UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

TFG del Grado en Ingeniería Informática

VRRecoveryGym: entorno de realidad virtual para personas con trastornos motores

Presentado por Rodrigo Grande Muñoz
en Universidad de Burgos — 7 de julio de 2025
Tutores: Álvar Arnaiz González, Héctor Royo Concepción y Luis Fuente Ibáñez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática

D. Álvar Arnaiz González, profesor del departamento de Ingeniería Informática, área de Lenguajes y Sistemas Informáticos.

Expone:

Que el alumno D. Rodrigo Grande Muñoz, con DNI 72901698R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado «VRRecoveryGym: entorno de realidad virtual para personas con trastornos motores».

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 7 de julio de 2025

Vº. Bº. del Tutor:

D. Álvar Arnaiz González

Vº. Bº. del co-tutor:

D. Héctor Royo Concepción

Vº. Bº. del co-tutor:

D. Luis Fuente Ibáñez

Resumen

El objetivo de este proyecto es desarrollar un *entorno virtual* en el que los usuarios puedan realizar estos ejercicios desde casa mediante herramientas de realidad virtual. Esta solución permitirá no solo la realización remota de los ejercicios, sino también un seguimiento preciso de la actividad de cada usuario, registrando parámetros que puedan ser consultados en remoto por profesionales del sector.

Este proyecto se realiza en colaboración con la fundación ASPAYM y HP, cuya misión es mejorar la calidad de vida de las personas con lesión medular y discapacidad física grave. Con el objetivo de favorecer su recuperación, la fundación ha desarrollado una serie de puzles impresos en 3D que fomentan movimientos específicos, como la extensión del codo, la abducción del brazo y el giro de la muñeca. Estos ejercicios han demostrado ser efectivos, pero requieren la presencia física de los usuarios en la asociación y la supervisión de un profesional.

Para el desarrollo de la aplicación se utilizará el motor Unreal Engine, empleando tanto C++ como *Blueprint* para la programación de la lógica del sistema.

Se busca ofrecer una solución innovadora y accesible que facilite la rehabilitación motriz, mejorando la autonomía de los usuarios y optimizando el seguimiento terapéutico a través de la tecnología de realidad virtual.

Descriptores

puzles, entorno virtual, seguimiento remoto, rehabilitación motriz, Unreal Engine

Abstract

The goal of this project is to develop a *virtual environment* where users can perform these exercises from home using virtual reality tools. This solution will not only enable remote execution of exercises but also allow for precise tracking of each user's activity, recording parameters that can be remotely accessed by healthcare professionals.

This project is carried out in collaboration with the ASPAYM Foundation and HP, whose mission is to improve the quality of life of people with spinal cord injuries and severe physical disabilities. In order to support their recovery, the foundation has developed a series of 3D-printed puzzles designed to encourage specific movements, such as elbow extension, arm abduction, and wrist rotation. These exercises have proven to be effective, but they require users to be physically present at the association and supervised by a professional.

The application will be developed using the Unreal Engine, utilizing both C++ and *Blueprint* for system logic programming.

This project aims to provide an innovative and accessible solution that facilitates motor rehabilitation, enhances users' autonomy, and optimizes therapeutic monitoring through virtual reality technology.

Keywords

puzzles, *virtual environment*, remote monitoring, motor rehabilitation, Unreal Engine

Índice general

Índice de figuras

Índice de tablas

1. Introducción

Vivimos en una época donde la tecnología se ha convertido en una herramienta clave para mejorar la calidad de vida de las personas. En particular, el desarrollo de soluciones tecnológicas aplicadas al ámbito de la salud ha abierto nuevas posibilidades para abordar desafíos de rehabilitación física mediante enfoques innovadores. En este contexto, la realidad virtual (VR) ha demostrado un gran potencial para motivar y asistir a pacientes en procesos de recuperación motriz, especialmente en colectivos con discapacidades físicas que requieren terapias continuadas y personalizadas.

Actualmente, se utilizan puzles físicos impresos en 3D para promover movimientos específicos que permiten la rehabilitación de estas personas. Sin embargo, este enfoque presencial limita la autonomía de los usuarios y requiere supervisión constante por parte de profesionales.

El presente proyecto busca trasladar estos ejercicios a un entorno virtual accesible desde el domicilio del usuario. Mediante el uso de cascos de realidad virtual y controladores (o *hand tracking* en un futuro), se desarrollarán actividades interactivas que simulan los puzles físicos. Además, se recopilarán métricas clave sobre la actividad motriz y el progreso del usuario, que podrán ser consultadas por profesionales de forma remota mediante un *CSV* exportado con todos los datos relevantes sobre el progreso del paciente en los puzles.

El desarrollo se llevará a cabo utilizando Unreal Engine, combinando programación en C++ y *Blueprint*, así como recursos 3D entregados, ya realizados por la fundación, y el *marketplace* de Unreal. El proyecto se beneficia del asesoramiento de los profesionales de ASPAYM para garantizar la validez terapéutica del sistema.

2. Objetivos del proyecto

Este trabajo se enmarca en el ámbito de la rehabilitación física asistida por tecnologías inmersivas. Se propone el desarrollo de una herramienta innovadora en forma de entorno de realidad virtual para favorecer la recuperación motora de personas con lesión medular o discapacidad física en las extremidades superiores mediante la realización de puzles.

2.1. Objetivos

Los objetivos se dividen en dos categorías: técnicos y de desarrollo software.

Objetivos técnicos

1. **Simular puzles físicos en un entorno de realidad virtual:** reproducir los ejercicios físicos diseñados por ASPAYM mediante interacción virtual precisa.

2. **Registrar métricas motrices relevantes:** como número de repeticiones, rangos de movimiento, tiempos de resolución o errores durante la actividad.

3. **Facilitar la evaluación del progreso del usuario:** mediante informes visuales y datos exportables para su análisis por parte de profesionales.

4. **Permitir el uso remoto de la herramienta:** asegurando accesibilidad desde entornos domésticos sin supervisión directa constante.

5. **Explorar la viabilidad del *hand tracking*:** como alternativa a los mandos que acompañan las gafas de realidad virtual, buscando una experiencia más natural y accesible.

6. **Garantizar la estabilidad y rendimiento del entorno virtual:** con pruebas de usabilidad y detección temprana de errores.

Objetivos de desarrollo software

1. **Desarrollar el entorno de realidad virtual utilizando Unreal Engine:** combinando C++ y *Blueprints* para lograr un equilibrio entre rendimiento y modularidad.

2. **Implementar y optimizar modelos 3D con Blender:** representando fielmente los puzles físicos y el entorno de trabajo.

3. **Seguir un marco ágil de desarrollo (SCRUM):** organizando el trabajo en *sprints* y manteniendo una comunicación clara mediante GitLab.

4. **Realizar pruebas funcionales:** que permitan verificar la funcionalidad del sistema en distintas etapas de desarrollo.

5. **Preparar el sistema para su despliegue en dispositivos de realidad virtual:** inicialmente en una versión específica de dispositivo, con posibilidad de extenderlo a más dispositivos.

3. Conceptos teóricos

En esta sección se presentan los fundamentos teóricos necesarios para comprender el desarrollo y la implementación del presente Trabajo de Fin de Grado. Se abordan las principales definiciones, teorías y modelos que sustentan la investigación, así como las tecnologías y herramientas empleadas.

3.1. Realidad virtual

Se entiende por realidad virtual (VR) la tecnología que permite crear entornos digitales tridimensionales, interactivos y en tiempo real, de forma que el usuario perciba la sensación de "estar presente" en dicho entorno. Este grado de inmersión y "telepresencia" se consigue mediante la combinación de hardware (cascos HMD, sistemas de tracking, interfaces hápticas) y software (motores gráficos, algoritmos de simulación) que actualizan la escena virtual en función de los movimientos y acciones del usuario [32, 29].

Dentro del estudio de la VR suelen distinguirse tres características fundamentales:

Inmersión: grado en que el sistema aísla y sustituye estímulos sensoriales reales por estímulos generados artificialmente, creando la impresión directa de "estar dentro" del entorno virtual [21].

Presencia: sensación subjetiva de "estar realmente" en el escenario virtual, más allá de la mera percepción sensorial [29, 31].

Interactividad: capacidad del sistema para registrar las acciones del usuario y modificar el entorno virtual de forma coherente y en tiempo real [32, 21].

Estas características hacen de la VR una herramienta poderosa en ámbitos tan diversos como la simulación de entrenamiento, la rehabilitación médica, la visualización arquitectónica o la investigación científica.

Componentes de un sistema de VR

Un sistema de VR típico se compone de los siguientes elementos [29, 2]:

- **Dispositivo de visualización (Head-Mounted Display, HMD):** muestra las imágenes estereoscópicas e incorpora sensores de orientación (giroscopios, acelerómetros).
- **Sistema de seguimiento (tracking):** rastrea la posición y orientación de la cabeza, manos u otros objetos mediante cámaras, emisores infrarrojos o sistemas magnéticos.
- **Dispositivos de interacción:** mandos, guantes hápticos o superficies táctiles que permiten manipular objetos virtuales.
- **Motor gráfico y software de simulación:** genera las escenas 3D, gestiona física, colisiones y comportamientos de los elementos del entorno.

Taxonomía: Continuo de la virtualidad

Milgram y Kishino propusieron el concepto de continuo de la virtualidad, que va desde la realidad completamente física hasta la realidad completamente virtual, englobando la realidad aumentada (AR) y la realidad mixta asistida [21]:

- **Realidad Aumentada (AR):** superposición de información virtual sobre el entorno real.
- **Realidad Mixta (MR):** fusión de elementos reales y virtuales que interactúan entre sí.
- **Realidad Virtual (VR):** reemplazo total del entorno real por uno virtual.

Modalidades de uso y aplicaciones

Según el grado de inmersión y el tipo de interacción, los sistemas de VR pueden clasificarse en:

- **VR de escritorio:** utiliza pantallas convencionales y controladores estándar; menor inmersión, pero accesible.
- **VR inmersiva:** emplea HMDs y sistemas de tracking avanzado; alta inmersión y presencia [2].
- **CAVEs:** espacios cerrados con pantallas proyectadas en paredes y suelo; múltiples usuarios simultáneos.

Las aplicaciones más difundidas incluyen simuladores de vuelo y conducción, terapias de exposición en salud mental, entrenamiento quirúrgico y entornos de aprendizaje inmersivo.

3.2. Diseño de videojuegos para rehabilitación

El uso de videojuegos para la rehabilitación física se enmarca dentro del concepto de *serious games*, definido como aquellos juegos cuyo principal objetivo va más allá del entretenimiento, buscando también la educación, la formación o la mejora de la salud [5].

Principios de diseño

El desarrollo de videojuegos para rehabilitación debe considerar aspectos tanto lúdicos como terapéuticos [14]. Entre los principales principios de diseño se encuentran:

- **Interactividad:** permite al usuario manipular y explorar el entorno, facilitando el aprendizaje y la mejora de habilidades motoras.
- ***Feedback* inmediato:** el sistema debe ofrecer respuestas instantáneas a las acciones del usuario, reforzando el progreso [3].
- **Dificultad progresiva:** las tareas deben adaptarse al nivel de capacidad del paciente, aumentando la complejidad de forma gradual para evitar la frustración [34].

- **Repetición y refuerzo positivo:** elementos clave en la rehabilitación física para promover la neuroplasticidad [15].

Gamificación y motivación

La gamificación introduce mecánicas de juego (puntos, logros, niveles) que aumentan la motivación y el compromiso del usuario [26]. En el caso del presente proyecto, la superación de puzles y la obtención de logros sirven como incentivo para continuar con las sesiones terapéuticas.

Accesibilidad y consideraciones éticas

Los videojuegos de rehabilitación deben garantizar la accesibilidad a personas con diferentes grados de discapacidad. Además, se debe asegurar que el uso de la tecnología no sustituya la supervisión profesional, sino que actúe como complemento [25].

3.3. Teoría del aprendizaje motor

La teoría del aprendizaje motor se basa en el modelo trifásico de Fitts y Posner [10], que describe la adquisición de habilidades motoras en tres etapas:

1. **Fase cognitiva:** el aprendiz construye una representación mental de la tarea, explorando secuencias de movimiento y cometiendo errores frecuentes. La ejecución es lenta y requiere alta atención consciente [27].
2. **Fase asociativa:** se establecen mapeos más precisos entre estímulos y respuestas motoras. La variabilidad de la ejecución disminuye y se corrigen errores de forma más autónoma, consolidándose patrones de movimiento [27].
3. **Fase autónoma:** la habilidad está completamente automatizada; la ejecución es rápida, fluida y apenas requiere atención consciente, liberando recursos cognitivos para otras tareas [10].

El *feedback* inmediato es crítico para la consolidación de estos patrones. Winstein y Schmidt demostraron que una frecuencia reducida de conocimiento de resultados (*knowledge of results*) optimiza el aprendizaje a largo plazo, evitando la dependencia excesiva de la retroalimentación externa [33].

En entornos de realidad virtual, el *feedback* inmediato y multimodal (visual, auditivo y háptico) permite detectar y corregir errores en tiempo real, reforzando los esquemas motores y favoreciendo la neuroplasticidad [27]. Además, la inmersión en VR incrementa la motivación y la adherencia al entrenamiento, potenciando la repetición deliberada y la automatización de la habilidad.

3.4. Gamificación y motivación intrínseca

La *gamificación* consiste en la aplicación de elementos de diseño de juegos (p. ej., puntos, niveles, insignias, narrativa, retos) en contextos no lúdicos con el objetivo de aumentar la implicación, la motivación y la adherencia de los usuarios a una tarea [7]. En el ámbito de la rehabilitación en VR, la gamificación permite transformar ejercicios repetitivos en experiencias más atractivas, favoreciendo la práctica continua y la mejora progresiva de las capacidades motoras y cognitivas.

La *motivación intrínseca* se sustenta en la Teoría de la Autodeterminación de Deci y Ryan [6], que postula tres necesidades psicológicas básicas:

- **Autonomía:** sensación de control y elección sobre la propia conducta.
- **Competencia:** percepción de eficacia y dominio en la realización de tareas.
- **Relación:** conexión social y sentido de pertenencia con otros participantes o terapeutas.

Cuando un sistema gamificado satisface estas necesidades, la motivación intrínseca aumenta, conduciendo a una práctica más sostenida y a mejores resultados terapéuticos.

Kapp [16] propone que la gamificación efectiva integra mecánicas (reglas del juego), dinámicas (comportamientos de los usuarios) y elementos estéticos (presentación), de modo que:

1. **Mecánicas:** puntos, niveles, temporizadores y desafíos escalonados para construir progresión.
2. **Dinámicas:** recompensas variables, retroalimentación inmediata y metas claras que fomentan la repetición deliberada.

3. **Estética:** narrativa inmersiva y diseño coherente con la experiencia de usuario en VR.

En entorno de VR, el *feedback* inmediato (visual, auditivo y háptico) refuerza la sensación de competencia y contribuye a la consolidación de los aprendizajes motores y cognitivos.

3.5. Rotaciones en 3D y *Gimbal Lock*

En el desarrollo de entornos tridimensionales, uno de los aspectos clave es el tratamiento de las rotaciones de los objetos. A diferencia del entorno 2D, donde una rotación puede describirse mediante un único ángulo, en 3D existen múltiples formas de representar una orientación: ángulos de *Euler*, matrices de rotación y *quaternions*, entre otros.

En *Unreal Engine*, una de las representaciones más utilizadas son los ángulos de *Euler*, que definen una rotación como la combinación secuencial de tres giros sobre los ejes principales (*pitch*, *yaw*, *roll*), generalmente en el orden *X-Y-Z*. Sin embargo, esta representación presenta ciertas limitaciones, entre ellas el fenómeno conocido como ***gimbal lock***.

El *gimbal lock* ocurre cuando dos de los tres ejes de rotación se alinean, perdiendo un grado de libertad y, por tanto, provocando comportamientos inesperados en las rotaciones. Esto puede dificultar el control preciso de la orientación de un objeto, especialmente cuando se realizan rotaciones combinadas y complejas.

Para evitar este tipo de problemas, muchos motores gráficos —incluido *Unreal Engine* en sus operaciones internas— utilizan cuaterniones para representar rotaciones. Los cuaterniones permiten interpolaciones suaves (como la *slerp*) y evitan el *gimbal lock*, aunque su interpretación visual y matemática es más compleja [30].

3.6. Sistemas de colisiones y físicas en 3D

En los entornos tridimensionales interactivos como los desarrollados en *Unreal Engine*, la detección de colisiones y la simulación física juegan un papel fundamental tanto para la interacción con objetos como para la inmersión del usuario [12].

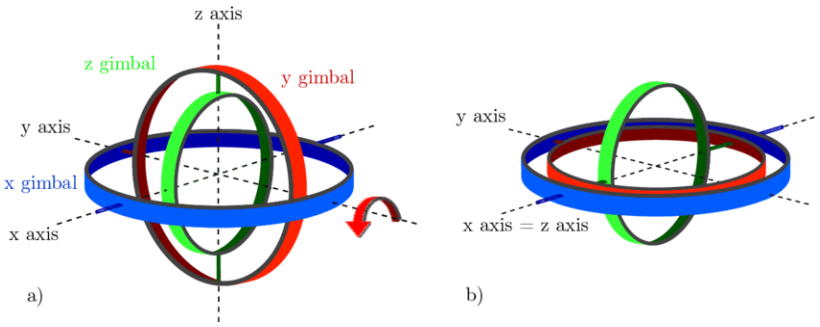

Figura 3.1: Ejemplo de *gimbal lock*. Imagen extraída de [24].

Detección de colisiones

La detección de colisiones es el proceso mediante el cual se determina si dos o más objetos en el espacio tridimensional se intersecan o entran en contacto. Para optimizar este cálculo, los motores como *Unreal Engine* utilizan primitivas de colisión simplificadas (por ejemplo, cajas, esferas o cápsulas) en lugar de las mallas complejas de los modelos. Cada objeto puede tener uno o varios componentes de colisión que definen su comportamiento físico y de colisión en el mundo virtual.

Existen distintos canales de colisión y respuestas configurables:

- **Bloqueo**: impide el paso del objeto.
- **Superposición**: detecta el contacto sin detener el movimiento.
- **Ignorar**: desactiva la colisión frente a determinados tipos de objetos.

Estas respuestas se configuran en el motor mediante el sistema de *Collision Presets* y *Object Channels*, permitiendo un control detallado sobre las interacciones entre objetos.

Simulación física

La simulación física busca replicar de manera realista el comportamiento de los objetos bajo leyes físicas básicas como la gravedad, la inercia, la fricción o los rebotes. *Unreal Engine* emplea el motor físico *Chaos Physics* para realizar estos cálculos en tiempo real.

Cada objeto puede tener activada o no la simulación física. Si está activada, se comporta como un cuerpo dinámico, reaccionando a fuerzas, colisiones y otras interacciones.

3.7. Comunicación entre objetos en entornos visuales

En el desarrollo de sistemas interactivos complejos dentro de *Unreal Engine*, especialmente mediante *Blueprints*, uno de los aspectos fundamentales es la capacidad de los distintos objetos (actores, componentes, interfaces) para comunicarse entre sí de manera eficiente, estructurada y mantenible. Esta comunicación define el flujo lógico del juego, la interacción del usuario con los elementos del entorno y la coordinación entre diferentes subsistemas del motor.

Lógica basada en eventos

El sistema de *Blueprints* se apoya en un modelo orientado a eventos (*event-driven*), en el cual los objetos reaccionan a estímulos como colisiones, entradas del usuario, temporizadores o condiciones personalizadas. Estos eventos se conectan con nodos de ejecución dentro del *Graph Editor*, lo que permite definir respuestas visuales sin necesidad de programación textual en *C++* [11].

Entre los eventos más comunes se encuentran:

- `BeginPlay`: se ejecuta al inicio del juego o cuando el actor es activado.
- `Tick`: se ejecuta cada fotograma y suele usarse para operaciones continuas.
- `OnOverlapBegin` / `OnHit`: se disparan en colisiones físicas o superposiciones.
- Eventos personalizados definidos por el usuario (*Custom Events*).

Interfaces en Blueprint

Para garantizar un bajo acoplamiento entre elementos y fomentar la reutilización del código, *Unreal Engine* proporciona el sistema de *Blueprint Interfaces*. Estas permiten definir contratos de funciones sin necesidad de conocer la implementación exacta del otro objeto [8].

Las interfaces son especialmente útiles cuando múltiples actores diferentes deben reaccionar a una misma instrucción sin necesidad de herencia

múltiple. Por ejemplo, una interfaz `Interactuable` podría declarar una función `Interactuar()`, que luego implementen tanto puertas, botones como personajes no jugables.

Despachadores de eventos (*Event Dispatchers*)

Otra herramienta poderosa para la comunicación asíncrona entre objetos es el uso de *Event Dispatchers*. Estos permiten declarar emisores de eventos personalizados dentro de un *Blueprint*, que otros actores pueden suscribirse para ejecutar funciones concretas cuando se produzca el evento.

Este patrón de diseño tipo *observador* facilita la coordinación entre componentes sin acoplamiento directo.

Flujos condicionales y comunicación entre componentes

Además de los mecanismos anteriores, el sistema de *Blueprints* permite acceder directamente a componentes de otros actores mediante referencias explícitas. Aunque esta técnica es más directa, puede generar dependencias rígidas si no se usa con cautela.

Para casos sencillos o controlados, acceder a propiedades públicas de otro actor (por ejemplo, cambiar el color de una luz, activar una animación o reproducir un sonido) es una forma válida y eficaz de comunicación.

Buenas prácticas de arquitectura

Para evitar problemas de escalabilidad y errores difíciles de depurar, se recomienda:

- Usar *Interfaces* para definir comunicación entre clases distintas.
- Utilizar *Dispatchers* cuando el evento debe propagar información a varios objetos.
- Minimizar referencias directas entre objetos no relacionados jerárquicamente.
- Modularizar la lógica en funciones reutilizables y macros claramente documentadas.

Estas herramientas permiten mantener un sistema organizado, extensible y coherente, especialmente en entornos de desarrollo visual como el que ofrece *Unreal Engine*, donde el control sobre la ejecución y la interacción entre objetos es esencial para la calidad final del proyecto.

4. Técnicas y herramientas

En esta sección se presentan las técnicas y herramientas fundamentales empleadas en el desarrollo del proyecto. A continuación, se presentan las herramientas empleadas durante el desarrollo del proyecto.

4.1. Técnicas

Scrum

La metodología ágil *Scrum* [28] es un marco de trabajo muy extendido en el ámbito del desarrollo de software, especialmente útil en proyectos donde los requisitos pueden variar con frecuencia o no están completamente definidos desde el inicio. Su principal objetivo es facilitar una entrega continua de valor al cliente mediante un proceso iterativo e incremental.

Scrum promueve la colaboración activa entre todos los miembros del equipo, fomenta la transparencia en cada fase del desarrollo y permite una adaptación rápida a cambios en las necesidades o prioridades. El trabajo se estructura en ciclos cortos llamados *sprints*, cuya duración habitual oscila entre una y cuatro semanas. Cada *sprint* comienza con una sesión de planificación (*sprint planning*), seguida de reuniones breves diarias (*daily stand-up*) para monitorizar el progreso (no realizadas en este proyecto), y finaliza con una revisión del trabajo desarrollado (*sprint review*) y una retrospectiva orientada a detectar oportunidades de mejora continua.

Programación modular

La programación modular es una técnica de diseño y desarrollo de software basada en la descomposición del sistema en unidades funcionales

independientes llamadas módulos. Cada módulo encapsula una parte específica de la lógica del programa, lo que permite dividir el desarrollo en componentes más manejables, reutilizables y fáciles de mantener [23].

Esta aproximación facilita la comprensión del sistema, promueve la reutilización de código y mejora la localización de errores, ya que cada módulo puede desarrollarse, probarse y depurarse de forma aislada. Además, favorece el trabajo colaborativo al permitir que distintos miembros del equipo trabajen en módulos separados sin interferencias.

En el contexto de este proyecto, se aplicó esta técnica organizando la lógica funcional del entorno virtual en subsistemas independientes mediante el uso de *Blueprints* en *Unreal Engine*. Por ejemplo, se definieron módulos separados para el sistema de interacción, la locomoción del usuario, la gestión de datos de rendimiento o la lógica de los puzles de rehabilitación. Esta estructura modular ha permitido realizar iteraciones sobre partes concretas del sistema sin afectar al resto de la aplicación.

4.2. Herramientas

Unreal Engine 5

Unreal Engine es un motor gráfico y de desarrollo de juegos en tiempo real desarrollado por Epic Games. Su arquitectura modular y su potente editor visual permiten diseñar, iterar y desplegar entornos 3D de alta fidelidad para aplicaciones que van desde videojuegos hasta simulaciones VR y AR. Para este proyecto se ha utilizado Unreal Engine 5, aprovechando sus siguientes características principales:

- **Editor de niveles y *blueprints*:** interfaz *What You See Is What You Get* para construir escenas y lógica de juego mediante nodos, sin necesidad de programar directamente en C++.
- **Renderizado de última generación:** soporte para Nanite (geometría virtualizada) y Lumen (iluminación global dinámica), que permiten obtener gráficos realistas en tiempo real.
- **Soporte nativo de Realidad Virtual:** integración con SDKs de dispositivos HMD (Oculus, SteamVR, etc.), sistemas de tracking y controladores hápticos.

Para la integración de la interfaz de usuario en VR se ha empleado el sistema UMG (Unreal Motion Graphics), lo que ha permitido diseñar menús y HUDs directamente dentro del entorno 3D.

Plantilla VR de Unreal Engine

La plantilla de VR incluida en Unreal Engine proporciona una base funcional para el desarrollo de aplicaciones en *realidad virtual*, permitiendo a los desarrolladores centrarse en la lógica y el diseño sin tener que implementar desde cero los sistemas de locomoción e interacción.

En este proyecto, la plantilla ha sido empleada como punto de partida para estructurar la escena inicial, configurar los controles específicos de las *Meta Quest 2*, y facilitar la integración con dispositivos de entrada de VR. Sus características principales incluyen:

- **Sistema de locomoción integrado:** rotación por grados y movimiento libre, adaptado para entornos de juego o simulación.
- **Interacción con objetos:** detección de colisiones, agarre mediante botones de los controladores y manipulación directa de objetos con físicas.
- **Compatibilidad inmediata con HMDs (*Head-Mounted Displays*):** configuración automática del entorno para su ejecución en dispositivos compatibles con *OpenXR* o *Oculus SDK*.
- **Estructura modular:** organización en componentes reutilizables como el *Motion Controller*, el sistema de agarre, la cámara estereoscópica y el controlador de entrada.

A partir de esta base, se han realizado múltiples modificaciones y ampliaciones específicas para la aplicación, adaptando la lógica de interacción a los requisitos.

Blueprints: sistema de scripting visual

El sistema de *Blueprints* constituye uno de los pilares fundamentales de la programación visual en *Unreal Engine*. Diseñado para facilitar la creación de lógica de juego sin necesidad de escribir código en C++, permite a desarrolladores, diseñadores y artistas implementar comportamientos complejos de forma visual, rápida y estructurada [11].

Blueprints se basa en un paradigma híbrido entre programación orientada a objetos y flujo de datos (*dataflow programming*), donde los elementos visuales (nodos) representan funciones, eventos, variables o estructuras de control, y las conexiones entre ellos definen la ejecución y el paso de información.

Estructura y componentes de un Blueprint

Un *Blueprint* es, en esencia, una clase que hereda de *UObject* o *AActor*, que puede contener:

- **Variables:** permiten almacenar datos (booleanos, enteros, vectores, estructuras, etc.) visibles o editables en el editor.
- **Componentes:** objetos adjuntos como mallas, colisionadores, luces, cámaras, etc., que definen su apariencia y funcionalidad.
- **Funciones:** encapsulan lógica reutilizable con entradas y salidas bien definidas.
- **Eventos:** puntos de entrada al flujo lógico, como *BeginPlay*, *OnOverlap*, o eventos definidos por el usuario.
- **Macros:** bloques reutilizables de lógica visual que pueden tener múltiples puntos de ejecución.
- **Interfaces:** contratos que permiten comunicación entre *Blueprints* sin necesidad de herencia directa.

Tipos de grafos en Blueprints

- ***Event Graph***: principal área de trabajo donde se definen eventos y flujo de ejecución. Los nodos se conectan para formar lógica condicional, secuencial o paralela.
- ***Construction Script***: grafo especial que se ejecuta en el editor (no en tiempo de juego), útil para inicializar geometría, materiales o transformaciones antes del *runtime*.
- **Anim Graph (solo en Blueprints de Animación)**: permite definir el comportamiento de estados de animación mediante máquinas de estados y transiciones.

Compilación y ejecución

Cuando se guarda un Blueprint, el sistema lo compila a un bytecode específico que se ejecuta dentro de la *Blueprint Virtual Machine (VM)*, una máquina virtual optimizada para mantener una alta compatibilidad con el motor nativo. Esto permite:

- Establecer *breakpoints* para depuración paso a paso.
- Visualizar el flujo de ejecución en tiempo real mediante el sistema de seguimiento visual.
- Usar herramientas de análisis como el *Profiler* de Blueprints.

Ventajas y limitaciones

El sistema *Blueprints* permite una rápida iteración y una cuVRa de aprendizaje accesible, especialmente para perfiles no programadores. Sin embargo, para operaciones intensivas, algoritmos complejos o gestión avanzada de memoria, se recomienda el uso de código C++ nativo, o su combinación con Blueprints a través del sistema de exposición de funciones y clases públicas [8].

Blender

Blender es una suite de creación 3D de código abierto que incluye herramientas para modelado, escultura, texturizado y exportación de geometrías. En este proyecto, su uso se ha centrado exclusivamente en la preparación de los modelos 3D proporcionados en formato STL y su posterior exportación para Unreal Engine:

- **Ajuste de escala y orientación:** normalización de las dimensiones del modelo y alineación de los ejes para que coincidan con las unidades y el sistema de coordenadas de Unreal Engine.
- **Conversión y exportación:** exportación del modelo en formatos compatibles (FBX u OBJ), asegurando que las normales de las caras permanezcan orientadas correctamente y que no se pierda información geométrica al importar en Unreal Engine.

4.3. Dependencias

Durante el desarrollo del proyecto se han empleado diferentes dependencias para facilitar determinadas funcionalidades:

Victory Plugin

Es un conjunto de herramientas desarrolladas por Rama que amplía las funcionalidades nativas del motor Unreal Engine mediante nodos personalizados para *Blueprints*.

Este plugin ha resultado especialmente útil en tareas relacionadas con la manipulación de archivos y operaciones del sistema que no están disponibles directamente en el sistema estándar de *Blueprints*. En el contexto del presente proyecto, se ha utilizado principalmente para implementar la funcionalidad de exportación de datos de usuario en formato *CSV*, permitiendo registrar métricas relevantes para la rehabilitación como el nombre del usuario, el tiempo de resolución de puzles, la fecha de la sesión, y el estado del resultado.

Gracias a esta librería, se ha podido implementar un sistema de almacenamiento externo de datos sin recurrir a programación en *C++*, integrando toda la lógica desde el entorno visual de *Blueprints*.

OpenXR

OpenXR es una especificación abierta y multiplataforma promovida por el consorcio *Khronos Group*, diseñada para proporcionar una interfaz común entre las aplicaciones de realidad extendida (*XR*) y los dispositivos de hardware compatibles [17]. Esta solución estandarizada permite que los desarrolladores construyan experiencias de realidad virtual (*VR*) y aumentada (*AR*) que funcionen sin cambios en múltiples plataformas y dispositivos, tales como *Meta Quest*, *HTC Vive*, *Windows Mixed Reality*, entre otros.

En el contexto de *Unreal Engine*, el plugin *OpenXR* sirve como puente entre la lógica del motor y el hardware XR, facilitando funcionalidades como la renderización estereoscópica, el seguimiento posicional, la gestión de controladores hápticos y la integración con los sistemas de entrada del usuario. Su arquitectura modular y abierta ha convertido a *OpenXR* en la solución recomendada por defecto para el desarrollo de aplicaciones *VR* en las últimas versiones del motor [9].

En este proyecto se utilizó el complemento *OpenXR* para garantizar la compatibilidad del entorno de rehabilitación con visores como *Meta Quest*

2, minimizando la dependencia de SDKs propietarios como *Oculus SDK* y asegurando una mayor portabilidad a futuro. Esta integración permitió ejecutar el sistema tanto en modo *VR Preview* dentro del editor como de forma autónoma en el visor, manteniendo una experiencia fluida y estable para el usuario final.

Dispositivos y hardware

Para la ejecución y pruebas del proyecto se ha utilizado el dispositivo de realidad virtual **Meta Quest 2**, un sistema autónomo de VR que combina alta calidad gráfica con un sistema de seguimiento preciso y facilidad de uso.

Figura 4.1: Gafas Meta Quest 2. Imagen extraída de [20]

Las gafas Meta Quest 2 permiten una experiencia inmersiva sin necesidad de conexión a un ordenador externo, lo que facilita su uso en entornos clínicos y domésticos, aportando flexibilidad y autonomía al paciente durante las sesiones de rehabilitación.

El dispositivo cuenta con dos controladores inalámbricos que permiten una interacción natural mediante detección de movimientos y pulsaciones, esenciales para la manipulación de los puzles y la navegación en el entorno virtual. Además, su sistema de seguimiento *inside-out* utiliza cámaras integradas para detectar la posición y orientación de la cabeza y las manos del usuario, eliminando la necesidad de estaciones base externas.

Para el desarrollo y pruebas se ha empleado un ordenador personal compatible con las exigencias de Unreal Engine y la transmisión vía *Link* para conectar las Meta Quest 2 al entorno de desarrollo cuando ha sido necesario.

La elección de este *hardware* se ha basado en la disponibilidad del mismo para ser utilizado durante el desarrollo.

Tabla 4.1: Herramientas empleadas en el desarrollo del proyecto

Herramienta	**Descripción**
Unreal Engine 5	Motor gráfico empleado para el desarrollo del entorno virtual interactivo en *realidad virtual*.
Plantilla VR de UE5	Plantilla base integrada en Unreal Engine para VR, utilizada como punto de partida para la lógica de interacción y locomoción.
Blueprints	Sistema de scripting visual de Unreal Engine que ha permitido desarrollar lógica de juego sin necesidad de código en C++.
Victory Plugin	Plugin para *Blueprints* que habilita la exportación de datos en formato *CSV*, entre otras funcionalidades extendidas.
OpenXR Plugin	Complemento oficial de Unreal Engine que permite la integración multiplataforma con dispositivos XR, garantizando compatibilidad con visores como *Meta Quest 2*.
Blender	Suite de creación 3D utilizada para ajustar y exportar modelos 3D al formato compatible con Unreal Engine.
GitLab	Plataforma de control de versiones empleada para la gestión del código y la colaboración durante el desarrollo del proyecto.
LaTeX	Sistema de preparación de documentos científicos utilizado para redactar la memoria del proyecto.
Meta Quest Link	Herramienta para vincular las Meta Quest 2 a un PC durante el desarrollo y pruebas del entorno VR.

Tabla 4.2: Técnicas empleadas en el desarrollo del proyecto

Técnica	**Descripción**
Metodología ágil *Scrum*	Técnica de gestión del desarrollo basada en iteraciones cortas denominadas *sprints*, con planificación, revisión y retrospectiva para mejorar el proceso de forma incremental.
Programación modular	División del sistema en módulos funcionales independientes y reutilizables, facilitando el mantenimiento, la escalabilidad y las pruebas unitarias.
Diseño centrado en el usuario (DCU)	Técnica que prioriza las necesidades, capacidades y limitaciones del usuario final durante todo el proceso de diseño e implementación.
Control de versiones con *Git*	Gestión del código fuente mediante una herramienta distribuida para registrar cambios y mantener trazabilidad del desarrollo.
Pruebas de sistema manuales	Aplicación de una técnica de verificación funcional mediante ejecución directa de escenarios de uso en el entorno real, validando el comportamiento del sistema con distintos perfiles de usuario.

5. Aspectos relevantes del desarrollo del proyecto

Durante el desarrollo del proyecto se han enfrentado diversos retos técnicos, metodológicos y humanos que han condicionado e impulsado la evolución del sistema. Este capítulo recoge los elementos más destacados del proceso, presentados a continuación.

5.1. Colaboración con el cliente y desarrollo del proyecto

Uno de los aspectos más significativos del presente proyecto ha sido la oportunidad de colaborar con un cliente externo real. En este caso, se trató de la asociación ASPAYM, cuyo responsable participó activamente en las reuniones de seguimiento. Esta colaboración permitió establecer una comunicación directa y continua, que fue clave para orientar el desarrollo del sistema hacia una solución alineada con las necesidades del colectivo al que va dirigido.

La implicación del cliente se tradujo en una serie de aportaciones valiosas, tanto a nivel conceptual como funcional. Durante el transcurso del proyecto, se presentaron propuestas preliminares que fueron debatidas en conjunto, recibiendo sugerencias, observaciones y validaciones por parte de la asociación y tutores. Esta dinámica de trabajo contribuyó a mejorar iterativamente el diseño del sistema, garantizando que cada funcionalidad implementada respondiera a un criterio de utilidad real.

Metodología de trabajo

Aunque no se realizó una prueba directa con usuarios finales, la visión experta del cliente aportó un conocimiento profundo de las barreras cotidianas que enfrentan los usuarios finales, lo cual fue determinante para orientar ciertas decisiones técnicas, como la configuración de los controles o la usabilidad.

Asimismo, la utilización de un marco de trabajo basado en *SCRUM* facilitó la adaptación continua a los cambios y sugerencias planteadas durante las reuniones. La división del trabajo en *sprints* y la planificación de entregas incrementales promovieron un entorno de desarrollo dinámico, flexible y centrado en resultados funcionales.

5.2. Desarrollo en entorno tridimensional

Otro punto destacable ha sido el proceso de aprendizaje en torno al desarrollo de aplicaciones en realidad virtual, un campo completamente nuevo al inicio del trabajo. Desde el primer momento, enfrentarse al desarrollo 3D supuso un reto conceptual importante. Comprender cómo funciona un entorno tridimensional implicó familiarizarse con nociones fundamentales como los sistemas de coordenadas, la orientación de los ejes (X, Y, Z), las posiciones relativas entre objetos, y especialmente la rotación de modelos, que no siempre es intuitiva debido a conceptos como el *gimbal lock* o el uso de *cuaterniones* en segundo plano explicados en los conceptos teóricos.

Uno de los mayores desafíos iniciales fue entender cómo calcular correctamente desplazamientos, trayectorias y movimientos dentro del espacio 3D, teniendo en cuenta tanto la posición absoluta como la local de los objetos, y cómo estas se ven afectadas por la jerarquía de actores o el uso de componentes. Adaptarse a este paradigma espacial llevó tiempo, especialmente al trabajar con interacciones físicas y colisiones dentro del entorno virtual.

5.3. Aprendizaje del motor *Unreal Engine*

Además, el modelo de desarrollo propuesto por *Unreal Engine* supuso una curva de aprendizaje pronunciada. Su sistema basado en *Levels*, *Blueprints*, *Actors* e *Interfaces* 3D resultó inicialmente complejo de asimilar, especialmente por la forma tan modular y a la vez interconectada en la que se organiza el proyecto. Por ejemplo, entender cuándo crear un *Blueprint* a nivel de actor, cuándo utilizar un *GameMode*, o cómo comunicar eventos

entre componentes mediante *Interfaces* o *Event Dispatchers*, supuso un ejercicio continuo de pruebas hasta conseguir el resultado deseado.

Todo este proceso requirió una formación autodidacta intensiva, apoyada en documentación oficial, foros, vídeos y la experimentación directa. A lo largo del proyecto se fue adquiriendo experiencia en el diseño de entornos virtuales inmersivos, en la implementación de mecánicas interactivas adaptadas a las capacidades del dispositivo *Meta Quest*, y en la gestión de la lógica del juego tanto en entornos *VR* como en el plano técnico de programación. Esto incluyó tanto el uso extensivo de *Blueprints* como la integración puntual de código en *C++* para extender la funcionalidad del motor donde fue necesario.

5.4. Implementación de modelos 3D

El uso de recursos tridimensionales ha sido un componente clave en el desarrollo de los puzles interactivos del sistema. Estos modelos no solo tienen una función estética, sino que también están intrínsecamente ligados a la lógica del juego, a través de sus propiedades físicas, su interacción con el entorno y su respuesta a los movimientos del usuario.

Cabe destacar que, al inicio del proyecto, no se contaban con conocimientos previos sobre modelado 3D ni sobre herramientas de edición como *Blender*. Esto implicó un proceso de aprendizaje intensivo para adquirir los fundamentos necesarios sobre manipulación de geometría en un espacio tridimensional.

El uso de *Blender* permitió realizar tareas fundamentales como:

- **Ajuste de escala y orientación:** fue necesario comprender cómo funcionan los sistemas de coordenadas en 3D, y cómo se relacionan con la escala y la rotación de los objetos. Especialmente importante fue aprender a posicionar correctamente el *pivote* o centro de transformación de cada modelo, ya que este punto determina el eje de rotación y desplazamiento dentro de *Unreal Engine*.
- **Texturizado básico:** aunque el enfoque principal del proyecto no era artístico, se trabajó también en la aplicación de imágenes y colores simples a los modelos para mejorar la identificación de piezas durante los ejercicios. Esto implicó la asignación de materiales y la comprensión básica del sistema de *UV Mapping*, que permite proyectar una imagen sobre la superficie del modelo.

- **Exportación adecuada:** finalmente, los modelos fueron exportados en formatos compatibles con *Unreal Engine*, como `.FBX` o `.OBJ`, asegurando que se mantuvieran las transformaciones aplicadas, el centro de masas y las normales orientadas correctamente.

El proceso de aprendizaje fue eminentemente práctico y se apoyó en recursos como documentación oficial, tutoriales en vídeo, foros y pruebas de ensayo-error. Gracias a ello, se logró integrar los modelos en el motor con la configuración adecuada de físicas, colisiones y puntos de anclaje, lo cual fue crucial para garantizar una experiencia fluida, funcional y segura dentro del entorno virtual.

5.5. Diseño accesible para usuarios con discapacidad

Por último, resulta crucial haber identificado desde el inicio las particularidades de los usuarios finales, muchos de los cuales presentan limitaciones motrices derivadas de su condición. Diseñar la jugabilidad pensando en ellas implicó adoptar mecánicas que reduzcan la necesidad de movimientos precisos y de alta velocidad, así como ofrecer múltiples modalidades de interacción—ya sea mediante gestos amplios y simplificados, menús accesibles o ayudas visuales y sonoras que guíen al jugador.

Este enfoque centrado en la accesibilidad no solo mejora la experiencia de quienes tienen dificultades físicas, sino que también enriquece el sistema general al fomentar una interfaz intuitiva y adaptable, capaz de ajustarse a distintos niveles de destreza y garantizar una participación inclusiva y satisfactoria.

6. Trabajos relacionados

Este proyecto se enmarca en la intersección de la realidad virtual, los videojuegos y la rehabilitación motora. Se han identificado diversas líneas de investigación relevantes, que se detallan a continuación.

Realidad virtual inmersiva para rehabilitación motora

La realidad virtual inmersiva ha demostrado ser efectiva en la rehabilitación de extremidades superiores, especialmente en pacientes que han sufrido accidentes cerebrovasculares. Estudios recientes como en el artículo *"Virtual Reality Therapy for Upper Limb Motor Impairments in Stroke Patients: A Meta-Analysis"* [13], indican que la combinación de VR con terapias tradicionales mejora la función motora y la destreza manual, siendo más efectiva durante las etapas aguda y subaguda de la recuperación.

Además, la VR permite a los pacientes realizar tareas funcionales repetitivas de manera lúdica, aumentando la adherencia al tratamiento y la motivación [22].

Desarrollo de videojuegos terapéuticos con Unity y Unreal Engine

En esta sección, es relevante citar el trabajo *"Design and usability evaluation of an immersive virtual reality mirrored hand system for upper limb stroke rehabilitation"* [18]. Se ha desarrollado un sistema de realidad virtual inmersiva con actividades de la vida real, como cocinar o conducir, para la rehabilitación del miembro superior en las personas con accidentes cerebrovasculares.

Asimismo, proponen un tipo de videojuego interactivo que combina el juego con la actividad física como terapia virtual que utiliza sensores portátiles inteligentes para la rehabilitación de la extremidad superior, capturando movimientos de la mano y el codo para personalizar la terapia, recogido en el siguiente trabajo [1].

Gamificación y motivación en la terapia VR

La gamificación en la VR ha mostrado beneficios en la rehabilitación, como la mejora de la función motora y la reducción del dolor percibido. Por ejemplo, en el artículo *"Playing your pain away: designing a virtual reality physical therapy for children with upper limb motor impairmen"* [19],se ha diseñado una terapia física de realidad virtual para niños con discapacidad motora en las extremidades superiores, mejorando la duración del ejercicio y produciendo emociones positivas hacia la terapia.

Integración de VR y robótica en la rehabilitación

La combinación de VR con dispositivos robóticos ha sido explorada para mejorar la rehabilitación de la extremidad superior. El estudio *"Immersive Virtual Reality and Robotics for Upper Extremity Rehabilitatio"* [4], introdujo una solución de rehabilitación virtual que combina VR con robótica y sensores portátiles para analizar los movimientos de la articulación del codo, mostrando ventajas potenciales en un enfoque inmersivo y multisensorial.

7. Conclusiones y Líneas de trabajo futuras

En este último apartado de la memoria se pretende exponer todo lo aprendido durante la realización del proyecto y mostrar ideas sobre mejoras para el presente proyecto.

7.1. Conclusiones

Debido a la diversidad de aprendizajes adquiridos durante el desarrollo del presente Trabajo de Fin de Grado, se ha optado por estructurar las conclusiones en tres bloques diferenciados: **conclusiones científicas**, relacionadas con los principios de rehabilitación y experiencia de usuario; **conclusiones técnicas**, centradas en el desarrollo del entorno VR; y **conclusiones personales**, vinculadas al crecimiento profesional y académico del desarrollador.

Conclusiones científicas

El desarrollo del entorno virtual de rehabilitación ha permitido profundizar en el impacto de la retroalimentación inmediata, la repetición y la interacción activa como elementos clave en el aprendizaje motor. Se ha confirmado que el diseño de ejercicios que exigen movimientos precisos y guiados favorece la consolidación de patrones motores, tal como apuntan estudios previos en neurorehabilitación.

Además, se ha evidenciado la necesidad de adaptar los entornos virtuales a las capacidades físicas y cognitivas del usuario final. El diseño debe contem-

plar tanto la accesibilidad como la variabilidad del ritmo de rehabilitación de cada paciente. Esto se traduce, por ejemplo, en la necesidad de permitir ajustes de dificultad o controlar el tiempo de sesión.

Finalmente, la experiencia con usuarios reales, aunque limitada, ha servido para validar la utilidad de la realidad virtual como herramienta de apoyo en terapias convencionales, destacando especialmente su potencial motivacional.

Conclusiones técnicas

Desde el punto de vista técnico, uno de los aprendizajes más significativos ha sido el dominio progresivo del motor *Unreal Engine 5.5*, junto con sus herramientas asociadas como *Blueprints* y el soporte parcial en *C++*. La utilización de la plantilla base para realidad virtual proporcionó una estructura inicial robusta, aunque fue necesario realizar múltiples adaptaciones para responder a los requerimientos específicos del proyecto.

Uno de los principales desafíos fue la creación de un sistema de guardado personalizado capaz de almacenar datos relevantes sobre el rendimiento del usuario. Asimismo, se implementó un mecanismo de exportación de datos en formato *CSV*, utilizando nodos extendidos a través del *Victory Plugin*, lo cual requirió una comprensión profunda del flujo de datos dentro del entorno VR.

También se abordó la configuración dinámica de niveles en función del tipo de ejercicio, lo que implicó una gestión precisa de variables globales, sistemas de navegación y condiciones de activación. La correcta orquestación de los eventos de realidad virtual y la comunicación entre actores mediante *Blueprint Interfaces* y eventos personalizados ha resultado esencial para mantener la coherencia del entorno y garantizar una experiencia fluida.

Por otro lado, se prestó especial atención a la construcción de interfaces accesibles, al tratamiento del input procedente de controladores hápticos, y a la optimización del rendimiento para mantener una tasa de refresco estable en el visor. Todos estos aspectos han contribuido al fortalecimiento del conocimiento sobre la arquitectura de *Unreal Engine*, el ciclo de vida de sus componentes y su modelo de ejecución en tiempo real.

Conclusiones personales

A nivel personal, este proyecto ha supuesto un reto en todas sus fases, que ha exigido aprender y aplicar conocimientos de programación, diseño de

UX/UI, metodologías ágiles y documentación técnica. Destacar además de que no se tenia ningún conocimiento sobre el diseño de un videojuego en realidad virtual, esto ha supuesto muchas horas de aprendizaje autodidacta y lectura de documentación.

Se ha aprendido a priorizar tareas, resolver bloqueos técnicos de forma autónoma y a colaborar con un cliente real (ASPAYM), ajustando el desarrollo a sus necesidades y expectativas. Esta experiencia ha resaltado la importancia de la comunicación clara, la empatía con el usuario final y la capacidad de adaptación ante nuevos desafíos.

Por último, este trabajo ha reforzado la idea de que la constancia y la disciplina son más determinantes que la motivación esporádica. La complejidad del desarrollo, junto con la responsabilidad de llevar a cabo un producto funcional en solitario, con todas las fases del desarrollo, ha permitido desarrollar una ética de trabajo sólida y resiliente. Aun así, destacar que la finalidad altruista de este proyecto ha motivado mucho, desde un principio, por sacar el proyecto adelante.

7.2. Líneas de trabajo futuras

El ámbito de la rehabilitación mediante realidad virtual continúa en expansión, y la integración de nuevas dinámicas interactivas, junto con la mejora técnica del sistema, puede marcar una diferencia significativa en su aplicabilidad real. Por ello, se plantean las siguientes líneas de trabajo que permitirían refinar, ampliar y optimizar la solución desarrollada. Estas propuestas se agrupan en diferentes categorías según su naturaleza.

Diseño de nuevos puzles rehabilitadores

Hasta el momento, se ha implementado únicamente el primer ejercicio interactivo, *El Palacio de Shahriar*, diseñado con prioridad alta por su aplicabilidad en etapas iniciales del proceso de rehabilitación. No obstante, el proyecto contempla la incorporación de un conjunto más amplio de puzles temáticos, que aporten diversidad y progresión en la dificultad. Las siguientes líneas de trabajo están orientadas a completar este objetivo:

- **Desarrollar nuevos puzles:** completar los ejercicios restantes según la lista establecida por ASPAYM.

- **Ajustar la dificultad progresiva:** diseñar cada puzle con niveles de exigencia crecientes, tanto a nivel cognitivo como motriz, adaptados a distintos perfiles de usuario.
- **Incorporar nuevos tipos de interacción:** explorar otras mecánicas más avanzadas (por ejemplo, movimientos finos de muñeca o rotaciones) que complementen el rango de movimientos trabajados.

Interfaz y experiencia de usuario

Aunque la interfaz actual es completamente funcional y ha sido diseñada en base a criterios de accesibilidad, se han detectado oportunidades de mejora a nivel estético y de experiencia de usuario. Algunas líneas a considerar son:

- **Revisión estética:** aplicar un rediseño visual de la interfaz, incorporando una paleta de colores más atractiva, iconografía adaptada y animaciones suaves que refuercen la inmersión.
- ***Feedback* adaptativo:** mejorar el sistema de retroalimentación visual y sonora en función del desempeño del usuario.
- **Personalización de la experiencia:** permitir al usuario configurar preferencias como el volumen o idioma.

Rendimiento y optimización

Pese a los esfuerzos realizados en mejorar el rendimiento durante el desarrollo, es un área que siempre admite margen de mejora, especialmente en entornos de realidad virtual donde los recursos son limitados.

- **Optimización del renderizado:** revisar los modelos y materiales utilizados para garantizar la máxima eficiencia sin comprometer la calidad visual.
- **Reducción del consumo de memoria:** auditar el uso de texturas, sonidos y blueprints para identificar elementos innecesarios o duplicados.
- **Pruebas de rendimiento sistemáticas:** implementar una batería de tests automatizados para comprobar la estabilidad del sistema en diferentes escenarios y niveles.

Incorporación de seguimiento de manos (*hand tracking*)

La incorporación de tecnologías de seguimiento de manos puede representar un salto cualitativo en cuanto a accesibilidad, naturalidad del movimiento y eliminación de barreras tecnológicas (como la necesidad de agarrar controladores físicos).

- **Compatibilidad con hand tracking nativo:** adaptar las interacciones del sistema para que puedan ser ejecutadas sin controladores físicos, empleando directamente los gestos naturales de las manos del usuario.
- **Diseño de gestos personalizados:** definir un conjunto de gestos que puedan ser utilizados en sustitución de acciones comunes como agarrar, seleccionar o mover objetos.
- **Evaluación de precisión y robustez:** analizar la fiabilidad del reconocimiento en distintos entornos y condiciones de iluminación, y su impacto en el proceso de rehabilitación.

Bibliografía

[1] L. Baron, J. Gutiérrez, and R. Ramírez. Virtual therapy exergame for upper extremity rehabilitation using smart wearable sensors. *arXiv preprint arXiv:2302.08573*, 2023.

[2] Grigore Burdea and Philippe Coiffet. *Virtual Reality Technology*. Wiley-Interscience, 2003.

[3] M. S. Cameirao et al. The impact of virtual reality based rehabilitation on stroke patients. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 18(5):531–538, 2010.

[4] V. Chheang, J. Sim, and H. Le. Immersive virtual reality and robotics for upper extremity rehabilitation. *arXiv preprint arXiv:2304.11110*, 2023.

[5] P. David et al. *Serious games: mechanisms and effects*. Routledge, 2016.

[6] Edward L. Deci and Richard M. Ryan. *Intrinsic Motivation and Self-Determination in Human Behavior*. Plenum, 1985.

[7] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart E. Nacke. From game design elements to gamefulness: Defining gamification. In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, pages 9–15, 2011.

[8] John P. Doran and Nitish Misra. *Blueprints Visual Scripting for Unreal Engine 5: Learn visual scripting and game development by building fun projects with Unreal Engine 5*. Packt Publishing, 2022.

[9] Epic Games. OpenXR plugin documentation, 2024. Último acceso: julio de 2025.

[10] P. M. Fitts and M. I. Posner. *Human Performance*. Brooks/Cole, 1967.

[11] Epic Games. Blueprints visual scripting, 2024. Accedido el 27 de junio de 2025.

[12] Epic Games. Chaos physics overview, 2024. Accedido el 27 de junio de 2025.

[13] M. Gandolfi, N. Valè, E. Dimitrova, et al. Virtual reality therapy for upper limb motor impairments in stroke patients: A meta-analysis. *Journal of Rehabilitation Research*, 2024.

[14] J. A. Greene et al. Virtual reality and interactive video game technology for arm and hand rehabilitation. *Stroke Research and Treatment*, 2016.

[15] M. K. Holden. Virtual environments for motor rehabilitation: review. *CyberPsychology and Behavior*, 8(3):187–211, 2005.

[16] Karl M. Kapp. *The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education*. Pfeiffer, 2012.

[17] Khronos Group. OpenXR: Open standard for AR/VR platforms, 2021. Último acceso: julio de 2025.

[18] S. Kim, Y. Yoon, and H. Jang. Design and usability evaluation of an immersive virtual reality mirrored-hand system for upper limb rehabilitation. *Scientific Reports*, 2025.

[19] J. Maldonado, L. Serrano, and H. Díaz. Playing your pain away: Designing a virtual reality physical therapy for children with upper limb motor impairment. *JMIR Serious Games*, 2023.

[20] Meta Platforms, Inc. Meta Quest – Gafas de realidad virtual todo en uno. https://www.meta.com/es/quest/, 2025. Accedido el 7 de julio de 2025.

[21] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE Transactions on Information Systems*, E77-D(12):1321–1329, 1994.

[22] D. S. Park, H. Lee, and K. Kim. Effects of immersive virtual reality on upper-extremity stroke rehabilitation: A systematic review and meta-analysis. *Frontiers in Neurology*, 2023.

[23] Roger S. Pressman and Bruce R. Maxim. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Education, 9th edition, 2019.

[24] ResearchGate. Illustration of the principle of gimbal lock, 2019. Accessed: 2025-06-24.

[25] A. Rodriguez et al. Designing accessible virtual rehabilitation environments. *Disability and Rehabilitation: Assistive Technology*, 15(2):145–153, 2020.

[26] L. Sardi et al. The role of gamification in health and wellness apps: review. *JMIR mHealth and uHealth*, 5(6):e94, 2017.

[27] R. A. Schmidt and T. D. Lee. *Motor Control and Learning: A Behavioral Emphasis*. Human Kinetics, 6 edition, 2019.

[28] Ken Schwaber and Jeff Sutherland. The scrum guide, 2020. Available at: https://scrumguides.org.

[29] William R. Sherman and Alan B. Craig. *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann, 2002.

[30] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 245–254. ACM, 1985.

[31] Mel Slater and Martin Usoh. Presence in immersive virtual environments. In *Proceedings of the 5th Annual Presence Workshop*, pages 27–56, 1994.

[32] Jonathan Steuer. Defining virtual reality: Dimensions determining telepresence. *Journal of Communication*, 42(4):73–93, 1992.

[33] C. J. Winstein and R. A. Schmidt. Reduced frequency of knowledge of results enhances motor skill learning. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 17(4):842–856, 1991.

[34] L. Zimmerli et al. Virtual reality based therapy for rehabilitation of motor function. *Journal of NeuroEngineering and Rehabilitation*, 9(1):1–15, 2012.