esferas sean atravesadas en el orden indicado mediante los números que aparecen al lado de cada una de ellas. Esto permite que, al mover una articulación, el paciente pueda seguir cualquier trayectoria tocando las esferas en el orden correcto. Seria ideal añadir un sistema que registre todos los puntos del espacio en los que la articulación ha estado posicionada, para de esta forma, poder dibujar una gráfica con los datos del movimiento realizado por el paciente. Así, el terapeuta pueda comprobar que los ejercicios de rehabilitación están siendo realizados de forma correcta.
- **Analizar automáticamente la movilidad del paciente durante un ejercicio**: Para cambiar el nivel de dificultad en el sistema actual, el terapeuta debe analizar el esfuerzo que necesita el paciente para realizar un ejercicio. Sería conveniente añadir un módulo que estudie al paciente durante la ejecución del ejercicio de rehabilitación para

adaptar el nivel de dificultad en base a su movilidad. Este módulo se implementaría con técnicas de inteligencia artificial.

7.4 Justificación de competencias adquiridas

El desarrollo de este proyecto ha permitido adquirir nuevas habilidades relacionadas con el mundo de la informática, además de reforzar todas las competencias que ya se habían estudiado durante el grado y en la intensificación de *Ingeniería del Sotfware*.

- **[IS1] Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la Ingeniería del Software.** Para llevar a cabo este proyecto se ha aplicado la metodología de desarrollo iterativo e incremental, lo que ha permitido aplicar teorías, principios, métodos y prácticas de la Ingeniería del Software.
- **[IS3] Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.** Para el desarrollo de este TFG se han utilizado diferentes tecnologías, entre las que destaca la biblioteca de *body tracking* de Azure Kinect DK que han sido integradas en el sistema para desarrollar el proyecto en Unity.
- **[IS4] Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.** Durante la elaboración de este trabajo han surgido problemas y retos de diferente naturaleza que se han abordado siguiendo un enfoque *top-down*: análisis del problema y elección de una solución de entre las diferentes valoradas, considerando las ventajas y desventajas desde el punto de vista técnico., Una vez analizados cada uno de estos problemas, sus soluciones fueron diseñadas, desarrolladas e implementadas en el sistema. Después de verificar su correcto funcionamiento, finalmente fueron registradas en el documento actual.

ANEXOS

Anexo A

Contenido del repositorio

En este anexo se describe el contenido de las carpetas más relevantes del repositorio encargado de alojar el proyecto, al que podemos acceder a través del siguiente enlace:

`https://github.com/davidgarsilves/TFG-David-Garcia`

- **Assets**: Contiene los recursos del proyecto.
  - **Ejercicios**: Contiene el código C# del módulo encargado de la gamificación, y los archivos JSON que definen los ejercicios.
  - **Materials**: Contiene los materiales utilizados sobre los objetos.
  - **Plugins**: Contiene algunas de las librerías que necesita la aplicación.
  - **Prefab**: Contiene los objetos que representan los escenarios y el avatar.
  - **Resources**: Contiene las imágenes utilizadas para mostrar la postura a adoptar en el ejercicio.
  - **Scenes**: Contiene las escenas del proyecto.
  - **Scripts** Contiene el código C# del módulo encargado del *body tracking*.
  - **Sonidos**: Contiene los archivos de sonido utilizados por la aplicación.
- **ProjectSettings**: Contiene la configuración de Unity.

Anexo B

Manual de usuario

En primer lugar, el usuario debe conectar el dispositivo de Azure Kinect a su ordenador mediante los dos cables disponibles. Hay que tener en cuenta que el cable de datos necesita un puerto USB 3.0, mientras que al de alimentación le sirve cualquier USB.

A continuación, debe colocar el dispositivo a una altura aproximada de un metro sobre el suelo, apuntando con la cámara a donde estará la articulación que va a capturarse. El usuario tiene que colocarse a tres metros del dispositivo de Azure Kinect mirando de frente o de lado en función del ejercicio

Los *exergames* se encuentran guardados en carpetas, una para cada *exergame*, donde en su interior se encontrarán los archivos necesarios para ejecutar el juego. El archivo con el que da comienzo la partida es el ejecutable de la carpeta raíz llamado igual que el ejercicio a realizar.

Una vez ejecutado el juego, se muestra una pantalla que da unas indicaciones sobre cómo realizar el *exergame*. A continuación, comienza el juego, donde el usuario tiene que atravesar las esferas con la articulación indicada. La partida termina cuando se completen todas las repeticiones o se acabe el tiempo.

Para cambiar la dificultad de una partida, hay que pulsar las flechas hacia arriba o hacia abajo del teclado para subir o bajar la dificultad, respectivamente.

Referencias

[65y20] Telerehabilitación en personas mayores: ¿en qué consiste y cuáles son sus ventajas?, 2020. (Consultado el 15-06-2021). url: https://www.65ymas.com/salud/preguntas/telerehabilitacion-personas-mayores-en-consiste-ventajas_11641_102.html.

[A+17] R Agredo et al. Comparación de dos Sistemas de Captura de Movimiento por medio de las Trayectorias Articulares de Marcha. *Revista mexicana de ingeniería biomédica*, 37(2):149–160, 2017.

[abc15] El 80 % de la población va a sufrir dolor de espalda en algún momento de su vida, 2015. (Consultado el 5-07-2021). url: https://tinyurl.com/jes5b3s5.

[abi14] Tipos de rehabilitación, 2014. (Consultado el 5-07-2021). url: https://abilita.com.mx/tipos-de-rehabilitacion/.

[Abi21] Centro de rehabilitación, 2021. (Consultado el 10-06-2021). url: https://abilita.com.mx/centro-de-rehabilitacion/.

[Abt87] Clark C Abt. *Serious games*. University press of America, 1987.

[AGSS20] Marcos Alonso-García y Antonio Sarría-Santamera. The economic and social burden of low back pain in Spain: a national assessment of the economic and social impact of low back pain in Spain. *spine*, 45(16):E1026–E1032, 2020.

[BDMM14] Stefano Bragaglia, Stefano Di Monte, y Paola Mello. A distributed system using MS kinect and event calculus for adaptive physiotherapist rehabilitation. En *2014 Eighth International Conference on Complex, Intelligent and Software Intensive Systems*, páginas 531–538. IEEE, 2014.

[BLLU14] John Bolton, Mike Lambert, Denis Lirette, y Ben Unsworth. PaperDude: a virtual reality cycling exergame. En *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, páginas 475–478. 2014.

[BRP+19] Meryene BARRIOS, Liliana RODRIGUEZ, Claudia PACHON, Boris MEDINA, y Javier E SIERRA. Telerehabilitación funcional en entornos virtuales interactivos como propuesta de rehabilitación en pacientes con discapacidad. *Revista ESPACIOS*, 40(25), 2019.

[Cho19] Yu-kai Chou. *Actionable gamification: Beyond points, badges, and leaderboards*. Packt Publishing Ltd, 2019.

[CSC20] Enrique Cruz, Andrea Simian, y Andrés Chahin. Dolor lumbar en niños. *Revista Médica Clínica Las Condes*, 31(5-6):404–416, 2020.

[dan19] ¿Cuánto tiempo me llevará recuperarme de una lesión?, 2019. (Consultado el 5-07-2021). url: https://www.clinicadanireig.com/cuanto-tiempo-me-llevara-recuperarme-de-una-lesion/.

[Dis08] Diseño de sistemas, 2008. (Consultado el 20-06-2021). url: https://www.slideshare.net/punk.kekito/diseo-de-sistemas-presentation-667665.

[EHM+18] Lesly Lisbeth Gómez Echeverry, Anyi Melissa Jaramillo Henao, Madeleine Angelica Ruiz Molina, Sandra Milena Velásquez Restrepo, Camilo Andrés Páramo Velásquez, y Gabriel Jaime Silva Bolívar. Sistemas de captura y análisis de movimiento cinemático humano: una revisión sistemática. *Prospectiva*, 16(2):24–34, 2018.

[EML21] Enciclopedia médica: Lordosis, 2021. (Consultado el 4-07-2021). url: https://healthtools.aarp.org/es/salud/lordosis.

[Eng20] Unreal Engine. Unreal Engine 5 Revealed! | Next-Gen Real-Time Demo Running on PlayStation 5, 2020. url: https://www.youtube.com/watch?v=qC5KtatMcUw&ab_channel=UnrealEngine.

[FCFC14] Ángeles Forner Cordero y Isabel Forner Cordero. Concepto de Medicina Física y Rehabilitación. 2014.

[GL20] Mario García Lázaro. Games development engines share evolution, 2020. (Consultado el 26-06-2021). url: https://tinyurl.com/tksjgy5.

[GM11] Yue Gao y Regan L Mandryk. GrabApple: the design of a casual exergame. En *International Conference on Entertainment Computing*, páginas 35–46. Springer, 2011.

[GS18] Nicolas González Soto. Motores de Videojuegos, 2018. (Consultado el 23-06-2021). url: http://jeuazarru.com/wp-content/uploads/2018/11/Motores-de-Videojuegos.pdf.

[HRLRFB+19] Erwin Hernando Hernández-Rincón, Catalina Leaño-Ramírez, Yuli Viviana Fuentes-Barreiro, María Fernanda Barrera-Orduz, y Jhosep Andres Blanco-Mejia. Telemedicina en procesos de rehabilitación en pacientes con paraplejia bajo el contexto de Atención Primaria de Salud. *Revista Cubana de Información en Ciencias de la Salud*, 30, 09 2019. url: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2307-21132019000300006&nrm=iso.

[LLK+19] Edward Liu, Ignacio Llamas, Patrick Kelly, et al. Cinematic rendering in UE4 with real-time ray tracing and denoising. En *Ray Tracing Gems*, páginas 289–319. Springer, 2019.

[LV14] B Loreto Vergara. Desarrollo de la Medicina Física y Rehabilitación como especialidad médica. 2014.

[MDNG12] Stéphanie Mader, Jérôme Dupire, Stéphane Natkin, y Emmanuel Guardiola. Designing therapeutic games for seniors: case study of "le village aux oiseaux"(birds village). *Modelling, Measurement and Control*, 73:n.c., 01 2012.

[MHK14] Hossein Mousavi Hondori y Maryam Khademi. A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. *Journal of medical engineering*, 2014.

[MM98] M Martínez Morillo. Manual de medicina física. 1998.

[MMK+16] Alejandro Mendelevich, Mariela Módica, Marcia Kramer, Sabrina Gallo, y Marco Ostolaza. Efectividad de la Nintendo Wii Fit® en la rehabilitación del equilibrio en sujetos con amputación unilateral de miembro inferior. *Boletín del Departamento de Docencia e Investigación IREP*, 15(1), 2016.

[MSK15] La magia sanitaria de Kinect, 2015. (Consultado el 30-06-2021). url: https://news.microsoft.com/es-es/2015/06/17/magia-sanitaria-kinect/.

[reh18] ¿Por qué es tan importante la rehabilitación y más en la discapacidad?, 2018. (Consultado el 4-07-2021). url: https://www.infosalus.com/asistencia/noticia-tan-importante-rehabilitacion-mas-discapacidad-20180827081432.html.

[RLS09] Daniel Roetenberg, Henk Luinge, y Per Slycke. Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*, 1, 2009.

[SMI14] I Setuain, N Millor, y M Izquierdo. Diferencias en el rendimiento y biomecánica del salto en jugadoras de balonmano femenino de élite con o sin

reconstrucción previa del LCA. Un estudio basado en sensores inerciales. *Trauma*, 25(3):150–156, 2014.

[TJCR17] L-M Torres, A-J Jiménez, A Cabezón, y M-J Rodríguez. Prevalencia del dolor irruptivo asociado al dolor crónico por lumbalgia en Andalucía (estudio COLUMBUS). *Revista de la Sociedad Española del Dolor*, 24(3):116–124, 2017.

[VER18] At-Home Physical Therapy: How VERA May Change Healthcare As We Know It, 2018. (Consultado el 5-6-2021). url: https://tinyurl.com/z38mmyhw.

[VGPA+20] David Vallejo, Cristian Gmez-Portes, Javier Albusac, Carlos Glez-Morcillo, y José Jesús Castro-Schez. Personalized Exergames Language: A Novel Approach to the Automatic Generation of Personalized Exergames for Stroke Patients. *Applied Sciences*, 10(20):7378, Oct 2020. url: http://dx.doi.org/10.3390/app10207378.

[WR220] What is rehabilitation?, 2020. (Consultado el 15-04-2021). url: https://www.who.int/news-room/fact-sheets/detail/rehabilitation.

Este documento fue editado y tipografiado con LaTeX empleando
la clase **esi-tfg** (versión 0.20181017) que se puede encontrar en:
`https://bitbucket.org/esi_atc/esi-tfg`