Universidad de Valladolid

UNIVERSIDAD DE VALLADOLID

ESCUELA DE INGENIERIAS INDUSTRIALES

Grado en Ingeniería Electrónica Industrial y Automática

DESARROLLO DE UN JUEGO SERIO DE REALIDAD AUMENTADA PARA REHABILITACIÓN NEUROMOTORA

Autor:

MATILLA PLAZA, Alberto

Tutor:

FUENTE LÓPEZ, Eusebio De La

RESUMEN

Este Trabajo de Fin de Grado se centra en el diseño y desarrollo de un juego serio de realidad aumentada con el propósito de facilitar la rehabilitación neuromotora. La utilización de la tecnología de realidad aumentada junto con las terapias de rehabilitación tradicionales puede mejorar de forma significativa la motivación y compromiso de los pacientes durante el proceso de recuperación.

El juego se desarrolla con el motor de videojuegos Unity, ampliamente reconocido por su versatilidad y potencia en la creación de experiencias interactivas. Para controlar la lógica dentro del videojuego se utilizan scripts escritos en C#, un lenguaje de programación compatible con el motor Unity y adecuado para el desarrollo de aplicaciones interactivas.

ABSTRACT

This Final Degree Project focuses on the design and development of a serious augmented reality game aimed at facilitating neuromotor rehabilitation. The use of augmented reality technology combined with traditional rehabilitation therapies can significantly enhance patient motivation and engagement during the recovery process.

The game is developed using the Unity game engine, widely recognized for its versatility and power in creating interactive experiences. To control the logic within the game, scripts written in C# are used, a programming language compatible with the Unity engine and suitable for developing interactive applications.

ÍNDICE PRINCIPAL

1. INTRODUCCIÓN Y OBJETIVOS. 13
1.1. MARCO DEL PROYECTO. 13
1.2. Contexto del Problema. 14
1.3. Objetivos del Proyecto. 16
1.4. Descripción de la Memoria. 17
1.4.1. Estado del arte. 17
1.4.2. Herramientas para el desarrollo del juego serio. 17
1.4.3. Estructura principal en el desarrollo del juego “Coins” para la rehabilitación del miembro superior. 18
1.4.4. Interacción de los objetos principales para el buen funcionamiento del juego serio “Coins” para la rehabilitación del miembro superior. 18
1.4.5. Desarrollo de las funciones para el buen funcionamiento del juego serio “Coins” para la rehabilitación del miembro superior. 19
1.4.6. Resultados Obtenidos. 19
1.4.6. Conclusiones. 19
2. ESTADO DEL ARTE. 21
2.1. Innovaciones Tecnológicas en la Rehabilitación. 22
2.1.1. Realidad virtual y aumentada en rehabilitación. 22
2.1.2. Telefisioterapia. 23
2.1.3. Exoesqueletos y dispositivos robóticos. 24
2.1.4. Biofeedback, aplicaciones móviles y sensores vestibles (wearables). 25
2.1.5. Inteligencia artificial en el diagnóstico y pronóstico. 26
2.1.6. Terapia genética. 26
2.1.7. Terapia basada en células. 27
2.2. Transformación tecnológica en la fisioterapia. 28
2.3. Realidad Aumentada. 29
2.4. Influencia de los videojuegos en el ámbito de la medicina. 31
2.5. Rehabilitación del miembro superior y uso de Realidad Aumentada. 33
3. HERRAMIENTAS PARA EL DESARROLLO DEL JUEGO SERIO. 36
3.1. M3Display. 36
3.1.1. Estructura Hardware. 37
3.1.2. Detección de la mano. 38
3.1.3. Procesamiento de datos de la mano. 39

3.1.4. Interfaz de selección de juegos. 40
3.1.5. Juegos serios para la rehabilitación. 41
3.1.5.1. Motor Unity. 42
4. ESTRUCTURA DEL JUEGO. 46
4.1. Concepto General. 46
4.2. Implementación M3Display. 47
4.2.1. Instalaciones necesarias. 47
4.2.1.1. Unity. 47
4.2.1.2. Python. 48
4.3. Unity. 48
4.3.1. Unity Hub. 48
4.3.2. Interfaz Unity. 49
4.3.2.1. Ventana Scene. 50
4.3.2.2. Ventana Hierarchy. 50
4.3.2.3. Ventana Project. 51
4.3.2.4. Ventana Inspector. 51
4.3.2.4. Ventana Console. 51
4.3.2.4. Ventana Game. 52
4.3.3. *GameObjects*. 52
4.3.4. Interacciones de *GameObjects*. 52
4.3.4.1. Unity UI. 53
4.3.4.2. UnityEngine.UI. 53
4.3.4.3. UnityEngine.Events. 54
4.3.5. *GameObjects* principales. 54
4.3.5.1. Table. 54
4.3.5.2. Canvas. 55
4.3.5.2.1. Menú principal. 56
4.3.5.2.2. Menú opciones. 56
4.3.5.2.3. Menú *InGame*. 57
4.3.5.2.4. Menú pausa. 58
4.3.5.2.5. Menú de carga. 58
4.3.5.2.6. Menú final. 59
4.3.5.3. Games. 59
4.3.5.4. SaveData. 62

4.3.5.5. Hand. 63
4.3.5.6. Audio Source. 63
5. INTERACCIÓN DE LOS OBJETOS EN EL JUEGO. 65
5.1. *GameObject* Botón. 65
5.1.1. Función *On Click()* 66
5.1.2. Transición entre menús. 67
5.1.3. Selección de trayectorias. 68
5.2. *GameObject* Slider. 69
5.2.1. Función *On Value Changed()*. 70
5.2.2. Variables del juego. 71
6. DESARROLLO DE LAS FUNCIONES DEL JUEGO. 73
6.1. Puntos de referencia de la mano. 73
6.2. Captura de monedas. 74
6.3. Puntuación. 76
6.4. Temporizador 76
6.5. Sonido. 77
6.6. Recopilación de datos. 78
6.7. Finalización de la partida. 79
7. RESULTADOS OBTENIDOS. 81
7.1. Pruebas funcionamiento M3Display. 81
7.2. Pruebas funcionamiento interfaz de usuario. 82
7.3. Pruebas funcionamiento acciones *InGame*. 82
7.4. Pruebas funcionamiento recogida de datos. 83
8. CONCLUSIONES. 87
8.1. Posibles mejoras en el proyecto. 87
8.1.1. Implementación Inteligencia Artificial. 87
8.1.2. Implementación de gafas de Realidad Virtual. 88
8.2. Conocimientos aplicados y validación del sistema 89
BIBLIOGRAFÍA 91
ANEXO I – *Script PlayerHand* 94
ANEXO II – *Script Score* 96
ANEXO III – *Script GameTimer* 99
ANEXO IV – *Script EndGame* 102
ANEXO V – *Script UICollisionDetection* 106

ANEXO VI – *Script SaveDataCoins* 109
ANEXO VII – *Script SoundManager* 114
ANEXO VIII – *Script CoinAnimation* 116
ANEXO IX – *Script SliderDifficultyValue* 118
ANEXO X – *Script SliderTimer* 120
ANEXO XI – Manual de Usuario 122

ÍNDICE DE FIGURAS

Figura 1: Variación Porcentual del mercado global de tecnología médica de 2010 a 2022 ..... **15**

Figura 2: Evolución anual de la inversión en investigación y desarrollo en sector de tecnología médica a nivel global desde 2011 a 2024 ..... **28**

Figura 3: Diferencias entre realidad virtual, realidad aumentada y realidad mixta **29**

Figura 4: Número de usuarios de realidad aumentada móvil a nivel mundial desde 2019 hasta 2024 ..... **30**

Figura 5: Miembro superior **33**

Figura 6: Distribución en la posición de la mano. [A] muestra los objetivos alcanzados con realidad aumentada. [B] muestra los objetivos alcanzados con versión PC. [C] muestra el movimiento con realidad aumentada. [D] muestra el movimiento en la versión PC **34**

Figura 7: Diferencias en la puntuación obtenida **35**

Figura 8: Estructura M3Display. (a) Estructura. (b) Reflexión de la imagen sobre el espejo. (c) Punto de vista del paciente **37**

Figura 9: (a) Puntos de referencia de las manos con profundidad relativa presentada en diferentes tonos. (b) Seguimiento en tiempo real de múltiples manos en Pixel 3 ..... **39**

Figura 10: Procesamiento de datos para la detección de la mano **40**

Figura 11: Interfaz M3Display para la selección de juego y registros de usuario **41**

Figura 12: Motores de desarrollo de videojuegos más populares **43**

Figura 13: Aumento en el número de videojuegos desarrollados con el uso del motor Unity **44**

Figura 14: Gameplay de varios videojuegos desarrollados con Unity. (a) Pokémon GO. (b) Among Us. (c) Fall Guys. (d) Subnautica. **44**

Figura 15: Oficina virtual de Meta Slap desarrollada por LG U+ **45**

Figura 16: Versión Unity utilizada y Unity Hub **49**

Figura 17: Gestión de proyectos en Unity Hub **49**

Figura 18: Interfaz del motor Unity **50**

Figura 19: Inicialización UnityEngine.UI **53**

Figura 20: *GameObject* Table **55**

Figura 21: Menús dentro de Canvas **55**

Figura 22: Menú Principal **56**

Figura 23: Menú Opciones **57**

Figura 24: Menú *InGame* **57**

Figura 25: Menú Pausa **58**

Figura 26: Menú de carga **58**

Figura 27: Menú final de la partida **59**

Figura 28: *GameObject* Games **60**

Figura 29: Trayectorias disponibles. (a) Trayectorias dificultad fácil. (b) Trayectorias dificultad media. (c) Trayectorias dificultad difícil **60**

Figura 30: Proyección de los elementos virtuales sobre la mesa **61**

Figura 31: *Asset* textura moneda **61**

Figura 32: Imagen de la trayectoria **62**

Figura 33: *GameObject SaveData* **62**

Figura 34: *GameObject* Hand **63**

Figura 35: *GameObject* Audio Source **64**

Figura 36: Sonido utilizado para el feedback auditivo **64**

Figura 37: Configuración de botón **65**

Figura 38: Programación de funciones de un *GameObject* **66**

Figura 39: Función *SetActive(bool)* para activar o desactivar un *GameObject* **67**

Figura 40: Botón de selección de trayectoria **68**

Figura 41: Diseño de un *Slider* **69**

Figura 42: Configuración *Slider* **70**

Figura 43: Función *On Value Changed()* de un *Slider* **71**

Figura 44: Transformación de puntos de referencia de la mano en *GameObjects* **74**

Figura 45: Lógica de una moneda **75**

Figura 46: Script *Score* **76**

Figura 47: Script *GameTimer* **77**

Figura 48: Script *SoundManager* **78**

Figura 49: Script *SaveData* **78**

Figura 50: Script *EndGame* **79**

Figura 51: Creación carpeta *CoinData* **83**

Figura 52: Archivos .CSV guardados dentro de *CoinData* **83**

Figura 53: Tablas con los datos guardados para el seguimiento de la evolución **84**

Figura 54: Obtención de puntos a lo largo de la partida **84**

Figura 55: Obtención de gráficas para la comparación de resultados **85**

Figura 56: Comparación de la trayectoria descrita por el paciente con la trayectoria objetivo **85**

Figura 57: Manual de usuario. Menú inicial **122**

Figura 58: Manual de usuario. Selección de parámetros **122**

Figura 59: Manual de usuario. Trayectorias disponibles **123**

Figura 60: Manual de usuario. Inicio de partida **123**

Figura 61: Manual de usuario. Transcurso de la partida **124**

Figura 62: Manual de usuario. Trayectoria **125**

Figura 63: Manual de usuario. Juego pausado **125**

Figura 64: Manual de usuario. Partida finalizada **126**

Figura 65: Manual de usuario. Carpeta *CoinData* **126**

1. INTRODUCCIÓN Y OBJETIVOS.

1.1. MARCO DEL PROYECTO.

El Daño Cerebral Adquirido (DCA) engloba diversas lesiones cerebrales causadas por traumas, ictus u otras complicaciones postparto, afectando a una parte significativa de la población española y mundial anualmente, constituyendo un desafío significativo para el sistema de salud.

Las secuelas neuromotoras derivadas del DCA, como la paresia y pérdida de fuerza en el agarre puede suponer en un impacto considerable en la bienestar y calidad de vida de los pacientes, dificultando así su capacidad para realizar ciertas actividades cotidianas. (Astudillo Aguiar, 2023)

La paresia es el término general para referirse a un grado de debilidad muscular leve a moderado. La paresia, a contraposición de la parálisis que implica una pérdida total o severa de la función motora, se caracteriza por una disminución parcial de la fuerza muscular. Algunos de los síntomas presentes son la debilidad en los músculos, pérdida de equilibrio y coordinación. (BVS, 1963)

Durante una evaluación clínica se pueden detectar diferentes signos que sugieren la presencia de DCA. Estos signos incluyen anomalías en los reflejos musculares, presencia de contracciones involuntarias, como el clonus, y la observación de respuestas anormales a estímulos específicos, como el signo de Babinski. También se pueden apreciar alteraciones en la musculatura, movimientos involuntarios como las sincinesias y fasciculaciones, atrofia muscular, trastornos de micción y defecación, así como ataxia, afasia o apraxia.

Las causas de aparición de la paresia pueden variar según su presentación. La paresia espástica puede estar asociada con un accidente cerebrovascular, tumores, esclerosis múltiple, traumatismo de la médula espinal. La paresia flácida puede surgir como resultado de condiciones como el Síndrome de Guillain-Barré, reacciones adversas a medicamentos, o trastornos neuromusculares hereditarios. Mientras que para la paresia simultánea puede ser consecuencias de enfermedades neurodegenerativas, como la esclerosis lateral amiotrófica, o infecciones virales que afectan la médula espinal.

El tratamiento fisioterapéutico para la paresia se adapta individualmente para cada paciente, según su condición y el estado de la enfermedad. Este tratamiento individualizado puede incluir una combinación de ejercicios de fortalecimiento muscular, terapia ocupacional, y técnicas de rehabilitación específicas para mejorar el equilibrio y la coordinación. El objetivo principal del tratamiento debe ser optimizar la función muscular y mejorar la calidad de vida

del paciente, en busca de la independencia en las actividades cotidianas. Algunos métodos para la rehabilitación del paciente serían: (FisioOnline, 2024):

· Movilizaciones activas y pasivas de los miembros superiores e inferiores.

· Ejercicios activos asistidos.

· Método de Rood.

· Método de Kabat.

· Facilitación neuromuscular propioceptiva.

· Aplicación de agentes físicos como la termoterapia, electroterapia, crioterapia, etc.

· Técnica de miofeedback.

· Reeducación postural global.

· Ejercicios respiratorios.

1.2. Contexto del Problema.

La rehabilitación tradicional para pacientes con DCA puede ser monótona y pueden llegar a provocar una falta de motivación, que puede llegar a afectar a la adherencia y empeorar los resultados en el tratamiento. Para solucionar este problema la integración de la realidad aumentada en la rehabilitación neuromotora presenta una oportunidad para hacer que las sesiones sean más interactivas, personalizadas y atractivas para los pacientes, lo que potencialmente mejora los resultados de la rehabilitación.

El uso de realidad aumentada es posible gracias al auge de la tecnología en la medicina en estos últimos años. Según “The future Unmasked. Predicting the future of healthcare and life sciences in 2025” la industria de telemedicina crecerá un 19,3% a nivel mundial hasta el año 2025 (APD, 2023).

En esta gráfica publicada por EvaluateMedTech, se observa la variación porcentual del mercado global de tecnología médica desde el año 2010 hasta el año 2020.

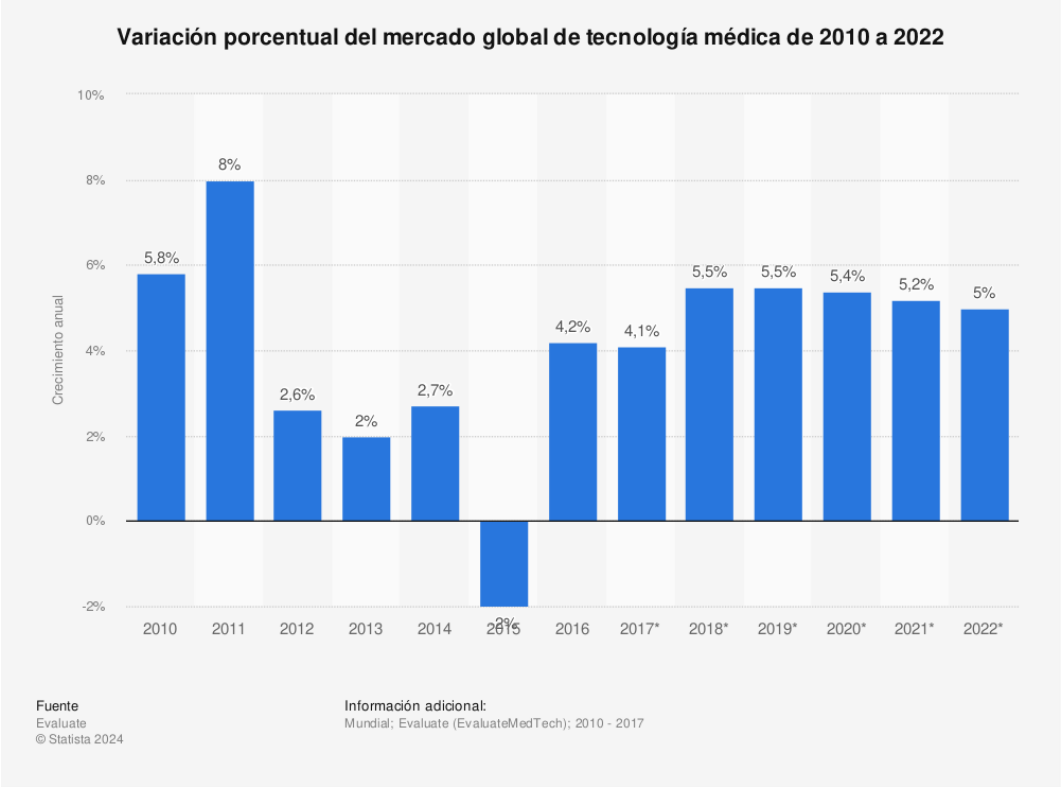

Figura 1: Variación Porcentual del mercado global de tecnología médica de 2010 a 2022. Fuente: (EvaluateMedTech, 2024).

Se puede observar que, a excepción del año 2015, donde cae un 2%, año tras año el porcentaje de utilización de la tecnología en la medicina aumenta.

La realidad virtual y la realidad aumentada representan tecnologías innovadoras que están revolucionando el campo de la rehabilitación neurológica, ofreciendo experiencias inmersivas e interactivas complementando los enfoques de los tratamientos de rehabilitación tradicionales. Al proporcionar simulaciones y entornos virtuales personalizados, la realidad virtual y la realidad aumentada pueden permitir a los pacientes participar de forma más estimulante y motivadora en las actividades terapéuticas.

La combinación de realidad virtual y realidad aumentada con la robótica y la inteligencia artificial abre aún más las posibilidades para la innovación dentro del campo de la rehabilitación neurológica. Un ejemplo podría ser la integración de prótesis robóticas con entornos virtuales cada vez más inmersivos y realistas, realizando tratamientos más funcionales y, por tanto, produciéndose una aceleración y mejora de resultados durante el periodo de rehabilitación. (ClinicaUner, 2023)

1.3. Objetivos del Proyecto.

El objetivo principal de este proyecto es diseñar y desarrollar un videojuego que utilice un sistema de rehabilitación basado en la realidad aumentada para pacientes con DCA, enfocado en la rehabilitación del miembro superior. El videojuego deberá tener los siguientes objetivos específicos:

· Crear un entorno de realidad aumentada que ofrezca unas actividades de rehabilitación personalizadas para cada paciente. Esto incluirá el diseño de escenarios virtuales y la implementación de actividades adaptativas que se ajusten al nivel de habilidad y proceso de cada paciente.

· Implementar herramientas de seguimiento y progreso para monitorear la evolución de los pacientes a lo largo del tratamiento. Se desarrollarán sistemas de registro de datos que permitan al terapeuta recopilar información sobre el desempeño de los pacientes en cada una de las sesiones de rehabilitación. En estos sistemas de registro de datos se incluirán gráficas de movimiento, tiempos de reacción y progreso.

· Integrar el videojuego de realidad aumentada desarrollado en este proyecto en el entorno virtual y soporte físico diseñado en el laboratorio. Esto implicará el ajuste y la coordinación del videojuego con el hardware físico utilizado en el proceso de rehabilitación, asegurando una integración fluida y efectiva entre el software y el hardware.

El cumplimiento de estos objetivos permitirá no solo el desarrollo y la implementación exitosa del sistema de rehabilitación con el uso de realidad aumentada, sino también la evaluación de su efectividad y potencial para mejorar los resultados en la rehabilitación frente al utilización de terapias tradicionales en pacientes con DCA.

1.4. Descripción de la Memoria.

La memoria de este proyecto se estructura en base a varios apartados, donde se detalla el desarrollo del juego serio *Coins* para la rehabilitación del miembro superior.

1.4.1. Estado del arte.

En este apartado se hablará de las principales tecnologías existentes que han influido en el desarrollo del juego. Se trata de explicar cómo han influido las nuevas tecnologías en la rehabilitación y el incremento de estas, la realidad aumentada, y el aumento del uso de videojuegos en el ámbito de la medicina. Para comprender el contexto de estas innovaciones, se realizará un análisis de artículos y estudios previos que abordan el uso de estas tecnologías en la rehabilitación. Este análisis ayudará a explicar la justificación del desarrollo de *Coins*, destacando cómo estas tecnologías contribuyen a mejorar los resultados de los pacientes y a hacer que el proceso de rehabilitación sea más accesible y efectivo.

1.4.2. Herramientas para el desarrollo del juego serio.

Para este punto se hará énfasis en las herramientas que han sido utilizadas para el desarrollo del juego serio. Se proporcionará una descripción detallada de las plataformas, software y bibliotecas que han facilitado el diseño, programación y prueba del juego. Se explicará cómo se ha implementado el sistema M3Display, cubriendo tanto al parte del software como la del hardware, que en conjunto proporcionan el entorno necesario para la rehabilitación.

1.4.3. Estructura principal en el desarrollo del juego “Coins” para la rehabilitación del miembro superior.

En el apartado siguiente se describirá la estructura principal que se ha seguido para el desarrollo de *Coins*. Mostrando los elementos clave que componen el juego, así como la organización de los menús y lógica interna para el buen funcionamiento de la aplicación.

Se definirán los componentes básicos del juego, como lo son los *GameObjects*, que forman los menús, y como estos forman la interfaz de usuario.

1.4.4. Interacción de los objetos principales para el buen funcionamiento del juego serio “Coins” para la rehabilitación del miembro superior.

En el punto anterior se muestra la organización, en cuanto a menús se refiere, de la interfaz de usuario del juego, que facilitan la navegación y selección de opciones para el usuario. En este apartado de explicará el funcionamiento de los elementos interactivos que se insertan en los menús para el buen funcionamiento del juego.

Estos elementos, compuestos por botones, *Sliders* o paneles, permiten la selección de menús y otras opciones dentro del juego. Deben aportar una operabilidad fácil e intuitiva para el terapeuta, permitiendo así agilizar las sesiones de terapia y mejorando el proceso de rehabilitación.

1.4.5. Desarrollo de las funciones para el buen funcionamiento del juego serio “Coins” para la rehabilitación del miembro superior.

Para que los elementos clave insertados en los menús funcionen correctamente, se tiene que programar la lógica de estos. En este apartado se explican los scripts en lenguaje C# y la programación interna de bloques que ofrece la interfaz de Unity para programar esta lógica.

1.4.6. Resultados Obtenidos.

Para comprobar el buen funcionamiento de todos los elementos que componen la interfaz de usuario y la lógica de estos, se deben realizar las pruebas oportunas. En este apartado se mostrarán los resultados obtenidos sobre las pruebas realizadas durante el desarrollo del juego.

El objetivo de estas pruebas será validar que todos los aspectos del juego, desde la interfaz de usuario hasta la lógica de funcionamiento y la interacción con los *GameObjects*, operen según lo previsto. Durante el proceso de prueba, se compararán los resultados obtenidos con los resultados esperados, para identificar cualquier error de programación o fallo en la interfaz de usuario.

Una vez el proceso de desarrollo haya concluido, se tendrá que simular el desempeño que tendrá el terapeuta a la hora de utilizar el juego, y visualizar los archivos con los datos generados para asegurar un correcto funcionamiento.

1.4.6. Conclusiones.

En este apartado se presentarán las conclusiones derivadas de los resultados obtenidos en las pruebas de funcionamiento. Se evaluarán los aspectos clave del desarrollo y la implementación del juego, considerando su efectividad y desempeño.

Los resultados del proyecto mostrarán si el juego cumple con los objetivos propuestos, proporcionando una herramienta interactiva y atractiva para la

rehabilitación. Sin embargo, como cualquier proyecto, siempre existen oportunidades de mejora, por lo que también se tratarán las diferentes tecnologías que podrían añadirse para una futura evolución, con el objetivo de una mejora en la funcionalidad y jugabilidad del juego.

2. ESTADO DEL ARTE.

En la rehabilitación, y en la medicina en general, se ha experimentado una notable transformación con la incorporación progresiva de nuevas tecnologías. Con el paso de los años, para explorar en detalle la evolución histórica de la rehabilitación, podemos segmentarla en tres períodos clave, reflejando su desarrollo y creciente complejidad. Los períodos se definen por cambios significativos en la historia, tanto en la percepción social de las personas con discapacidades, como en las prácticas médicas asociadas con la rehabilitación.

La primera fase se podría situar desde la antigüedad hasta la Primera Guerra Mundial. Hay que remontarse a tiempos prehistóricos para comenzar a estudiar la progresión de la medicina de rehabilitación. El ser humano primitivo, movido por instinto, empleaba métodos simples para aliviar el dolor y preparar su cuerpo para las exigencias físicas que afrontaba en su día a día. Este enfoque intuitivo incluía el frotamiento en áreas doloridas, o la aplicación de frío o calor a su disposición en la naturaleza. Durante esta era, la rehabilitación estaba menos formalizada y se centraba en la integración o exclusión social de las personas con discapacidades. Las prácticas de rehabilitación eran rudimentarias y a menudo se basaban en enfoques ad hoc que variaban enormemente de una cultura a otra. Sin embargo, este tiempo también sirvió para asentar las bases de la comprensión de la necesidad de asistencia y métodos para mejorar la vida de las personas con discapacidades físicas o mentales.

La segunda fase podría corresponder al periodo entre guerras, entre la Primera Guerra Mundial y la Segunda Guerra Mundial. El dramático aumento en el número de soldados que regresaban del frente con discapacidades físicas y traumas psicológicos llevó a una necesidad urgente de desarrollar estrategias de readaptación médica. Durante este tiempo se hicieron avances significativos en técnicas ortopédicas, terapéuticas y quirúrgicas. Esta fase también vio el nacimiento de la rehabilitación profesional como una disciplina médica, con la creación de instituciones dedicadas y programas sistemáticos destinados a la reintegración de veteranos en la sociedad y en mercado laboral.

La tercera fase se resume en la fase contemporánea de la rehabilitación, caracterizada por la formalización y profesionalización del campo. Esta era está definida por la creación de políticas institucionales y el desarrollo de diversas profesiones dentro del ámbito de la rehabilitación, como la terapia física, la terapia ocupacional, la terapia del habla y la audiología, entre otras. Además, ha habido un creciente enfoque en los derechos de las personas con discapacidades, impulsando legislaciones que promueven la accesibilidad y la igualdad de oportunidades. La tecnología moderna, incluyendo la informática y la ingeniería biomédica, ha jugado un papel crucial en el desarrollo de nuevas

técnicas y herramientas para la rehabilitación, lo que ha permitido avances significativos en la calidad de la atención y los resultados de los pacientes.

Cada una de estas fases refleja un avance progresivo en cómo la sociedad y la comunidad médica entienden y abordan las necesidades de las personas con discapacidades, marcando hitos importantes en la evolución de la rehabilitación como un campo médico esencial y altamente especializado. (Wirotius, 1999)

2.1. Innovaciones Tecnológicas en la Rehabilitación.

Como se expone anteriormente, el aumento en el uso de tecnología moderna en el campo de la medicina, y más concretamente en la rehabilitación, ha producido un incremento en el desarrollo de nuevas técnicas, facilitando el proceso de recuperación de los pacientes.

El uso intensivo de la tecnología moderna en la rehabilitación no solo ha mejorado las técnicas existentes, sino que también ha revolucionado completamente le enfoque hacia el tratamiento y la integración de las personas con discapacidades.

A continuación se exponen varios ejemplos de implementación de tecnología moderna en la rehabilitación: (Toro, 2023) (Fisioform, 2023).

2.1.1. Realidad virtual y aumentada en rehabilitación.

La realidad virtual representa una tecnología avanzada que crea entornos completamente virtuales, donde los usuarios se sumergen en un espacio generado por computadora, experimentando una sensación de presencia física. Esta tecnología utiliza dispositivos especializados que permiten interacciones visuales y, en algunos casos, auditivas y táctiles, simulando una experiencia realista en un entorno controlado.

Por su parte, la realidad aumentada integra elementos virtuales con el mundo real, enriqueciendo la percepción del usuario. Mediante el uso de dispositivos como teléfonos inteligentes y gafas de realidad aumentada, la realidad

aumentada añade información visual y gráfica al entorno físico, mejorando la interacción del usuario con el entorno inmediato.

Estas dos tecnologías están revolucionando el campo de la rehabilitación mediante el desarrollo de programas que ofrecen entornos interactivos y adaptativos. La realidad virtual y la realidad aumentada facilitan la creación de ejercicios terapéuticos personalizados que pueden ser ajustados para mejorar aspectos específicos como el equilibrio, la coordinación en los movimientos y la fuerza muscular. Además, estas tecnologías tienen la capacidad de atenuar la percepción del dolor durante los ejercicios de rehabilitación, lo que puede hacer el proceso más cómodo y efectivo para el paciente.

Adicionalmente, la realidad virtual y realidad aumentada son de gran utilidad en la rehabilitación de pacientes con movilidad reducida o aquellos que deben permanecer encamados. Permiten realizar actividades simuladas que promueven el movimiento sin el dolor asociado a la actividad física convencional, evitando así el deterioro físico que podría resultar de largos períodos de inactividad.

El uso de la realidad virtual y la realidad aumentada en programas de rehabilitación no solo está cambiando la manera en que se implementan los tratamientos terapéuticos, sino que también está ampliando las posibilidades para la personalización del cuidado médico. Cada paciente puede recibir un tratamiento específicamente adaptado a sus necesidades, lo que maximiza las posibilidades de recuperación y mejora sustancialmente su calidad de vida.

2.1.2. Telefisioterapia.

Es un enfoque moderno en el tratamiento terapéutico que utiliza tecnologías de comunicación digital, como videollamadas y otras plataformas para la interacción a distancia, ofreciendo servicios de fisioterapia de forma remota. Esta tecnología permite al profesional realizar evaluaciones, proporcionar orientación y monitorear a los pacientes sin necesidad de contacto físico directo, facilitando así la rehabilitación desde cualquier lugar accesible para el paciente.

Aunque en la terapia muchas técnicas y tratamientos requieren contacto directo con el paciente para ser efectivos, en algunos casos podría no ser crucial. Por ello, este sistema de atención a distancia es especialmente beneficioso para pacientes que enfrentan dificultades de movilidad o viven en áreas con acceso limitado a servicios de rehabilitación especializados. Las videollamadas y plataformas en línea brindan la posibilidad de que los

pacientes reciban instrucciones claras y realicen sus ejercicios bajo la supervisión de un profesional, desde la seguridad y comodidad de sus hogares.

Las ventajas de la telefisioterapia incluyen una mayor accesibilidad a servicios especializados, conveniencia para los pacientes y la capacidad de realizar un seguimiento efectivo del progreso del paciente. Además, esta forma de fisioterapia facilita la comunicación fluida y constante entre el fisioterapeuta y el paciente, permitiendo conocer el estado y evolución de la terapia en todo momento, lo cual es esencial para adaptar los tratamientos a las necesidades cambiantes del paciente y mejorar los resultados terapéuticos. Estos beneficios hacen que la telefisioterapia sea una alternativa eficiente y práctica para la rehabilitación moderna.

La telefisioterapia emergió como una solución crucial durante la pandemia de COVID-19, para continuar ofreciendo servicios de rehabilitación mientras se cumplían las restricciones de distanciamiento social y confinamiento.

2.1.3. Exoesqueletos y dispositivos robóticos.

Los exoesqueletos para rehabilitación están revolucionando el tratamiento de individuos con discapacidades motoras o lesiones severas. Estos dispositivos mecánicos se adaptan externamente al cuerpo del paciente y son fundamentales para asistir en la ejecución de movimientos complejos, facilitando así la recuperación de la movilidad y las funciones físicas esenciales.

Los exoesqueletos se colocan sobre las partes afectadas del cuerpo, proporcionando el soporte estructural necesario para realizar actividades que implican movimientos sofisticados. Esto es particularmente beneficioso para pacientes cuyas capacidades motoras están comprometidas debido a lesiones o condiciones médicas.

El empleo de estos dispositivos en la fisioterapia se centra en fortalecer la musculatura, mejorar la coordinación y estabilizar el equilibrio del paciente. Además, son cruciales para fomentar la neuroplasticidad, es decir, la capacidad del cerebro de adaptarse y reorganizarse en respuestas a nuevas experiencias o daños neuronales.

Una de las mayores ventajas de los exoesqueletos es su capacidad para monitorear con precisión el avance del paciente mediante sensores integrados que registran datos de rendimiento en tiempo real. Esta funcionalidad permite a los terapeutas ajustar los tratamientos de forma precisa, garantizando que se adapten a las necesidades y progresos específicos de cada individuo.

Los exoesqueletos permiten que los pacientes realicen ejercicios terapéuticos de manera más exacta y segura, minimizando el riesgo de lesiones adicionales y maximizando los beneficios del tratamiento. El soporte que proporcionan estos dispositivos no solo mejora la efectividad de la rehabilitación, sino que también puede acelerar el proceso de recuperación, contribuyendo a una mejora significativa en la calidad de vida del paciente.

2.1.4. Biofeedback, aplicaciones móviles y sensores vestibles (wearables).

El biofeedbak es una técnica de retroalimentación biológica que ayuda a los usuarios a controlar funciones fisiológicas normalmente involuntarias, como las de contracciones musculares en la fisioterapia del suelo pélvico. Este método convierte las contracciones en señales auditivas o visuales, permitiendo tanto a los fisioterapeutas como a los pacientes observar y analizar la actividad muscular efectiva durante los ejercicios.

Los wearables, integrados en accesorios personales como relojes o pulseras, facilitan el monitoreo constante de indicadores de salud clave como la frecuencia cardíaca y saturación de oxígeno. Estos dispositivos proporcionan datos valiosos que permiten seguir de cerca el estado físico del usuario.

Ambas tecnologías, los wearables y el biofeedback, ofrecen a los fisioterapeutas herramientas poderosas para personalizar y ajustar los tratamientos según las respuestas y el avance de cada paciente. Además, el desarrollo de aplicaciones móviles de fisioterapia complementa estas tecnologías, permitiendo a los pacientes seguir rutinas de ejercicios específicas, recibir retroalimentación en tiempo real y monitorear su progreso de forma continua y motivadora.

La combinación de dispositivos vestibles, biofeedback y aplicaciones móviles están transformando la práctica de la fisioterapia en una más dinámica, interactiva y centrada en el paciente, optimizando los tratamientos y mejorando los resultados de rehabilitación.

2.1.5. Inteligencia artificial en el diagnóstico y pronóstico.

La inteligencia artificial se ha consolidado como uno de los avances tecnológicos más impactantes en tiempos recientes, y su aplicación en el ámbito de la fisioterapia está comenzando a mostrar beneficios considerables. La capacidad que tiene la inteligencia artificial para procesar y analizar grandes conjuntos de datos permite proporcionar diagnósticos más acertados y desarrollar pronósticos personalizados que son cruciales para la eficiencia del tratamiento. Esto contribuye no solo a una mejora en la precisión del diagnóstico, sino que también facilita una reducción en los tiempos de recuperación y mejora la toma de decisiones clínicas.

Es evidente que la fisioterapia está evolucionando rápidamente gracias a los avances tecnológicos, y la adopción de la inteligencia artificial es una muestra de cómo estos progresos pueden optimizar los resultados de los tratamientos. La aplicación de la inteligencia artificial está enriqueciendo la calidad del tratamiento y la experiencia del paciente, además de extender el alcance de la fisioterapia a comunidades más aisladas o individuos con restricciones de movilidad.

La incorporación de esta tecnología no solo mejora la precisión y la personalización de los tratamientos de fisioterapia, sino que también contribuye a hacer la atención más accesible y continua para todos los pacientes. Se anticipa que, en el futuro, la fisioterapia virtual y la integración de soluciones tecnológicas avanzadas serán elementos esenciales del cuidado fisioterapéutico. Estas innovaciones aseguran que los métodos de rehabilitación se vuelvan cada vez más eficaces y satisfactorios para los pacientes, marcando un avance significativo en la práctica fisioterapéutica tanto en el presente como en el futuro.

2.1.6. Terapia genética.

La terapia genética es un enfoque innovador que implica alterar los genes dentro de las células de un paciente para abordar diversas enfermedades. En el ámbito de la fisioterapia, esta técnica abre nuevas vías para el tratamiento de condiciones como enfermedades inflamatorias crónicas, afecciones musculoesqueléticas y lesiones del sistema nervioso.

Actualmente, se están desarrollando investigaciones sobre terapias genéticas que podrían ser cruciales para la reparación y regeneración de tejido muscular y nervioso dañado. Estos estudios están dirigidos a crear métodos genéticos que puedan mejorar efectivamente la recuperación y rehabilitación de los tejidos afectados. Por ejemplo, en la práctica fisioterapéutica, podrían emplearse vectores virales para introducir genes específicos en las células afectadas, con la finalidad de promover la regeneración o reducir la inflamación adversa.

La implementación de la terapia genética en el campo de la fisioterapia podría revolucionar el manejo de lesiones graves y enfermedades crónicas, mejorando significativamente los resultados para muchos pacientes. Este enfoque tiene el potencial de ofrecer soluciones más efectivas y duraderas que los métodos tradicionales, marcando un progreso significativo en la capacidad para tratar y manejar condiciones que hasta ahora han sido difíciles de abordar eficazmente.

2.1.7. Terapia basada en células.

Esta técnica se basa en la utilización de células extraídas del cuerpo del paciente para promover la curación. En el ámbito de la fisioterapia, esta estrategia se está investigando activamente para su aplicación en el tratamiento de lesiones musculares y articulaciones.

Actualmente, se están llevando a cabo investigaciones para evaluar la efectividad de terapias basadas en células, especialmente células madre, en la reparación y regeneración de tejidos musculares y articulares lesionados. Las células madre poseen la capacidad distintiva de transformarse en diversos tipos de células específicas, lo que se convierte en un recurso invaluable para la medicina regenerativa. Por ejemplo, existen estudios que exploran cómo las células madre pueden ser utilizadas para mejorar significativamente la recuperación de tejidos dañados, ofreciendo nuevas vías para tratamientos más eficientes y menos invasivos.

El uso de terapias basadas en células en fisioterapia ofrece un gran potencial, no solo para facilitar la curación de lesiones agudas, sino también para mejorar los procesos de recuperación en afecciones crónicas, reduciendo potencialmente la dependencia de cirugías o tratamientos farmacológicos a largo plazo. Este enfoque regenerativo hacia la rehabilitación de lesiones musculares y articulares marca un avance prometedor, abriendo nuevas posibilidades para un tratamiento más natural y eficaz.

2.2. Transformación tecnológica en la fisioterapia.

El campo de la fisioterapia está experimentando una transformación radical gracias a la integración de tecnologías avanzadas, que están redefiniendo las posibilidades de tratamiento y rehabilitación para pacientes con diversas condiciones médicas. Estas tecnologías no solo están cambiando la forma en que se implementan los tratamientos de fisioterapia, sino que también están mejorando significativamente los resultados de los pacientes. Mirando hacia el futuro, es evidente que la integración de estas innovaciones se convertirá en un componente estándar del cuidado fisioterapéutico, promoviendo tratamientos más efectivos, accesibles y personalizados que mejoren sustancialmente la calidad de vida de los pacientes. En resumen, la fisioterapia está en camino a una era de mayor precisión, eficacia y alcance gracias a la continua evolución de la tecnología.

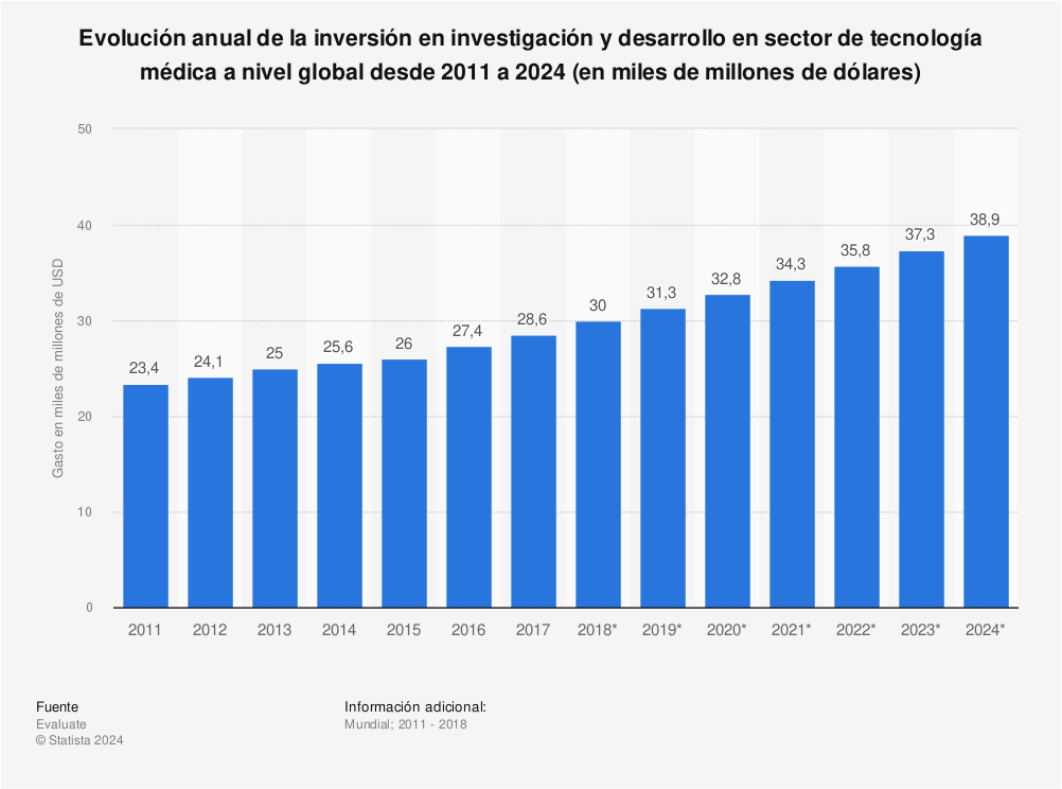

Figura 2: Evolución anual de la inversión en investigación y desarrollo en sector de tecnología médica a nivel global desde 2011 a 2024. Fuente: (EvaluateMedTech, 2018).

En esta estadística de EvaluateMedTech se indica la evolución de la inversión en investigación y desarrollo en el sector de la tecnología médica a nivel global desde 2011 a 2024 a nivel global, medida en millones de dólares. La inversión global en investigación y desarrollo en el sector de la tecnología médica ha mostrado una tendencia ascendente clara y consistente. El incremento anual en la financiación refleja el creciente reconocimiento de la importancia de la innovación en el campo médico.

Este patrón de crecimiento refleja no solo la evolución de las necesidades sanitarias globales sino también una mayor confianza en el potencial retorno económico y beneficio social que las innovaciones médicas pueden ofrecer. La tendencia refleja que la inversión en I+D no muestra signos de disminuir, señalando un compromiso continuo y fortalecido hacia la mejora expansión de la tecnología médica a nivel mundial.

2.3. Realidad Aumentada.

La realidad aumentada se encuentra dentro del ámbito de a tecnología de la realidad extendida, se distinguen tres conceptos fundamentales: La realidad aumentada, la realidad mixta y la realidad virtual.

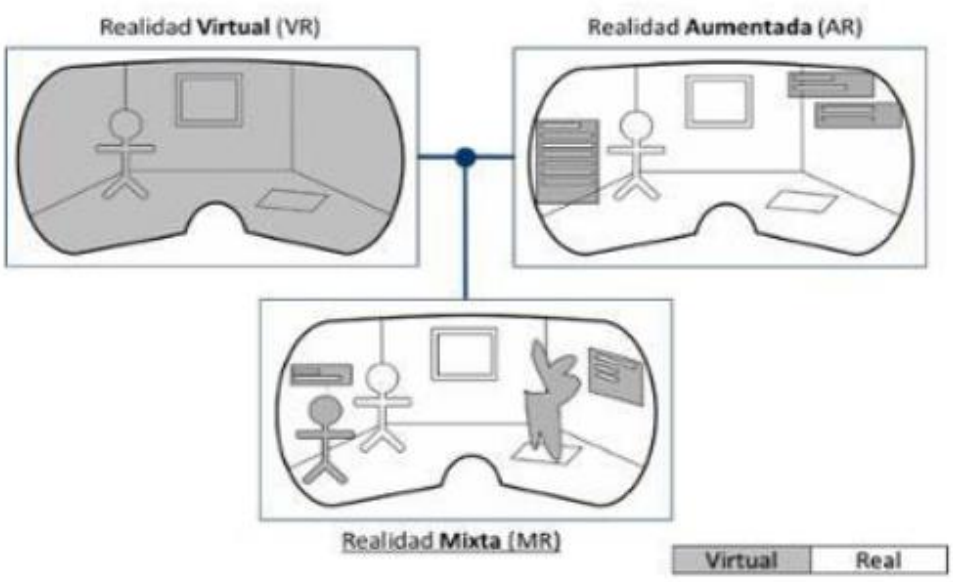

Figura 3: Diferencias entre realidad virtual, realidad aumentada y realidad mixta. Fuente: (Ordoñez, 2020).

Como se observa en la Figura 3, dentro del espectro donde se encuentran estas tecnologías de la realidad extendida, se podría decir que la realidad aumentada se sitúa en un extremo del espectro, donde la información digital se superpone al mundo real, mientras que la realidad virtual se encuentra en el extremo opuesto, donde el usuario es completamente inmerso en un entorno simulado. En el medio se encuentra la realidad mixta, que combina elementos de ambos, permitiendo la interacción con objetos digitales dentro del entorno físico. Esta disposición muestra cómo estas tecnologías ofrecen diferentes grados de inmersión y experiencias, cada una con sus propias aplicaciones y ventajas.

La realidad aumentada se refiere a un conjunto de tecnologías que posibilitan la visualización de una parte del entorno físico a través de dispositivos tecnológicos, mientras se superpone información gráfica generada por estos dispositivos. En esencia, la realidad aumentada integra elementos virtuales con la realidad existente, permitiendo que elementos digitales aparezcan y se interactúe con ellos en el mundo real. Esta tecnología utiliza dispositivos especializados para superponer información virtual, creando así una experiencia en la que elementos virtuales se integran de manera coherente con el entorno físico.

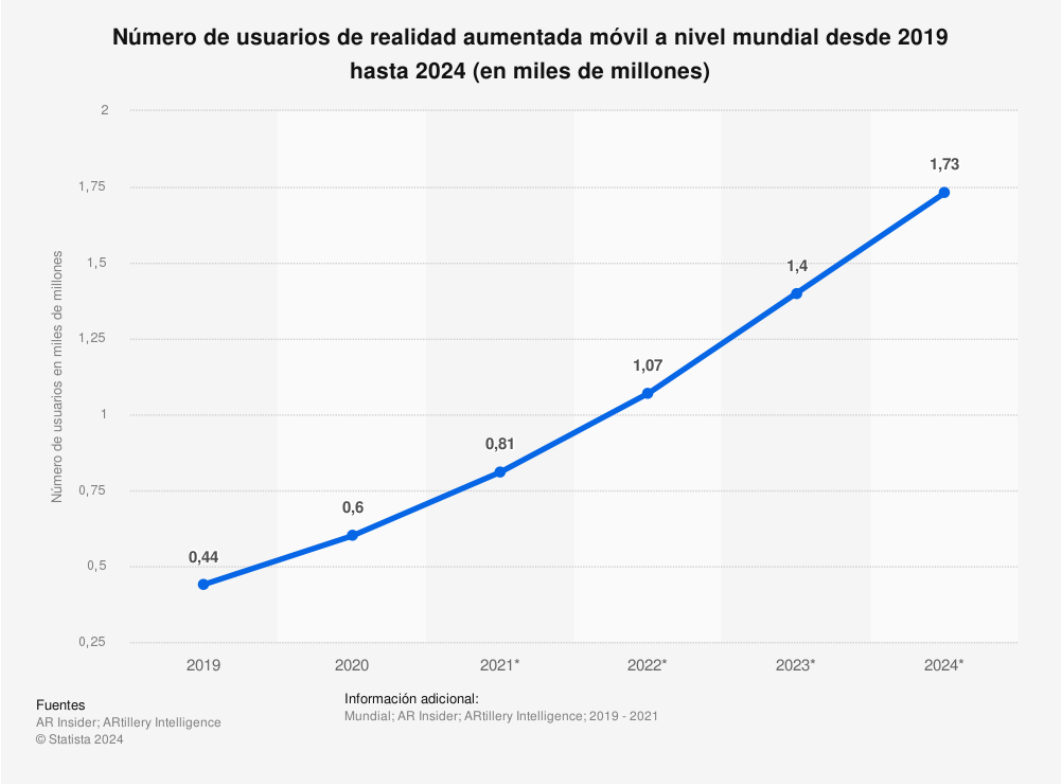

Figura 4: Número de usuarios de realidad aumentada móvil a nivel mundial desde 2019 hasta 2024. Fuente: (Insider & Intelligence, 2024).

Observando la Figura 2, donde se muestra un aumento en la inversión de tecnología médica a nivel mundial, y la Figura 4, que indica un aumento del número de usuarios de realidad aumentada móvil a nivel mundial, es razonable inferir que existe un crecimiento en el uso de realidad aumentada en la medicina. Esta correlación sugiere que la realidad aumentada está siendo cada vez más adoptada en el campo de la medicina, posiblemente debido a sus beneficios en la mejora de la atención médica, el diagnóstico, la formación médica y la rehabilitación, entre otros aspectos.

2.4. Influencia de los videojuegos en el ámbito de la medicina.

Según el diccionario de la Real Academia Española un videojuego es un "dispositivo electrónico que permite, mediante mandos apropiados, simular juegos en las pantallas de un televisor, una computadora u otro dispositivo electrónico" (RAE, 2024). Aunque la Real Academia Española pueda definir los videojuegos principalmente como medios de entretenimiento, esta visión puede ser considerada restrictiva dado el amplio rango de usos y beneficios que los videojuegos han demostrado tener más allá del simple ocio.

El uso de videojuego para propósitos beneficiosos, como herramientas para promover salud y prevenir riesgos, está aumentando. A medida que avanza el campo de los "juegos serios", juegos que se definen como aquellos cuyo principal objetivo no es simplemente el entretenimiento, emergiendo así nuevas oportunidades y un creciente interés por diseñar y utilizar videojuegos que tengan un impacto positivo en la salud.

Los elementos fundamentales de los videojuegos, que incluyen reglas definidas, metas claras, y en muchos casos, sistemas de puntos y niveles, tienen el potencial de ser utilizados para fomentar resultados de salud positivos. En el ámbito de los juegos serios, se persigue el objetivo de emplear metodologías científicamente validadas, tales como contenidos que se fundamentan en teorías específicas y la realización de ensayos clínicos controlados y aleatorizados. Estos enfoques se destinan a desarrollar y evaluar la efectividad de los juegos, y también para aprovechar la capacidad de estos de generar datos detallados dentro de entornos simulados de juego, los cuales imitan comportamientos humanos reales. (Lynn E. Fiellin, 2014).

Las características intrínsecas de los videojuegos, como tener reglas claras, objetivos definidos, y sistemas de puntos y niveles, contribuyen significativamente a la mejora de adherencia y el compromiso de los pacientes con sus tratamientos de rehabilitación. A continuación, se detallan algunas

formas en que estas características impactan positivamente en la participación del paciente:

1. Objetivos claros y orientados a metas: Los videojuegos estructurados con objetivo específicos proporcionan a los pacientes metas claras y alcanzables, lo que fomenta un sentido de propósito y dirección del tratamiento. Esta estructura orientada a objetos ayuda a los pacientes a visualizar su progreso y éxito, incrementando su motivación y compromiso.

2. Reglas definidas: Las reglas en los videojuegos crean un entorno predecible y controlado en el que los pacientes pueden operar. Esto reduce la ansiedad y la incertidumbre al proporcionar un conjunto claro de expectativas y limitaciones, lo que facilita que los pacientes comprendan lo que se espera de ellos y cómo deben proceder en sus terapias.

3. Sistemas de puntos y niveles: Introduce un elemento de recompensa y reconocimiento que es crucial para la motivación del paciente. Estos sistemas recompensan el progreso, celebran logros y, a menudo, permiten que los pacientes superen sus propios récords personales. Además, estos videojuegos al ofrecer sistemas de puntuación permiten al terapeuta controlar la evolución del paciente de forma más efectiva. Es posible recopilar datos precisos y detallados sobre el desempeño y el progreso del paciente durante el tratamiento.

4. Feedback inmediato: Los videojuegos proporcionan retroalimentación instantánea sobre las acciones y decisiones del jugador. En un contexto terapéutico, esto permite a los pacientes ajustar rápidamente sus acciones, aprendiendo y mejorando en tiempo real. Este feedback continuo ayuda a los pacientes a entender mejor cómo sus acciones afectan su recuperación, lo cual es esencial para el aprendizaje y la mejora de la adherencia del tratamiento.

5. Entornos atractivos y simulados: Los videojuegos crean entornos ricos y estimulantes que pueden hacer que la experiencia de tratamiento sea más agradable y menos tediosa. Al sumergir a los pacientes en un entorno interativo y atractivo, los videojuegos pueden hacer que el tiempo dedicado a la terapia pase más rápido y sea más gratificante, lo que ayuda a mantener a los pacientes comprometidos en tratamientos de largo plazo.

2.5. Rehabilitación del miembro superior y uso de Realidad Aumentada.

La rehabilitación, según el diccionario de la Real Academia Española, es el "conjunto de métodos que tiene por finalidad la recuperación de una actividad o función perdida o disminuida por traumatismo o enfermedad". (RAE, 2024).

El miembro superior se compone por cuatro diferentes regiones: el hombro, el brazo, el antebrazo y la mano.

· El hombro es una articulación compleja que conecta el brazo con el tronco. Facilita la amplitud de movimiento del brazo en múltiples direcciones. Participa en la elevación y descenso del brazo.

· El brazo facilita la flexión y extensión del codo, permitiendo actividades como doblar y estirar el brazo.

· El antebrazo permite la pronación y supinación. Además de facilitar la flexión y extensión de la muñeca, lo que es esencial para realizar actividades de la vida cotidiana.

· La mano proporciona destreza y presión a la hora de la manipulación de objetos. Permitiendo la presión y liberación de los objetos, incluyendo agarres en pinza y agarres de presión.

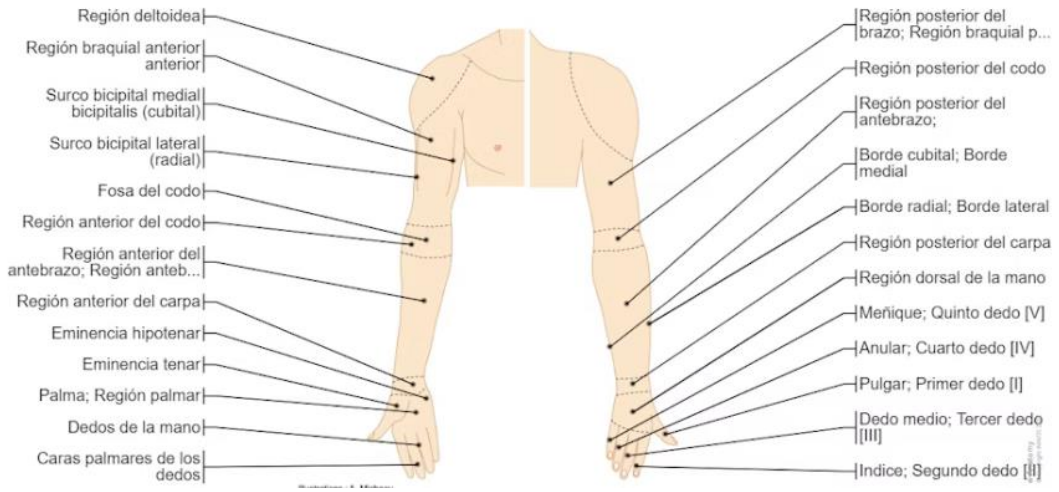

Figura 5: Miembro superior. Fuente: https://www.imaios.com/es/e-anatomy/miembro-superior/miembro-superior

Después de recibir un daño en el sistema nervioso, se produce un fenómeno conocido como neuroplasticidad, que se refiere a la capacidad inherente del

cerebro y la médula espinal para reorganizarse y establecer nuevas conexiones neuronales. El organismo, a menudo, activa procesos de recuperación que reestablecen ciertas funciones después de la lesión. Tras esta primera etapa, las mejoras suelen darse debido a la capacidad del sistema nervioso para adaptarse y cambiar en respuesta a estímulos internos y externos. (Díez, 2020)

Estos estímulos internos y externos provienen de la propia actividad física, la terapia ocupacional, la fisioterapia, la estimulación eléctrica, entre otros métodos utilizados en el proceso de rehabilitación. La idea es que, al exponer al sistema nervioso a estos estímulos de manera repetida y sistemática, se promueva la reorganización neuronal y se facilite la recuperación de las funciones afectadas.

Como se ha dicho anteriormente, el uso exclusivo de terapias de rehabilitación convencionales puede llevar al paciente a una pérdida de interés, y, por tanto, a una posible deceleración en el proceso. Por ello se proponen la integración de diferentes tecnologías. Según (M. Khademi, 2012), entre las nuevas tecnologías utilizadas en el campo de la medicina, el interés en la investigación de aplicaciones de realidad aumentada en Ingeniería de Rehabilitación y Tecnología Asistencial está creciendo, demostrándose que estos sistemas pueden mejorar enormemente la calidad de vida de personas con discapacidades.

En un estudio realizado en, (Hossein Mousavi Hondori, 2016), donde un total de 18 pacientes, con una edad media de 57 años, una media de 70 meses postaccidente cerebrovascular, y con déficits motores de leves a moderados, tenían que jugar a un juego basado en *Fruit Ninja* en un entorno PC y en un entorno informático con realidad aumentada. Este estudio concluyó que los pacientes que utilizaron la versión del juego en realidad aumentada, en comparación con la versión en PC, obtuvieron puntuaciones más altas, movimientos más rápidos y consistentes.

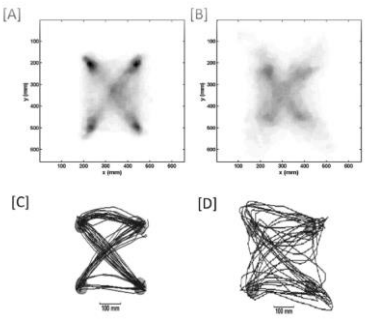

Figura 6: Distribución en la posición de la mano. [A] muestra los objetivos alcanzados con realidad aumentada. [B] muestra los objetivos alcanzados con versión PC. [C] muestra el movimiento con realidad aumentada. [D] muestra el movimiento en la versión PC. Fuente: (Hossein Mousavi Hondori, 2016).

En la Figura 6 se puede observar cómo en [A], los pacientes que utilizaron la versión en realidad aumentada alcanzaron de forma más precisa los objetivos que los pacientes en la gráfica [B], que utilizaron la versión en PC. Al igual que en la gráfica [C], que representa el movimiento del brazo de los pacientes que utilizaron la versión en realidad aumentada, que comparada con la gráfica [D], que representa el mismo movimiento, pero en este caso en la versión PC, los pacientes que utilizaron la versión en realidad aumentada realizan movimientos más precisos.

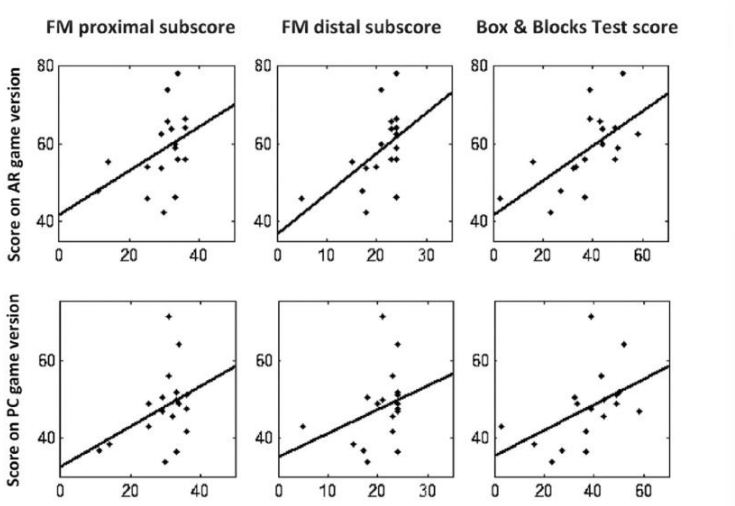

Figura 7: Diferencias en la puntuación obtenida. Fuente: (Hossein Mousavi Hondori, 2016).

En el caso de la puntuación también se puede apreciar como los pacientes que utilizaron la versión del juego en realidad aumentada obtienen puntuaciones más altas, debido a que estos alcanzaban los puntos de interés en menor tiempo.

3. HERRAMIENTAS PARA EL DESARROLLO DEL JUEGO SERIO.

La rehabilitación de un paciente suele dividirse en tres fases distintas, cada una con sus propios objetivos y desafíos específicos. En la primera fase, el paciente generalmente experimenta una movilidad muy limitada, y el principal objetivo es incrementar progresivamente su rango de movimiento. En la segunda fase, el paciente ha logrado cierta mejora en su movilidad, pero aún enfrenta dificultades para realizar movimientos precisos y coordinados. Finalmente, en la tercera fase, el paciente enfrenta la falta de motricidad fina, lo que implica la necesidad de trabajar en la mejora de habilidades más detalladas y específicas. Estas etapas proporcionan una guía estructurada para el proceso de rehabilitación, adaptándose a las necesidades y capacidades cambiantes del paciente a lo largo de su recuperación.

El enfoque en el desarrollo de un juego serio con realidad aumentada se sitúa en la segunda fase del proceso de rehabilitación, cuando el paciente ha avanzado lo suficiente como para tener cierta mejoría en su movilidad, pero aún enfrenta dificultades con movimientos imprecisos. En este punto, el desafío principal radica en la capacidad del paciente para coordinar y disociar los movimientos de las articulaciones, lo que requiere intervenciones específicas y adaptadas a sus necesidades individuales. El uso de realidad aumentada en esta etapa busca proporcionar un entorno interactivo y motivador que fomente el desarrollo de habilidades motoras más precisas y coordinadas.

3.1. M3Display.

M3Display es un sistema de realidad mixta, desarrollado por el Instituto de las Tecnologías Avanzadas de la Producción (ITAP), concretamente en la línea de investigación de robótica médica, dentro de la Universidad de Valladolid. Este sistema utiliza tecnología de realidad aumentada para proporcionar una experiencia interactiva y personalizada durante las sesiones de terapia. Los juegos incorporados en M3Display están diseñados para mejorar la movilidad, coordinación y fuerza del miembro superior, adaptándose a las necesidades y capacidades individuales de cada paciente.

3.1.1. Estructura Hardware.

En el Instituto de las Tecnologías Avanzadas de la Producción se ha optado por diseñar una estructura no inmersiva basada en la proyección de las imágenes sobre un monitor, en lugar de unas gafas de realidad virtual, debido a su sencillez, y a que los entornos inmersivos que utilizan gafas no tienen una gran aceptación por algunos pacientes, especialmente los de mayor edad.

Esta estructura se compone de una cámara Logitech HD Pro C920 (resolución de 1920 x 1080 a 30 fps y campo visual diagonal de 78°), un ordenador Intel Core i5 11400 CPU (6 núcleso, 2,6GHz, 16GB de RAM) y una pantalla de 27" (Phillips 273V7QDSB, resolución 1920 x 1080, @75 Hz). El procedimiento implicará que el paciente se siente en una silla con los brazos colocados sobre una mesa. En la mesa, se dispondrá de un soporte para la pantalla que contará con una cámara integrada, la cual estará enfocada hacia un espejo situado detrás de la pantalla. El reflejo de los brazos del paciente en el espejo se proyectará en tiempo real en la pantalla, donde se supondrán los elementos virtuales proporcionados mediante la realidad aumentada, con el fin de cumplir con los requisitos establecidos por los juegos serios.

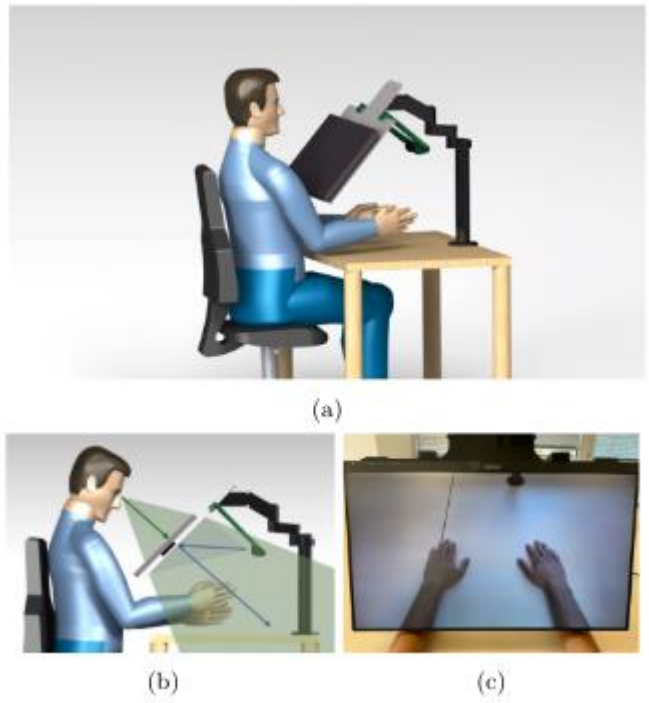

Figura 8: Estructura M3Display. (a) Estructura. (b) Reflexión de la imagen sobre el espejo. (c) Punto de vista del paciente. Fuente: (A. Cisnal, 2023).

3.1.2. Detección de la mano.

La detección de la mano es fundamental para el juego serio en realidad aumentada, ya que permite una interacción más precisa y natural entre el usuario y el entorno virtual. Al reconocer y seguir los movimientos de las manos, el sistema puede interpretar los gestos del usuario y responder en consecuencia, lo que mejora la inmersión y la experiencia de juego. Además, la detección de las manos facilita el control y la manipulación de objetos virtuales en el espacio, lo que resulta esencial para la efectividad de la rehabilitación y el cumplimiento de los objetivos terapéuticos.

Con el sistema M3Display, la detección de las manos se llevará a cabo mediante MediaPipe Hands (Team M. , 2021). Esto garantiza una identificación precisa de las manos del usuario, lo que es crucial para una interacción fluida y efectiva con los elementos virtuales del juego.

MediaPipe Hands representa una herramienta esencial en el reconocimiento y seguimiento preciso de las manos, lo cual es esencial para garantizar una interacción natural y fluida en los entornos de realidad aumentada y virtual. Este sistema se basa en un modelo de inteligencia artificial y se compone de dos elementos interrelacionados: un detector de palmas que opera de un marco delimitador específico para manos y un modelo de puntos de referencia de manos que trabaja en conjunto con el detector de palmas para proporcionar información detallada sobre la posición y orientación de las manos en un espacio tridimensional, devolviendo puntos de referencia 2.5D de alta fidelidad. Este enfoque de seguimiento de manos en dos etapas es altamente eficiente y puede gestionar múltiples manos simultáneamente en tiempo real en dispositivos móviles. Utilizando únicamente información de color RGB, el sistema puede prever con precisión la posición tridimensional y predecir la postura de manos 2.5D. Además, MediaPipe Hands está disponible como una solución de código abierto, lo que facilita su integraciónen una variedad de plataformas, incluyendo Android, iOS y Tensorflow, entre otras. (Fan Zhang, 2020).

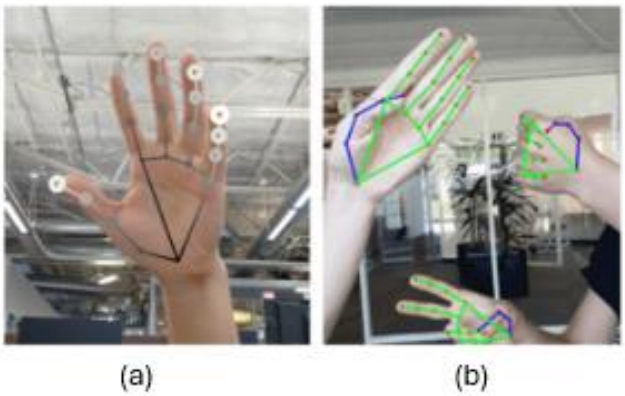

Figura 9: (a) Puntos de referencia de las manos con profundidad relativa presentada en diferentes tonos. (b) Seguimiento en tiempo real de múltiples manos en Pixel 3. Fuente: (Fan Zhang, 2020).

3.1.3. Procesamiento de datos de la mano.

En el contexto del desarrollo del sistema de seguimiento de manos, el objetivo principal de utilizar MediaPipe Hands radica en la obtención de puntos de referencia precisos. Estos puntos actúan como la base para llevar a cabo un análisis de los datos recolectados, con el objetivo de generar un esqueleto virtual que represente de manera fiel la estructura y los movimientos de la mano en tiempo real. Este proceso es esencial, ya que permite transformar la información bruta obtenida de la detección de manos en una representación más organizada y significativa. De esta manera, se facilita su integración y aplicación en sistemas de realidad aumentada y virtual, mejorando así la experiencia del usuario con una mayor precisión y realismo.

Para el procesamiento de los datos, para definir el esqueleto de la mano, primero se realizan una serie de operaciones lógicas y morfológicas mediante la dilatación con kernels cuadrados de tamaño considerable, buscando una primera delimitación de la segmentación final. Luego, se procede a realizar un muestreo del espectro de colores de puntos cercanos al esqueleto de la mano para establecer los umbrales de discriminación en el espacio de representación CIELAB, que permite discernir entre los puntos correspondientes de la mano y aquellos que forman parte del fondo. (A. Cisnal, 2023).

Esta combinación de técnicas de procesamiento de imágenes y análisis de color permite una segmentación efectiva de la mano y el fondo. En este sentido, el procesamiento de los datos desempeña un papel crucial para facilitar una

interacción fluida y precisa del paciente con los elementos virtuales presentes en los juegos serios de realidad aumentada.

Figura 10: Procesamiento de datos para la detección de la mano. Fuente: (A. Cisnal, 2023).

3.1.4. Interfaz de selección de juegos.

La implementación de una interfaz visual para los terapeutas, mediante la cual puedan seleccionar los juegos disponibles y registrar la información de cada paciente, desempeña un papel fundamental en el contexto de la rehabilitación con realidad aumentada. Este componente adquiere una relevancia destacada al considerar la complejidad y diversidad de las actividades terapéuticas, así como la necesidad, así como la necesidad de personalizar el tratamiento para cada individuo. La creación de una interfaz intuitiva y eficiente no solo facilita la navegación y la selección de juegos por parte del profesional de la salud, sino que también optimiza la gestión y el seguimiento de la evolución de cada paciente. Además, esta herramienta puede contribuir significativamente a mejorar la calidad y la eficiencia de las sesiones de rehabilitación, al tiempo que ofrece una experiencia más fluida y personalizada tanto para los terapeutas como para los pacientes.

El Plan España Digital 2025, de la agenda 2030 del Gobierno de España, busca la implementación de este tipo de tecnologías en el sector sanitario, con el objetivo principal de mejorar la eficiencia y calidad de la atención médica mediante la agilización de los sistemas de información y promoción de la compartición segura de datos, lo que también permitirá personalizar los servicios de salud ofrecidos. (Gobierno de España, 2020)

M3Display cuenta con una selección de juegos serios de realidad aumentada, y se pueden visualizar en la interfaz de usuario del sistema. Estos juegos están integrados en la interfaz para simplificar su acceso y facilitar la interacción entre diferentes ejecuciones de este. A través del uso de perfiles de usuario y registros de sesiones anteriores, se logra una adaptación personalizada para

cada paciente, ajustando la dificultada de los juegos según su rendimiento. (A. Cisnal, 2023).

Figura 11: Interfaz M3Display para la selección de juego y registros de usuario. Fuente: (A. Cisnal, 2023).

3.1.5. Juegos serios para la rehabilitación.

La línea de trabajo entre el laboratorio de robótica médica de ITAP con el Instituto de Rehabilitación Funcional de La Salle consiste en la realización de una serie de juegos serios empleando realidad aumentada. Cada uno de estos juegos está diseñado con un propósito específico, centrándose en la rehabilitación del miembro superior. Aunque varían en sus enfoques y movilidad de esta área del cuerpo.

En esta propuesta de juegos serios para la rehabilitación del miembro superior, uno de los juegos desafía al paciente a atrapar un ratón virtual, lo que implica medir la distancia entre el paciente y el ratón, así como evaluar sus reflejos y movimientos. Otro juego simula la disposición de las horas de un reloj, permitiendo al paciente realizar movimientos verticales, horizontales y diagonales con la mano. En este juego, para pacientes con dificultades cognitivas debido a una discapacidad neuromotora, si la persona presenta dificultades para prestar atención a una parte específica del reloj, el juego centrará el número de objetivos de este en esa parte para fortalecer estas capacidades cognitivas. Otro de los juegos desafiará al paciente a trazar diferentes trayectorias, variando en dificultad, dentro de un tiempo determinado (Plaza, 2023). Con esta variedad de juegos y enfoques, el objetivo

es proporcionar una experiencia de rehabilitación integral y efectiva para cada paciente.

En esta colección de juegos que ofrece el sistema M3Display es crucial la retroalimentación visual y auditiva, junto con una evaluación continua del desempeño del paciente en los juegos de rehabilitación, resulta fundamental para mantener la alta motivación y compromiso del paciente.

Definiendo unos objetivos específicos en cada partida, ya sea porque el paciente deba obtener una puntuación determinada, o que este complete con éxito una acción en un cierto tiempo, se fomenta una motivación para que la persona trate de esforzarse por mejorar a lo largo de su periodo de terapia, contribuyendo significativamente a la obtención de mejoras en los resultados.

3.1.5.1. Motor Unity.

La colección de juegos serios para la rehabilitación, además de utilizar la misma segmentación de la mano del paciente, están desarrollados con el motor de videojuego multiplataforma Unity.

Unity, desarrollado por Unity Technologies en 2005, es un motor de videojuego multiplataforma. En el ámbito de la creación de videojuegos y experiencias interactivas, Unity emerge como una herramienta fundamental. Este motor ha revolucionado la forma en que se construyen y distribuyen aplicaciones digitales. Unity brinda la versatilidad para la creación de juegos en 2D y 3D, así como su influencia en diversas industrias más allá del entretenimiento. Como expresó David Helgason, CEO de Unity, Unity “es un conjunto de herramientas utilizado para construir juegos, y es la tecnología que ejecuta los gráficos, el audio, la física, las interacciones y la red”.

Desde sus inicios, el crecimiento de este motor se ve reflejado gracias a la disponibilidad de una versión gratuita, la creación de una comunidad activa de desarrolladores, la introducción del *Asset Store*, su flexibilidad multiplataforma y su compromiso con la innovación continua. Este modelo, con licencia gratuita, permitió duplicar su base de usuarios. La comunidad ha facilitado el aprendizaje y la colaboración, mientras que el *Asset Store* ha ahorrado tiempo y esfuerzo a los desarrolladores. Su capacidad multiplataforma ha ampliado su alcance, y las actualizaciones constantes han mantenido a Unity a la vanguardia de la tecnología de desarrollo de juegos y aplicaciones interactivas. Marc Whitten, responsable de Unity Create dijo: “Ya sea por el crecimiento en estudios establecidos o por aficionados que dan el salto al desarrollo de juegos como un trabajo a tiempo completo, me asombra su pura creatividad al dar

vida a sus visiones. Estamos orgullosos de ofrecer herramientas que ayudan a los creadores de todo el mundo a hacer grandes juegos".

Unity tiene la mayor participación en el mercado de motores. El 38% de los desarrolladores de juegos que utilizan motores para desarrollar videojuegos utilizan Unity como su motor principal.

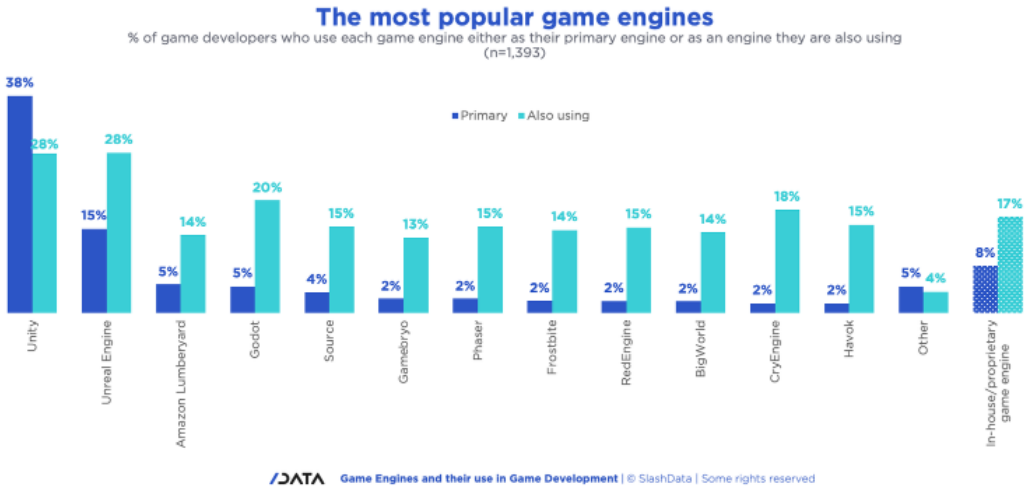

Figura 12: Motores de desarrollo de videojuegos más populares. Fuente: (Team S. , 2022).

Según *The Unity Gaming Report 2022*, publicada por Unity, en 2021 el número de juegos hechos con Unity aumentó en un 93% y el número de creadores aumentó en un 31%. Este aumento, a diferencia de otros sectores, fue favorable en parte gracias a la pandemia de COVID-19. Estos datos sugieren una tendencia ascendente significativa en el uso de Unity. Dados estos incrementos, es razonable prever que esta tendencia continúe en aumento, a medida que más desarrolladores reconozcan las ventajas y flexibilidad que ofrece Unity como motor de desarrollo de videojuegos.

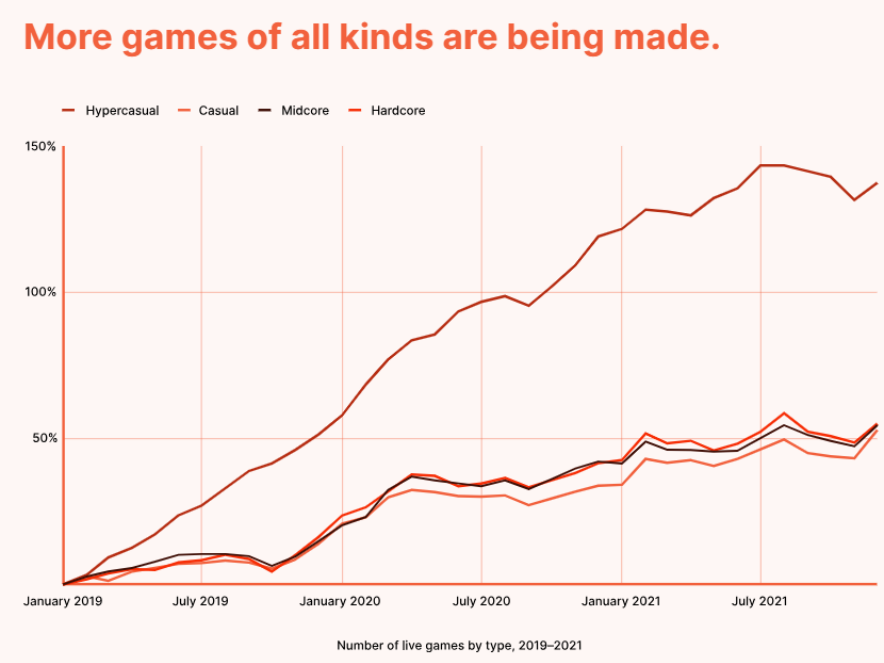

Figura 13: Aumento en el número de videojuegos desarrollados con el uso del motor Unity. Fuente: (Report, 2022).

Unity es ampliamente reconocido como uno de los motores gráficos más populares en la industria del desarrollo de videojuegos, habiendo ganado una reputación notable por su versatilidad y facilidad de uso. A lo largo de los años, Unity ha sido utilizado para crear una amplia variedad de videojuegos, desde simples juegos móviles hasta títulos de gran envergadura para consolas y PC. Unos ejemplos de videojuegos desarrollados con el motor Unity serían el Pokémon GO, Subnautica, Among Us, Fall Guys, entre otros.

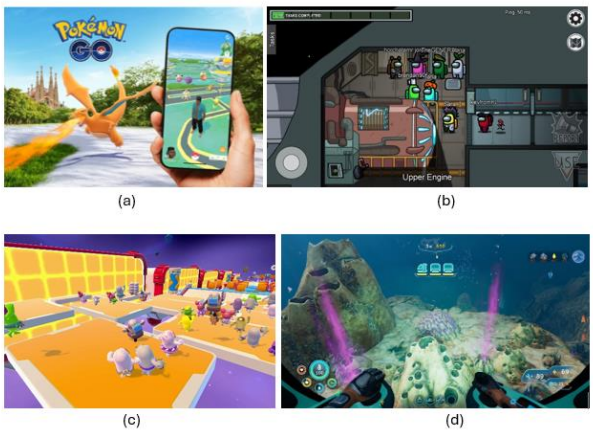

Figura 14: Gameplay de varios videojuegos desarrollados con Unity. (a) Pokémon GO. (b) Among Us. (c) Fall Guys. (d) Subnautica. Fuente: (Google Images).

A pesar de que el motor Unity mayoritariamente se utiliza en la industria del entretenimiento para el desarrollo de videojuegos, también se pueden crear experiencias 3D en tiempo real. Meta Slap, desarrollada por LG U+, es una oficina virtual, en la que es posible interactuar con los compañeros de trabajo. A raíz de la pandemia de COVID-19, donde la gran mayoría de la población mundial vio limitada las interacciones cara a cara con el resto del mundo, LG U+ decidió crear esta oficina virtual para simular la sensación de estar en el mismo espacio de trabajo que el resto de los compañeros, mientras cada uno está realizando teletrabajo en sus respectivos hogares, y así hacer más amena y plácida la jornada laboral. Hyun -woo Lee, jefe de proyecto de Meta Slap en LG U+ dijó: “Creo que la era de Web 3.0 va a ser tridimensional, y Unity es la base de los motores 3D. En el futuro, se espera que los casos de uso de Unity se expandan, no solo a computadoras y teléfonos inteligentes de mayor rendimiento, sino también a servicios cotidianos que naturalmente ampliarán la experiencia del mundo real a medida que los dispositivos experienciales se vuelvan más comunes”. (Park, 2023).

Figura 15: Oficina virtual de Meta Slap desarrollada por LG U+. Fuente: (Park, 2023).

4. ESTRUCTURA DEL JUEGO.

El desarrollo de *Coins* se encuentra dentro de una colección de juegos serios para la rehabilitación del miembro superior. Estos juegos serios se encuentran dentro del sistema M3Display, sistema desarrollado por la línea de investigación de robótica médica del Instituto de las Tecnologías Avanzadas de la Producción (ITAP) para el Instituto de Rehabilitación Funcional (IRF) La Salle.

4.1. Concepto General.

Con el propósito de obtener una mejora en la rehabilitación del miembro superior de los pacientes, en este proyecto entre ITAP e IRF, se propone el desarrollo de una serie de juegos basados en la realidad aumentada, con sistemas de puntuación y de dificultad, para tratar de conseguir una mejora en la adherencia y motivación en el proceso de rehabilitación. Cada uno de estos juegos serios basa su jugabilidad en potenciar un aspecto específico dentro del proceso de rehabilitación del paciente.

La rehabilitación de un paciente cuenta con tres fases. En la primera fase el paciente tiene una movilidad muy limitada y el objetivo es que el paciente aumente su rango de movimiento. En la segunda fase el paciente cuenta con cierta movilidad, pero realiza movimientos imprecisos. En la tercera fase el paciente no tiene la motricidad fina. El juego serio *Coins* se desarrolla con un enfoque hacia la segunda fase, ya que el paciente no es capaz de disociar los movimientos de las articulaciones.

Este juego consiste en retar al paciente a dibujar diferentes trayectorias con el brazo afectado en un tiempo determinado. Estas trayectorias pueden variar en dificultad dependiendo de la evolución del paciente en el proceso de terapia.

El desarrollo del videojuego se llevó a cabo en Unity, un potente motor que permite la creación y ejecución de contenido 3D en tiempo real. En este entorno, se programaron los scripts necesarios para controlar todos los aspectos del juego, desde la navegación por los menús hasta las funciones durante la partida, garantizando que tanto el paciente como el médico puedan visualizar la puntuación y el tiempo restante durante la experiencia de juego. Además, se generan archivos tipo .CSV que registran la fecha y hora de inicio de cada sesión de juego, así como las coordenadas de la posición de la mano en cada instante. Estos datos permiten la creación de gráficas que representan el movimiento de la mano, facilitando la comparación del resultado con la

trayectoria deseada y proporcionando información valiosa para el seguimiento y la evaluación del progreso del paciente en su rehabilitación.

4.2. Implementación M3Display.

Para poder desarrollar un juego basado en la realidad aumentada, donde el paciente tenga que seguir las diferentes trayectorias proyectadas sobre el entorno, y así cumplir los objetivos, se necesitará un sistema de visión artificial para la detección de la mano. En ITAP se ha desarrollado el sistema M3Dispaly, que cuenta con el entorno M3D, donde se incluye la detección de la cámara que se utilizará en el juego y la detección mediante visión artificial de la mano utilizando Python. Este entorno fue desarrollado por Guillermo Sánchez Brizuela. (Brizuela, 2021).

4.2.1. Instalaciones necesarias.

Para el desarrollo de este juego ha habido que instalar dos paquetes de software: Unity y Python.

4.2.1.1. Unity.

Debido a su versatilidad a la hora de desarrollar aplicaciones para experiencias en 3D, se utilizará el motor Unity. Se deberá instalar la versión 2020.3.0f1, además de diferentes paquetes.

Se deberá instalar la versión 2.1.1 de *Naughty Attributes*. *Naughty Attributes* es una extensión para el Inspector de Unity. Amplía el rango de atributos que Unity proporciona para que se puedan crear potentes inspectores sin necesidad de editores personalizados o cajones de propiedades. También proporciona atributos que pueden aplicarse a diferentes campos.

Para la gestión de paquetes dentro del entorno de desarrollo de Unity se necesita *NuGetForUnity*. Permite acceder a la amplia variedad de paquetes disponibles de *NuGet* directamente desde el Editor de Unity. *NuGetForUnity* facilita la adquisición e integración de la biblioteca *NetMQ*, implementación nativa en C# de la biblioteca de mensajería *ZeroMQ*, permitiendo utilizar sus

funciones avanzadas de mensajería para crear aplicaciones de red robustas y escalables con facilidad. Se deberá instalar la versión 4.0.1.6 de *NetMQ*.

4.2.1.2. Python.

Para completar la detección de los puntos de referencia de la mano con MediaPipe Hands, y su posterior segmentación, con la que más tarde se podrá interactuar con los objetos virtuales del juego serio, se necesitará un sistema de visión artificial en Python, que extraiga la información de la señal de vídeo y poder mandar esa información a Unity.

Para poder ejecutar este proceso en Python, será necesaria la inicialización previa de un entorno de anaconda o miniconda. Este entorno anaconda o miniconda se inicializa con un archivo .batch que se invoca desde *PythonServiceManager* dentro de Unity.

Para definir el entorno anaconda, utilizando un archivo YAML se pueden especificar fácilmente los paquetes de Python necesarios, sus versiones y cualquier otra dependencia del entorno, todo en un formato claro y fácil de entender. En este caso, el entorno anaconda para el funcionamiento del juego viene definido por el archivo M3D.yml.

4.3. Unity.

Una vez instalado el entorno virtual y el programa de Unity, se puede dar inicio al desarrollo del juego serio.

4.3.1. Unity Hub.

Unity Hub es una aplicación que facilita la gestión de los proyectos y versiones de Unity en un mismo lugar.

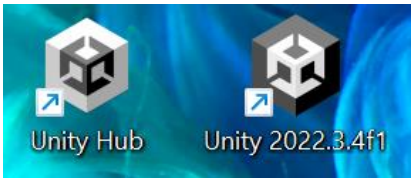

Figura 16: Versión Unity utilizada y Unity Hub. Fuente: Obtención Propia.

En Unity Hub se pueden crear nuevos proyectos, abrir proyectos existentes, descargar e instalar diferentes versiones de Unity, y acceder a documentación y recursos de aprendizaje.

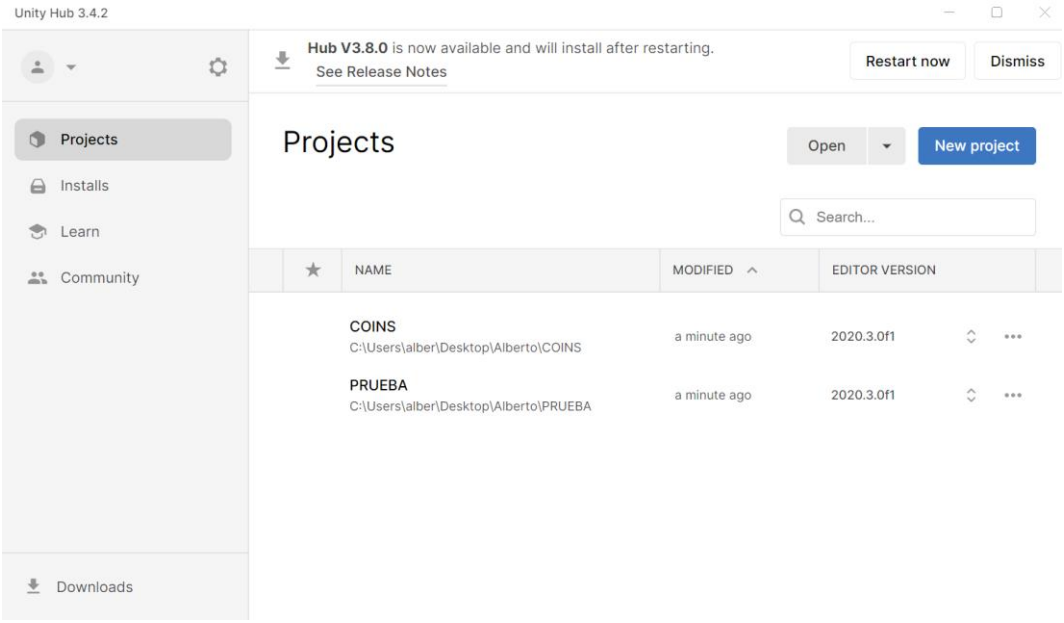

Figura 17: Gestión de proyectos en Unity Hub. Fuente: Obtención Propia.

4.3.2. Interfaz Unity.

Tras haber creado o abierto un proyecto en *Unity Hub*, se abre la aplicación con la versión deseada, donde se puede desarrollar el juego.

Unity es un motor para crear aplicaciones en 2D y 3D con una interfaz sencilla e intuitiva, que se basa en la interacción de objetos mediante la unión de bloques dentro de la interfaz y scripts escritos en lenguaje C#.

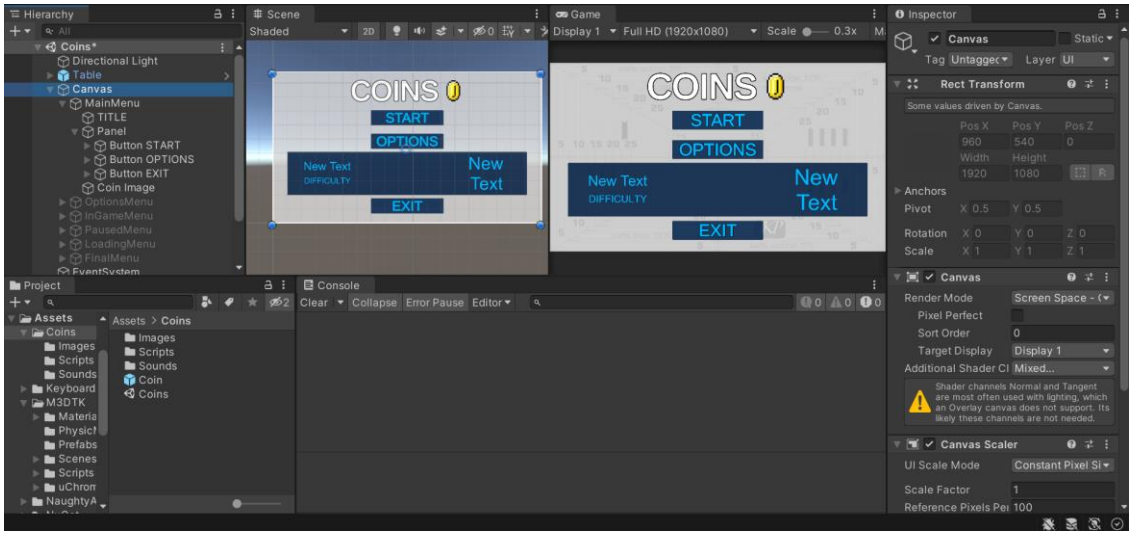

Figura 18: Interfaz del motor Unity. Fuente: Obtención propia.

La interfaz de Unity se divide en ventanas.

4.3.2.1. Ventana *Scene*.

Las escenas en Unity son Assets, que al igual que ocurren en el cine o teatro, representa donde ocurre la acción dentro del juego. Representan diferentes momentos del juego, una pantalla de menú, una cinmática, entre otros. Al abrir un proyecto nuevo, Unity te muestra una escena de muestra básica con una cámara y una luz. A esta escena de muestra se le incluirán los diferentes *GameObjects* que forman la estructura del juego. El juego tendrá una única escena llamada *Coins*.

La ventana *Scene* muestra la escena actual. Esta ventana se utiliza para posicionar los objetos en el espacio donde más adelante se mostrarán en la interfaz de usuario al utilizarse el juego.

4.3.2.2. Ventana *Hierarchy*.

La ventana *Hierarchy* muestra la jerarquía de los objetos en el proyecto. En esta ventana se puede seleccionar que objeto aparece en escena para así poder

configurar su disposición en la escena escogida, o como debe interactuar con otros objetos.

4.3.2.3. Ventana *Project*.

La ventana *Project* muestra todas las carpetas que contiene el proyecto. Dentro de estas carpetas se encontrarán los *Assets* que vienen por defecto en el motor Unity o creados fuera de Unity.

Los *Assets* representan la imagen y la lógica que tendrá el juego a la hora de ser utilizado. Estos van desde imágenes, texturas o sonidos, hasta scripts.

4.3.2.4. Ventana *Inspector*.

La ventana *Inspector* muestra las acciones que puede realizar el objeto seleccionado desde la ventana *Hierarchy*, además de mostrar las características relacionadas cuando se selecciona un *Asset* en la ventana *Project*.

Los objetos son los elementos principales del juego, y desde esta ventana se pueden configurar sus materiales, propiedades y los scripts que permiten la interactuación entre ellos.

4.3.2.4. Ventana *Console*.

La ventana *Console* muestra los mensajes generados por el motor cuando el juego está en curso, tanto errores y advertencias, como mensajes programados en los scripts. Esta ventana es fundamental durante el desarrollo del proyecto para detectar los errores que van surgiendo durante todo su proceso.

4.3.2.4. Ventana *Game*.

La ventana *Game* muestra la última actualización en el desarrollo de la aplicación. La principal utilidad de esta ventana es poder comprobar el estado del proyecto en la interfaz de Unity, y así no tener que exportar este cada vez que se realice una prueba de funcionamiento.

4.3.3. *GameObjects*.

Dentro de la escena de *Coins* se encuentran los *GameObjects*, siendo estos la base del buen funcionamiento del juego. Cualquier objeto dentro del juego se considera un *GameObject*, ya sea un personaje, un objeto que interacciona con otros, una luz, o el entorno donde se proyectarán las imágenes captadas por la cámara. Los *GameObjects* se tienen que configurar, ya sea mediante la programación en la interfaz, como aplicando códigos y scripts, para poder interactuar con otros *GameObjects* o el entorno.

4.3.4. Interacciones de *GameObjects*.

Los *GameObjects*, siendo los elementos fundamentales para el correcto funcionamiento del juego, no realizan ninguna función por si mismos, por lo que es necesario progamar sus funciones, ya sean mediante scripts o la interfaz del motor Unity.

Un ejemplo de *GameObject* podría ser un botón. Un botón puede tener diferentes funciones, desde cambiar de menú hasta poder escoger la trayectoria deseada para la sesión de rehabilitación, por lo que se tiene que programar para que este realice la función deseada. Este enfoque modular y programable permite una gran flexibilidad en el diseño y la funcionalidad del juego, ya que cada GameObject puede ser personalizado para adaptarse a las necesidades y objetivos del proyecto.

4.3.4.1. Unity UI.

Unity UI (Interfaz de Usuario), es un conjunto de herramientas que facilita el desarrollo de interfaces de usuario para juegos y aplicaciones. Se trata de un sistema basado en *GameObjects* que emplea componentes y la vista de juego para organizar, ubicar y estilizar las interfaces de usuario.

4.3.4.2. UnityEngine.UI.

UnityEngine.UI es un namespace esencial en Unity que engloba clases y funcionalidades relacionadas con el interfaz de usuario. Al incorporar este namespace en tus scripts, obtienes acceso a una amplia gama de variables y métodos que te permiten crear y manipular elementos de la interfaz de usuario en tus proyectos.

Por defecto, al crear un nuevo script en Unity, este incluirá automáticamente los namespaces System.Collections, System.Collections.Generic y UnityEngine, pero para trabajar específicamente con la interfaz de usuario, es necesario utilizar el namespace UnityEngine.UI.

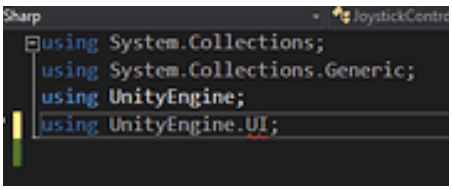

Figura 19: Inicialización UnityEngine.UI. Fuente: Obtención propia.

Este namespace proporciona las herramientas necesarias para diseñar interfaces de usuario intuitivas y funcionales, lo que resulta fundamental para la experiencia del usuario en aplicaciones y juegos.

4.3.4.3. UnityEngine.Events.

El espacio de nombres *UnityEngine.Events* proporciona una infraestructura robusta para implementar eventos y callbacks en Unity. Con *UnityEngine.Events* se pueden crear callbacks personalizados que se pueden asignar a *GameObjects* y ejecutar en respuestas a diferentes eventos en tiempo de ejecución. Esta funcionalidad es fundamental para la creación de sistemas interactivoa y dinámicos en los juegos y aplicaciones. Además, *UnityEngine.Events* permite la configuración de callbacks tanto estáticos co o dinámicos, lo que brinda flexibilidad para adaptarse a diversas situaciones de desarrollo. Los eventos estáticos son preconfigurados con valores establecidos en la interfaz de usuario, mientras que los eventos dinámicos pueden recibir argumentos enviados desde el código, lo que permite una mayor personalización y control en la ejecución de los callbacks.

4.3.5. *GameObjects* principales.

Los *GameObjescts* son la base del juego, y mediante el uso de la venta *Hierarchy*, podemos seleccionar y organizar estos *GameObjects* según su jerarquía.

4.3.5.1. Table.

El *GameObject* principal es el objeto llamado *Table*. Este *GameObject* simula la posición en donde se va a encontrar la mesa cuando se proyecte la imagen de la cámara, además de donde se incluyen los scripts que controlan el funcionamiento del entorno de visión artificial para la detección de la mano, y que asignan como objetos los puntos de referencia de la mano o *landmarks* encontrados con MediaPipe Hands.

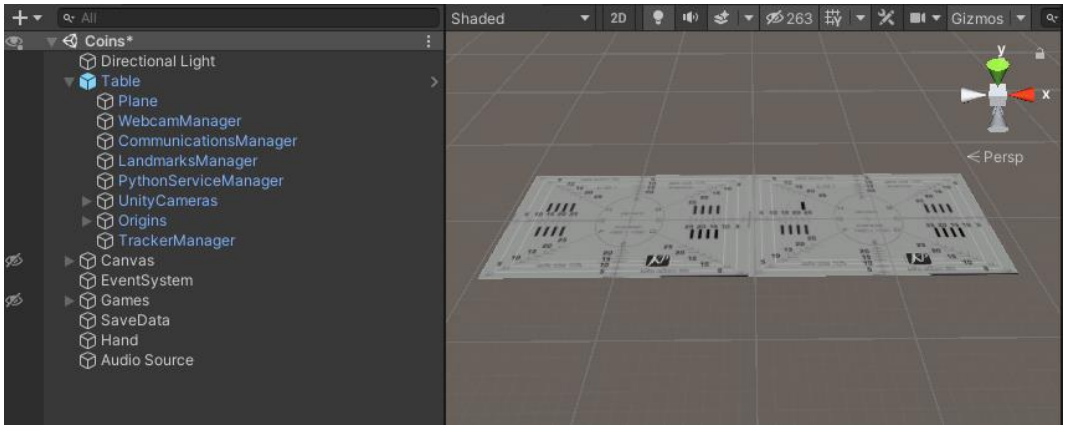

Figura 20: *GameObject* Table. Fuente: Obtención Propia.

4.3.5.2. Canvas.

Para poder controlar el funcionamiento en un juego, y que este sea intuitivo, el juego debe contar con diferentes menús. El *GameObject* llamado *Canvas* tendrá incluidos todos los menús.

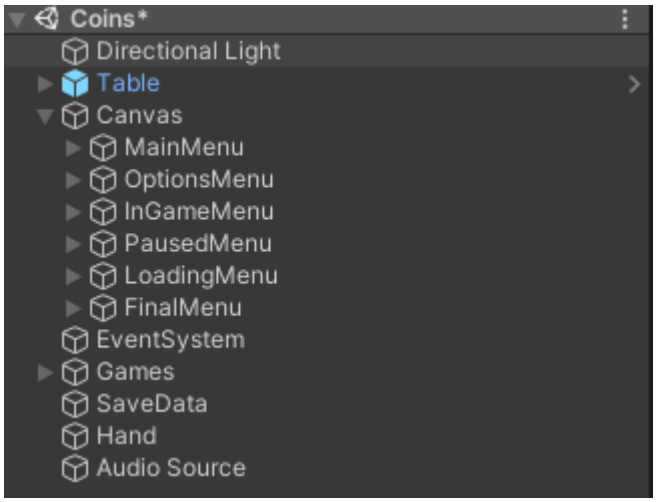

Figura 21: Menús dentro de Canvas. Fuente: Obtención Propia.

4.3.5.2.1. Menú principal.

En *MainMenu* se encuentra el menú principal del juego. Este es el menú que se encontrará abierto al inicializar el juego. El profesional encargado de la rehabilitación del paciente, en cada sesión que necesite utilizar este juego, podrá controlar y acceder al resto de menú de forma intuitiva desde este.

El menú principal se compone del título del juego, además de los botones *Start* para inicializar cada partida, el botón *Options* para seleccionar las diferentes opciones que presenta el juego, y el botón *Exit* para salir. Además, se dispone de un cuadro donde se puede visualizar la dificultad, forma de la trayectoria y tiempo máximo previamente escogido por el médico.

Figura 22: Menú Principal. Fuente: Obtención propia.

4.3.5.2.2. Menú opciones.

En *OptionsMenu* se encuentra el menú donde el médico podrá escoger las diferentes opciones de partida que presenta el juego. En este menú el terapeuta se encontrará dos *sliders*, un *slider* con el que podrá elegir el tiempo que máximo que tendrá el paciente en trazar la trayectoria con éxito y tratar de lograr la puntuación máxima. Con el otro *slider* se puede cambiar la dificultad que tendrá la partida, dependiendo del estado y la evolución del paciente a la hora de realizar la sesión. Según que dificultad haya escogido el médico con el *slider*, el panel cambiará para mostrar que trayectorias están disponibles. En este panel se encuentran los botones que seleccionan la trayectoria que deberá realizar el paciente una vez inicializado el juego.

Figura 23: Menú Opciones. Fuente: Obtención propia.

4.3.5.2.3. Menú *InGame*.

En *InGameMenu* se encuentra el menú que muestra por pantalla la trayectoria propuesta por el médico para que el paciente realice durante la sesión. En este menú el fondo es transparente, para poder mostrar el *GameObject Table* que simula la mesa sobre la que se proyectará la trayectoria escogida.

Este menú, además, para poder ofrecer un *feedback* visual al paciente mientras realiza la sesión, incluye un contador que muestra el tiempo restante para finalizar la partida, y un contador del puntaje que lleva el paciente en todo momento.

Se contará con un botón *PAUSE* para poder pausar la partida, si es necesario para el paciente, o si el terapeuta necesita dar alguna indicación al paciente durante la partida.

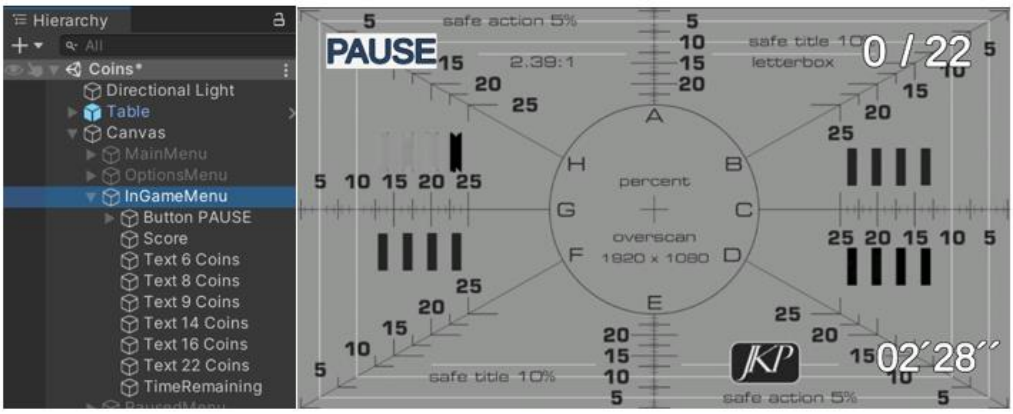

Figura 24: Menú *InGame*. Fuente: Obtención propia.

4.3.5.2.4. Menú pausa.

En *PausedMenu* se encuentra el menú con la partida pausada. Este menú tiene la única función de detener el transcurso de la partida, manteniendo el tiempo restante, para poder reanudar este cuando el médico desee. Se cuenta con dos botones, el botón *CONTINUE* para reanudar la partida, y el botón *EXIT* para finalizar la partida.

Figura 25: Menú Pausa. Fuente: Obtención propia.

4.3.5.2.5. Menú de carga.

En *LoadingMenu* se muestra el menú que carga la partida. Este menú aparece durante el intervalo de tiempo que tardan los elementos virtuales de la realidad aumentada en aparecer en el menú *InGame*. En el menú solo se encuentra una animación del logo de *M3ROB* DE ITAP.

Figura 26: Menú de carga. Fuente: Obtención Propia.

4.3.5.2.6. Menú final.

En *FinalMenu* se muestra el menú del final de la partida, se visualiza al concluir esta, ya sea por haber conseguido la puntuación máxima o porque se haya terminado el tiempo antes de haber llegado a esa puntuación.

Dentro de este menú, tanto el paciente como el terapeuta podrán observar un panel con los resultados obtenidos durante la partida, mostrando el tiempo restante y la puntuación final. Además, el médico se encontrará con dos botones, un botón *MAIN MENU* con el que podrá volver al menú inicial en caso de querer realizar otra trayectoria, y otro botón *EXIT* si este quiere dar por finalizada la sesión y salir del juego.

Figura 27: Menú final de la partida. Fuente: Obtención propia.

4.3.5.3. Games.

En este *GameObject* se han incluido todos los elementos virtuales que proyectarán sobre la mesa para el correcto funcionamiento del juego en realidad aumentada. Estos elementos virtuales se componen de una imagen de la trayectoria deseada sobre la mesa, para que al paciente le resulte más intuitivo el movimiento que tiene que realizar, y sobre esta trayectoria estarán situadas las monedas que tiene que recoger el paciente para tratar de lograr la puntuación máxima.

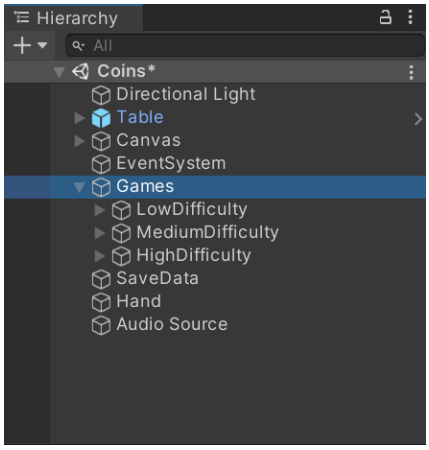

Figura 28: *GameObject* Games. Fuente: Obtención propia.

Para regular la evolución del paciente en el proceso de rehabilitación se han dividido las trayectorias en tres dificultades: dificultad fácil, dificultad media y dificultad difícil. De entre cada una de estas tres dificultades, el terapeuta tendrá la posibilidad de escoger entre un número de trayectorias.

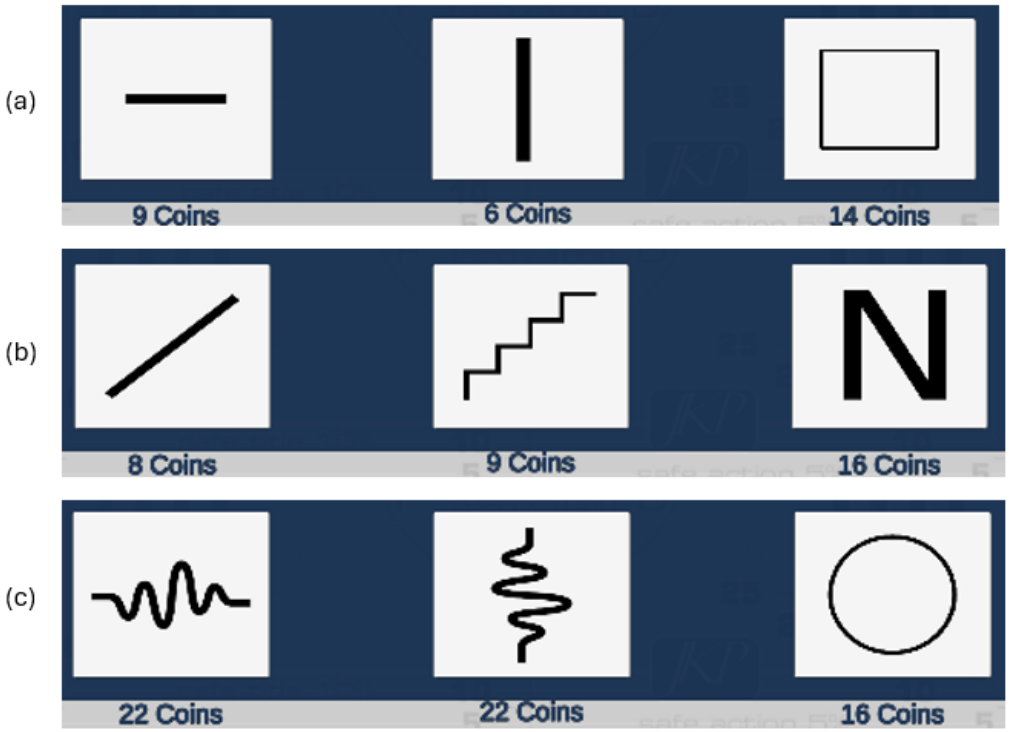

Figura 29: Trayectorias disponibles. (a) Trayectorias dificultad fácil. (b) Trayectorias dificultad media. (c) Trayectorias dificultad difícil. Fuente: Obtención propia.

Cada dificultad dispondrá de los *GameObjects* que representarán las monedas y la imagen de la trayectoria.

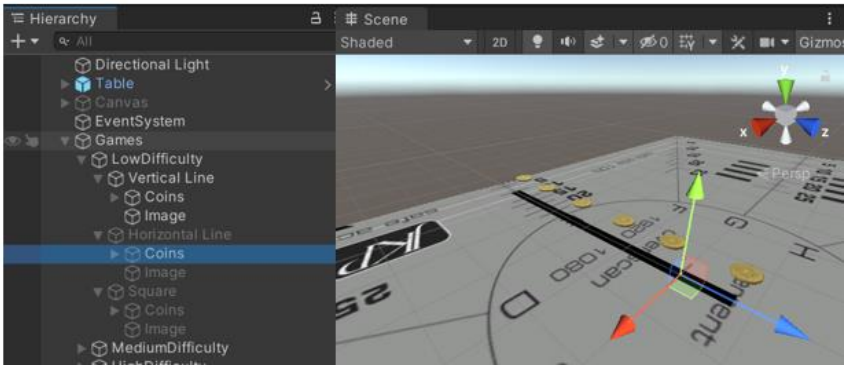

Figura 30: Proyección de los elementos virtuales sobre la mesa. Fuente: Obtención propia.

Al *GameObject* que representa una moneda se le incluido un *Asset* con la textura. Este *Asset* se puede encontrar y seleccionar desde la ventana *Project* en la carpeta M3DTK.

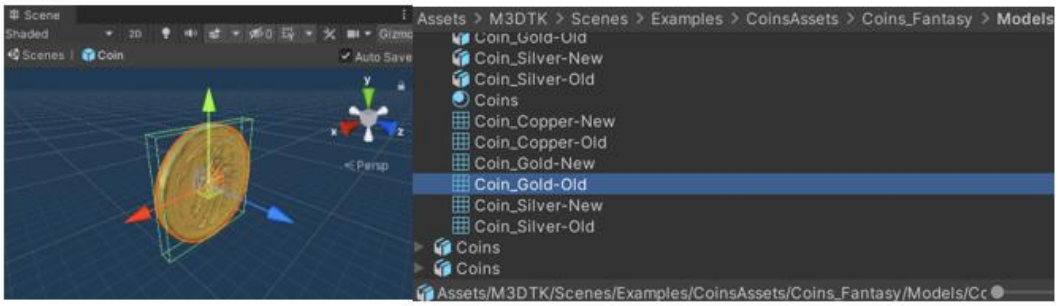

Figura 31: *Asset* textura moneda. Fuente: Obtención propia.

Para las imágenes superpuestas sobre la mesa donde se situará el paciente, también se importará un *Asset* al *GameObject*, en este caso una imagen que se encuentra en la misma carpeta que el resto de las imágenes utilizadas en el proyecto.

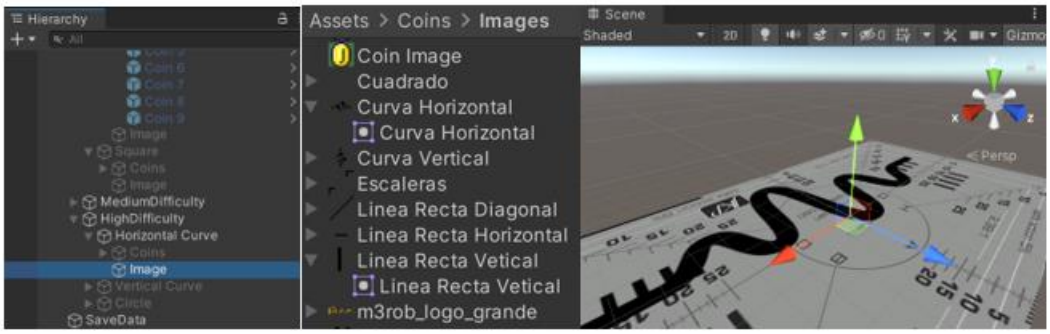

Figura 32: Imagen de la trayectoria. Fuente: Obtención propia.

4.3.5.4. SaveData.

Para garantizar un seguimiento óptimo del progreso del paciente, es fundamental contar con un registro actualizado de su evolución. Con este fin, se ha implementado un *GameObject* específico que alberga los scripts dedicados a la adquisición de datos. Este componente es esencial para recopilar información relevante durante las sesiones de terapia de rehabilitación, permitiendo así evaluar el rendimiento del paciente, realizar análisis comparativos y ajustar el plan de tratamiento según sea necesario.

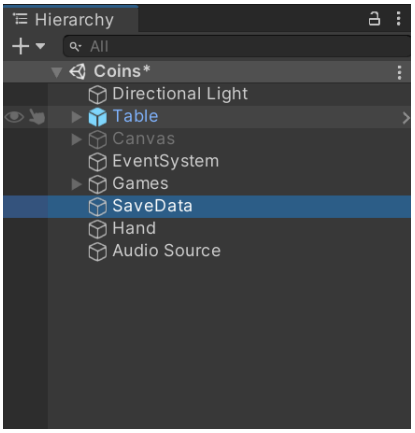

Figura 33: *GameObject SaveData.*Fuente: Obtención propia.

4.3.5.5. Hand.

Para que el paciente pueda interactuar con elementos virtuales del juego de realidad aumentada, se tiene que realizar un proceso de detección de la mano. Se dispone de un *GameObject* para englobar las funciones de la mano del paciente.

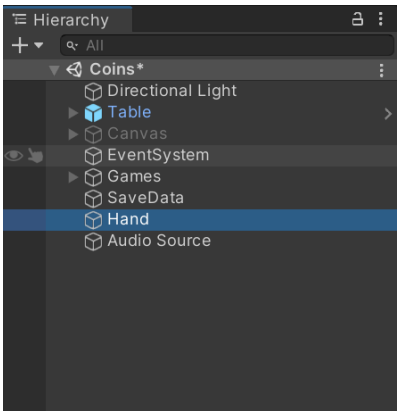

Figura 34: *GameObject* Hand. Fuente: Obtención propia.

4.3.5.6. Audio Source.

Junto con el feedback visual, es fundamental integrar un feedback auditivo durante las sesiones de rehabilitación para mejorar la experiencia del paciente. Para ello, se ha incluido un *GameObject* encargado de proporcionar el componente de audio al juego. Esta funcionalidad permite enriquecer la interacción del paciente con el entorno virtual, ofreciendo señales auditivas que complementan y refuerzan la retroalimentación visual proporcionada por el sistema.

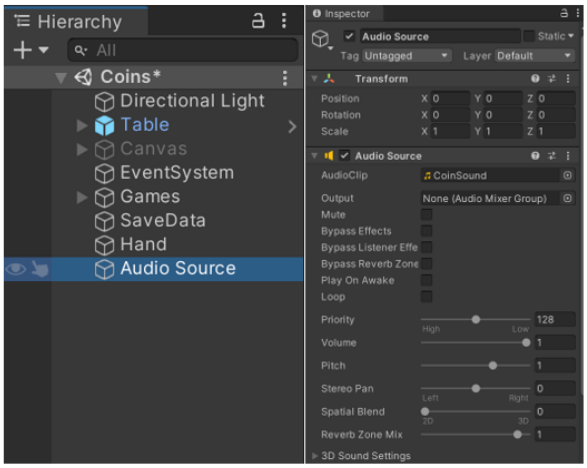

Figura 35: *GameObject* Audio Source. Fuente: Obtención propia.

El sonido escogido se ha descargado en freesound.org, el creador es Taira Komori. Este sonido está dispuesto de una licencia *Attribution*, por lo que se tiene la disponibilidad total de uso siempre que se de crédito al autor de este.

Figura 36: Sonido utilizado para el feedback auditivo. Fuente: http://taira-komori.jpn.org/freesounden.html

5. INTERACCIÓN DE LOS OBJETOS EN EL JUEGO.

Una vez se ha establecido la estructura principal del juego con los *GameObjects* necesarios, el siguiente paso crucial es programar la lógica del juego. Esto implica desarrollar las funciones y comportamientos específicos que cada *GameObject* debe ejecutar durante la utilización del juego. Desde la navegación del terapeuta entre los menús del juego, hasta la interacción del paciente con los elementos virtuales dispuestos por la realidad aumentada en el entorno,

Cada aspecto del juego debe ser meticulosamente programado para garantizar una experiencia de juego fluida y atractiva al usuario, y así provocar un mayor compromiso y una mayor adherencia del paciente en el proceso de rehabilitación.

5.1. *GameObject* Botón.

Un ejemplo de *GameObject* serían los botones. Estos elementos son esenciales para la interacción con el juego, ya que permiten activar acciones, cambiar entre menús y realizar diversas operaciones clave.

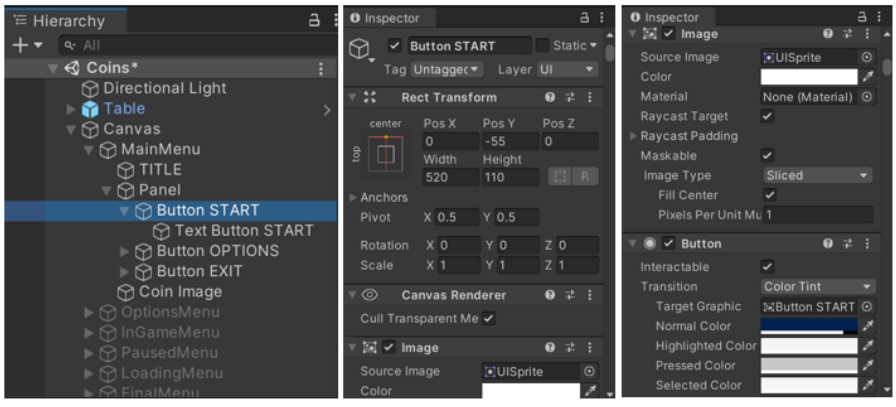

Figura 37: Configuración de botón. Fuente: Obtención propia.

La interfaz del motor Unity nos brinda herramientas visuales intuitivas para modificar la apariencia y el comportamiento de los elementos del juego, como el botón *START* ubicado en el menú principal. En la Figura 37, se puede observar cómo en la ventana *Inspector*. La pestaña *Inspector* en Unity es una

herramienta fundamental que proporciona acceso a las propiedades y componentes de un *GameObject* seleccionado.

5.1.1. Función *On Click()*.

Para que el terapeuta pueda controlar de manera efectiva el juego, por ejemplo, pulsando los botones que se encuentran en su interfaz, y estos respondan de la manera que deben, es necesaria una correcta configuración de la función *On Click()* en la ventan *Inspector*.

El *Inspector* es capaz de realizar cualquier acción o evento que herede de *UnityEvent*, en este caso, al tratarse de un botón, esta función se invoca al pulsarse el botón en la interfaz de usuario.

Como se observa en la Figura 38, se puede configurar que funciones se deben realizar al invocarse la función *On Click()* a través de la interfaz de Unity, arrastrando los *GameObjects* que deben interactuar con el botón, y seleccionando la acción a realizar.

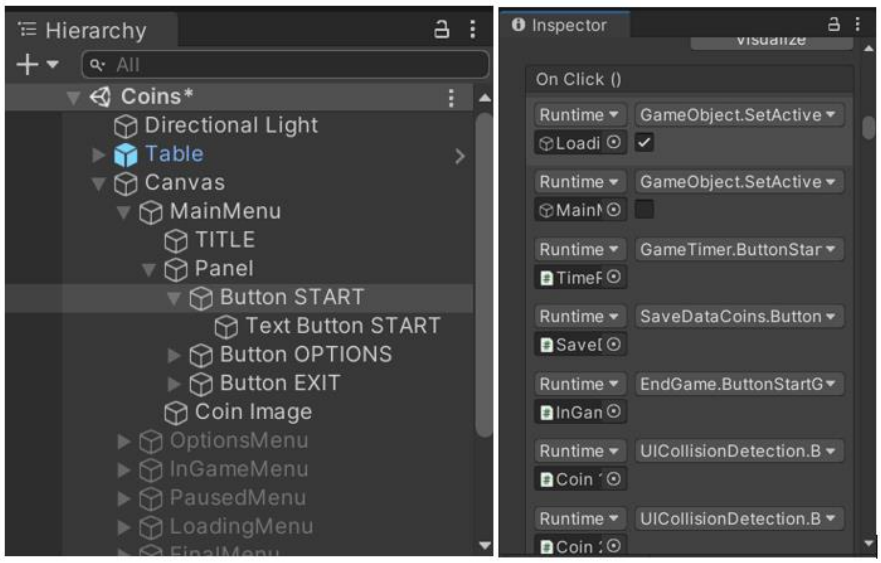

Figura 38: Programación de funciones de un *GameObject*. Fuente: Obtención propia.

Al igual que el aspecto, en la ventana *Inspector* se encuentra la función *On Click()*. Esta función forma parte del sistema de programación de funciones de *GameObjects* incluidos en el motor de Unity. Permite asignar acciones específicas que se ejecutarán cuando se presione un *GameObject*, como un

botón. Al configurar la función *On Click()*, se puede definir fácilmente la interactividad y el comportamiento de los elementos de la interfaz de usuario, lo que contribuye a una experiencia de juego más dinámica y envolvente.

5.1.2. Transición entre menús.

Para una interacción fluida entre el usuario, en este caso el terapeuta, y el juego es imprescindible que navegar entre los menús de este sea sencillo e intuitivo. Por ello en el desarrollo de "Coins" se ha provisto de botones expresen de forma clara la acción que supone el pulsarlos.

Para que se produzca de forma correcta la transición entre dos menús dentro del juego, tiene que dejar de aparecer en la interfaz de usuario el menú donde se encuentra el botón que se pulsa y aparecer el menú al que se desea acceder.

Los menús son *GameObjects* dentro del juego, y existe una función booleana llamada *SetActive(bool)*, que al invocar la función *OnClick()* del botón, permite activar o desactivar un *GameObject* dependiendo del valor que se le de a esa variable *bool*.

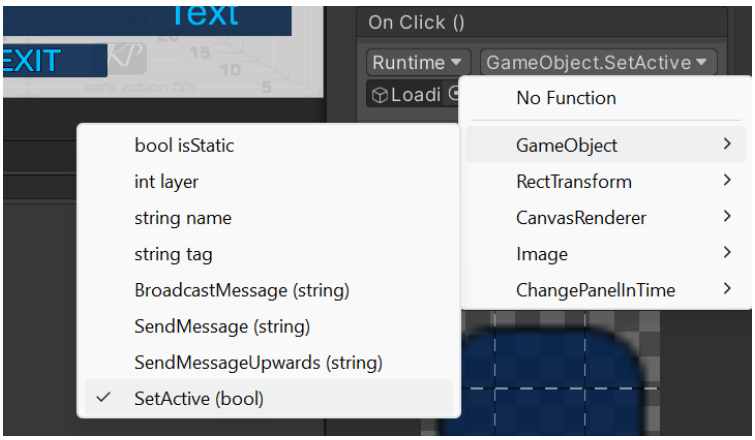

Figura 39: Función *SetActive(bool)* para activar o desactivar un *GameObject*. Fuente: Obtención propia.

Al ser una función de uso común en el desarrollo de juegos con el motor de Unity, la programación de la activación y desactivación de los *GameObjects* viene incluida en la interfaz del motor. Gracias a la integración de esta función en la interfaz de Unity, basta con seleccionar el *GameObject* sobre el que actúa la función y arrastrar este hasta la ventana *Inspector*, además de dar un valor

“positivo” o “negativo” a la variable *bool* dependiendo si se desea la activación o desactivación de este.

5.1.3. Selección de trayectorias.

En el menú destinado a que el terapeuta elija las opciones disponibles para cada sesión de terapia, se presenta un panel que contiene un botón correspondiente a cada una de las trayectorias disponibles. Esta disposición facilita al terapeuta la elección y configuración de la trayectoria adecuada según las necesidades y objetivos de cada paciente.

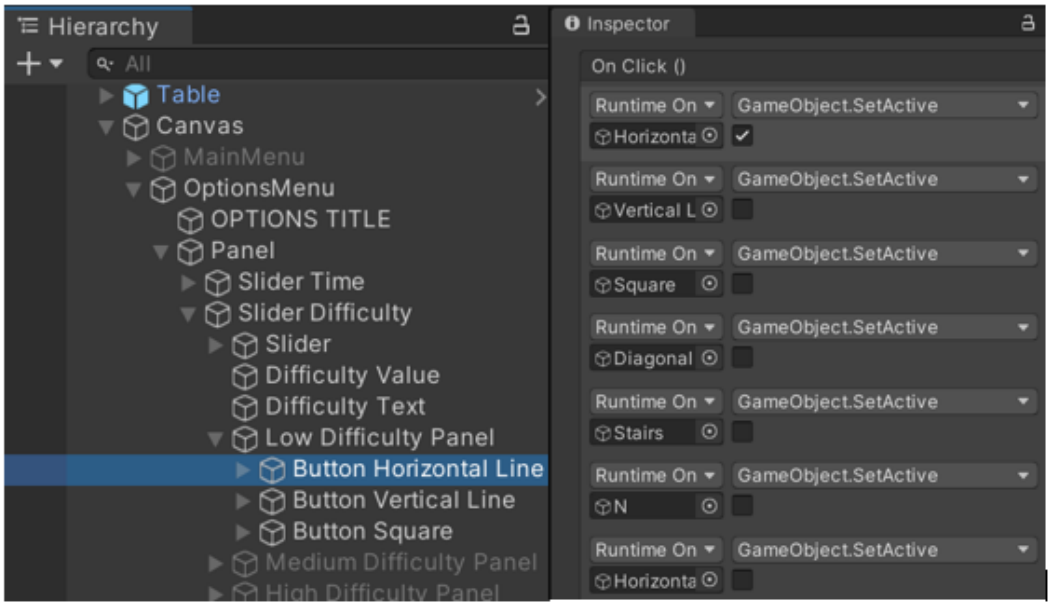

Figura 40: Botón de selección de trayectoria. Fuente: Obtención propia.

Presionando este botón se tienen que cumplir varias funciones, que como se indica en la Figura 40, se invocan con la función *On Click()* de la ventana *Inspector*.

Se tiene que activar el *GameObject* que muestra la imagen de la trayectoria seleccionada en el menú principal, lo que permite al terapeuta y al paciente visualizar claramente la opción elegida. Al igual que se debe mostrar la trayectoria en el menú *InGame*, para que pueda ejecutar la acción debidamente el paciente.

Cada trayectoria presenta un nivel de dificultad, y por lo tanto cuentan con un diferente número de monedas proyectadas sobre la mesa donde se sitúa el paciente. Para proporcionar un feedback visual a la hora de realizar las terapias de rehabilitación, en el menú *InGame* se dispone de un contador del puntaje en todo momento, y de la puntuación máxima que tiene la trayectoria. Presionando el botón que selecciona cada trayectoria se activa el *GameObject* que muestra esta puntuación máxima durante la sesión, lo que ayuda al paciente a tener claro el objetivo a alcanzar y a mantenerse motivado durante la terapia.

5.2. *GameObject* Slider.

Los *Sliders* son otro ejemplo importante de *GameObject* en Unity. Estos elementos permiten al usuario ajustar un valor específico asociado a una variable mediante una interfaz gráfica intuitiva. Esta funcionalidad es fundamental en la creación de interfaces de usuario interactivas y personalizadas en aplicaciones y juegos de Unity.

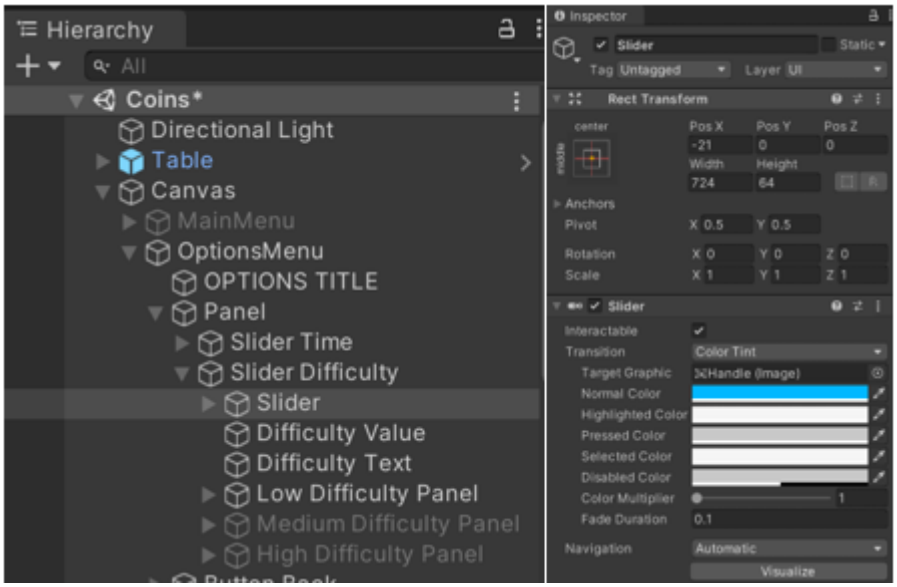

Figura 41: Diseño de un *Slider*. Fuente: Obtención propia.

Al igual que el resto de *GameObjects*, se puede modificar el aspecto de los los *Sliders* en la ventana *Inspector*. Se puede cambiar tanto el aspecto como el color, para contar con un aspecto acorde a la estética del resto de la interfaz.

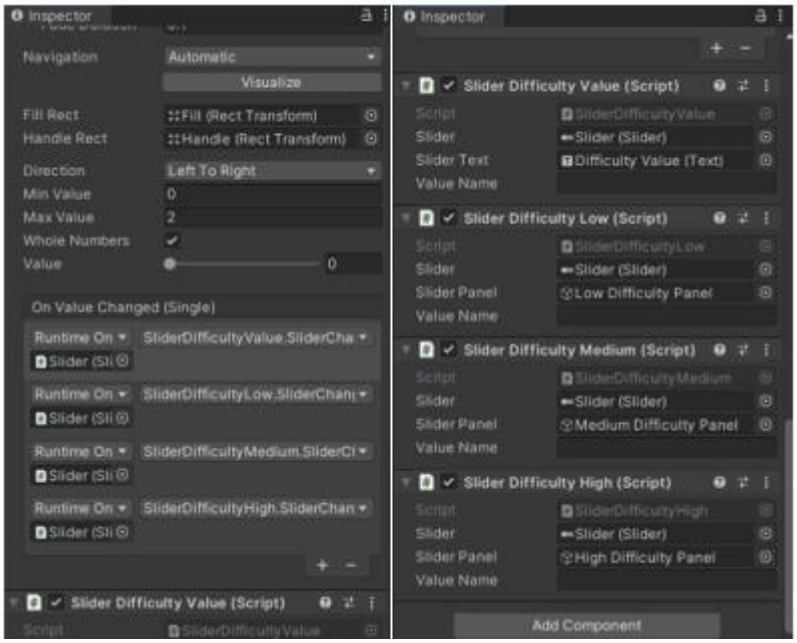

Figura 42: Configuración *Slider*. Obtención propia.

Los *Sliders* se utilizarán para dar un valor determinado a una variable. En el caso del juego *Coins*, el terapeuta hará uso de estos a la hora de escoger el tiempo máximo de duración de una partida, y para seleccionar la dificultad de la trayectoria a realizar.

El valor del *Slider* viene determinado por la posición en la que se encuentra la manija, situada a lo largo de su longitud. Al mover la manija a lo largo de la longitud del *Slider*, el valor de la variable asociada a este cambiará en un rango de un valor mínimo *Min Value* y un valor máximo *Max Value*.

5.2.1. Función *On Value Changed()*.

En la Ventana *Inspector*, al insertar un *Slider* en la interfaz del usuario, aparece una pestaña para hacer uso de la función *On Value Changed()*.

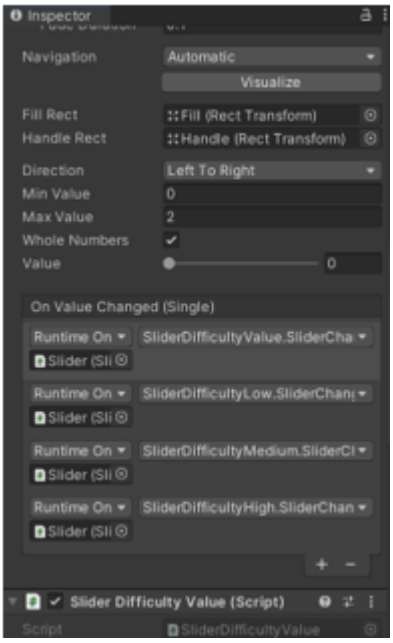

Figura 43: Función *On Value Changed()* de un *Slider.* Fuente: Obtención propia.

Al igual que con los botones y su *On Click()*, la función *On Value Changed()* en los *Sliders* viene predeterminada en la interfaz de Unity, para un manejo más sencillo de la función, pudiendo arrastrar los *GameObjects* a esta pestaña para que sus varibales asociadas puedan variar a la hora de interactuar con el *Slider*.

Esta función es fundamental para controlar y responder a los cambios en el valor del *Slider* durante la interacción del usuario con la interfaz del juego. De esta manera los *Sliders* se convierten en herramientas versátiles para ajustar parámetros, configurar opciones y realizar acciones específicas dentro del juego.

5.2.2. Variables del juego.

Para personalizar la experiencia de cada paciente durante la terapia, se implementa la opción de seleccionar una trayectoria con una dificultad y tiempo máximo específicos. Esta personalización se logra utilizando la función *On Value Changed()* en combinación con scripts que gestionan los *GameObjects* asociados.

Como se observa en la Figura 42, en la interfaz de Unity, se pueden modificar los elementos que muestran el tiempo máximo y la dificultad seleccionada en

el menú principal, como se muestra en la Figura 22, lo que permite al terapeuta ajustar la sesión según las necesidades del paciente. Además, se actualizan las variables de tiempo y puntaje en los scripts del juego, para ajustar la lógica de funcionamiento de este.

6. DESARROLLO DE LAS FUNCIONES DEL JUEGO.

En el desarrollo del juego serio con realidad aumentada *Coins* para la rehabilitación del miembro superior, la programación de la lógica es un aspecto fundamental que define su funcionalidad y jugabilidad. Para esto, se emplean una serie de scripts escritos en lenguaje C#, los cuales contienen clases, métodos y variables que controlan el comportamiento de los elementos del juego. En Unity, estos scripts están estrechamente relacionados con los *GameObjects* de la interfaz, ya que es necesario ajustar las variables y funciones definidas en el código con los objetos visuales y funcionales presentes en la escena del juego.

Esta integración entre el código y los elementos de la interfaz es esencial para asegurar una lógica de funcionamiento coherente y efectiva, permitiendo que el juego responda adecuadamente a las acciones del usuario y cumpla con los objetivos de rehabilitación de manera satisfactoria.

6.1. Puntos de referencia de la mano.

El sistema M3Display de realidad mixta cuenta con un proceso avanzado de detección y procesamiento de datos de la mano del paciente. Para lograr esto, utiliza MediaPipe Hands, una tecnología de visión artificial que identifica puntos de referencia clave en la mano del usuario.

En el contexto del motor de videojuegos Unity, el funcionamiento se centra en la interacción de *GameObjects*, elementos fundamentales que representan objetos y funcionalidades en el entorno virtual. Como se ilustra en la Figura 34, se emplea un *GameObject* específico para integrar las funciones de la mano del paciente. Este *GameObject* cuenta con el script que favorece a la interacción de la mano con la interfaz de usuario, convirtiendo los puntos de referencia detectados con MediaPipe Hands en objetos virtuales capaces de interactuar con los elementos virtuales presentes en la aplicación de realidad mixta. De esta manera, se establece una conexión efectiva entre el mundo físico del paciente y el entorno virtual, permitiendo una experiencia de usuario inmersiva y funcional.

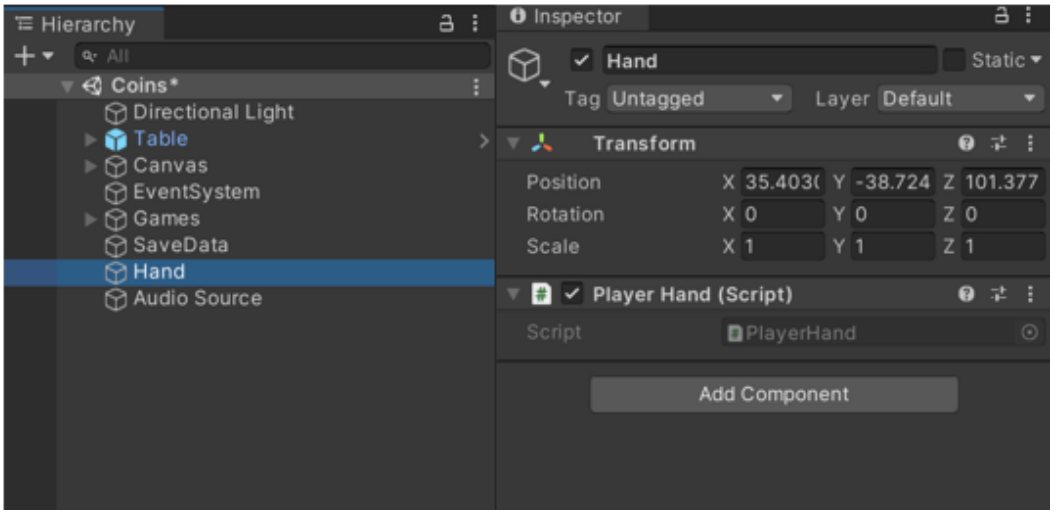

Figura 44: Transformación de puntos de referencia de la mano en *GameObjects*. Fuente: Obtención propia.

En la ventana *Inspector* se incluye el script denominado *PlayerHand*, el cual desencadena un proceso fundamental en la aplicación. En este script se recogen todos los puntos de referencia clave de la mano del paciente y convierte cada uno de estos puntos en un *GameObject* dentro del entorno de Unity, además de obtener un vector con sus coordenadas tridimensionales (x,y,z) en el espacio virtual.

A la hora de registrar el proceso del paciente, uno de los datos obtenidos y que se tendrá en cuenta para que el terapeuta compruebe si existe mejora en su rendimiento, es la posición de la mano en todo momento. El sistema de visión artificial, para poder procesar los datos y recrear el esqueleto de la mano, obtiene la posición de varios puntos. Para poder exportar la posición de la mano, se recopilan las coordenadas espaciales de todos los puntos obtenidos, y se realiza la media de cada coordenada. Este método proporciona una representación precisa de la posición media de la mano del paciente, lo que permite un seguimiento efectivo de esta y así poder comparar la trayectoria descrita por el paciente con la trayectoria ideal propuesta por el terapeuta por medio de la interfaz.

6.2. Captura de monedas.

Para mantener la motivación del paciente en todo momento, es importante que el juego tenga un sistema de puntaje, con el que el paciente se esfuerce por mejorar sus marcas, y por lo tanto acelerar el proceso de rehabilitación.

Para este juego de realidad aumentada *Coins*, como su nombre indica, se ha pensado en colocar monedas virtuales sobre la trayectoria escogida. Con este sistema, la intención del paciente de recoger todas las monedas para tratar de obtener la máxima puntuación posible obligará a este a tener que seguir la trayectoria deseada.

Como se indica en la Figura 31, cada una de las monedas es un *GameObject* al que se le han insertado los *Assets* para obtener la forma y la textura necesarias, y los scripts necesarios para implementar la lógica de la moneda.

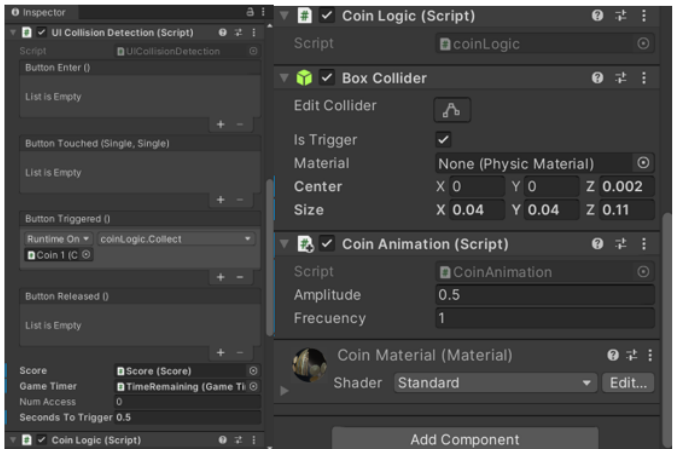

Figura 45: Lógica de una moneda. Fuente: Obtención propia.

Cada una de las monedas cuenta con un *Box Collider*. Este *Box Collider* representa el espacio de coordenadas en el que cada moneda interactúa con el resto de *GameObjects* del juego.

También se incluirá el script *UI Collision Detection*, que incluirá la lógica de colisiones de cada moneda. En este script se incluye la función *Button Triggered()*, donde cuando choca cualquier *GameObject* obtenido de los puntos de referencia obtenidos de la mano con el *Box Collider* de la moneda, se invoca la función *Collect()* del script *coinLogic*. Además de invocar esta función, cada vez que se ejecuta *Button Triggered()*, se suma una unidad a la variable que indica la puntuación actual, y se invoca el script asociado al sonido del juego.

A cada moneda también se le asocia el script *coinLogic*. En este se encuentra la función *Collect()*, que se invoca en cuanto algún punto de referencia de la mano golpea. Esta función desactiva el *GameObject* de la moneda, ya que esta acaba de ser recogida.

6.3. Puntuación.

Como se ha explicado, la puntuación es un aspecto crucial en el desarrollo del juego, un buen sistema de puntaje provocará un mejor desarrollo de la aplicación. Del valor de la variable que representa la puntuación en cada momento del desarrollo de la partida dependen muchas de las funciones que engloban la lógica del juego.

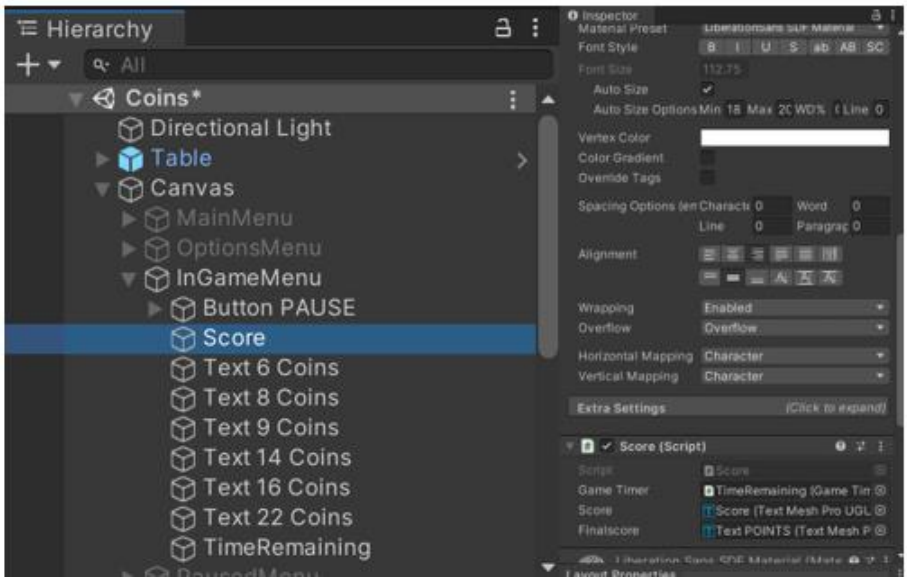

Figura 46: Script *Score*. Fuente: Obtención propia.

Se crea un script *Score*, donde se inicializa la variable *points*, que tendrá como valor la puntuación que lleva el paciente en todo momento. También se inicializa la variable *MaxScore*, que como su nombre indica es la puntuación máxima, y depende de la trayectoria elegida por el terapeuta.

Este script, como se indica en la Figura 46, se agregará en la ventana *Inspector* de varios *GameObjects*. Tanto el valor de la puntuación máxima, como el valor de la puntuación actual, constituyen un alto porcentaje de la lógica que hace funcionar de forma correcta y fluida el juego.

6.4. Temporizador.

Al igual que el puntaje, el temporizador es crucial a la hora de forzar al paciente a lograr su objetivo, y por tanto de agilizar el proceso de rehabilitación con un mayor control en el progreso.

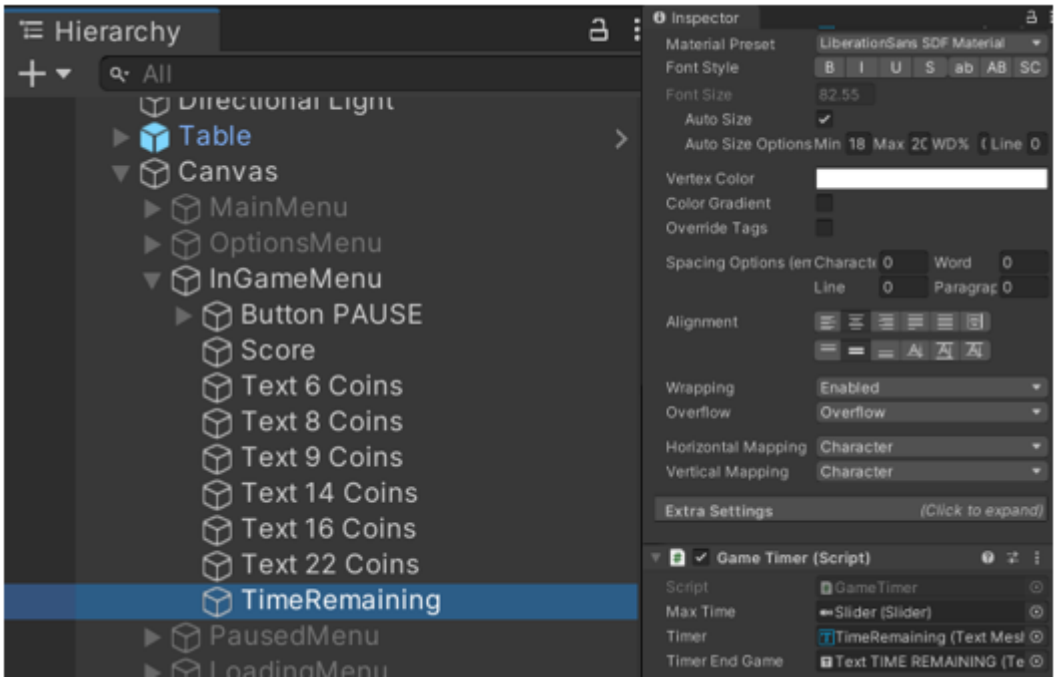

Figura 47: Script *GameTimer*. Fuente: Obtención propia.

Como ocurre con la puntuación, en el script *GameTimer* se inicializa la variable *ActualTime* con el valor del temporizador.

Como se indica en la Figura 47, este *script* se incluye en la ventana *Inspector* de varios *GameObjects*, esto es debido a que del valor del temporizador depende el funcionamiento de la mayor parte de la lógica del juego.

6.5. Sonido.

El feedback auditivo es muy importante a la hora de agilizar el proceso de rehabilitación. Para este juego se ha decidido implementar el sonido mostrado en la Figura 36.

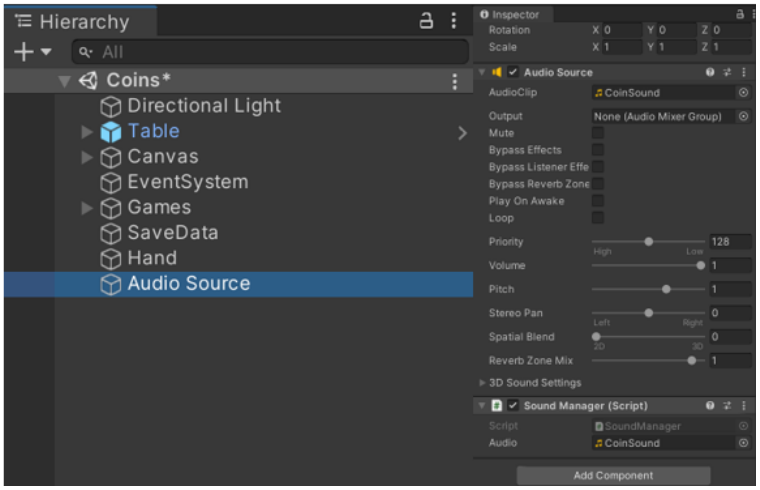

Figura 48: Script *SoundManager*. Fuente: Obtención propia.

En el *GameObject* denominado *Audio Source* se inserta el sonido, además del script que inicializa la variable que controla el sonida y la función con la que se ejecuta. Cuando Un punto de referencia de la mano choca con el *Box Collider* de una moneda como se aprecia en la Figura 45, aparte de sumar una unidad a la variable de la puntuación, se ejecutará la función que controla el sonido.

6.6. Recopilación de datos.

Para ayudar al terapeuta a controlar el progreso del paciente, y poder realizar terapias personalizadas, se debe hacer un seguimiento de cada una de las sesiones.

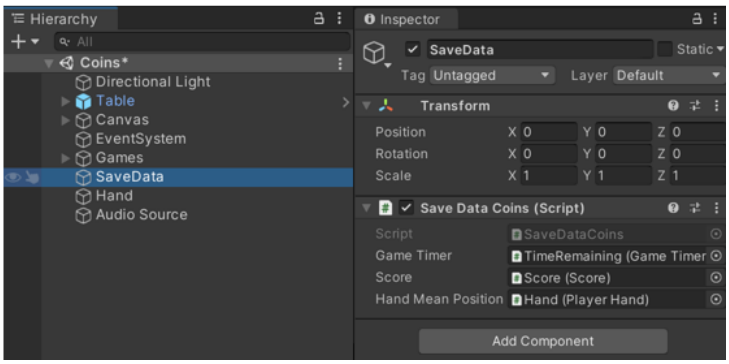

Figura 49: Script *SaveData*. Fuente: Obtención propia.

En el *GameObject* llamado *SaveData* adjuntaremos el script *SaveData*. En este script se generará un archivo .CSV de cada partida en una carpeta llamada *CoinData*. Para diferenciar las partidas, el nombre de cada uno de estos archivos será la fecha y la hora del comienzo de estas.

Al script para guardar datos se le pasan los valores de las variables que representan la puntuación y el tiempo. En cada frame de vídeo capturado por la cámara del sistema se recogen los datos requeridos.

Para lograr un seguimiento óptimo de la evolución del paciente se ha decidido recoger datos desde que el paciente recoge la primera moneda, así se puede observar el tiempo de reacción de este en el momento en el que aparecen las monedas sobre la mesa.

También se recogen las coordenadas espaciales de la posición de la mano del paciente, y así poder representar gráficamente el movimiento de la mano, para comparar este con la trayectoria objetivo.

Otro de los datos importantes puede ser el instante en el cual se recoge cada moneda, así el terapeuta puede calcular el tiempo que transcurre entre que recoge una moneda y la siguiente, y así observar que tramos de la trayectoria son más fáciles y cuales más difíciles para el paciente.

6.7. Finalización de la partida.

Cada partida tiene que suponer un reto por llegar a un objetivo para el paciente, en el caso de este juego, el paciente tiene que obtener la puntuación máxima antes de cumplir el tiempo máximo requerido.

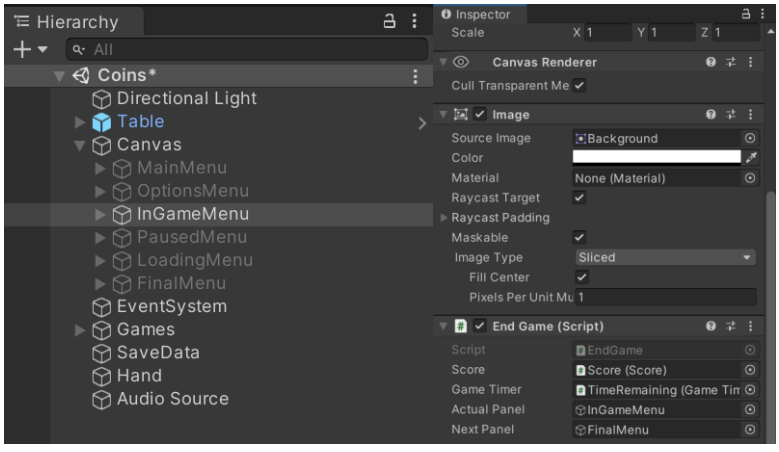

Figura 50: Script *EndGame*. Fuente: Optención propia.

El script *EndGame* se incluye en la ventana *Inspector* del menú *InGame*, al que se le incluyen las variables que representan la puntuación y el temporizador. Este script es el encargado de dirigir al terapeuta al menú final una vez el paciente haya obtenido la puntuación máxima, o el temporizador haya llegado a un valor nulo. Más adelante, en el menú final, el terapeuta podrá observar de manera rápida los resultados obtenidos en la partida.

7. RESULTADOS OBTENIDOS.

Para lograr un desarrollo completo y efectivo del juego serio de realidad aumentada *Coins* para la rehabilitación del miembro superior, ha sido fundamental realizar pruebas continuas y exhaustivas. En cada fase del proyecto, se han comparado los resultados obtenidos con los resultados esperados, lo que ha permitido identificar y corregir de manera óptima cualquier error de programación o interfaz de usuario. Este proceso de evaluación constante ha sido crucial para solucionar problemas y refinar el juego, asegurando una versión final que sea tanto eficaz como fluida en su funcionamiento. La iteración entre pruebas y ajustes ha garantizado que el juego cumpla con los objetivos terapéuticos previstos y ofrezca una experiencia positiva y efectiva para los usuarios de este.

7.1. Pruebas funcionamiento M3Display.

Como se explica en (A. Cisnal, 2023), para que el paciente tenga una buena experiencia utilizando el juego para agilizar su proceso de rehabilitación, se necesita un correcto funcionamiento del sistema M3Display, donde se encuentran recogidos la colección de juegos serios para la rehabilitación del miembro superior, donde se encuentra el juego *Coins*.

M3Display se compone tanto de la parte hardware, que cuenta con la estructura que incluye el monitor y la cámara como se muestra en la Figura 8, y la parte del software que incluye la detección de los puntos de referencia de la mano para incluir estos como *GameObjects* en Unity.

El funcionamiento eficiente de este sistema es esencial para garantizar que el juego *Coins* sea eficaz durante las sesiones de terapia, proporcionando al paciente una experiencia fluida y contribuyendo al proceso de rehabilitación.

7.2. Pruebas funcionamiento interfaz de usuario.

Una vez funcione correctamente el sistema M3Display, para ofrecer una experiencia satisfactoria a la hora de interactuar con la interfaz de usuario, se tiene que comprobar que todos los *GameObjects* integrados en el juego funcionan correctamente

Como se explicó en el quinto apartado, *Coins* incluye una variedad de elementos interactivos, como botones, *sliders*, contadores, temporizadores y paneles. Estos componentes deben funcionar sin problemas y responder adecuadamente a las acciones del usuario. Para lograr esto, se tienen que realizar pruebas exhaustivas en cada elemento de la interfaz. Esto incluye verificar que los botones ejecuten las funciones designadas, que los *sliders* ajusten las variables asociadas de manera precisa, y que los contadores y temporizadores reflejen correctamente el progreso y el tiempo transcurrido durante la sesión de juego.

Para poder asegurar que estas funciones se están ejecutando correctamente, a la hora de introducir nuevos *GameObjects* se tendrá que comprobar su funcionamiento en la ventana *Game*, para comprobar en todo momento que la programación ha sido correcta, y si hubiese algún problema poder corregir la lógica del objeto para corregir este funcionamiento.

7.3. Pruebas funcionamiento acciones *InGame*.

El objetivo principal del juego es que el paciente trate de llegar a la puntuación final antes de que se acabe el tiempo máximo permitido impuesto por el terapeuta, recogiendo todas las monedas que flotan sobre la trayectoria propuesta.

En las acciones *InGame*, para el buen funcionamiento del juego, interactúan una gran cantidad de *GameObjects* tanto en la interfaz de Uniy como dentro de los scripts que controlan la lógica del juego, por lo que es crucial que estos *GameObjects* interactúen entre sí de forma óptima.

Las monedas están todas colocadas en la mesa y dejan de estar activas las que no son necesarias en función de la trayectoria escogida, por lo que siempre hay que comprobar que este proceso ocurra de manera correcta.

A la hora de atrapar cada una de las monedas se activan varias funciones, para hacer desaparecer la moneda cuando ya ha sido cogida, la función que activa el sonido para el feedback auditivo, y sumar un punto a la puntuación. Además, para que el paciente, debido a algún fallo en la trayectoria de su miembro superior, no recoja una moneda no debida, se ha dispuesto a cada moneda una variable que solo permita recoger esa moneda en el instante adecuado, por lo que hay que comprobar también el buen funcionamiento de esa función.

7.4. Pruebas funcionamiento recogida de datos.

Para hacer el correcto seguimiento de la evolución del paciente, se crean archivos .CSV donde se recoge información esencial para comprobar el proceso. Cuando se ejecuta una partida se crea una carpeta llamada *CoinData*, si no está ya creada con el inicio de otras partidas ya finalizadas.

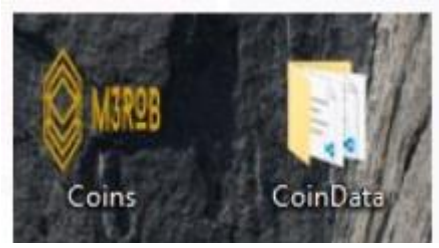

Figura 51: Creación carpeta *CoinData*. Fuente: Obtención propia.

Dentro de esta carpeta se guardarán los archivos .CSV con los datos obtenidos durante las partidas. Para poder diferenciar cada uno de estos archivos se guardarán con la fecha y la hora correspondientes al inicio de cada partida.

Nombre	Fecha de modificación	Tipo	Tamaño
17-08-2023 12.34.21	17/08/2023 13:18	Archivo de valores se...	15 KB
17-08-2023 12.48.52	17/08/2023 13:18	Archivo de valores se...	17 KB
17-08-2023 12.51.22	17/08/2023 13:18	Archivo de valores se...	15 KB
17-08-2023 13.09.26	17/08/2023 13:18	Archivo de valores se...	17 KB
17-08-2023 13.15.40	17/08/2023 13:18	Archivo de valores se...	38 KB

Figura 52: Archivos .CSV guardados dentro de *CoinData*. Fuente: Obtención propia.

En estos archivos se guardará la información necesaria para que el terapeuta pueda llevar un seguimiento óptimo de la evolución del paciente.

	A	B	C	D
1	DATE	17-08-2023 13.15.40		
2	Time	HandX	HandZ	points
3	0	0	0	0
4	6,593748	-96,04524	29,43095	1
5	6,611197	-96,04524	29,43095	
6	6,627581	-96,04524	29,43095	
7	6,650599	-95,41666	29,65952	
8	6,681465	-95,41666	29,65952	
9	6,708045	-95,41666	29,65952	
10	6,732762	-95,41666	29,65952	
11	6,792665	-95,41666	29,65952	
12	6,805486	-95,41667	29,65952	
13	6,828539	-95,30714	29,13334	
14	6,851794	-95,30714	29,13334	
15	6,878802	-95,30714	29,13334	
16	6,906843	-95,30714	29,13334	
17	6,929013	-95,30714	29,13334	
18	6,987245	-95,30714	29,13334	

	A	B	C	D
81	8,737147	-88,37621	25,60952	
82	8,75724	-88,15715	25,07381	
83	8,783719	-88,15715	25,07381	
84	8,810081	-88,15715	25,07381	
85	8,838423	-88,15715	25,07381	
86	8,871371	-88,15715	25,07381	2
87	8,931819	-88,15715	25,07381	
88	8,944055	-87,1619	24,00714	
89	8,96688	-87,1619	24,00714	
90	8,989382	-87,1619	24,00714	
91	9,014552	-87,1619	24,00714	
92	9,045021	-87,1619	24,00714	
93	9,104516	-87,1619	24,00714	
94	9,111672	-87,1619	24,00714	
95	9,130597	-86,61191	23,53571	
96	9,156287	-86,61191	23,53571	
97	9,182359	-86,61191	23,53571	
98	9,21066	-86,61191	23,53571	

Figura 53: Tablas con los datos guardados para el seguimiento de la evolución. Fuente: Obtención propia.

Como se observa en la Figura 53, en una prueba realizada el día 17/08/2023 se han recogido los datos de forma óptima. El terapeuta obtiene una tabla con la posición de la mano en todo momento, el tiempo y el punto que se recoge en cada instante.

El programa comienza a tomar datos en el momento que se recoge la primera moneda, para que se pueda visualizar de forma rápida el tiempo de respuesta del paciente. El terapeuta también puede observar los puntos que obtiene el paciente a lo largo de la partida, y en qué momento obtiene se obtienen, para poder estudiar qué tramos de la trayectoria le cuestan más o menos al paciente.

	A	B	C	D
1	DATE	17-08-2023 13.15.40		
2	Time	HandX	HandZ	points
3	0	0	0	0
4	6,593748	-96,04524	29,43095	1
5	6,611197	-96,04524	29,43095	
6	6,627581	-96,04524	29,43095	

	A	B	C	D
82	8,75724	-88,15715	25,07381	
83	8,783719	-88,15715	25,07381	
84	8,810081	-88,15715	25,07381	
85	8,838423	-88,15715	25,07381	
86	8,871371	-88,15715	25,07381	2
87	8,931819	-88,15715	25,07381	
88	8,944055	-87,1619	24,00714	

Figura 54: Obtención de puntos a lo largo de la partida. Fuente: Obtención propia.

Con los datos obtenidos podemos realizar todo tiempo de gráficos que servirán para comparar los resultados obtenidos con los objetivos.

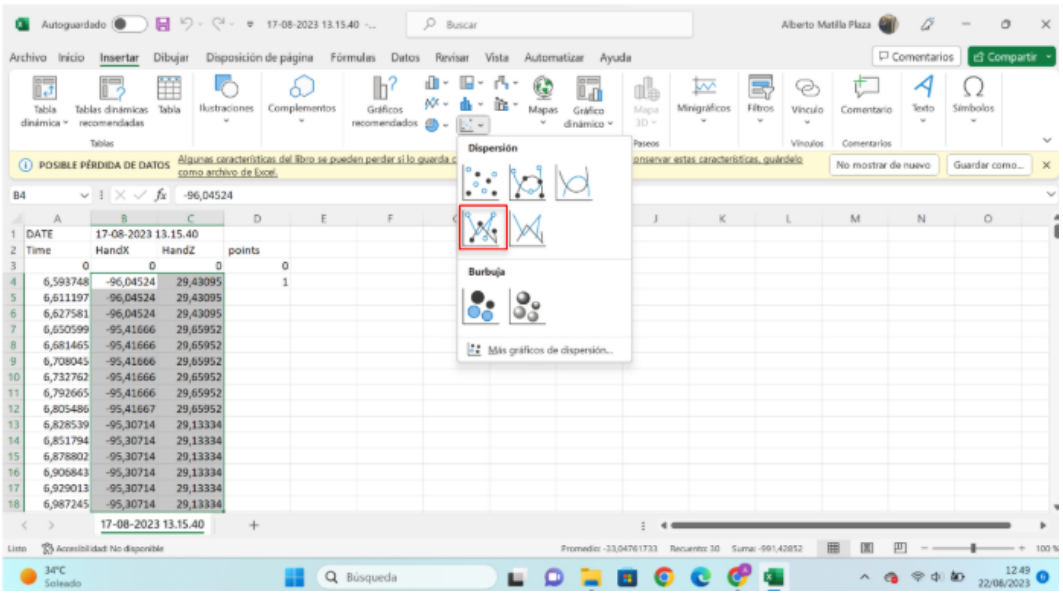

Figura 55: Obtención de gráficas para la comparación de resultados. Fuente: Obtención Propia.

Un ejemplo de gráfica que se puede generar, como se muestra en la Figura 55, sería la posición de la mano en cada momento, para poder recrear la trayectoria descrita por el paciente.

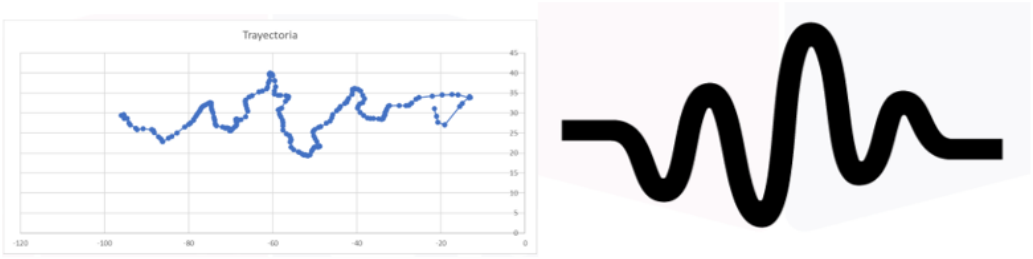

Figura 56: Comparación de la trayectoria descrita por el paciente con la trayectoria objetivo. Fuente: Obtención propia.

En este experimento de prueba se escogió una de las trayectorias de la mayor dificultad, en concreto en la que el paciente tiene que describir una trayectoria curva de forma horizontal. Como se observa en la Figura 56, al graficar los datos obtenidos de la posición de la mano, se puede comparar de forma aproximada la trayectoria descrita con la trayectoria objetivo. En este caso se

puede concluir que se ha realizado una partida con un resultado positivo, obteniendo la puntuación máxima al concluir la partida y habiendo completado el movimiento con precisión.

8. CONCLUSIONES.

Tras la finalización de este proyecto, y observando los resultados obtenidos en el apartado anterior, se puede concluir que se ha cumplido la finalidad principal de este con éxito. Desarrollar un juego serio de realidad aumentada, *Coins*, como parte integral del sistema de rehabilitación M3Display para el miembro superior.

Las pruebas realizadas han confirmado que tanto la interfaz de usuario, como el sistema de obtención de datos funcionan correctamente y responden adecuadamente a las acciones del usuario. Esto asegura que *Coins*, además de ofrecer un entorno de juego eficaz e intuitivo, también sirve como una herramienta precisa para la terapia del miembro superior, y poder compaginar su uso con métodos más tradicionales con el mismo propósito, permitiendo al paciente a adherirse a este proceso, mejorando los resultados de este en un menor tiempo.

8.1. Posibles mejoras en el proyecto.

El proyecto ha logrado el propósito de combinar la tecnología emergente de la realidad aumenta con el sistema M3Display desarrollado por ITAP, y así creando una solución innovadora para complementar a las técnicas tradicionales de la rehabilitación del miembro superior. La integración del juego *Coins* en este sistema demuestra que la industria de los videojuegos llega más allá del mundo del entretenimiento, y se puede utilizar en otros entornos, como en este caso, orientado a la sanidad. La adaptación y mejora continua de estas tecnologías permite contemplar un aumento de sistemas similares a este, e incluso la integración de nueva tecnología a este proyecto proporcionando mejoras.

8.1.1. Implementación Inteligencia Artificial.

Con el gran incremento que está sufriendo el uso de la inteligencia artificial en una gran cantidad de industrias, es obvio pensar que la implementación de esta tecnología en el sistema M3Display, y en particular en el juego *Coins*.

La idea de inteligencia artificial se puede resumir en incorporar a una computadora la forma de "pensar" de un ser humano. Una idea de implementación de inteligencia artificial al juego *Coins* podría ser la utilización de *Machine Learning*, o aprendizaje automático. En el campo de la inteligencia artificial, el *Machine Learning* consiste en la utilización de diferentes algoritmos para que el sistema obtenga un aprendizaje para realizar tareas de forma autónoma.

Como el juego *Coins* consiste en tratar de describir con el miembro superior una trayectoria definida y elegida por el terapeuta, se puede implementar un algoritmo de *Machine Learning* para que el sistema esté constantemente obteniendo datos de las partidas que se realizan y poder dibujar por si solo trayectorias nuevas, de diferentes dificultades, en función del desempeño del paciente en sesiones previos, o incluso en esa misma sesión.

8.1.2. Implementación de gafas de Realidad Virtual.

La integración de tecnología en la rehabilitación del miembro superior se ha materializado con el sistema M3Display, el cual incluye una mesa donde se proyectan las monedas, y un brazo que sostiene un monitor y una cámara, como se describe anteriormente. Este hardware permite una implementación eficaz del juego *Coins* en el centro de rehabilitación.

Sin embargo, en ciertas ocasiones, la necesidad de opciones de terapia a distancia ha llevado a explorar alternativas innovadoras para aquellos pacientes que enfrentan dificultades para asistir a las sesiones presenciales. En este caso puede ser de gran ayuda la utilización de gafas de realidad virtual en algunas de estas sesiones, ofreciendo una solución práctica y avanzada.

En este escenario, el centro de rehabilitación podría proporcionar estas gafas de realidad virtual a estos pacientes, garantizando así la continuidad del tratamiento en la comodidad de su hogar. Con las gafas de realidad virtual, se crea un entorno simulado similar al sistema M3Display, proyectando una mesa virtual sobre la que flotarán un número de monedas dibujando una trayectoria, al igual que el juego original disponible en el centro de rehabilitación. Esto permite que el paciente continúe su progreso en la terapia de rehabilitación, siguiendo las mismas trayectorias a las disponibles en el entorno físico del centro, realizando los ejercicios de forma efectiva.

8.2. Conocimientos aplicados y validación del sistema.

El desarrollo de este proyecto ha requerido adquirir habilidades de programación en lenguajes como Python, C# y el motor de videojuegos Unity. Han sido esenciales conocimientos de Python para entender el procesamiento de las imágenes para la posterior manipulación de los datos obtenidos. También ha sido necesaria la compresión del funcionamiento de la programación en bloques propuesta por el motor Unity, al igual que el aprendizaje del lenguaje C# para programar la lógica y comportamiento del juego.

Con estos conocimientos adquiridos se ha logrado un correcto funcionamiento del juego en las pruebas realizadas en el laboratorio. Aunque estas simulaciones han resultado satisfactorias, estas han sido efectuadas por personas ajenas al campo de la medicina, y especialmente de la rehabilitación. Además, se han llevado a cabo con personas sanas, sin discapacidades neuromotoras que afecten la movilidad del miembro superior. Para una validación completa, será esencial obtener una retroalimentación en un entorno clínico similar al expuesto en los objetivos del proyecto, probando el sistema con fisioterapeutas y pacientes que presenten necesidades reales de rehabilitación. Esto permitirá una evaluación más precisa y la adaptación necesaria del sistema en condiciones reales.

BIBLIOGRAFÍA

A. Cisnal, G. A.-L. (2023). *M3Display: Sistema de realidad aumentada para la rehabilitación de la función motora del miembro superior.* Instituto de las Tecnologías Avanzadas de la Producción (ITAP), Universidad de Valladolid, Valladolid, España. XLI Congreso Anual de la Sociedad Española de Ingeniería Biomédica. 22-24 Noviembre 2023. pp. 161-164.

APD. (2023). *Tecnología en la medicina: beneficios y avances más importantes.* Redacción APD.https://www.apd.es/tecnologia-medicina-beneficios-avances/. Último acceso: 26/6/2024.

M. Khademi, H. Mousavi Hondori, C. V. Lopes, L. Dodakian and S. C. Cramer (2012). *Haptic Augmented Reality to Monitor Human Arm's. 2012 IEEE-EMBS Conference on Biomedical Engineering and Sciences*, Langkawi, Malaysia. pp. 892-895, doi:10.1109/IECBES.2012.6498168

Astudillo Aguiar, A. (2023). *Dispositivo de ayuda a la rehabilitación neuromotora de personas con daño cerebral adquirido.* Proyecto Fin de Grado, Arquitectura y Tecnología de Sistemas Informáticos (UPM).

Brizuela, G. S. (2021). *Documentación M3Display - Toolkit de desarrollo.*

BVS. (1963). *Paresia.* https://decs.bvsalud.org/es/ths/resource/?id=10480. Último acceso: 26/6/2024.

ClinicaUner. (2023). *Realidad virtual y aumentada: mejorando la rehabilitación con nuevas tecnologías.* Recuperado el Agosto de 2023. https://clinicauner.es/realidad-virtual-y-aumentada-mejorando-la-rehabilitacion-neurologica-con-nuevas-tecnologias/. Último acceso: 26/6/2024.

Díez, Ú. C. (2020). *Robótica para la rehabilitación.*

EvaluateMedTech. (2018). *Evolución anual de la inversión en investigación y desarrollo en sector de tecnología médica a nivel global desde 2011 a 2024.*

EvaluateMedTech. (2024).

EvaluateMedTech. (2024). *Variación Porcentual del mercado global de tecnología médica de 2010 a 2022.*

Fan Zhang, V. B.-L. (2020). *MediaPipe Hands: On-device Real-time Hand Traking.* CVPR Workshop on Computer Vision for Augmented and Virtual Reality, Seattle, WA, USA.

Fisioform. (2023). *Avances en fisioterapia y tendencias emergentes.* https://fisioformcursos.com/avances-en-fisioterapia-y-tendencias-emergentes/. Último acceso: 26/6/2024.

FisioOnline. (2024). *Paresia.* https://www.fisioterapia-online.com/glosario/paresia. Último acceso: 26/6/2024.

Gobierno de España. (2020). *Plan España Digital 2025.* Programas para el Avance Digital.

Hossein Mousavi Hondori, M. K. (2016). *Choice of Human–Computer Interaction Mode in Stroke Rehabilitation.* SAGE journals. Neurorehabilitation and Neural Repair. doi:10.1177/1545968315593805

Insider, A., & Intelligence, A. (2024). *Número de usuarios de realidad aumentada móvil a nivel mundial desde 2019 hasta 2024.*

Lynn E. Fiellin, K. D. (2014). *Videogames, Here for Good.* Artículo, Pediatrics, 134(5), pp. 849-851. doi:10.1542/peds.2014-0941

Ordoñez, J. L. (2020). *Realidad Virtual y Realidad Aumentada.* CEDRO.

Park, S. (2023). *From pixels to high-fives: LF U+ fosters real connectiones for virtual offices.* Unity blog.

Plaza, D. M. (2023). *Entorno de Realidad Aumentada para Rehabilitación de Miembro Superior.* Trabajo Fin de Grado, Universidad de Valladolid, Departamento de Ingeniería de Sistemas y Automática, Valladolid, España.

RAE. (2024).

Report, U. G. (2022). *Number of games by type.*

Team, M. (2021). *MediaPipe Hands.*

Team, S. (2022). *Game Engines and their use in Game Development.* DATA.

Toro, P. d. (2023). *Avances Tecnológicos en fisioterapia y su impacto en la rehabilitación.* Recuperado el 21 de Agosto de 2023

Wirotius, J. M. (1999). *Historia de la rehabilitación.* París: Editions Scientifiques et Médicales. Elsevier SAS.

ANEXO I – *Script PlayerHand*

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

/// <summary>
/// Class Name: PlayerHand
/// Author: Alberto Matilla Plaza
/// Function: Gets the spatial coordinates of the mean position of the hand reference points.
/// </summary>

public class PlayerHand : MonoBehaviour
{

    // Start is called before the first frame update
    void Start()
    {
        transform.position = new Vector3(0, 0, 0);
    }

    // Update is called once per frame
    void Update()
    {

    }

    /// <summary>
    /// Called to return the spatial coordinates of the mean position of the hand reference
points.
    /// </summary>
    /// <returns></returns>
    public Vector3 HandMeanPosition()
    {
        GameObject[] objects = HandLandmarks();
        Vector3 AllPositions = Vector3.zero;
        int _Pos = 0;

        foreach (GameObject obj in objects)
        {
            AllPositions += obj.transform.position;
            _Pos++;
        }

        Vector3 MeanPosition = AllPositions / (float)_Pos;
        return MeanPosition;
    }

    /// <summary>
    /// Called to return the hand landmarks as an object for the game.
    /// </summary>
    /// <returns></returns>
    private GameObject[] HandLandmarks()
    {
        LayerMask layer;
        int layerNumber = LayerMask.NameToLayer("Landmark");
        layer = 1 << layerNumber;

        Collider[] colliders = Physics.OverlapSphere(Vector3.zero, Mathf.Infinity, layer);
        GameObject[] objects = new GameObject[colliders.Length];
        for(int i = 0; i < colliders.Length; i++)
        {
            objects[i] = colliders[i].gameObject;
        }

        return objects;
    }

}
```

ANEXO II – *Script Score*

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

/// <summary>
/// Class Name: Score
/// Author: Alberto Matilla Plaza
/// Function: Counts and write the current score in the interface during the game.
/// </summary>

public class Score : MonoBehaviour
{
    // Class Variables
    public GameTimer gameTimer;
    private float points;
    public TextMeshProUGUI score;
    public TextMeshProUGUI finalscore;
    private float button;
    private float MaxScore;
    private float MaxScoreAnterior = 0;
    private float timer;

    // Start is called before the first frame update
    private void Start()
    {
        points = 0;
        timer = gameTimer.TimeValue();
        score.text = points.ToString("0");
    }

    // Update is called once per frame
    private void Update()
    {
        if (button == 4)
            MaxScore = 8;
        else if (button == 2)
            MaxScore = 6;
        else if (button == 5 || button == 1)
            MaxScore = 9;
        else if (button == 3)
            MaxScore = 14;
        else if (button == 9 || button == 6)
            MaxScore = 16;
        else if (button == 8 || button == 7)
            MaxScore = 22;
        else
            MaxScore = MaxScoreAnterior;

        timer = gameTimer.TimeValue();
        score.text = points.ToString("");
        finalscore.text = points.ToString("");
        MaxScoreAnterior = MaxScore;

    }

    /// <summary>
    /// Called when a coin is collected to add a point to the score.
    /// </summary>
    /// <param name="NewPoints"></param>
    public void SumPoints(float NewPoints)
    {
        points += NewPoints;
    }

    /// <summary>
    /// Called to return the current score.
    /// </summary>
    /// <returns></returns>
    public float PointsValue()
    {
        return points;
    }

    /// <summary>
```

```
    /// Called when the horizontal line button is pressed.
    /// </summary>
    public void ButtonHorizontalLine()
    {
        button = 1;
    }

    /// <summary>
    /// Called when the vertical line button is pressed.
    /// </summary>
    public void ButtonVerticalLine()
    {
        button = 2;
    }

    /// <summary>
    /// Called when the square button is pressed.
    /// </summary>
    public void ButtonSquare()
    {
        button = 3;
    }

    /// <summary>
    /// Called when the diagonal button is pressed.
    /// </summary>
    public void ButtonDiagonal()
    {
        button = 4;
    }

    /// <summary>
    /// Called when the stairs button is pressed.
    /// </summary>
    public void ButtonStairs()
    {
        button = 5;
    }

    /// <summary>
    /// Called when the N button is pressed.
    /// </summary>
    public void ButtonN()
    {
        button = 6;
    }

    /// <summary>
    /// Called when the horizontal curve button is pressed.
    /// </summary>
    public void ButtonHorizontalCurve()
    {
        button = 7;
    }

    /// <summary>
    /// Called when the vertical curve button is pressed.
    /// </summary>
    public void ButtonVerticalCurve()
    {
        button = 8;
    }

    /// <summary>
    /// Called when the circle button is pressed.
    /// </summary>
    public void ButtonCircle()
    {
        button = 9;
    }
    /// <summary>
    /// Called when the MainMenu button is pressed. The score take a value of 0
    /// and the time value is restarted.
    /// </summary>
    public void ButtonMainMenuPressed()
    {
        points = points - points;
        timer = gameTimer.TimeValue();
        score.text = points.ToString("0");
    }

}
```

ANEXO III – *Script GameTimer*

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

/// <summary>
/// Class Name: GameTimer
/// Author: Alberto Matilla Plaza
/// Function: Creates a counter and a countdown of the game time.
/// </summary>

public class GameTimer : MonoBehaviour
{
    // Class Variables
    [SerializeField]
    public Slider MaxTime;
    [SerializeField]
    public TextMeshProUGUI Timer;
    [SerializeField]
    public Text TimerEndGame;
    private float ActualTime;
    private float ActualTimeEnd;
    private float TimeLeft;
    private bool ActivateTime = false;

    // Start is called before the first frame update
    public void Start()
    {
        ActivateTimer();
    }

    // Update is called once per frame
    public void Update()
    {
        if(ActivateTime)
        {
            ChangeTimer();
        }
    }

    /// <summary>
    /// Called to change the time text in the in game menu
    /// </summary>
    private void ChangeTimer()
    {
        ActualTime -= Time.deltaTime;
        ActualTimeEnd += Time.deltaTime;
        TimeLeft = (float)MaxTime.value * 60f - ActualTimeEnd;

        if(ActualTime >= 0)
        {
            double ActualTimeD = double.Parse(ActualTime.ToString());
            double TimeLeftD = double.Parse(TimeLeft.ToString());
            double minutes = Math.Ceiling((ActualTimeD / 60) - 1);
            double minutesLeft = Math.Ceiling((TimeLeftD / 60) - 1);
            float seconds = ActualTime % 60;
            float secondsLeft = TimeLeft % 60;
            Timer.text = minutes.ToString("00´") + seconds.ToString("00´´");
            TimerEndGame.text = minutesLeft.ToString("00´") +
secondsLeft.ToString("00´´");
        }

        if(ActualTime <= 0)
        {
```

```
            Debug.Log("Game Lost");
            DesactivateTimer();
        }
    }

    /// <summary>
    /// Called to know if the game is set to activate the time
    /// </summary>
    /// <param name="state"></param>
    public void GameState(bool state)
    {
        ActivateTime = state;
    }

    /// <summary>
    /// Called to active the timer
    /// </summary>
    public void ActivateTimer()
    {
        ActualTime = (float)MaxTime.value;
        ActualTime *= 60f;
        GameState(true);
    }

    /// <summary>
    /// Called to desactivate the timer
    /// </summary>
    public void DesactivateTimer()
    {
        GameState(false);
    }

    /// <summary>
    /// Called to return the current value of the time countdoewn
    /// </summary>
    /// <returns></returns>
    public float TimeValue()
    {
        return ActualTime;
    }

    /// <summary>
    /// Called to return the current value of the time counter
    /// </summary>
    /// <returns></returns>
    public float TimePassed()
    {
        return ActualTimeEnd;
    }

    /// <summary>
    /// Called when the start button is pressed.
    /// The time for the countdown takes the value of the Slider Time.
    /// </summary>
    public void ButtonStartPressed()
    {
        ActualTime = (float)MaxTime.value;
        ActualTime *= 60f;
    }

    /// <summary>
    /// Called when the MainMenu button is pressed.
    /// The time for the counter takes the value 0.
    /// </summary>
    public void ButtonMainMenuPressed()
    {
        ActualTimeEnd = 0;
        ActivateTimer();
    }
}
```

ANEXO IV – *Script EndGame*

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

/// <summary>
/// Class Name: EndGame
/// Author: Alberto Matilla Plaza
/// Function: Change Panel to the Final Menu when the player reach de Maximum score
/// or runs out of time.
/// </summary>

public class EndGame : MonoBehaviour
{
    // Class Variables
    public Score score;
    public GameTimer gameTimer;
    private float button;
    private float buttonAnterior;
    private float points;
    private float timer;
    private float MaxScore;
    private float MaxScoreAnterior = 0;
    public GameObject ActualPanel;
    public GameObject NextPanel;

    // Start is called before the first frame update
    void Start()
    {
        points = score.PointsValue();
        timer = gameTimer.TimeValue();
    }

    // Update is called once per frame
    void Update()
    {
        if (button == 4)
            MaxScore = 8;
        else if (button == 2)
            MaxScore = 6;
        else if (button == 5 || button == 1)
            MaxScore = 9;
        else if (button == 3)
            MaxScore = 14;
        else if (button == 9 || button == 6)
            MaxScore = 16;
        else if (button == 8 || button == 7)
            MaxScore = 22;
        else
            MaxScore = MaxScoreAnterior;

        points = score.PointsValue();
        timer = gameTimer.TimeValue();
        MaxScoreAnterior = MaxScore;
        buttonAnterior = button;


        ChangePanel(MaxScore, points, timer);
    }

    /// <summary>
    /// Called to change panel when the player has finished the game.
    /// </summary>
    /// <param name="MaxScore"></param>
    /// <param name="score"></param>
    /// <param name="time"></param>
```

```
void ChangePanel(float MaxScore, float score, float time)
{
    if(score == MaxScore || time <= 0)
    {
        ActualPanel.SetActive(false);
        NextPanel.SetActive(true);
    }
}

/// <summary>
/// Called when the horizontal line button is pressed
/// </summary>
public void ButtonHorizontalLine()
{
    button = 1;
}

/// <summary>
/// Called the vertical line button is pressed
/// </summary>
public void ButtonVerticalLine()
{
    button = 2;
}

/// <summary>
/// Called when the square button is pressed
/// </summary>
public void ButtonSquare()
{
    button = 3;
}

/// <summary>
/// Called when the diagonal button is pressed
/// </summary>
public void ButtonDiagonal()
{
    button = 4;
}

/// <summary>
/// Called when the stairs button is pressed
/// </summary>
public void ButtonStairs()
{
    button = 5;
}

/// <summary>
/// Called when the N button is pressed
/// </summary>
public void ButtonN()
{
    button = 6;
}

/// <summary>
/// Called when the horizontal curve button is pressed
/// </summary>
public void ButtonHorizontalCurve()
{
    button = 7;
}

/// <summary>
/// Called when the vertical curve button is pressed
/// </summary>
public void ButtonVerticalCurve()
{
    button = 8;
}
```

```
    /// <summary>
    /// Called when the circle button is pressed
    /// </summary>
    public void ButtonCircle()
    {
        button = 9;
    }

    /// <summary>
    /// Returns the maximum score
    /// </summary>
    /// <returns></returns>
    public float FinalScore()
    {
        return MaxScore;
    }

    /// <summary>
    /// Puts the score to zero when the Main Menu Button is pressed
    /// </summary>
    public void ButtonMainMenuPressed()
    {
        points = 0;
    }

    public void ButtonStartGamePressed()
    {
        if (button == buttonAnterior)
        {
            Debug.Log("Maxima Puntuacion Actual:" + MaxScore);
            Start();
        }
    }
}
```

ANEXO V – *Script UICollisionDetection*

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

public class UICollisionDetection : MonoBehaviour
{

    [System.Serializable]
    public class TouchedButtonEvent : UnityEvent<float, float> { }
    [SerializeField]
    private UnityEvent ButtonEnter = new UnityEvent();
    [SerializeField]
    private TouchedButtonEvent ButtonTouched = new TouchedButtonEvent();
    [SerializeField]
    private UnityEvent ButtonTriggered = new UnityEvent();
    [SerializeField]
    private UnityEvent ButtonReleased = new UnityEvent();
    public Score score;
    public GameTimer gameTimer;
    private SoundManager coinSound;
    public float NumAccess;
    private float access;

    private float _elapsedTime = 0f;
    private bool _alreadyTriggered = false;
    [SerializeField]
    private float _secondsToTrigger = 3f;
    private int _numLandmarks = 0;

    // Start is called before the first frame update
    void Start()
    {
        float time = gameTimer.TimePassed();
        coinSound = FindObjectOfType<SoundManager>();
    }

    // Update is called once per frame
    void Update()
    {

        if (!_alreadyTriggered && _elapsedTime > _secondsToTrigger &&
score.PointsValue() == NumAccess)
        {

            ButtonTriggered.Invoke();
            _alreadyTriggered = true;
            score.SumPoints(1);
            coinSound.AudioSelected(1f);
            Debug.Log("Puntos Actuales: " + score.PointsValue());
        }
    }



    public void testPressed(float t, float total)
    {
        if (t < total)
        {
            byte temp = (byte)(255 - t / total * 255);
            gameObject.GetComponent<Renderer>().material.color = new Color32(temp,
255, 255, 255);
        }
    }

    private void OnTriggerEnter(Collider other)
```

```
    {
        if (other.gameObject.layer == LayerMask.NameToLayer("Landmark"))
        {
            print("Entrando");
            _numLandmarks += 1;
            if (_numLandmarks == 1)
            {
                ButtonEnter.Invoke();
            }
        }
    }

    private void OnTriggerStay(Collider other)
    {
        if (other.gameObject.layer == LayerMask.NameToLayer("Landmark"))
        {
            _elapsedTime += Time.deltaTime / _numLandmarks;
            ButtonTouched.Invoke(_elapsedTime, _secondsToTrigger);
        }
    }

    private void OnTriggerExit(Collider other)
    {
        if (other.gameObject.layer == LayerMask.NameToLayer("Landmark"))
        {
            if (_numLandmarks == 1)
            {
                _elapsedTime = 0;
                _alreadyTriggered = false;
                ButtonReleased.Invoke();
            }
            _numLandmarks -= 1;
        }
    }
    public void ButtonMainMenuPressed()
    {
        gameObject.SetActive(true);
        float time = gameTimer.TimePassed();
        score.SumPoints(-score.PointsValue());
    }
    public void ButtonStartPressed()
    {
        _elapsedTime = 0f;
        _numLandmarks = 0;
}
}
```

ANEXO VI – *Script SaveDataCoins*

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.IO;
using UnityEngine;
using UnityEngine.SocialPlatforms.Impl;

/// <summary>
/// Class Name: SaveDataCoins
/// Author: Alberto Matilla Plaza
/// Function: Save the Data collected during the game in a .CSV
/// </summary>

public class SaveDataCoins : MonoBehaviour
{
    // Class Variables
    private float _time;
    private float _currentTime;
    private StreamWriter _file;
    private string _filePath = "";
    public GameTimer gameTimer;
    public Score score;
    public PlayerHand HandMeanPosition;
    private float MaxScore;
    private float button;
    private float points;
    private bool inGame;
    private float ContPoints;

    // Start is called before the first frame update
    void Start()
    {
        _time = 0f;
        _currentTime = 0f;
        points = 0;
        ContPoints = 1;
        inGame = false;

    }

    // Update is called once per frame
    private void Update()
    {
        _time = gameTimer.TimeValue();
        _currentTime = gameTimer.TimePassed();
        points = score.PointsValue();

        if (button == 4)
            MaxScore = 8;
        else if (button == 5 || button == 1 || button == 2)
            MaxScore = 9;
        else if (button == 9 || button == 6 || button == 3)
            MaxScore = 16;
        else if (button == 8 || button == 7)
            MaxScore = 22;

        if (inGame == true)
        {

            if (points >= 1 && points <= MaxScore && _time > 0 &&
Double.Parse(_currentTime.ToString()) > 0.01)
            {
                SaveOnCSV();
                if(points == MaxScore)
                {
                    CloseFile();
```

```
                    inGame = false;
                }
            }
        }

    }

    /// <summary>
    /// Called every frame to save the data in the .CSV
    /// </summary>
    private void SaveOnCSV()
    {
        Vector3 handPosition = HandMeanPosition.HandMeanPosition();

        if (!float.IsNaN(handPosition.x) || !float.IsNaN(handPosition.y) ||
!float.IsNaN(handPosition.z))
        {
            if(points != ContPoints)
            {
                _file.WriteLine($"{Double.Parse(_currentTime.ToString())};
{handPosition.x}; {handPosition.z}");
                Debug.Log(ContPoints + "  " + points);
            }
            else
            {
                _file.WriteLine($"{Double.Parse(_currentTime.ToString())};
{handPosition.x}; {handPosition.z}; {points}");
                Debug.Log(ContPoints + "  " + points);
                ContPoints++;

            }

        }

    }

    /// <summary>
    /// Called to create the .CSV and open it to save the data.
    /// </summary>
    private void CreateAndOpenCSV()
    {
        string folder = "CoinData";
        if (!Directory.Exists(folder))
        {
            Directory.CreateDirectory(folder);
        }
        string actualDate = DateTime.Now.ToString("dd-MM-yyyy HH.mm.ss");
        string fileName = actualDate + ".csv";

        _filePath = Path.Combine(folder, fileName);

        _file = new StreamWriter(_filePath, true);

        _file.WriteLine($"DATE; {actualDate}");
        _file.WriteLine($"Time; HandX; HandZ; points");
        _file.WriteLine($"0; 0; 0; 0");
    }

    /// <summary>
    /// Called to close the .CSV when the game is over.
    /// </summary>
    private void CloseFile()
    {
        bool fileExist = File.Exists(_filePath);
        if(fileExist)
        {
            _file.Close();
            gameObject.SetActive(false);
        }
    }

    /// <summary>
```

```
/// Called when the exit button is pressed so the .CSV can be closed.
/// </summary>
public void ButtonExitPressed()
{
    CloseFile();
    inGame = false;
}

/// <summary>
/// Called when the start buton is pressed so the .CSV can be opened.
/// </summary>
public void ButtonStartPressed()
{
    _time = 0f;
    CreateAndOpenCSV();
    inGame = true;
}

/// <summary>
/// Called when the pause button is pressed to not save data during that time.
/// </summary>
public void ButtonPausePressed()
{
    inGame = false;
}

/// <summary>
/// Called when the continue button is pressed to restart saving data.
/// </summary>
public void ButtonContinuePressed()
{
    inGame = true;
}

/// <summary>
/// Called when the horizontal line button is pressed.
/// </summary>
public void ButtonHorizontalLine()
{
    button = 1;
}

/// <summary>
/// Called when the vertical line button is pressed.
/// </summary>
public void ButtonVerticalLine()
{
    button = 2;
}

/// <summary>
/// Called when the square button is pressed.
/// </summary>
public void ButtonSquare()
{
    button = 3;
}

/// <summary>
/// Called when the diagonal button is pressed.
/// </summary>
public void ButtonDiagonal()
{
    button = 4;
}

/// <summary>
/// Called when the stairs button is pressed.
/// </summary>
public void ButtonStairs()
{
    button = 5;
```

```
    }

    /// <summary>
    /// Called when the N button is pressed.
    /// </summary>
    public void ButtonN()
    {
        button = 6;
    }

    /// <summary>
    /// Called when the horizontal curve button is pressed.
    /// </summary>
    public void ButtonHorizontalCurve()
    {
        button = 7;
    }

    /// <summary>
    /// Called when the vertical curve button is pressed.
    /// </summary>
    public void ButtonVerticalCurve()
    {
        button = 8;
    }

    /// <summary>
    /// Called when the circle button is pressed.
    /// </summary>
    public void ButtonCircle()
    {
        button = 9;
    }
}
```

ANEXO VII – *Script SoundManager*

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

/// <summary>
/// Class Name: SoundManager
/// Author: Alberto Matilla Plaza
/// Function: Controls the audio of the game
/// </summary>

public class SoundManager : MonoBehaviour
{
    // Class Variables
    [SerializeField] private AudioClip audio;

    private AudioSource controlAudio;

    // Start is called before the first frame update
    void Start()
    {
        controlAudio = GetComponent<AudioSource>();
    }

    // Update is called once per frame
    void Update()
    {

    }

    /// <summary>
    /// Called when a coin is collected to play the sound
    /// </summary>
    /// <param name="volumen"></param>
    public void AudioSelected(float volumen)
    {
        controlAudio.PlayOneShot(audio, volumen);
    }
}
```

ANEXO VIII – *Script CoinAnimation*

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

/// <summary>
/// Class Name: CoinAnimation
/// Author: Alberto Matilla Plaza
/// Function: Creates an animation to make the coins move during the game.
/// </summary>

public class CoinAnimation : MonoBehaviour
{
    // Class Variables
    public float amplitude = 0.5f;
    public float frecuency = 1.0f;
    Vector3 posOrigin = new Vector3();
    Vector3 temPos = new Vector3();

    // Start is called before the first frame update
    void Start()
    {
        posOrigin = transform.position;
    }

    // Update is called once per frame
    void Update()
    {
        temPos = posOrigin;
        temPos.y += Mathf.Sin(Time.fixedTime * Mathf.PI * frecuency) * amplitude;
        transform.position = temPos;
        transform.Rotate(new Vector3(0f, 0f, 50f) * Time.deltaTime);
    }
}
```

ANEXO IX – *Script SliderDifficultyValue*

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

/// <summary>
/// Class Name: SliderDifficultyValue
/// Author: Alberto Matilla Plaza
/// Function: Shows a text indicating the chosen difficulty level with the Slider.
/// </summary>

public class SliderDifficultyValue : MonoBehaviour
{
    // Class Variables
    public Slider Slider;
    public Text SliderText;
    public string ValueName;

    // Start is called before the first frame update
    void Start()
    {
        float minValue = Slider.minValue;
        Slider.minValue = minValue;
        PlayerPrefs.SetInt(ValueName, (int)minValue);
        int minValueInt = (int)minValue;
        Visualize(minValueInt);
    }
    // Update is called once per frame
    void Update()
    {

    }
    /// <summary>
    /// Called to visualize the value of the Slider Difficulty.
    /// </summary>
    public void SliderChange()
    {
        int value = (int)Slider.value;
        Visualize(value);
    }
    /// <summary>
    /// Called to visualize the difficulty selected near the Slider Difficulty.
    /// </summary>
    /// <param name="value"></param>
    public void Visualize(int value)
    {
        PlayerPrefs.SetInt(ValueName, value);
        string Difficulty;
        if(value == 0)
        {
            Difficulty = "Low";
            SliderText.text = Difficulty;
        }
        else if(value == 1)
        {
            Difficulty = "Medium";
            SliderText.text = Difficulty;
        }
        else if(value == 2)
        {
            Difficulty = "High";
            SliderText.text = Difficulty;
        }
    }
}
```

ANEXO X – *Script SliderTimer*

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

/// <summary>
/// Class Name: SliderTime
/// Author: Alberto Matilla Plaza
/// Function: Shows a text indicating the chosen maximum time with the Slider.
/// </summary>

public class SliderTime : MonoBehaviour
{
    // Class Variables
    public Slider Slider;
    public Text SliderTextTime;
    public string ValueName;

    // Start is called before the first frame update
    void Start()
    {
        float minValue = Slider.minValue;
        Slider.minValue = minValue;
        PlayerPrefs.SetInt(ValueName, (int)minValue);
        int minValueInt = (int)minValue;
        Visualize(minValueInt);
    }

    // Update is called once per frame
    void Update()
    {

    }

    /// <summary>
    /// Called to visualize the value of the Slider Time.
    /// </summary>
    public void SliderChange()
    {
        int value = (int)Slider.value;
        Visualize(value);
    }

    /// <summary>
    /// Called to visualize the maximum time selected near the Slider Time.
    /// </summary>
    /// <param name="value"></param>
    public void Visualize(int value)
    {
        PlayerPrefs.SetInt(ValueName, value);
        string TimeText = string.Format($"{value}´");
        SliderTextTime.text = TimeText;
    }
}
```

ANEXO XI – Manual de Usuario

MANUAL DE USUARIO

1.- ¿Qué parámetros tiene el juego y cómo se configuran?

El juego tiene dos parámetros, el tiempo máximo de duración de partida y la dificultad. Dentro de la dificultad además hay que seleccionar la trayectoria deseada.

Figura 57: Manual de usuario. Menú inicial. Fuente: Obtención propia.

Antes de comenzar la partida hay que configurar los parámetros, para ello hay que pulsar el botón “OPTIONS” que se encuentra en el menú inicial del videojuego.

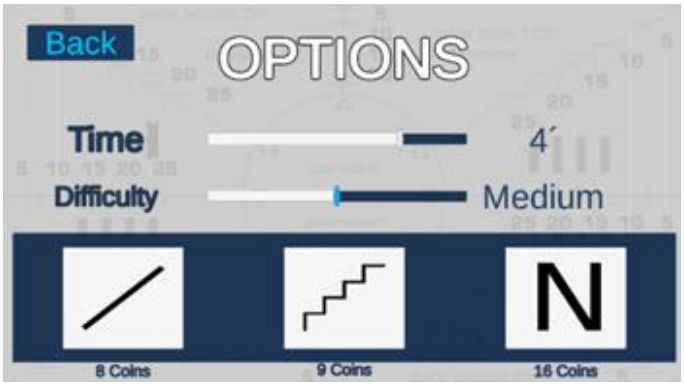

Figura 58: Manual de usuario. Selección de parámetros. Fuente: Obtención propia.

Dentro del menú donde escogemos los parámetros, se cuenta con *Sliders* para seleccionar el tiempo máximo de partida y la dificultad de las trayectorias.

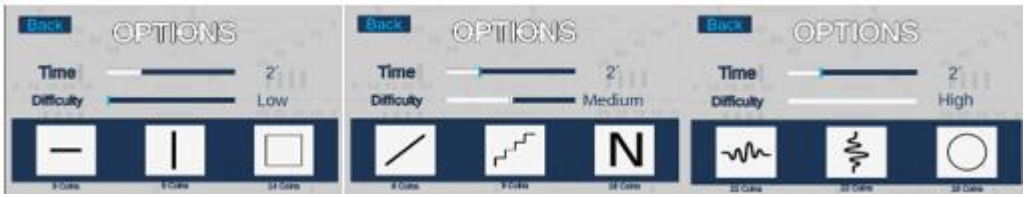

Figura 59: Manual de usuario. Trayectorias disponibles. Fuente: Obtención propia.

Moviendo el *Slider* que configura el tiempo de izquierda a derecha o viceversa, se da un valor al tiempo máximo, desde un minuto hasta cinco minutos. Al igual que con el *Slider* del tiempo, si se mueve el *Slider* de la dificultad de la misma manera, se podrá escoger entre las trayectorias que se muestra en la Figura 59.

Una vez configurados los parámetros deseados dentro del menú opciones, se tendrá que pulsar el botón *Back* situado en la esquina superior izquierda para poder volver al menú inicial y poder comenzar la partida.

2.- ¿Cómo iniciar la partida?

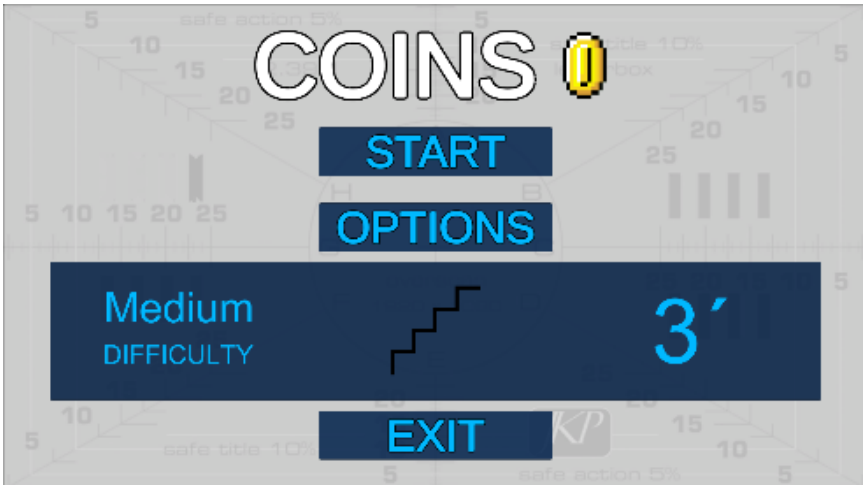

Figura 60: Manual de usuario. Inicio de partida. Fuente: Obtención propia.

Una vez se han escogido los parámetros deseados, se mostrarán en el panel del menú inicial. Si se desea iniciar la partida con los parámetros escogidos se deberá pulsar el botón *START*.

3.- ¿Cuál es el objetivo principal de la partida?

Al pulsar el botón *START* del menú inicial, dará comienzo la partida, con el marcador a cero, y empezará la cuenta atrás para concluir el tiempo máximo. Cuando se comienza una partida, aparecerá en la pantalla la trayectoria con sus respectivas monedas, el tiempo restante de la partida, la puntuación en todo momento y el botón para pausar la partida.

Figura 61: Manual de usuario. Transcurso de la partida. Fuente: Obtención propia.

Para saber cómo comenzar el movimiento, la primera moneda de la trayectoria será de color verde, además estar sobreimpresionado una señal a cuadros y una flecha que indica la dirección de la trayectoria.

Figura 62: Manual de usuario. Trayectoria. Fuente: Obtención propia.

4.- ¿Cuándo finaliza la partida?

La partida puede finalizar de dos maneras. La partida puede finalizar cuando se cumple el objetivo de la partida, el cual es recoger todas las monedas siguiendo la trayectoria indicada antes de que se agote el tiempo límite. La otra forma de finalizar la partida es que se agote el tiempo impuesto por el médico antes de recoger todas las monedas.

5.- ¿Se puede pausar la partida?

Al comenzar la partida, se podrá seguir su desarrollo a través del monitor, donde aparecerá un botón *PAUSE* en la esquina superior izquierda.

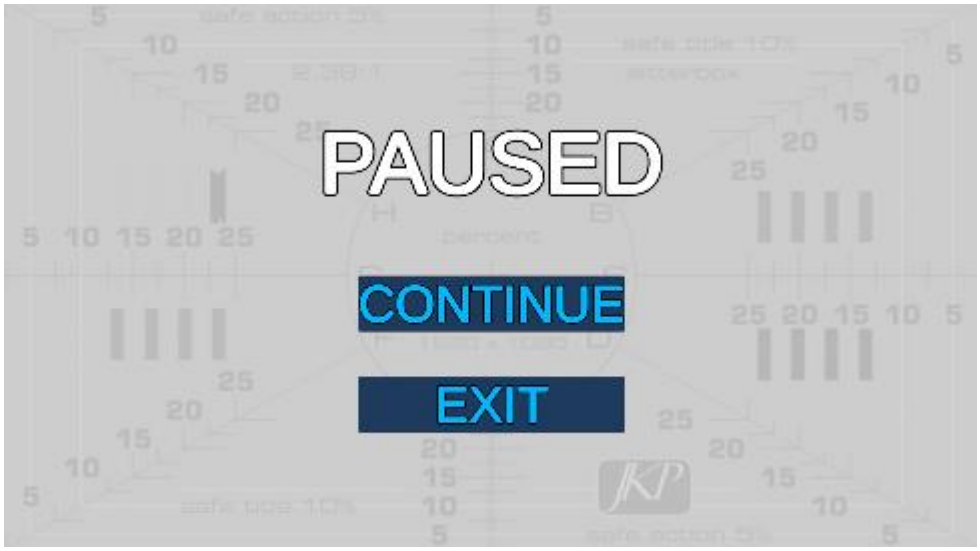

Figura 63: Manual de usuario. Juego pausado. Fuente: Obtención propia.

Si se quiere reanudar la partida se deberá presionar el botón *CONTINUE*, y si se quiere finalizar la partida en ese instante se tendrá que pulsar el botón *EXIT*.

6.- ¿Qué información se obtiene al finalizar la partida?

Al finalizar la partida aparecerá automáticamente el menú final. En este aparecerá un panel con la puntuación final y el tiempo restante.

Figura 64: Manual de usuario. Partida finalizada. Fuente: Obtención propia.

7.- ¿Dónde revisar los datos obtenidos durante la partida?

Figura 65: Manual de usuario. Carpeta *CoinData*. Fuente: Obtención propia.

Al comenzar una partida se creará, si no se creó anteriormente, una carpeta llamada *CoinData*. En esta carpeta se encontrarán los archivos .CSV que tendrán como nombre la fecha y la hora de comienzo de cada partida.