Universidad Veracruzana
Instituto de Investigaciones en Inteligencia Artificial

Ajuste dinámico de la dificultad en un juego serio para la rehabilitación de muñeca utilizando lógica difusa

TESIS PARA OBTENER EL TÍTULO DE:

Maestro en Inteligencia Artificial

PRESENTA:

Juan Ignacio Vargas Bustos

Asesor:
Dra. Ericka Janet Rechy Ramírez

Febrero 2023

Índice

1. Capítulo 1: Introducción **6**
- 1.1. Introducción . . . 6
- 1.2. Motivación . . . 8
- 1.3. Planteamiento del problema . . . 8
- 1.4. Preguntas de investigación . . . 8
- 1.5. Objetivos . . . 9
  - 1.5.1. Objetivo general . . . 9
  - 1.5.2. Objetivos específicos . . . 9
- 1.6. Variable dependiente e independiente . . . 9
  - 1.6.1. Variable independiente . . . 9
  - 1.6.2. Variable dependiente . . . 9
- 1.7. Hipótesis de investigación . . . 9
- 1.8. Contribución . . . 10

2. Capítulo 2: Estado del arte **12**
- 2.1. Metodología de búsqueda . . . 12
- 2.2. Criterios de selección . . . 12
- 2.3. Parámetros analizados . . . 13
- 2.4. Trabajos relacionados . . . 14
- 2.5. Juegos con DDA utilizando ratón . . . 14
- 2.6. Juegos con DDA utilizando cámaras . . . 14
- 2.7. Afecciones estudiadas . . . 15
- 2.8. Métodos propuestos . . . 16
- 2.9. Parámetros ajustados . . . 16
- 2.10. Uso de sensores . . . 17
- 2.11. Panorama general . . . 18

3. Capítulo 3: Metodología **38**
- 3.1. Etapas . . . 38
- 3.2. Sensores y software . . . 38
  - 3.2.1. Sensor: Leap Motion . . . 38
  - 3.2.2. Unity . . . 39
- 3.3. Definición breve del videojuego . . . 39
  - 3.3.1. Movimientos . . . 39
  - 3.3.2. Jugabilidad . . . 41
  - 3.3.3. DDA con lógica difusa tipo I . . . 43
  - 3.3.4. DDA en el videojuego . . . 44
- 3.4. Modalidades del videojuego . . . 48
- 3.5. Diseño experimental . . . 49
- 3.6. Métricas para medir el rendimiento de las modalidades . . . 49

4. Capítulo 4: El videojuego **51**
- 4.1. Jugabilidad . . . 51
  - 4.1.1. Obstáculos . . . 51
  - 4.1.2. Recompensas . . . 52
- 4.2. Puntuación . . . 52
- 4.3. Modos de juego . . . 53
  - 4.3.1. Modo de juego 1 . . . 53
  - 4.3.2. Modo de juego 2 . . . 53

5. Capítulo 5: Experimentación y resultados **56**
5.1. Participantes 56
5.2. Resultados de la modalidad sin DDA 56
5.3. Resultados de la modalidad basada en DDA 57
5.4. Cuestionario GEQ 59
5.5. Análisis estadístico 62

6. Capítulo 6: Conclusiones y trabajo futuro **67**
6.1. Conclusiones 67
6.2. Limitaciones 67
6.3. Trabajo futuro 67

Abstract

Abstract

La rehabilitación tradicional suele carecer de motivación y adaptabilidad. Los pacientes pueden frustrarse y aburrirse con los ejercicios tradicionales de la rehabilitación a su vez no es un proceso continuo ya que el fisioterapeuta debe ajustar los ejercicios a los avances del paciente. Es por esto por lo que actualmente se ha estado investigando el uso de videojuegos para propósitos serios como lo es la rehabilitación. Sin embargo, aunque está demostrado que es más entretenido para los pacientes, aún carece de adaptabilidad. Las personas pueden tener diferentes grados de movilidad en la muñeca, sobre todo si ha recibido una lesión, en consecuencia sería conveniente que el juego pudiera adaptarse según los rangos de cada jugador. Esta investigación implementó un juego serio para la rehabilitación de muñeca con dos modos de juego. El modo de juego 1 no ajusta la dificultad del juego; mientras que el modo 2 ajusta la dificultad del juego basándose en el rango de movimiento del jugador. Se hizo uso de los movimientos de flexión, extensión, desviación cubital y radial. La dificultad del juego se ajustó utilizando lógica difusa tipo 1 para calcular las posiciones en las que se encontrarían las recompensas a lo largo del juego (fácil, medio y difícil de recolectar). Cuatro participantes jugaron ambas modalidades. Las pruebas t de student revelaron que no hubo diferencias significativas entre ambos modos en términos de recompensas recolectadas ($p = 0.6621$), tiempo de juego ($p = 0.8178$) y el cuestionario "Game Engament Questionarie"($p = 0.1383$)

Capítulo 1:

Introdución

1. Capítulo 1: Introducción

1.1. Introducción

Las manos juegan un papel importante en la vida humana, son nuestro principal modo de interacción con el mundo. Además una parte importante de la movilidad de éstas, son las muñecas. Por lo que tras recibir una lesión es importante llevar a cabo un proceso de rehabilitación para ser reincorporado para tareas cotidianas. La muñeca consta de 2 articulaciones radio carpiana y medio carpiana como se aprecia en la figura 1 [Norkin and White, 2016].

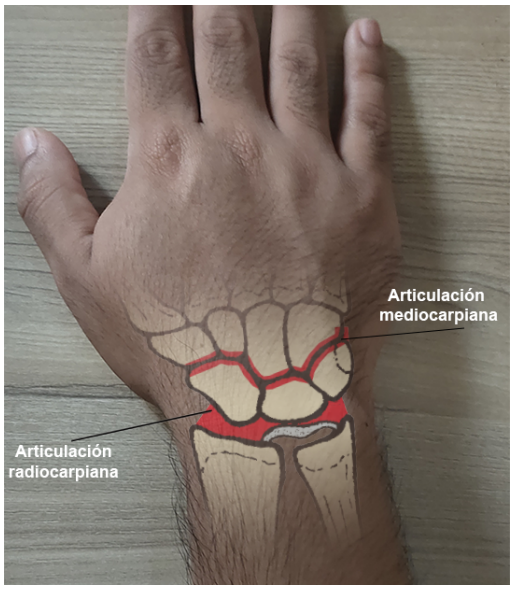


Figura 1: Articulaciones de la muñeca

La causa más común de lesiones en el lugar de trabajo es por el uso de equipo mecánico, estas lesiones son las que causan el daño más significativo y pueden dar lugar a secuelas graves que comprometen la función de la mano.
De acuerdo con [[Zárate-Ramírez and Espinosa-Gutiérrez, 2013]] "En el año 2011, el total de accidentes laborales en el IMSS (Instituto Mexicano del Seguro Social) fue de 422,043 de los cuales 113,511 presentaron lesiones en la región de muñeca y mano, lo que representa el 26.9 % del total de accidentes laborales a nivel nacional."
La rehabilitación se define como un proceso dinámico de cambio adaptativo planificado en el estilo de vida en respuesta a un cambio no planificado impuesto al indicado por una enfermedad o accidente traumático. La rehabilitación tradicional carece de falta de motivación hacia los pacientes y falta de adaptabilidad. Si algún ejercicio es muy difícil de realizar puede resultar en frustrante. Actualmente los expertos han estudiado métodos para evaluar cómo los videojuegos afectan el aprendizaje motor y su alcance [[Lohse et al., 2013]].

Estos juegos forman parte de los "juegos serios" los cuales como su nombre dice, tienen un enfoque serio más allá del entretenimiento. De acuerdo con [Zyda, 2005] "son vistos como un ejercicio mental jugado en una computadora, de acuerdo con unas normas específicas basadas en el entretenimiento para promover los objetivos gubernamentales, corporativos, capacitación, educación, salud, política, comunicación estratégica, etc." Sin embargo que se utilice un videojuego lo hace más atractivo pero no necesariamente más desafiante. [[Csikszentmihalyi and Csikzentmihaly, 1990]] introdujo por primera vez el concepto de "flow channel" donde se busca un equilibrio entre aburrimiento y frustración.

Se han hecho revisiones como es el caso en [Zohaib, 2018] donde utilizan este concepto para darle forma a lo que llamamos DDA ("Dynamic Difficulty Adjustment") donde la dificultad del juego serio se adapta en función de la habilidad del jugador.

Existen diversas maneras de aplicar DDA en un juego serio: redes neuronales [[Wang and Tseng, 2013]], algoritmo de Monte Carlo [[Hocine et al., 2015]], máquinas de soporte vectorial [[Verhulst et al., 2015]], algoritmos evolutivos [[Andrade et al., 2016]], lógica difusa [[Valencia Lemos et al., 2019]].

La lógica difusa se utilizará en esta investigación para implementar DDA en el juego serio. Asimismo, la lógica difusa nos ayuda a manejar grados de incertidumbre con muy poca información. Nosotros como seres humanos manejamos el razonamiento de manera ambigua y hay veces que se requiere ser exacto. La lógica difusa hace posible manejar esta información ambigua en sistemas expertos [Ross, 2005]. "Los conjuntos difusos están definiéndolos como una clase de conjunto con grados de pertenencia que van desde 0 a 1. Se busca aproximar el razonamiento humano." [Ponce-Cruz et al., 2016]

El juego propuesto en esta investigación, busca adaptar la dificultad en función de grados de flexión del jugador, para conseguir que el jugador se mantenga en estado de "flow". Además se comparará con 2 modos de juego, uno que contenga DDA y otro que mantenga el método de dificultad tradicional (selección de dificultad previa al juego).

1.2. Motivación

El avance tecnológico en los últimos años nos ha permitido obtener sensores más accesibles y potentes. Estos son algunos ejemplos de dichos avances: El Kinect, el cual tiene un sensor de movimiento, cámaras y detecta gestos, inclusive la voz del usuario; El *Tobbi eye tracker*, ayuda a darle un seguimiento al movimiento del ojo mediante sus cámaras; Existen otros tipos de sensores que calculan las variables biométricas del cuerpo, como lo es Plux Biosignals. Es un sensor modular, el cual puede medir: ritmo cardiaco, presión sanguínea, oxigenación en sangre, actividad electro dérmica de la piel, actividad electromiográfica, actividad electroencefalográfica. A lo largo de esta investigación utilizaremos el sensor Leap Motion, el cual por medio de su cámaras infrarrojas hace un seguimiento de los movimientos de las manos inclusive los gestos, pero nos centraremos en la muñeca.

Sin embargo, estos sensores han sido desaprovechados para el uso de rehabilitación médica. Esto ayuda a los pacientes con su proceso de rehabilitación, haciéndola más dinámica y divertida. Con la ayuda de DDA lo que se busca es que el paciente evite sentirse abrumado o frustrado por las limitaciones de movilidad de su muñeca.

1.3. Planteamiento del problema

Las lesiones en muñeca puede conllevar a tener secuelas que comprometen la función de la mano. La funcionalidad de la mano, es de suma importancia para nuestras tareas cotidianas y por ende la muñeca funge para la utilización de la mano.

De acuerdo con [Zárate-Ramírez and Espinosa-Gutiérrez, 2013] durante el año 2011, los accidentes en la región muñeca y mano representan 26.9 % a nivel nacional, debido a accidentes laborales. Por lo que la pronta rehabilitación de este miembro es importante, ya que de no ser tratada, el costo y tiempo de recuperación aumentan de manera significativa[Zárate-Ramírez and Espinosa-Gutiérrez, 2013].

Sin embargo el proceso de rehabilitación podría ser tedioso por sus repetición monótona de movimientos, y hasta frustrante. A su vez esto puede conllevar a no completar la terapia [Tageldeen et al., 2017]. Por lo que la investigación en juegos serios puede ser la manera viable de realizar dicha terapia sin ser aburrida.

En este estudio nos concentraremos además de aplicar un juego serio para rehabilitación de muñeca, implementamos DDA para apoyar al jugador a encontrar su dificultad personal de forma automática, esto basándose en la teoría del "flow"propuesta por Csikszentmihaly, el cual indica que es un punto medio entre frustración y aburrimiento y aplicada al gaming en [Koster, 2014]. Existen diversas formas de poder aplicar DDA. Esto se revisará más adelante en el estado del arte, la mayoría de los métodos propuestos son algoritmos propios, que solo sirven para sus videojuegos, el otro algoritmo más utilizado es aprendizaje por refuerzo. Para fines de esta investigación abordaremos un algoritmo de zona de habilidad, basado en lógica difusa, un método poco visto en la literatura. Se hará comparación entre un modo con DDA y otro sin DDA.

1.4. Preguntas de investigación

Con base al estado del arte y al planteamiento de problema, se plantearon una serie de preguntas que el presente estudio se encargará de responder. Las preguntas serán las siguientes:

- ¿Existe diferencia significativa en el puntaje del juego en emplear el modo con DDA y modo sin DDA?
- ¿DDA permite al jugador disfrutar más el juego (con base al cuestionario CEQ [1])?

[1]Este cuestionario es utilizado para medir el nivel de compromiso en el juego [Brockmyer et al., 2009]

1.5. Objetivos

1.5.1. Objetivo general

El objetivo de la investigación será comparar dos modos de juego, para rehabilitación de muñeca, uno con DDA y otro sin DDA.

1.5.2. Objetivos específicos

- Identificar los movimientos de muñeca.
- Desarrollar un juego serio para rehabilitación de muñeca.
- Desarrollar 2 modos de juego (Uno con DDA y otro sin DDA).
- Implementar DDA utilizando lógica difusa.
- Medir el rendimiento del jugador mediante el puntaje y un cuestionario.
- Determinar si existen diferencias significativas, comparando resultados de ambos modos de juego mediante pruebas estadísticas.

1.6. Variable dependiente e independiente

1.6.1. Variable independiente

La variable independiente de la investigación será el modo de juego. Puesto que ésta será la intercambiada al jugador para realizar comparaciones.

1.6.2. Variable dependiente

Se establecieron como variables dependientes:

- El puntaje del jugador (Niveles pasados con éxito).
- El cuestionario GEQ (Game Engagement Questionnaire) para encontrar el nivel de compromiso.

1.7. Hipótesis de investigación

A lo largo de esta investigación sostendremos una hipótesis de investigación con respecto al uso de DDA. No hay diferencias significativas entre los dos modos de juego (uno con DDA y otro sin DDA), en términos de cuestionario CEQ y puntaje del jugador.

1.8. Contribución

La contribución de esta investigación es analizar el estado del arte de los juegos serios con DDA y desarrollar nuestro juego serio para rehabilitación de muñeca con lógica difusa y determinar si existe un a diferencia significativa entre la puntación de 2 modos que se desarrollaran. A su vez se añadirá el uso de un cuestionario GEQ que se utilizará para detectar los estados de ánimo de los jugadores.

La investigación busca ofrecer una alternativa a la rehabilitación tradicional de manera divertida y adaptable a las habilidades del jugador. Busca con ayuda de un cuestionario tener más claramente que emociones experimenta el jugador después de cada sesión para darle una mejor experiencia.

El videojuego desarrollado aquí pone las bases para una investigación más profunda sobre el uso de DDA para rehabilitación de muñeca y su vez incentiva el monitoreo de los estados de ánimo para realmente comprender si el videojuego produce mayor compromiso en el juego. Además de que demuestra que la lógica difusa tipo I puede usarse en algoritmos de DDA con gran facilidad en los juegos serios.

Capítulo 2:

Estado del arte

2. Capítulo 2: Estado del arte

En este capítulo busca dar construcciones de sentido sobre los datos que apoyan nuestra hipótesis de investigación y explicar el panorama de juegos serios para la rehabilitación en los últimos años. Se comenzará delimitando el tema con una metodología de búsqueda. Se hará un resumen de las tendencias en esta área. Para posteriormente dar una conclusión general.

2.1. Metodología de búsqueda

Nuestra búsqueda se basó en la pregunta: ¿Actualmente que estudios se han realizado que contengan DDA para rehabilitación?. Además e realizó siguiendo los lineamientos "PRISMA" (Preferred Reporting Items for Systematic Reviews and Meta Analyses) [Page et al., 2021]. Se ha realizado una búsqueda de la literatura relacionada con el uso de "Dynamic Difficulty Adjustment" para rehabilitación motora mediante juegos serios. Se utilizaron tres bases de datos electrónicas:

- IEEE
- SpringerLink
- GoogleScholar

Para la búsqueda se utilizaron las siguientes palabras clave "Dynamic Difficulty Adjustment", "rehabilitation", "Serious game" con diferentes combinaciones:

1. "Dynamic Difficulty Adjustment" "rehabilitation"
2. "Dynamic Difficulty Adjustment" "rehabilitation" "Serious game"
3. "Dynamic Difficulty Adjustment" "Serious game"

2.2. Criterios de selección

Se seleccionaron los artículos que proporcionaban un uso de DDA dentro de una actividad o videojuego enfocado a la rehabilitación motora. Otros criterios de inclusión añadidos fueron:

- La investigación debe estar enfocada a rehabilitación motora.
- Debe contener juegos serios o actividades.
- Contener al menos un método de DDA.
- Deben estar escritos en idioma Inglés.
- Los artículos deben ser recientes 2010-2021.

Podemos ver en la Figura 2 la búsqueda arrojó 161 artículos (39 de SpringerLink, 2 de IEEE y 120 de Google Scholar). Después de remover duplicados y escritos inaccesibles, se obtuvieron 138 artículos. Solo 27 artículos cumplieron con los criterios explicados anteriormente.

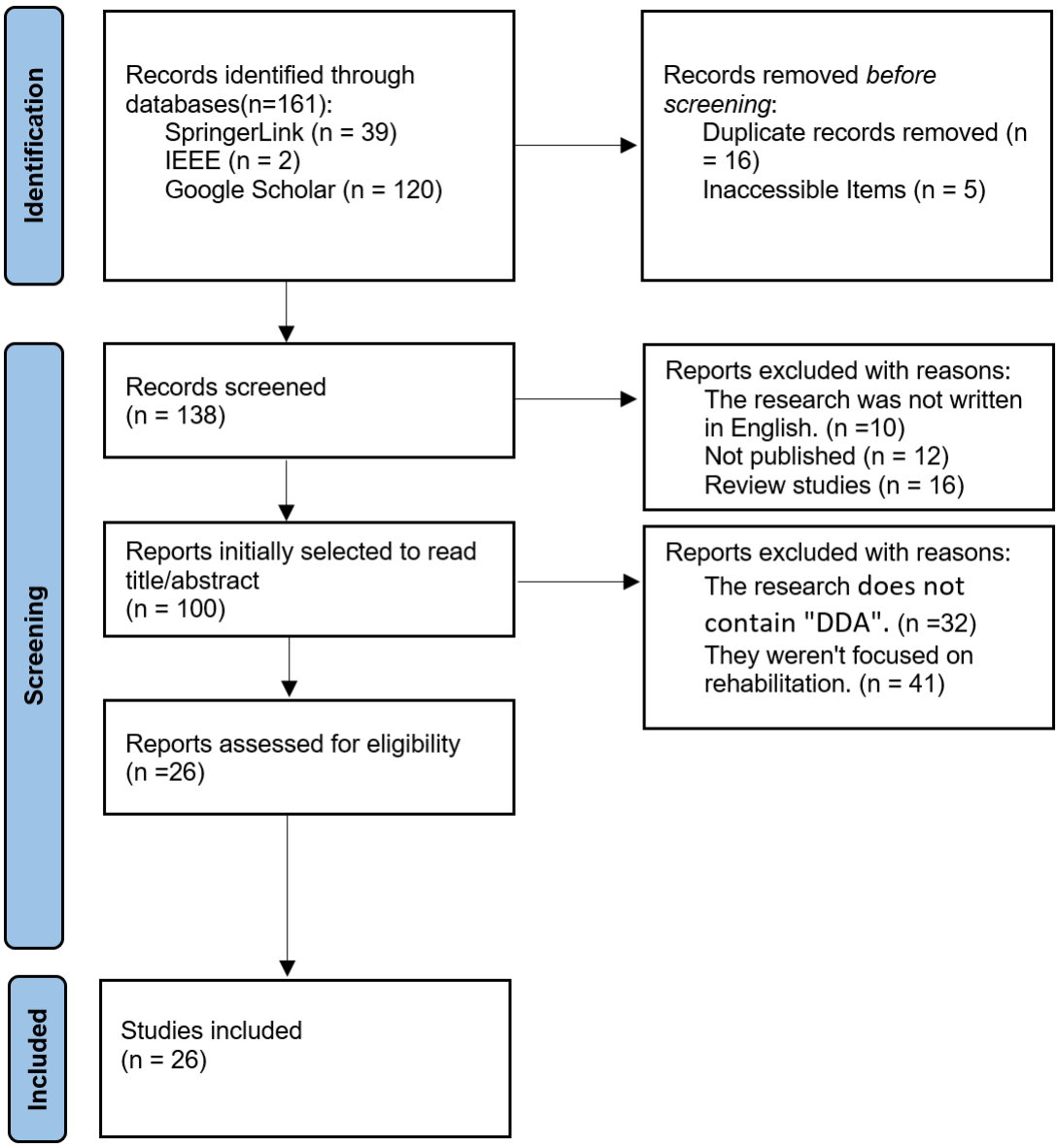


Figura 2: Diagrama de flujo PRISMA[Page et al., 2021]

2.3. Parámetros analizados

Esta revisión del estado del arte recabó 16 elementos de cada uno de los artículos revisados, los cuales serán enumerados a continuación:

1. Año.
2. Meta del estudio.
3. Métodos de DDA.
4. Parámetro ajustado.
5. Afección estudiada.
6. Tipo de ejercicio.

7. Resultados.
8. Sensores utilizados.
9. Tipo de videojuego.
10. Si es 2D o 3D.
11. Modalidad del juego.
12. Juego en línea.
13. Utilización de realidad virtual.
14. Plataforma de programación.
15. Total de sujetos de prueba.
16. Juego en solitario o con multijugador.

2.4. Trabajos relacionados

Se han encontrado 2 artículos relacionados con esta investigación. En [da Cruz et al., 2020] se utilizó el Leap Motion, su videojuego solo buscaba mejorar la coordinación mano-ojo. Su objetivo era reventar globos que aparecían en la pantalla. Y utilizaba un algoritmo basado en Aprendizaje por refuerzo. En[Valencia Lemos et al., 2019] se utilizó también el Leap Motion, su videojuego va enfocado a enfermedades cerebrovascualres. El objetivo era depositar todos los bichos que aparecen en escena en cajas que los clasifican por colores. Su método de DDA es un sistema de lógica difusa.

Sin embargo ninguno de estos 2 estudios se ha centrado en la rehabilitación de muñeca, uno iba enfocado a la coordinación y otro a el agarre. Nuestra investigación contempla más de 2 movimientos, en concreto son 4 movimientos y la posición central de descanso.

2.5. Juegos con DDA utilizando ratón

Un juego llamado "Prehab" se juega a través de ratón [Hocine et al., 2015]. El jugador debe eliminar enemigos y recolectar piedras preciosas y monedas con el movimiento del ratón. La dificultad del juego se ajustó utilizando una búsqueda de árbol de Monte Carlo seleccionar una zona de habilidad (es decir, un área en la que las monedas se muestran en las posiciones fácil, media, difícil, muy difícil y experta para que el jugador las recoja) en función de las habilidades del jugador.

2.6. Juegos con DDA utilizando cámaras

Estos juegos utilizan un Kinect (es decir, un sensor basado en cámaras RGB e infrarrojas que proporciona un esqueleto del usuario) o un controlador de movimiento Leap (sensor basado en cámaras infrarrojas que proporciona las coordenadas X,Y,Z de las falanges, la muñeca, la palma de la mano y el codo). para recopilar los datos del movimiento de la mano. Las actividades de estos juegos serios eran recolectar objetos de la escena del juego y colocarlos en diferentes posiciones del escenario. Los objetivos de estos juegos eran ayudar a las personas que padecían Parkinson [Siegel and Smeddinck, 2012] y enfermedad cerebro-vascular [Valencia Lemos et al., 2019]. Estos juegos utilizaron sistemas difusos, Q-Learning y calculo de variables en función de su comportamiento anterior.

En [Siegel and Smeddinck, 2012] utilizaron un Kinect para identificar los movimientos de brazos y manos. Estos movimientos se emplearon para recolectar estrellas que se mostraban secuencialmente en la escena del juego. Los autores ajustaron la dificultad adaptando el tiempo que las estrellas están disponibles para recolectar y la aptitud del movimiento del jugador. Otro estudio [Valencia Lemos et al., 2019] implementó un juego llamado "Bug Catcher" donde el jugador debe recolectar insectos de diferentes colores que se muestran en la escena del juego y colocarlos en unas casillas correspondientes por color del insecto. Los insectos se

mostraban durante cierto tiempo. El tiempo y la configuración de las casillas se ajustaron mediante lógica difusa y a su ves se utilizó el sensor Leap Motion para los movimientos de tomar y soltar.

Existe un estudio [da Cruz et al., 2020] donde se propuso un juego donde el jugador debe explotar globos que se mueven en la escena del juego. El tiempo en el que se movían los globos se ajustó mediante Q-Learning. Además el movimiento de la mano se identificó utilizando un sensor Leap Motion.

[Verhulst et al., 2015] implementó un juego donde el jugador controla un avatar y toca objetos con las manos. Los movimientos del cuerpo fueron reconocidos mediante un kinect. En este juego la dificultad se adaptó utilizando umbrales para la ansiedad y el aburrimiento del jugador. Las emociones se recopilar con señales electrocardiografía y de activad electro dérmica a través de un sensor BItalino. Estas señales se procesaron utilizando máquinas de vectores de soporte para reconocer estas emociones.

2.7. Afecciones estudiadas

De los artículos revisados, 9 están enfocados a rehabilitación tras derrame cerebral [[Choi et al., 2011], [Hocine and Gouaïch, 2011], [Hocine et al., 2011], [Gouaïch et al., 2012], [Ines et al., 2011],[Knobel et al., 2021], [Hocine et al., 2015], [Sekhavat, 2017], [Huygelier et al., 2017], [Hocine et al., 2014]]. Se revisaron 2 artículos que ven rehabilitación para Parkinson [[Smeddinck et al., 2013],[Siegel and Smeddinck, 2012]]. Se ha explorado igualmente rehabilitación no motora, como lo es para le enfermedad de Alzheimer que se han encontrado 2 artículos que tratan esta afección [[Bouchard et al., 2012], [Imbeault et al., 2011]]. Sin embargo poco se ha abundado a miembros específicos como las manos o la muñeca. Estos juegos serios van más enfocados a enfermedades crónicas como se aprecia en la figura 3. En este diagrama la categoría "Otros" contiene afecciones como: sobrepeso, rehabilitación militar para veteranos, esclerosis múltiple, declive cognitivo, etc.

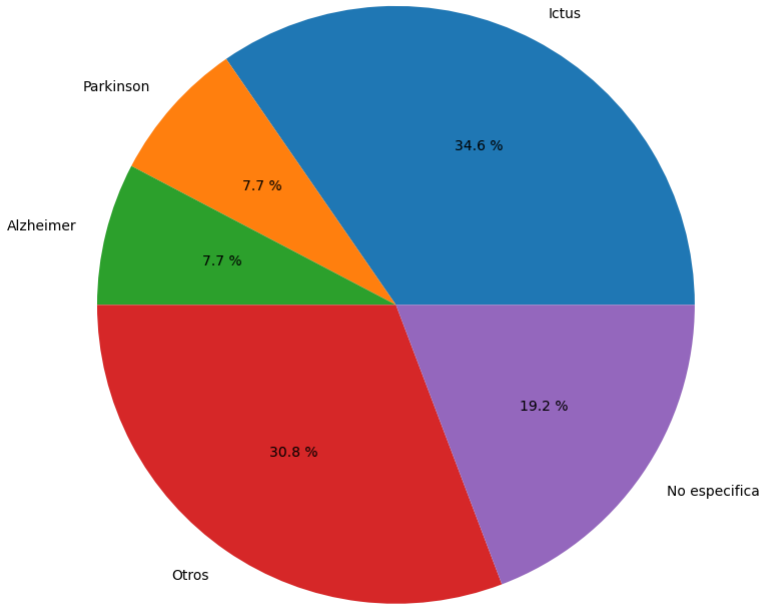


Figura 3: Diagrama de afecciones estudiadas

2.8. Métodos propuestos

Existen 12 artículos que utilizan métodos de DDA propuestos que funcionan específicamente con su juego, por lo que no se pueden estandarizar a otros géneros [[Siegel and Smeddinck, 2012],[Parsons and Reinebold, 2012], [Costa et al., 2017],[Nagle et al., 2015],[Valencia Lemos et al., 2019],[Ozkul et al., 2019], [El-Habr et al., 2019], [Choi et al., 2011] [Smeddinck et al., 2013] [Pezzera et al., 2019], [Hocine and Gouaïch, 2011], [Di Loreto et al., 2012]]. El método conocido, más utilizado fue aprendizaje por refuerzo, con un total de 3 artículos [[Sekhavat, 2017], [da Cruz et al., 2020], [Andrade et al., 2014]]. Seguido por el sistema de ELO[2], algoritmos evolutivos, y el algoritmo de Monte Carlo, ambos con 2 artículos. Se han encontrado otros métodos utilizados con menor frecuencia como lo son: máquinas de estados; máquina de soporte vectorial; sistema difuso y umbrales. Ver figura 4

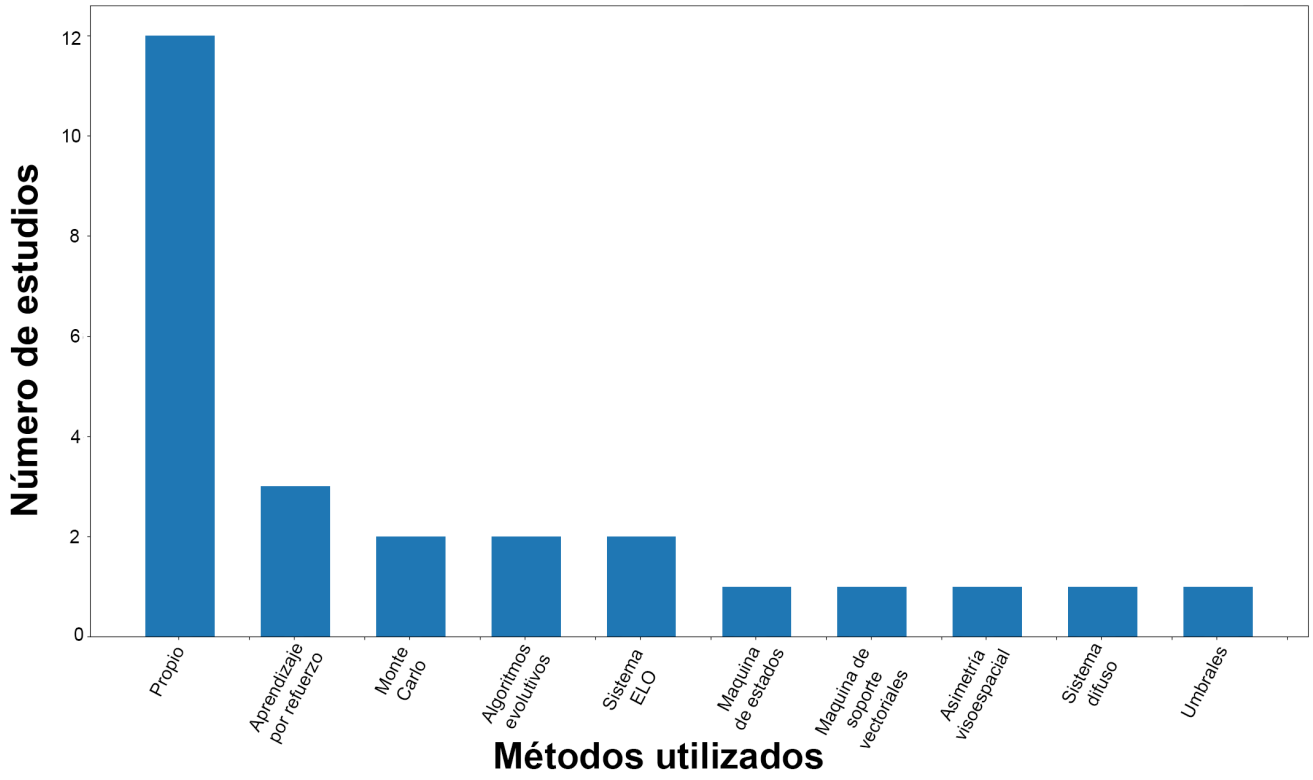


Figura 4: Métodos propuestos

2.9. Parámetros ajustados

Los parámetros que se utilizaban para ajustar la dificultad fueron variados, existen estudios que manejan más de un parámetro como es el caso de [Ozkul et al., 2019] el cual utiliza 3 parámetros de ajuste de la dificultad (Velocidad, Tiempo y Posición de los objetivos). Por lo que en la figura 5 nos muestra todos los parámetros utilizados. Se aprecia cómo el parámetro utilizado con más frecuencia para el ajuste de la dificultad es la posición de los objetivos, seguido de la velocidad (esto varía entre velocidad de movimiento, de escenario, de objetivos, etc), el tiempo es el tercer parámetro más utilizado. Algunos autores no especificaron su parámetro a ajustar. La categoría "Otros" está conformada por: un personaje virtual contra el jugador; Cantidad de puntos, Cantidad de objetivos.

[2]El sistema de puntuación que se utiliza en el ajedrez para calcular la habilidad de los jugadores, ahora también es utilizado en videojuegos.[Cabonell-Sánchez, 2020]

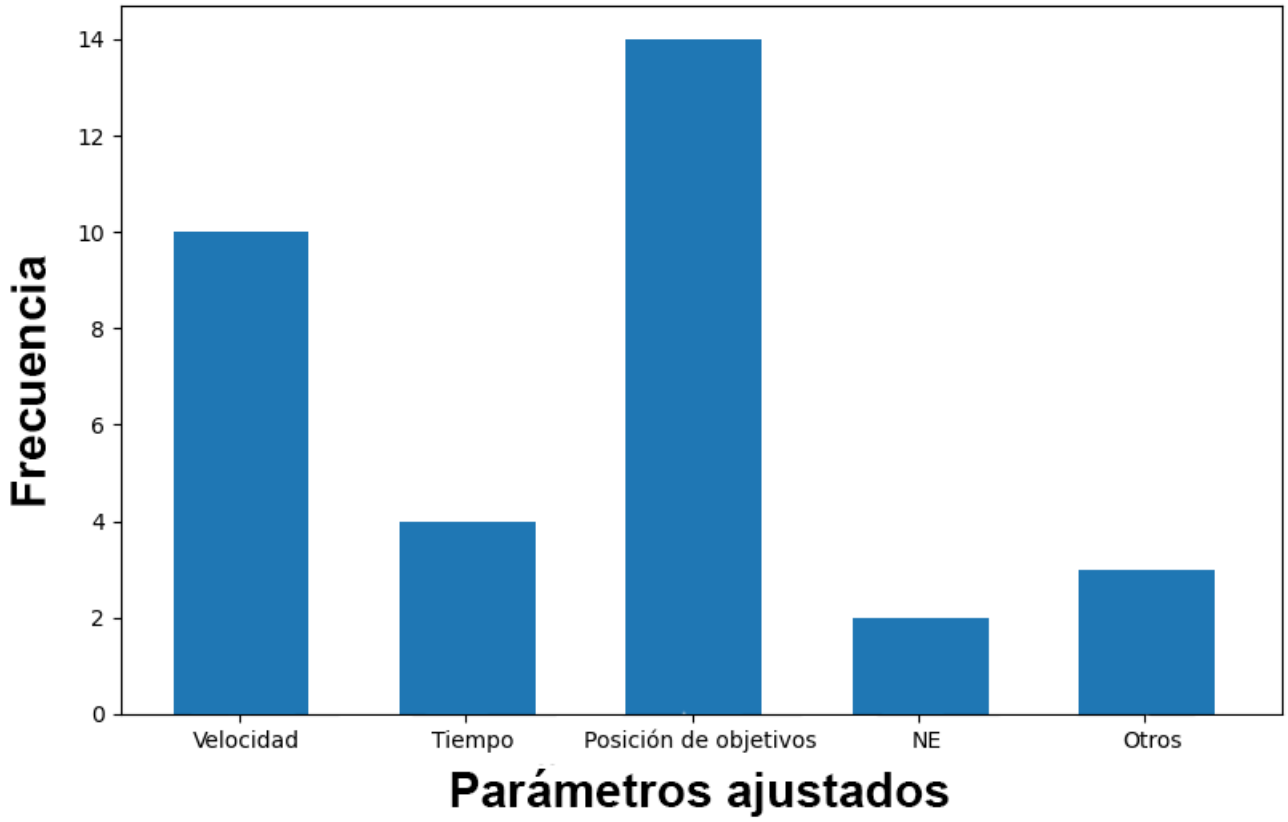


Figura 5: Parámetros ajustados con más frecuencia

2.10. Uso de sensores

El sensor más utilizado dentro de los estudios analizados fue el kinect de microsoft, con 8 artículos que lo utilizan [[Siegel and Smeddinck, 2012],[Sekhavat, 2017],[Verhulst et al., 2015],[El-Habr et al., 2019], [Smeddinck et al., 2013],[Pezzera et al., 2019],[Pinto et al., 2018]] El uso del sensor Leap Motion, para rehabilitación con juegos serios, se ha revisado con anterioridad en [[Valencia Lemos et al., 2019],[da Cruz et al., 2020]]. Sin embargo aunque contienen DDA, se enfocaron al agarre y señalamiento de objetos respectivamente con la mano. [Grubišić et al., 2015] identificó movimientos de muñeca para rehabilitación, sin embargo, no implementó un juego serio. En la figura 6 podemos apreciar como a lo largo de 10 años, se han usado distintos sensores, siendo el más utilizado el Kinect. El sensor Leap Motion lo hemos visto más en años recientes, entre el 2019 y 2020. Otro sensor que podría tener un buen uso para los juegos serios, es el HTC vive que también a su vez utiliza la realidad virtual[3].

[3]Es una interfaz avanzada persona-ordenador que simula un entorno realista.[Zheng et al., 1998]

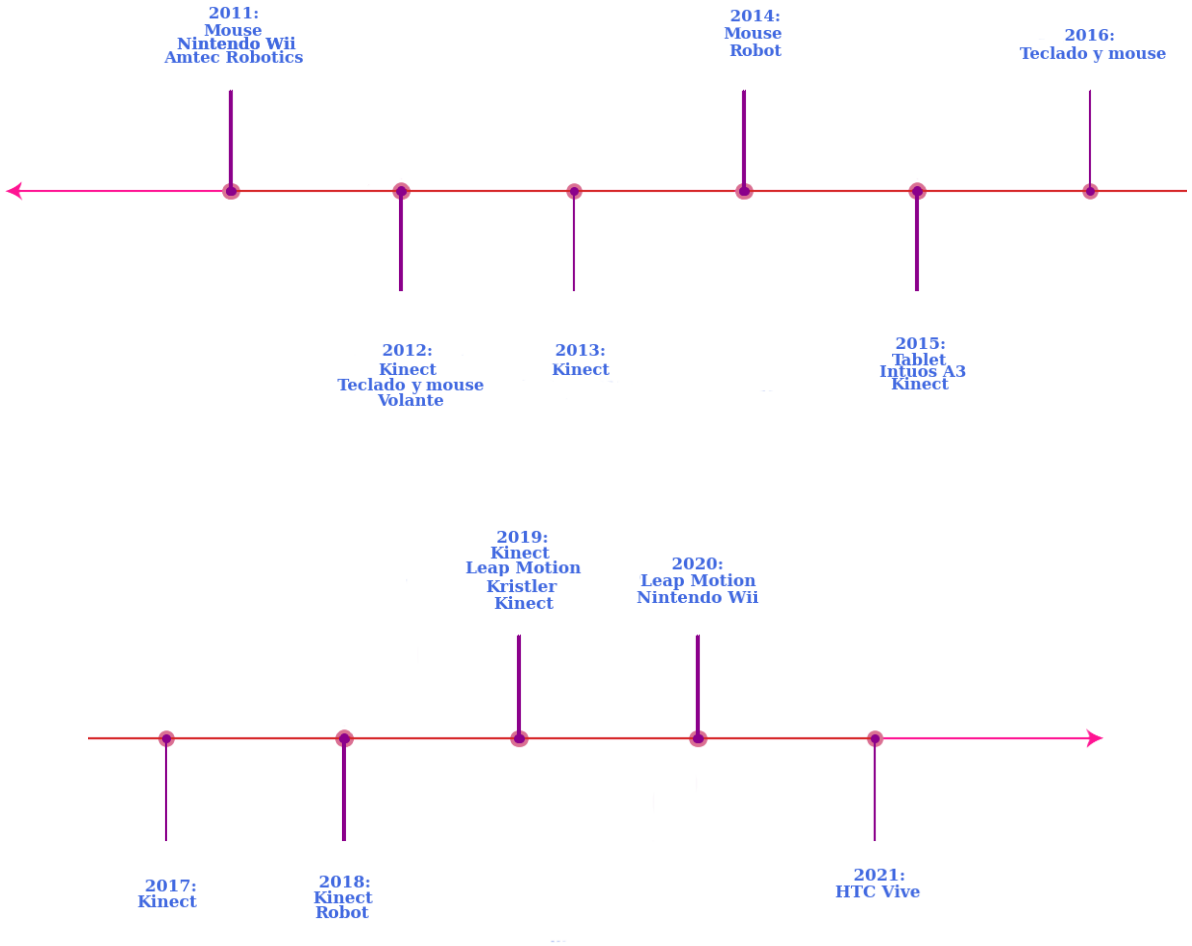


Figura 6: Línea de tiempo con el uso de sensores parte 2

2.11. Panorama general

Podemos ver que las afecciones relacionadas con los juegos serios es el Ictus o Derrame cerebral, dominando un 34.6 % de los estudios revisados. Poco se aborda acerca de lesiones y sobre todo en miembros específicos como la muñeca. Los métodos más utilizados para aplicar DDA están mayormente diseñados para funcionar con su videojuego. Esto lo hace poco replicable, el uso de un algoritmo más general como lo es el aprendizaje por refuerzo, hace que se pueda adaptar mejor a distintos tipos de juego. Sin embargo tampoco se ha profundizado tanto en la lógica difusa. A su vez los parámetros más utilizados han sido la posición de los objetivos. Esto puede ser un acierto, debido a que una persona sin su máxima habilidad no puede hacer esfuerzos rápidos. Así se concentran en que el juego sea disfrutable, siempre y cuando el jugador pueda cumplir sus objetivos dentro del juego.

Cuadro del estado del arte

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Dynamic difficulty adjustment in exer-games for rehabilitation: a mixed approach	El parámetro de información sobre el ejercicio se ajusta mediante un adaptador lineal o un adaptador difuso. El adaptador lineal simplemente ajusta la dificultad aumentando o disminuyendo los parámetros en una cantidad fija siempre que la tasa de éxito o de fracaso supere un umbral determinado. No se dan detalles sobre el adaptador difuso.	Alcance (control de la amplitud del movimiento), la velocidad (control de la frecuencia de los movimientos) y precisión (control del tamaño de la canasta en el juego).	Esclerosis múltiple	Nintendo Wii Balance Board	Juego de atrapar frutas: el paciente debe atrapar las manzanas que caen de las ramas de los árboles con una cesta en la cabeza.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Adaptive difficulty with dynamic range of motion adjustments in exergames for parkinson´s disease patients	Los tres parámetros de dificultad (velocidad, precisión y amplitud) se adaptan automáticamente después de cada ronda de juego en función del rendimiento del jugador en esa ronda. Los ajustes de velocidad y precisión cambian en función de la tasa de estrellas recogidas frente a las no recogidas y del tiempo de finalización. Para la adaptación del ajuste de amplitud, la tasa de recogida se pondera por la distancia de las estrellas desde la posición de reposo de la mano de juego.	El tiempo disponible para recoger todas las estrellas antes de que desaparezcan automáticament e (velocidad), -el tamaño del cursor con el que el jugador tiene que golpear las estrellas (precisión), y la amplitud de movimiento (amplitud) de los movimientos que el jugador tiene que realizar para alcanzar todas las estrellas.	Parkinson	Kinect	El usuario recoge estrellas usando la mano

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Dynamic difficulty adaptation in serious games for motor rehabilitation	Una búsqueda de Árboles Monte Carlo y un método de límites de confianza superior para árboles con el fin de seleccionar las "mejores" tareas de señalización para un nivel de juego.	Posición y número de objetivos en la zona y éxitos y fracasos de los usuarios	Derrame cerebral	Graphics tablet Intus A3 o Ratón	Se pide al jugador que elimine a los enemigos que constituyen los obstáculos para el personaje principal dado (tortuga, gato o conejo). El jugador puede recoger monedas para aumentar su puntuación. El jugador debe alcanzar la secuencia de objetivos (tarea de señalar, es decir: enemigos y monedas) colocados en diferentes puntos
Developing Adaptive Mental Health Games using Player-Patient Modeling	Simularon datos basados en un modelo que representa la asimetría visoespacial.	Las ubicaciones de los objetivos se desplazaron más hacia el lado periférico (desatendido) del espacio en el caso de los pacientes con un sesgo espacial más fuerte.	Negligencia hemiespacial después de un ictus.	NE	Los pacientes deben diferenciar entre 2 tipos diferentes de estímulo objetivo. Los estímulos objetivo pueden presentarse en diferentes lugares del entorno realista de RV. El objetivo es la orientación atencional.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Dynamic difficulty adjustment in virtual reality applications for upper limb rehabilitation	Sistema difuso con múltiples entradas y una única salida (MISO) como técnica de adaptación de la dificultad. El nivel de dificultad se obtiene en una variable discreta (L: nivel) con siete subconjuntos. Las funciones de pertenencia (triangulares) asociadas a cada subconjunto de las variables de entrada se obtienen realizando un análisis estadístico de los datos obtenidos en las pruebas con usuarios sanos.	La dificultad del juego para presentar diferentes configuraciones: las cajas pueden estar alineadas (todas situadas en la misma profundidad) o intercaladas (variación de profundidad entre cajas); los bichos tienen dos estados: caminando (diferentes velocidades) o quietos.	Enfermedades cerebrovasculares	Leap Motion LM-010	El usuario debe depositar todos los bichos que aparecen en la escena dentro de las cajas de su respectivo color, como parámetro de dificultad, los bichos tienen un tiempo de desaparición
Developing serious games specifically adapted to people suffering from Alzheimer	El sistema ELO hace uso de una función de distribución normal para predecir la superación de una partida entre dos jugadores, en función de sus respectivas habilidades relativas R1 y R2. Como la partida se juega en solitario, el jugador llega a "competir contra el juego".	NE	Alzheimer	NE	En este juego, se invita al jugador a preparar diferentes comidas en una cocina virtual utilizando los ingredientes y platos proporcionados. Cada nivel presenta al jugador una comida predefinida para preparar, en una curva de aprendizaje suave.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Exploring dynamic difficulty adjustment mechanism for rehabilitation tasks using physiological measures and subjective ratings	IDOL es un algoritmo de uso común, en el que el nivel de dificultad se decide directamente en función de la puntuación del sujeto, incrementando el nivel uno hacia arriba, disminuyendo el nivel uno hacia abajo o permaneciendo en el mismo nivel. El algoritmo POSM también controla la puntuación, pero esta actualiza primero los valores de creencia de todos los niveles de dificultad de la tarea y, a continuación, se propone un nuevo nivel de dificultad basado en estos valores de creencia.	Varios parámetros del juego, como la velocidad de la cesta (BS), tiempo de nivel (en segundos), número de frutas (FN), velocidad de las frutas (FS), número de rocas (RN), velocidad de las rocas (RS), tiempo de espera de reaparición (SWT) (en segundos) y tiempo de espera de oleada (WWT) (en segundos).	NE	Kistler-9313AA1	Los sujetos deben recoger las frutas y evitar las rocas en un tiempo determinado.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
MPRL: Multiple-Periodic Reinforcement Learning for difficulty adjustment in rehabilitation games	Para aplicar el ajuste dinámico de la dificultad, se utilizan ideas de aprendizaje por refuerzo (RL) para adaptar la dificultad del juego en función de las habilidades de un paciente. Este trabajo propone un aprendizaje por refuerzo de periodos múltiples (MPRL), en el que pueden evaluarse distintos objetivos en periodos separados.	Cambiar la velocidad de movimiento del personaje en el entorno de juego.	Ictus	Kinect	El juego consiste en un paisaje verde poblado de árboles y edificios. Los movimientos físicos de los brazos se mapean en los movimientos de los brazos virtuales, donde el jugador controla los brazos del personaje visto en una perspectiva en primera persona.
Adaptation in serious games for upper-limb rehabilitation: an approach to improve training outcomes	El modelo propuesto se basa en la predicción a corto plazo de las capacidades del jugador y de la condición física diaria del paciente con Ictus. El módulo de entrenamiento requiere la evaluación de las habilidades del jugador, conocida como "zona de habilidad". La zona de habilidad se define en el ejercicio de evaluación y se actualiza durante la sesión de juego. En este ejercicio, se pide al paciente que cubra todo el espacio de trabajo moviendo el ratón en 9 direcciones diferentes. La búsqueda de Árbol Monte Carlo en el juego se utiliza para calcular las mejores secuencias de tareas de puntería, con el fin de crear varios modos de dificultad en el juego.	Niveles, Se definen cinco modos de dificultad: fácil, medio, difícil, muy difícil y experto. Estos modos se basan en las limitaciones de espacio y, en particular, en la distancia entre la mano y el objetivo, así como en la dirección del movimiento.	Ictus	Tableta gráfica Intuos A3	El objetivo del juego es recoger todas las piedras preciosas. En cada nivel, el objetivo es eliminar a los enemigos para alcanzar la gema. Los enemigos actúan como obstáculos en la progresión del personaje principal. Además, el jugador puede recoger varias monedas para aumentar su puntuación.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
A Self-adaptive Serious Game for Eye-Hand Coordination Training	Aprendizaje Q: es un algoritmo de diferencia temporal fuera de política que se centra en los valores de acción del estado. utilizando el conjunto de herramientas ML-Agents como facilitador para el diseño del juego y la comunicación con el algoritmo	La velocidad v del globo como parámetro de dificultad del juego. Con un total de 10 velocidades.	NE	Leap Motion LM-010	El juego requiere un jugador reventar determinados globos que aparecen en una pantalla con el objetivo de evitar que vuelen.
Physiological-based Dynamic Difficulty Adaptation in a Theragame for Children with Cerebral Palsy.	Support vector machine (SVM): A partir del estado fisiológico del usuario (sólo ansiedad y aburrimiento, con señales de actividad electrocardiográfica y electro dérmica), un clasificador de aprendizaje identifica el estado afectivo del usuario, que se transmite al algoritmo DDA, que adapta en consecuencia el nivel de dificultad del juego. A continuación, utilizaron 14 características para entrenar 2 SVM, uno entrenado para reconocer la ansiedad, y el otro para reconocer el aburrimiento. El algoritmo DDA puede explicarse en esas pocas líneas: la aplicación cambia la dificultad hasta que ansiedad/aburrimiento no se detecta y la dificultad está por encima de un umbral dado.	El DDA cambia el intervalo de tiempo entre cada mensaje (color objetivo)	Parálisis cerebral	Kinect, BITalino	En este juego, el usuario controla el avatar a través de un Kinect, y debe tocar los objetos moviendo su brazo izquierdo.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Runner: A 2D platform game for physical health promotion	PCG4H & DDA4H, se refiere a la creación de contenidos de juego de forma automática mediante algoritmos. La creación de una nueva plataforma tiene en cuenta la plataforma precedente. Cada plataforma recibe una lista generada automáticamente de plataformas vecinas La altura se establece en función de la altura de la plataforma precedente, y ésta es una propiedad importante para determinar si una plataforma candidata puede incluirse en la lista actual. El DDA utilizado en este experimento funciona ajustando la aparición de plataformas trampa y plataformas con monedas amarillas a la tasa de mortalidad del jugador en todas sus partidas anteriores.	Puede cambiar la velocidad de carrera del personaje del jugador y ajustar una serie de umbrales que se utilizan para controlar la generación de trampas (por ejemplo, reducir la frecuencia de aparición de trampas o aumentar la distancia mínima entre trampas consecutivas). Velocidad de carrera: la velocidad del avatar aumenta con el tiempo, lo que hace que la puntuación crezca más rápido.	Sobrepeso	Kinect	Runner es un juego de plataformas en 2D del tipo *endless-running,* en el que el personaje del jugador avanza continuamente por un mundo de juego generado proceduralmente y teóricamente infinito. El objetivo del juego es conseguir una puntuación lo más alta posible, antes de que el personaje muera, recogiendo monedas y evitando obstáculos.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Feasibility of the adaptive and automatic presentation of tasks (ADAPT) system for rehabilitation of upper extremity function post-Ictus	ADAPT es un robot de uso general (Amtec Robotics) con una muñeca 3-DOF montada en un actuador lineal 1-DOF. El robot genera pares elevados. El programador de tareas adaptativo de alto nivel de ADAPT selecciona tanto la tarea a practicar dentro del banco de tareas como la dificultad de la tarea en función del rendimiento previo del paciente. Se utilizan cuatro herramientas (pomo, timbre, jarra y destornillador) para realizar seis tareas. Las herramientas están dispuestas en un estante del que el robot coge una herramienta para una tarea funcional seleccionada. El modelo de tarea funcional genera una trayectoria deseada para simular la dinámica de la tarea seleccionada con la dificultad especificada por el programador de tareas adaptativo de alto nivel.	Ángulo, velocidad y aceleración, respectivamente, del movimiento de la tarea	Ictus	Amtec Robotics con una muñeca 3-DOF montada sobre un actuador lineal 1-DOF. El DOF lineal solo se utiliza para posicionar la herramienta, no para simular la dinámica de la tarea.	NA

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Adaptive difficulty in exergames for Parkinson's disease patients.	El parámetro de amplitud se elabora a partir de la distancia deseada de las manos por encima del hombro que se necesita para alcanzar la estrella más alta (el ajuste implica una amplitud aproximadamente equidistante en Otras direcciones). Se define como: *AmplitudDeeficacia* = 2 * (*distanciaSobreHombro* - 0,15). 0,15 induce una amplitud mínima de 15 cm. La velocidad se incrementa en 0,1 si el rendimiento de la recogida supera 0,93. La velocidad se incrementa en 0,2, si el tiempo performance supera 0,8 al mismo tiempo.	Velocidad y amplitud de las estrellas	Parkinson	Kinect	Los jugadores utilizan sus manos para recoger estrellas que aparecen secuencialmente, siguiendo un conjunto predefinido de trayectorias que reflejan movimientos de fisioterapia.
Approaches for increasing patient's engagement and motivation in exer-games-based autonomous telerehabilitation	El segundo enfoque que descubrieron es el ajuste dinámico de la dificultad en tiempo de ejecución. La Estación Hospitalaria permite a los terapeutas establecer un valor mínimo y un valor máximo para cada parámetro relacionado con la dificultad del ejercicio, con el fin de permitir que cada paciente juegue a los *exergames* con una dificultad que se ajuste a sus capacidades. Partiendo de estos valores preseleccionados, la Estación del Paciente los ajusta en función del rendimiento en tiempo real del paciente: esto ayudará a ofrecer constantemente un reto satisfactorio al paciente, cuyas capacidades pueden cambiar día a día.	La cantidad de puntos que son necesarios para mejorar su granja en el juego, sólo se puede obtener por los *exergames* y están vinculados a la calidad del ejercicio	NE	Kinect o Wii Balance	Es una granja virtual en la que los pacientes construyen su propia granja gastando puntos que sólo pueden obtenerse jugando. Hay dos monedas diferentes: monedas y puntos de granja (PF); sólo estos últimos se obtienen jugando, mientras que las monedas son la moneda necesaria para aumentar el valor de la granja. Las monedas sólo pueden obtenerse vendiendo recursos.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Adaptive gameplay and difficulty adjustment in a gamified upper-limb rehabilitation	Desarrollaron una máquina de estados responsable de la evaluación del rendimiento del paciente en cada minijuego y, en función de los resultados, de la transición del paciente de un estado de dificultad a otro. Consideraron la implementación de 3 estados diferentes: Fácil, Medio y Difícil. Los estados varían en función de los parámetros específicos de cada minijuego y también del punto del proceso de rehabilitación del paciente, es decir, los tres estados se personalizan para cada paciente.	La amplitud de los objetivos	NE	Kinect v2	**Obra abstracta:** Se espera que el paciente realice una flexión del miembro superior con el fin de golpear una bola de pintura que tiene delante, disparándola así hacia un lienzo vacío y creando un cuadro abstracto. **Navegar en barco:** El paciente controla el movimiento de un bote de remos utilizando su antebrazo y la dificultad del ejercicio viene impuesta por la posición de varios puntos de control situados con precisión. **Reloj clásico:** Se desarrolla en un salón con mobiliario clásico en el que el paciente debe imitar, lo más fielmente posible, el movimiento del péndulo de un reloj situado en el interior de la habitación. **Mariposas:** Aquí, el paciente tiene que tocar unas mariposas que están sobre una roca en un jardín, lo que las hará salir volando.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Evolutionary algorithms for a better gaming experience in rehabilitation robotics	Se adoptó un algoritmo evolutivo basado en una estrategia de optimización para ajustar la dificultad del juego al modelo de usuario. Se aplicó un enfoque de tres pasos. En el primer paso, se crea una población aleatoria con NCEA1 soluciones candidatas (cromosomas). En el segundo paso se evalúan las soluciones candidatas, y el tercero consiste en crear un nuevo conjunto de soluciones candidatas haciendo copias de los cromosomas seleccionados NSEA1 con los valores de aptitud más altos y aplicando el operador de mutación *Creep*.	Distancia entre objetivos	Ictus, parálisis cerebral y lesiones medulares.	Robot con 1 solo grado de libertad.	El objetivo del jugador en The Catcher es controlar a la ardilla (personaje del juego) hacia las avellanas que caen (objeto objetivo). El personaje sólo se mueve en el eje x de la pantalla, correspondiente a la flexión de la muñeca (palmar flexión) o a la extensión de la muñeca (dorsiflexión).
Dynamic Player Modelling in Serious Games Applied to Rehabilitation Robotics	El algoritmo de aprendizaje Q consiste en actualizar los valores descontados de las recompensas esperadas, Q(s,a). En cada iteración con el entorno, los valores Q se actualizan de acuerdo con una ecuación. Realizando una acción a, el agente cambia del estado s al estado s', y recibe una recompensa inmediata r. En el estado s' se realiza una búsqueda dentro de las acciones disponibles para encontrar la acción a' que lleva al agente a un estado con el mayor valor de recompensa.	Se adoptaron dos parámetros para ajustar la dificultad la "tasa de caída de nueces" (v) discretizada en valores m y la "distancia a la nuez" (d).	NE	Robot con 1 solo grado de libertad. Utiliza un sistema de impedancia para controlar la rigidez del movimiento.	El jugador controla los movimientos de una ardilla que camina sobre el eje horizontal (derecha e izquierda). La ardilla recoge las nueces que caen de los árboles. El juego se desarrolló teniendo en cuenta un robot conectado a la muñeca del usuario como dispositivo de entrada.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Therapeutic games' difficulty adaptation: An approach based on player's ability and motivation	Este paso se denomina evaluación y produce una matriz en la que cada lugar A (i , j) representa una tasa de éxito del paciente. Por último, el nivel de dificultad se expresa como una probabilidad de éxito deseada d para realizar una tarea por parte del paciente. En otras palabras, el juego ' propone una dificultad mientras que la probabilidad de éxito se estima en d% como mínimo.	El lugar donde se coloca la bola	Ictus	Wii Balance	El jugador debe alcanzar unos objetivos situados en distintas posiciones en el plano 2D. Se basa en una estructura orientada a tareas en la que el objetivo es estabilizar una bola lo más cerca posible de un objetivo en un tiempo limitado. Dependiendo de la ubicación del objetivo, al jugador le resultará más o menos difícil estabilizar la bola.
Digital-Pheromone Based Difficulty Adaptation in Post-Ictus Therapeutic Games	El primer paso del proceso de adaptación consiste en recopilar datos iniciales sobre las capacidades funcionales del paciente. Esto permite construir un modelo sobre las capacidades de movimiento del paciente en el plan de trabajo denominado "matriz de capacidad". Una zona de capacidad A(m,n) es una matriz de dimensión m x n donde m representa el número de filas y n el número de columnas. Cada valor de celda A [m, n]'' representa una puntuación de facilidad para que el paciente alcance el área mapeada del plan de trabajo. La zona de habilidad representa el modelo de capacidades de movimiento del paciente	Alcance (definido en función de la precisión deseada del movimiento)	Ictus	Teclado y Ratón	Para alcanzar un objetivo, el usuario controla una goma que debe extender pulsando continuamente la barra espaciadora del teclado con su mano no dominante. Cuanto mayor sea la frecuencia de pulsación, más se extenderá la goma; si la frecuencia de pulsación disminuye, la longitud de la goma se reduce a su tamaño mínimo. El ratón sólo proporciona la dirección de la goma.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Mixed reality serious games for post-Ictus rehabilitation	Matriz de probabilidades de acción (MAP). Esta matriz representa el plano 2D del juego y cada lugar A(i;j) contiene una probabilidad de que el paciente realice una acción en ese lugar. Cada lugar D(i;j) representa entonces la probabilidad de éxito del paciente en particular y no una genérica. Por último, el nivel de dificultad se expresa como una probabilidad de éxito deseada (d) por el paciente. Esto se interpreta como la voluntad de que el paciente tenga éxito con la probabilidad de d% como mínimo. El módulo de ajuste de la dificultad toma una de estas tres decisiones en función del perfil y la motivación del paciente: aumentar, disminuir o mantener el nivel de dificultad actual de la sesión de entrenamiento. Para ajustar el nivel de dificultad se tienen en cuenta tres criterios.	Sensibilidad de percepción de la velocidad de los peces: cuando la velocidad de la mano del paciente supera, durante un tiempo determinado, un umbral definido (vmax) -lo que se interpreta como que el paciente lo está haciendo demasiado bien-, entonces el agente informático pide a los peces que desafíen al paciente huyendo de su mano.	Ictus	Wii mote	Dos juegos que han desarrollado: en un caso, el objetivo es espantar a los cuervos que se comen la cosecha; en el segundo juego, el objetivo es pescar un pez animado.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Serious games in cognitive training for Alzheimer's patients	El sistema ELO hace uso de la función de distribución normal FN para predecir la superación de un partido entre dos jugadores basándose en sus respectivos niveles relativos de habilidad. Con dos rangos de jugador R1 y R2, el valor de expectativa se calcula evaluando la distribución normal de la diferencia de habilidades FN(R2 - R1). Por lo tanto, cuanto mayor sea la diferencia entre los rangos de los jugadores, mayor será el valor de la expectativa para el jugador mejor clasificado.	Crear un jugador virtual 2 con un ELO similar al paciente	Alzheimer	Ratón	El paciente puede hacer clic en la tostadora para meter las rebanadas o colocar el objeto en otro lugar. Por ejemplo, podríamos colocar rebanadas de pan en la tostadora, y aparecerá un temporizador para mostrar el tiempo restante mientras se tuestan. Cuando estuvieran listas, aparecería otra retroalimentación (por ejemplo, una flecha animada) para llamar la atención del paciente sobre el objeto.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Neurocognitive Stimulation Game: Serious Game for Neurocognitive Stimulation and Assessment	Plataforma ACT-Age: Los jugadores tendrán que completar los minijuegos y una prueba diagnóstica. Los resultados finales se comunican al experto, para que realice el diagnóstico. Tanto el personal informático como el experto proporcionarán asistencia a los jugadores, en caso de que tengan dudas relacionadas con las actividades que deberán realizar.	NE	Declive cognitivo	Ratón y teclado	*SynapseToLife* está organizado en cuatro escenarios diferentes, donde cada uno simulará situaciones del mundo real, por las que el usuario tendrá que pasar, mientras realiza las tareas de estimulación previstas. Cada escenario se encarga de pedir la realización de tareas relacionadas con ese escenario específico. Por ejemplo, en el escenario Cocina, el jugador sólo realiza tareas relacionadas con la cocina, como cocinar, promoviendo estímulos sobre procesos cognitivos específicos.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Development of a Search Task Using Immersive Virtual Reality: Proof-of-Concept Study	En cuanto a la mecánica de la tarea, esta adaptaba su dificultad (es decir, cambiaba el nivel de dificultad) automáticamente, en función del rendimiento en la ronda anterior. Cada nivel tenía un umbral de subida de nivel (es decir, si se alcanzaba el valor umbral, la siguiente ronda comenzaba en un nivel de dificultad superior) y un umbral de bajada de nivel (es decir, si no se alcanzaba el valor umbral, la siguiente ronda comenzaba en un nivel de dificultad inferior), definidos en función del porcentaje de objetivos encontrados. Si el porcentaje de objetivos encontrados en una ronda concreta no era superior al umbral de subida de nivel ni inferior al umbral de bajada de nivel, la dificultad de la ronda siguiente no cambiaba.	Duración y velocidad de los objetivos	Ictus	HTC Vive	Los objetivos (pájaros) podían marcarse alineando el mando con el objetivo y pulsando simultáneamente un gatillo en el mando. La aparición del objetivo se programó para que tuviera lugar aleatoriamente dentro de un área restringida frente al jugador (± 60° horizontalmente y ±50° verticalmente, definida con respecto al plano medio sagital del tronco). Para alertar al jugador de la aparición de un nuevo objetivo, se presentaba una señal auditiva breve (1 segundo) (cacareo de pollo) de forma binaural a través de auriculares. En caso de que el objetivo no fuera marcado en el tiempo máximo de presentación de 15 segundos, desaparecía.

Referencia	Método de DDA	Parámetro ajustado	Enfermedad tratada	Sensores utilizados	Tipo de juego
Adaptive virtual environments for neuropsychological assessment in serious games	El marco adaptativo VRCPAT 2.0. Para ello, es necesario que la simulación infiera primero el estado cognitivo actual del usuario basándose en las señales psicofisiológicas y en el rendimiento de la tarea y, a continuación, cree cambios en el entorno virtual para dirigir al usuario hacia el estado deseado. La determinación de si el nivel de excitación del usuario está por encima o por debajo de los umbrales preestablecidos se realiza mediante una comparación de características extraídas de los siguientes índices psicofisiológicos: Actividad cardiorrespiratoria, Actividad electro dérmica, respiración, pupilometría	La distancia y la velocidad del vehículo perseguido	(Militar)Lesión por explosión	*Head Mounted Display* (HMD), volante y pedales de acelerador y freno.	Los jugadores del juego tienen instrucciones de mantener su separación inicial del vehículo líder (LV) a pesar de los cambios en la velocidad del LV. El vehículo del conductor se sitúa 23 metros por detrás del VL y se le indica que mantenga la distancia mediante aceleraciones y deceleraciones.
High user control in game design elements increases compliance and in-game performance in a memory training game	La adaptación de la dificultad se realizaba después de cada ronda incrementando n en uno si había tres o menos errores y disminuyendo n en uno si había nueve o más errores, con el valor de n restringido al rango El valor inicial de n se fijaba en 2. Se trataba, por tanto, de una versión minimalista de DDA. Aunque el cambio de dificultad se realizaba después de cada ronda, el recuento de errores se reiniciaba sólo después de una instancia decreciente.	n aumenta el número de pruebas y objetivos	Vejez	Tablet	Presenta a los participantes estímulos en 1 de las 6 ubicaciones de la pantalla de la tableta, con una duración de presentación de 2 s y un intervalo entre estímulos de 2,5 s. Los participantes debían pulsar un botón "Sí" si el estímulo actual coincidía con la ubicación del presentado n elementos antes, y un botón "No" si el estímulo actual no coincidía.

Capítulo 3:

Metodología

3. Capítulo 3: Metodología

En este capítulo se detalla el proceso y la metodología a seguir para realizar la investigación y esta sigue la siguiente forma (ver figura 7).

3.1. Etapas

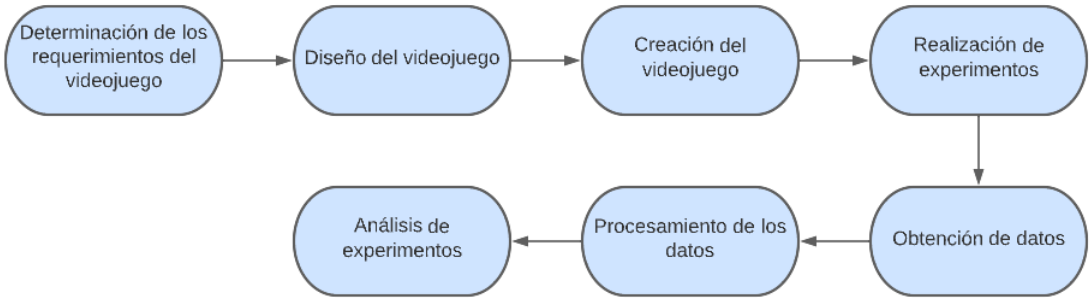


Figura 7: Etapas de la metodología

A continuación se especificará en que consiste cada uno de los puntos mencionados.

1. **Determinación de los requerimientos del videojuego:** Se estableció que datos requeríamos de los movimientos de la muñeca para la jugabilidad del juego serio.
2. **Diseño del videojuego:** Se comenzó a diseñar siguiendo los lineamientos del subsubgénero "endless runner" e implementado los movimientos de muñeca en al jugabilidad.
3. **Creación del videojuego:** Se realizo el videojuego en Unity utilizando el API para integrar el Leap-Motion.
4. **Realización de experimentos:** Los experimentos se realizarán en días separados a disposición del jugador con un tiempo de juego de 30 minutos(15 minutos por cada modo de juego) y se llevará una bitácora para posteriormente analizar.
5. **Obtención de datos:** Durante la realización de experimentos se recolecto toda la información necesaria para realizar la experimentación.
6. **Procesamiento de los datos:** Una vez obtenidos los datos en bruto, se procesaron para poder realizar las pruebas estadísticas.
7. **Análisis de experimentos:** Se realizarán pruebas estadísticas para analizar los resultados.

3.2. Sensores y software

3.2.1. Sensor: Leap Motion

El Leap Motion Controller (ver figura 8) es un módulo óptico de seguimiento de manos que captura los movimientos de sus manos. Consta de dos cámaras IR (Infrarrojas) monocromáticas y tres LED (Diodo emisor de luz) infrarrojos. El dispositivo observa un área semi hemisférica a una distancia menor de 1 metro; El fabricante sugiere una distancia de 30 cm de distancia del sensor para obtener mejores resultados al modelar las manos.

El funcionamiento de este sensor se basa en que los LED generan una luz IR y las cámaras generan casi 200 FPS (Fotogramas por segundo) de las cuales por medio de ambas cámaras generan la posición 3D de la mano.[[Weichert et al., 2013]] Para más especificaciones técnicas revisar el cuadro 1.

Figura 8: Sensor Leap Motion

Cuadro 1: Ficha técnica LeapMotion

Leap Motion especificacciones	
Dimensiones	80mm X 30mm X 11.3mm
Rango de seguimiento	Entre 10cm - 60 cm de profundidad
Campo de visión	140° X 120°
Modo de conexión	USB

3.2.2. Unity

De acuerdo con [Unity, 2022] "Unity es un motor de videojuego multiplatadorma creado por Unity Technologies." La plataforma tiene soporte de compilación con diversas plataformas como lo son:

- Computadora
- Consola
- Dispositivos móviles
- VR (Realidad virtual)

Unity puede ser utilizado junto con plataformas de modelado 3D, como lo es Blender o Maya. Sin embargo se puede manejar los objetos 3D que vienen incluidos en el mismo Unity.

El script en unity se basa en Mono, la implementación de código abierto de .NET Framework. Se puede programar utilizando UnityScript, C o Boo (Inspirado en Python).

3.3. Definición breve del videojuego

3.3.1. Movimientos

Se han contemplado cinco movimientos para la rehabilitación de muñeca extraídos de [Norkin and White, 2016]. Los cuales son: flexión (Ver figura 9), extensión (Ver figura 10), desviación radial (Ver figura 11), desviación cubital (Ver figura 12), posición de descanso central (Ver figura 13.

Figura 9: Flexión de muñeca

Figura 10: Extensión de muñeca

Figura 11: Desviación radial de muñeca

Figura 12: Desviación cubital de muñeca

Figura 13: Movimiento de descanso central

Estos movimientos tienen la particularidad de que el antebrazo debe tener tres cuartas partes apoyada sobre una mesa y que la muñeca quede libre para movimiento según el autor. Para la implementación de DDA con lógica difusa nos basamos en los movimientos en el ROM (Range Of Motion), De acuerdo con [Norkin and White, 2016] lo define como " el arco de movimiento en grados entre el comienzo y el final de un movimiento en un plano específico. El movimiento puede ocurrir en una sola articulación o en una serie de articulaciones" Se revisaron los ROM específicos para movimientos de muñeca (Ver cuadro 2[4]).

Cuadro 2: ROM de movimientos de muñeca

	AAOS	AMA	Boone y Azen	Greene y Wolf	Ryu
			20-54 años n=56	18-55 años n=20	n=40
Movimiento	ROM max	ROM min	Media (Desviación estándar)	Media	Media
Flexión	80°	60°	74.8° (6.6°)	73.3°	79.1°
Extensión	70°	60°	74.8° (6.6°)	64.9°	59.3°
Desviación radial	20°	20°	21.1° (4.0°)	25.4°	21.1°
Desviación cubital	30°	30°	35.3° (3.8°)	39.2°	37.7°

Una vez obtenidos los movimientos mediante el LeapMotion, se procesa el movimiento de la avioneta en el videojuego basándose en los movimientos de la muñeca. Se toman 2 variables directamente de la API del leap motion que son "Yaw" (Para la parte horizontal) y "Pitch" "Yaw" (Para la parte vertical) Gracias a esta variable obtenemos la dirección del movimiento basándose en la palma de la mano. Así nos proporciona la dirección como negativa o positiva y utilizamos dicha variable para ajustar la posición y rotación del personaje en pantalla tomando las posiciones en el videojuego como se muestra en la figura 14.

3.3.2. Jugabilidad

En la actualidad existen diversos géneros para los videojuegos, a continuación se listarán algunos ejemplos de la gran variedad de géneros hay.

- Acción
- Aventura
- Puzzles
- De rol
- Simulación
- Estrategia
- Deportes

[4]AAOS = Academy of Orthopaedic Surgeons; AMA = American Medical Association.

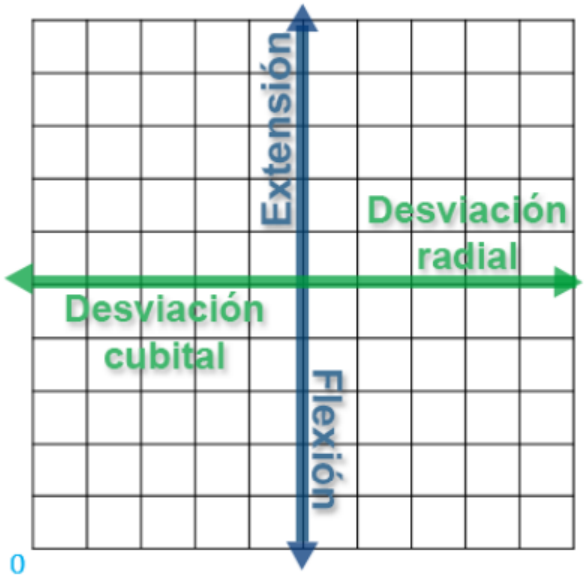


Figura 14: Plano cartesiano con la dirección de los movimientos

- MMO (Massively Multiplayer Online)
- Sandbox
- Con propósito

Los juegos serios entran en el género de juegos con propósito a su vez que los juegos de arte, juegos para mantenerse en forma y juegos educacionales. Sin embargo regularmente los juegos serios también pueden ser parte de más de 1 género. A lo largo de la investigación nos hemos encontrado con juegos de deporte, juegos de acción, puzzle, etc.

El videojuego que desarrollamos es del subsubgénero de los juegos de acción, “endless runner” como lo son títulos conocidos como “Subways surfers” ó “Temple run”. La jugabilidad se basa en que el escenario es el que se va hacia el jugador, dando la ilusión óptica de que el jugador está moviéndose a lo largo del escenario. A lo largo del juego, aparecerán obstáculos y recompensas, los cuales el jugador tendrá que obtener cambiando su posición en pantalla.

Nuestro videojuego fue desarrollado en Unity en la versión “2020.2.1f1”. Es un juego 3D (Ver figura 15) que se controla con los movimientos antes mencionados, los cuales son capturados por el Leap Motion. En el videojuego controlamos una pequeña avioneta con la cual deberemos evitar obstáculos (Ovnis) y tomar recompensas (Huesos). Al impactar con un obstáculo se va a la pantalla de inicio nuevamente, reiniciando todas las variables como: huestos tomados, el tiempo y el score.

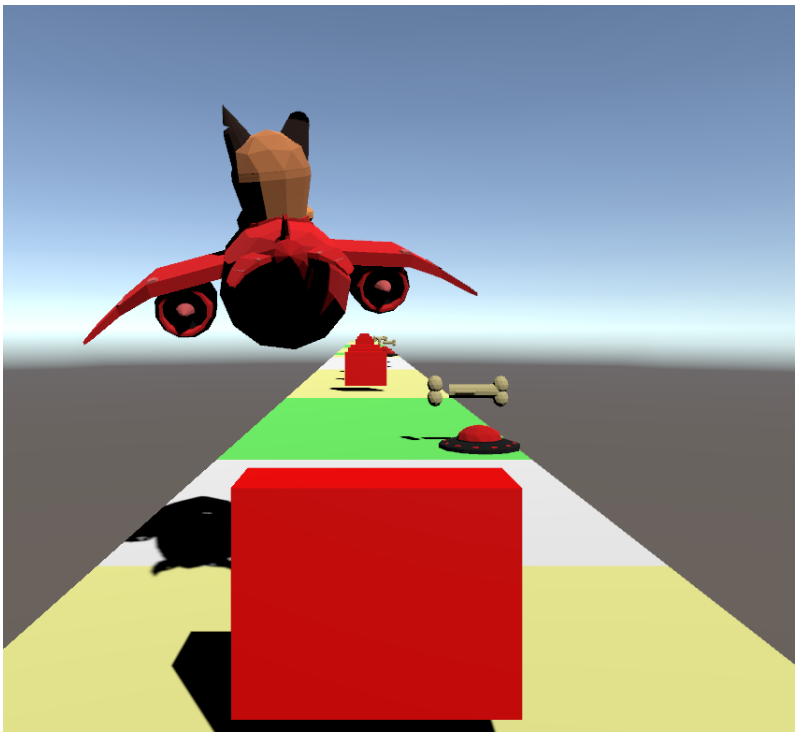


Figura 15: Implementación del videojuego en Unity

3.3.3. DDA con lógica difusa tipo I

Lógica difusa
Lofity Zadeh en 1965 propuso su teoría de conjuntos difuso; El cual definió como "una clase de conjunto con grados de pertenencia que van desde 0 a 1". Sistemas expertos, como las redes neuronales artificiales y los conjuntos difusos comparten la propiedad de ser aproximaciones sin modelo, lo que significa que no hay un modelo matemático exacto del sistema físico para controlar o para aproximar si es necesario. [Ponce-Cruz et al., 2016]

Los sistemas basados en lógica difusa conllevan ciertas etapas, las cuales están enumeradas a continuación.

1. Entrada difusa
2. Fusificación
3. Inferencia difusa
4. Defusificación
5. Salida difusa

Función de membresía
En la lógica booleana solo se utilizan dos valores, verdadero (1) y falso (0). Esta lógica no puede representar valores intermedios, los cuales representan conceptos vagos. Por medio de una función de membresía podemos ver el grado de pertenencia que tienen las variables que va de entre 0 a 1. Las funciones de membresía caracterizan lo difuso ya sea que los elementos de los conjuntos difusos sean discretos o continuos. Las funciones de membresía se representan mediante formas gráficas, como se puede apreciar en la figura 16.

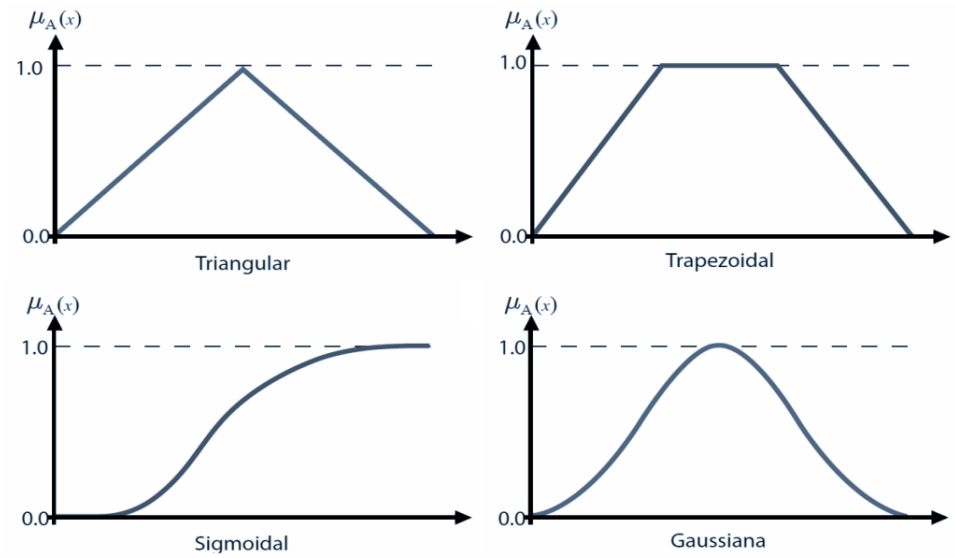
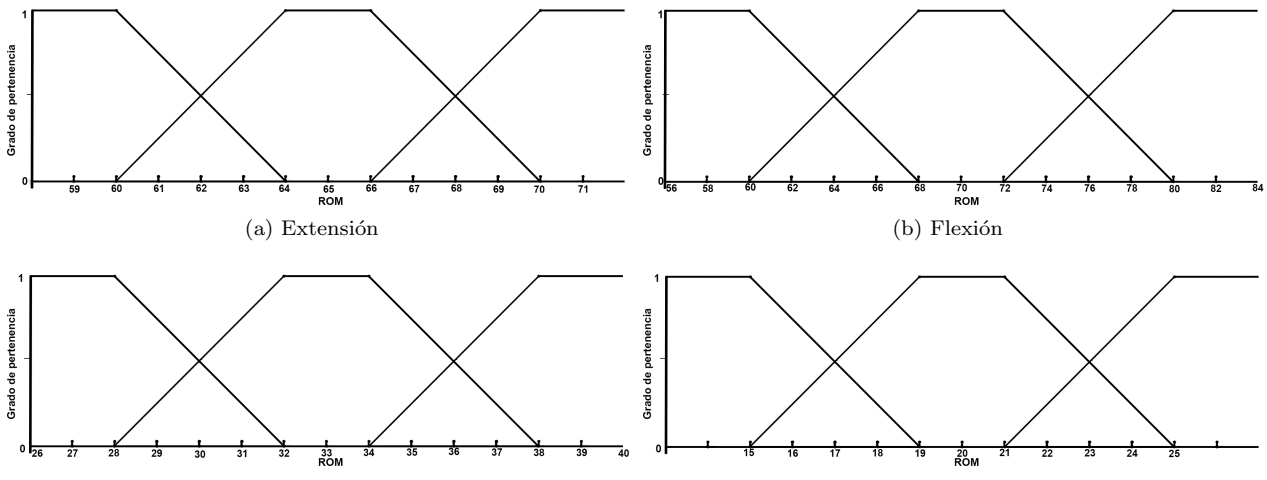


Figura 16: Funciones de membresía

Para esta investigación se ha seleccionado la función trapezoidal. Se ha seleccionado que las funciones de entrada serían la calidad de movimiento, donde manejaríamos como variables lingüísticas las siguientes:

1. Deficiente
2. Media
3. Buena

Las funciones establecidas con las variables lingüísticas antes mencionadas y los movimientos de muñeca con sus respectivos ROMs son las siguientes [17].



(a) Extensión

(b) Flexión

(c) Desviación cubital

(d) Desviación radial

Figura 17: Funciones de membresía para los movimientos

3.3.4. DDA en el videojuego

Para la implantación de DDA se seguirán los puntos enumerados a continuación.

1. **Entrada difusa:** En este etapa se establece un valor numérico como parámetro de entrada, el cual corresponderá a los grados de flexión. Con la muñeca esta entrada difusa será el ángulo que forma como se aprecia en la figura 18. En estas líneas de código es donde se calcula el ángulo, para ver a mas detalle el código revisar el anexo A (6.3).

```
anguloF = (Vector2.Angle(vectorM, vectorP) ) * pitch * 10;
anguloD = (Vector2.Angle(newxAxisVector, correct) - 123)*yaw;
```

Para el calculo de los ángulos se utiliza la formula *"Vector.Angle"* y se coloca el vector de la muñeca y el vector de la palma, luego se multiplica por la dirección el cual viene en la variable *"pitch"* y por último se hacen algunos ajustes para que esté dentro del rango establecido en la literatura y se multiplica por 10. Para los ángulos de desviaciones se utilizan los mismos principios sin embargo seguimos los lineamientos hechos en [Irving Herrera Luna, 2019]. Donde explican que hay que transponer los vectores para obtener los ángulos en la horizontal.

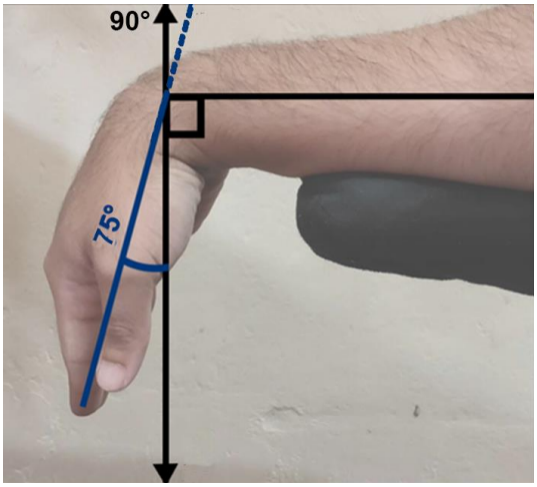


Figura 18: Aquí se puede apreciar un valor numérico de 75°para la flexión (Range Of Motion: ROM)

Teniendo la función trapezoidal para definir la velocidad se pasa a la etapa de fusificación.

2. **Fusificación:** Luego se determinará a qué tipo de flexión (deficiente, regular, buena) corresponde a dicha flexión como se ve en la figura 20. Se utilizará una función de membresía trapezoidal la cual estará definida por la ecuación (1).

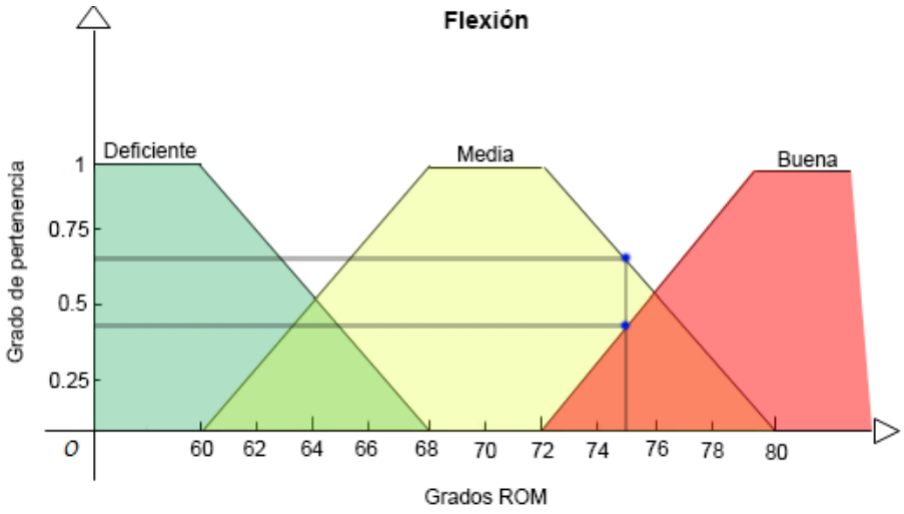


Figura 19: Función de membresía.

Si tenemos un ROM de 75° esto corresponde a 0.60 de flexión media y 0.46 de flexión buena.

$$f\left(x;a,b,c,d\right)=\begin{cases}0 & for \quad x<a\\ \frac{x-a}{b-a} & for \quad a\leq x\leq b\\ 1 & for \quad b\leq x<c\\ \frac{d-x}{d-c} & for \quad c\leq x<d\\ 0 & for \quad d\leq x\end{cases} \tag{1}$$

Donde:

- a es el límite inferior de la base mayor del trapecio.
- b es el límite inferior de la base menor del trapecio.
- c es el límite superior de la base menor del trapecio.
- d es el límite superior de la base mayor del trapecio.
- x es el valor tomado de las mediciones por el LeapMotion.

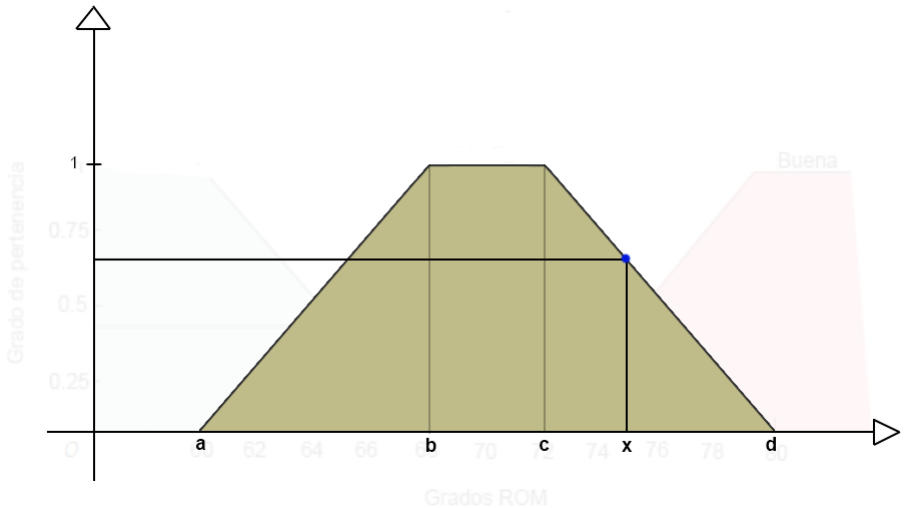


Figura 20: Puntos de la formula.

Si tenemos un ROM de 75° esto corresponde a 0.60 de flexión media y 0.46 de flexión buena.

3. **Inferencia difusa:** se establece la dificultad en valores lingüísticos(fácil,media, difícil).
   - Si la flexión es media, entonces la dificultad en el eje Y- será media.
   - Si la flexión es deficiente entonces la dificultad en el eje Y- será fácil.
   - Si la flexión es buena entonces la dificultad en el eje Y- será difícil.
   - Si la extensión es deficiente entonces la dificultad en el eje Y+ será fácil.
   - Si la extensión es media, entonces la dificultad en el eje Y+ será media.
   - Si la extensión es buena, entonces la dificultad en el eje Y+ será difícil.
   - Si la desviación cubital es deficiente, entonces la dificultad en el eje X- será fácil.
   - Si la desviación cubital es media, entonces la dificultad en el eje X- será media.
   - Si la desviación cubital es buena, entonces la dificultad en el eje X- será difícil.
   - Si la desviación radial es deficiente, entonces la dificultad en el eje X+ será fácil.
   - Si la desviación radial es media, entonces la dificultad en el eje X+ será media.
   - Si la desviación radial es buena, entonces la dificultad en el eje X+ será difícil
4. **Defusificación:** Se establece que tamaño en píxeles tomará la zona de habilidad con cada coordenada (Figura 21). Esto empleando las diferentes funciones de dificultades (Figura 22).

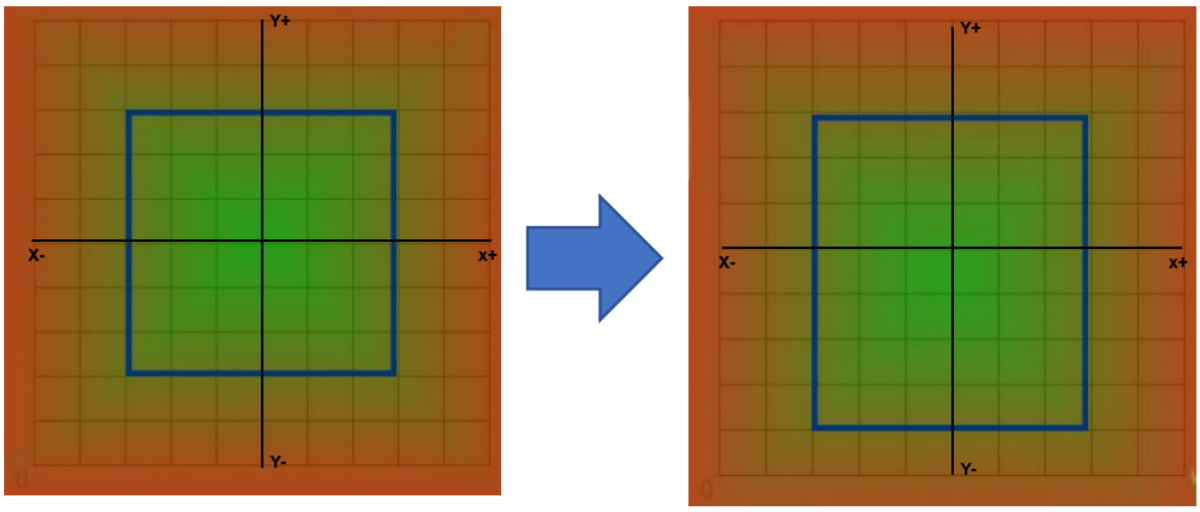
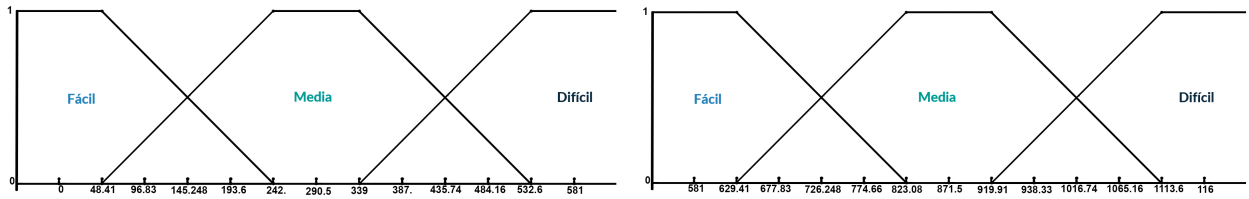
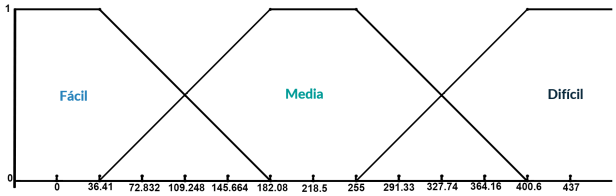


Figura 21: Reducción de zona de habilidad

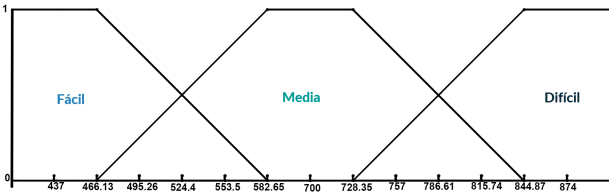


(a) X-

(b) X+



(c) Y-



(d) Y+

Figura 22: Funciones de membresía para las díficultades

Así se determina la coordenada exacta dentro del videojuego donde pueden aparecer huesos. Se utilizará el método de Center of sums (CS). La formula (2) describe este método, donde b1 y b2 son los límites del área del trapecio.

$$Área = \frac{(b_1 + b_2)}{2} \cdot altura \tag{2}$$

5. **Salida difusa:** Se establece un valor numérico concreto. Para este punto en específico se toma en cuenta el área de la matriz. En este caso en concreto el valor difuso que toma la dificultad sería fácil.

3.4. Modalidades del videojuego

Las modalidades del videojuego son 2, una con DDA y otro sin DDA. La que contiene DDA establece con base a la zona de habilidad las recompensas (los huesos). Si el jugador no puede alcanzar una recompensa esta se adaptará a su zona de habilidad, como se muestra en la figura 23.

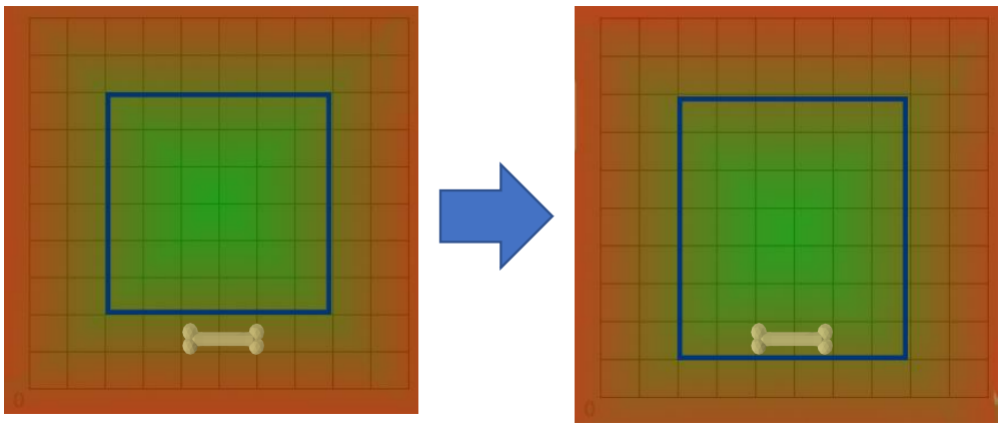


Figura 23: Zona alcanzable

Sin embargo en el modo sin DDA se está contemplando que la aparición de recompensas sea de manera aleatoria sin importar la habilidad del jugador. La velocidad de movimiento del escenario irá aumentando constantemente. Para revisar a detalle los códigos de generación de nivel de ambos modos revisar los anexos B y C.

3.5. Diseño experimental

El equipo para la adquisición de registros fue una computadora con sistema operativo Windows 11, con un procesador Intel Core i5 7ma generación, 4 núcleos a 3.5 Ghz, con una tarjeta gráfica dedicada Nvidia 1050Ti. Se considera una distancia entre 15 cm a 30 cm para el sensor Leap Motion y la mano.

Se registraron 4 usuarios (dos hombres y dos mujeres) con una edad promedio de 23.75 años. El número de usuarios es debido a la pandemia de COVID 19. Las pruebas se realizarán en la mano dominante que nos indiquen los mismo usuarios.

Cada sesión de juego será de 15 minutos, los jugadores al perder tendrán la oportunidad de volver a jugar con el tiempo límite antes mencionado.

3.6. Métricas para medir el rendimiento de las modalidades

El videojuego toma en cuenta el tiempo transcurrido y las recompensas obtenidas. Por lo que este puntaje del videojuego (Ver sección 4.2 para ver a detalle el cálculo de la puntación) se toma en cuenta para medir el rendimiento del jugador, en ambos modos de juego persisten estas variables.

Se mencionó anteriormente que se hará uso del cuestionario GEQ (Ver sección 5.4) para medir el nivel de compromiso del jugador.

Se compararán las variables obtenidas con análisis estadístico.

Capítulo 4:

El videojuego

4. Capítulo 4: El videojuego

4.1. Jugabilidad

El juego es de tipo *"endless runner"*, de acuerdo con [Törnquist, 2022] es donde el jugador constantemente se mueve hacia una dirección determinada y tiene la tarea de sobrevivir el mayor tiempo posible mientras que evita toparse con obstáculos que lo matarán. La puntación está basada en el tiempo que sobrevivió.

Teniendo definido el tipo de juego, se tiene que adaptar la idea de la rehabilitación de muñeca dentro de la jugabilidad del juego. Por lo que se estableció la serie de movimientos, mencionados en la sección 3.3.

La idea lógica del videojuego es bastante simple, se tiene una serie de objetos prefabricados, figura 24, los cuales se irán acercando al jugador para que al pasarlo se eliminarán para evitar utilización de recursos de manera innecesaria.

Para realizar una visualización de la jugabilidad puede consultar el siguiente link.

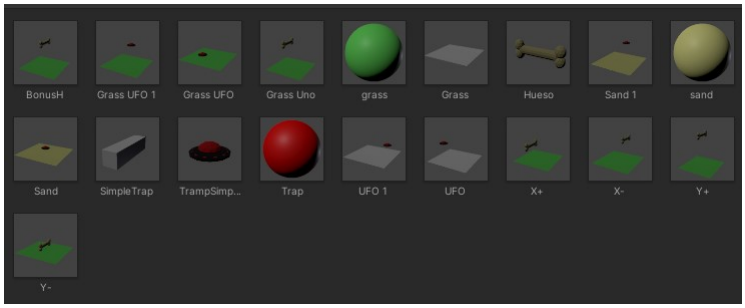


Figura 24: Objetos prefabricados del videojuego

4.1.1. Obstáculos

Con la intención de que el jugador se mueva de su posición original; se han empleado obstáculos a lo largo del nivel, los cuales tienen forma de platillo volador, como se ve en la figura 25. Si el jugador colisiona con estos, el nivel acaba y la puntuación se deja de acumular. Para evitar que el jugador se frustre demasiado pronto, los *hitbox*[5] han sido dimensionados un 3 % más pequeños que el modelo 3D.

Figura 25: Obstáculo con forma de platillo volador

[5] Área invisible en forma de caja que, cuando se toca, cuenta como golpe.[Lazaridis et al., 2021]

4.1.2. Recompensas

Para motivar al jugador a no solo moverse para esquivar obstáculos, si no moverse a zonas específicas, se han colocado recompensas, los cuales tienen forma de hueso, figura 26. Cuando impacta el jugador con ellas, aumenta la puntación en el juego.

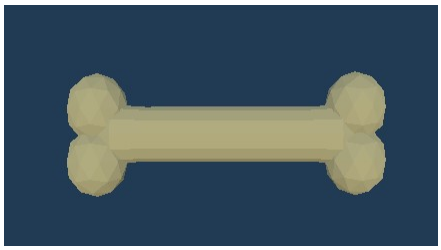


Figura 26: Recompensa con forma de hueso

4.2. Puntuación

La puntuación del jugador aparecerá siempre en la pantalla principal (Figura 27) y está basada en 2 variables, tiempo y recompensas recolectas. El tiempo es los segundos que ha estado jugando el nivel sin perder, ya que al perder (Colisionando con un obstáculo) este valor regresa a 0; y también las recompensas alcanzadas, las cuales te dan 5 puntos por cada recompensa recogida. La fórmula está representada en la ecuación: 3.

$$Score = Tiempo + (Huesos * 5) \tag{3}$$

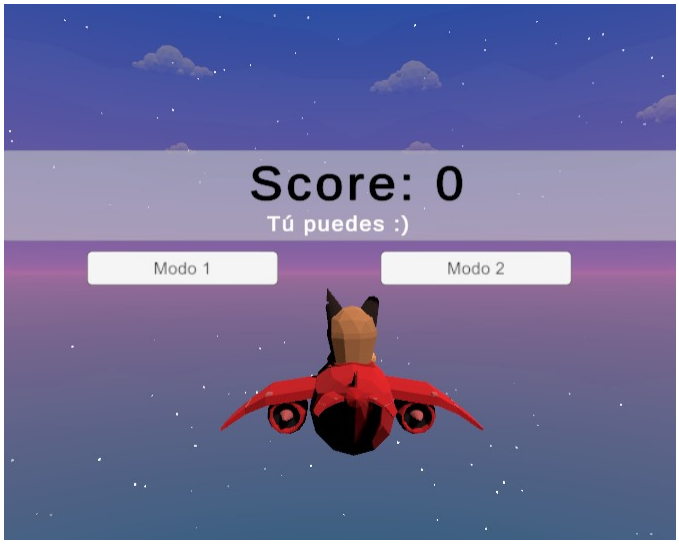


Figura 27: Pantalla principal, inicial

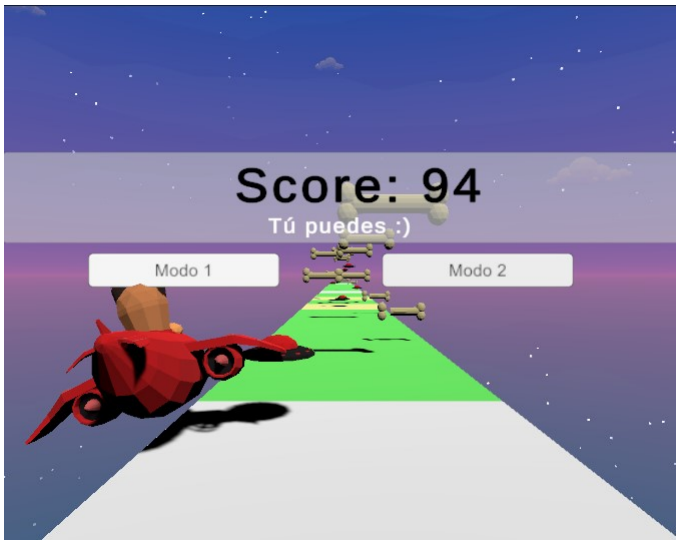


Figura 28: Pantalla principal, tras perder

4.3. Modos de juego

Se mencionó anteriormente en la sección 3.4, el juego contiene 2 modos los cuales se ejecutan por separado, esto es de muhca utilidad para la investigación, ya que de esta forma tendremos resultados de ambos modos de juego sin mezclar datos. Se podrá acceder a cada modo de juego con la ayuda de la interfaz del menú, donde éste contiene 2 botones, como se puede apreciar en la figura 28.

4.3.1. Modo de juego 1

Este modo de juego no contiene DDA, se utiliza una aceleración de 0.0005, la cual se aplica a la velocidad del juego. Por lo que con el paso del tiempo el juego se torna más difícil debido a que va más rápido. El sistema de puntación es el mismo para ambos modos de juego.

4.3.2. Modo de juego 2

Para este modo de juego el DDA está aplicado en unas recompensas fuera de los objetos prefabricados, los cuales se generan tras haber tomado muestras del rango de movilidad del jugador, estos se generan y se mueven hacía el jugador, estos se actualizan cada 22 segundos y si el jugador no los atrapa de igual forma desaparecen tras pasarlo.

La resolución del juego se fijó a 1162X874 píxeles (Figura 29). Esto para poder calcular la posición exacta de los huesos.

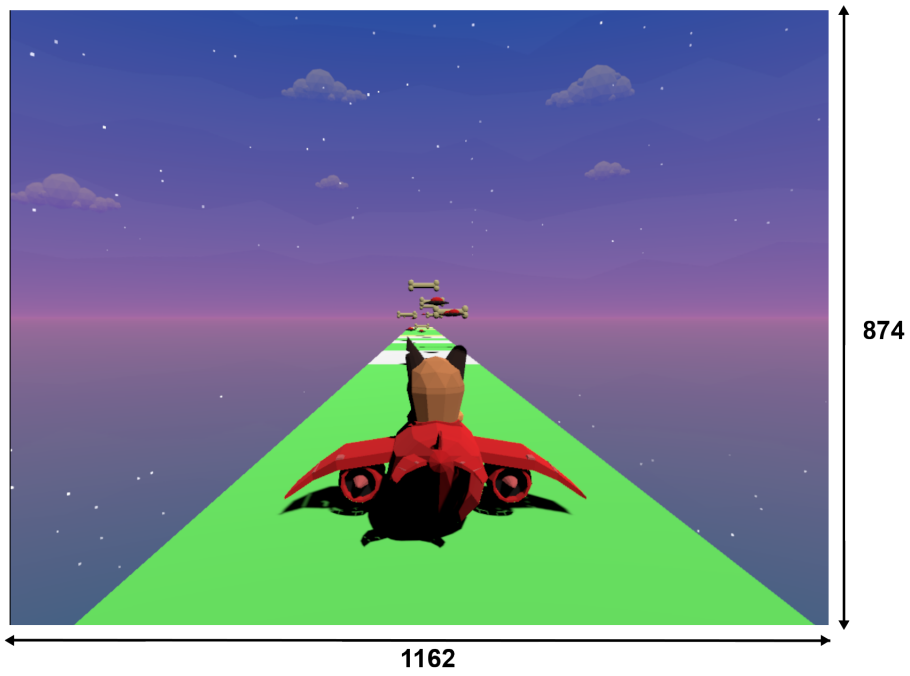


Figura 29: Resolución del juego

La colocación de estas recompensas en forma de hueso utilizan una serie de formulas para establecer cada coordenada:

- Y-: $v1 = \frac{(z1*5)}{437}$
- X+: $v2 = \frac{(z2*20)}{1162} - 10$
- X-: $v3 = \frac{(z3*10)}{581} - 10$
- Y+: $v4 = \frac{(z4*10)}{874}$

Donde cada Z representa el grado de pertenencia de la calidad del movimiento. Los "-10" son para que el *hitbox* esté al alcance del jugador.

Capítulo 5:

Experimentación y resultados

5. Capítulo 5: Experimentación y resultados

5.1. Participantes

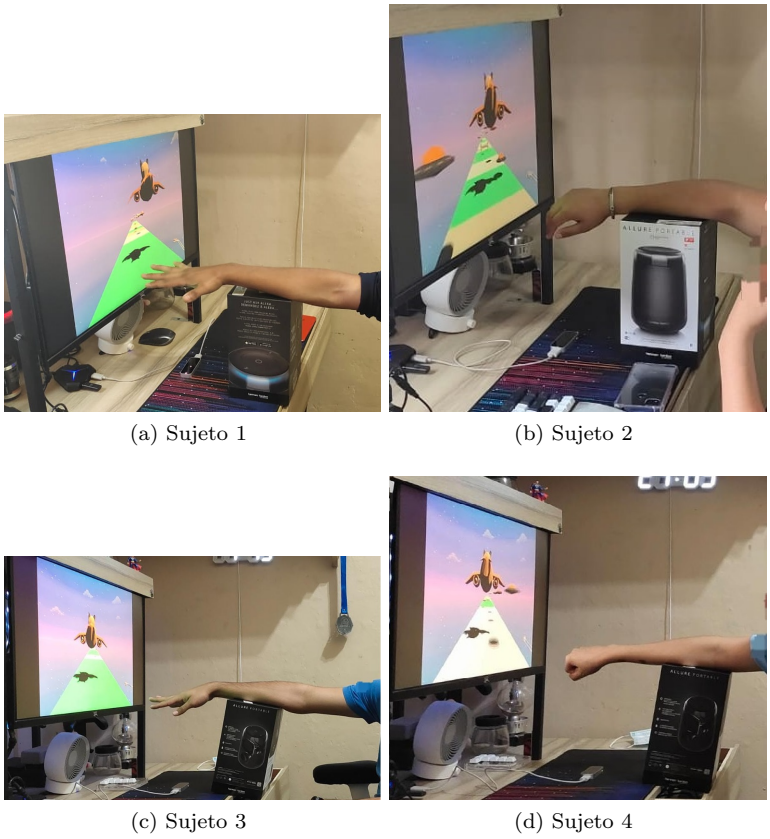
Todos los participantes eran diestros. Así que las pruebas fueron realizadas en la mano derecha. El uso de un apoyo es debido a 3 puntos.

1. El leap motion funciona a una distancia óptima de 30 cm como se menciono con anterioridad en la 3.2.1.

2. La autora Norkin sugiere usar un apoyo donde la muñeca quede suspendida 3.3.1.

3. El brazo de los jugadores al tener sesiones prolongadas puede llegar a cansarse.

Se les preguntó antes de iniciar la sesión de prueba, si con anterioridad habían jugado videojuegos; dos de ellos dijeron que sí y 2 de ellos no. En la figura 30 se aprecia la experimentación con cada uno de los participantes.



(a) Sujeto 1 (b) Sujeto 2

(c) Sujeto 3 (d) Sujeto 4

Figura 30: Sujetos realizando pruebas

5.2. Resultados de la modalidad sin DDA

Los resultados obtenidos fueron variados, debido a las diferencias entre cada sesión dentro del tiempo limite, por lo que se promediaron dichas sesiones y se manejaron estos promedios para las pruebas estadísticas como se aprecia en la cuadro 3. Este cuadro nos muestra que el score máximo para este modo, lo ha logrado el sujeto 1; tanto en tiempo como en huesos tomados.

Cuadro 3: Resultados de modo 1

	Modo 1		
Sujeto de prueba	Score	Tiempo	Huesos tomados
1	160.4285	68.2857	18.4285
2	116.8333	54.75	12.4166
3	135.3333	66.4444	13.7777
4	114.5454	60.9090	10.7272

Estos resultados obtenidos son para el modo 1, el cual no contiene DDA. El cuadro muestra las variables con las que está compuesta la variable score. El score se calcula sumando el tiempo transcurrido en segundos y el número de huesos tomados multiplicado por 5 como se explica en la ecuación 4.

5.3. Resultados de la modalidad basada en DDA

Para el modo 2 se obtuvieron los siguientes resultados representados en la cuadro 4. Este cuadro muestra que el score máximo lo ha logrado el sujeto 4; tanto en tiempo como en huesos tomados.

Cuadro 4: Resultados de modo 2

	Modo 2		
Sujeto de prueba	Score	Tiempo	Huesos tomados
1	159.125	73.75	17.075
2	83.25	37.625	9.125
3	62.9333	33.6	5.8666
4	219.5714	124.5714	19

Se han realizado comparaciones entre los sujetos por cada una de las variables como se pueden apreciar en las siguientes figuras.

Para los resultados de la variable score, representados en la figura 31 vemos el promedio de la variable score obtenida por cada uno de los sujetos, el sujeto 4 obtuvo el mayor score (219.5 pts) de los 4 sujetos, en el modo con DDA. Esto es debido a que obtuvo un mayor numero de huesos y se mantuvo mayor tiempo vivo dentro del videojuego.

Para los resultados de la variable tiempo, representados en la figura 32 se aprecia que igual el sujeto 4 tuvo mayor tiempo de permanencia en nivel (19 s) en el modo con DDA que a comparación con los demás sujetos.

Para los resultados de la variable huesos, representados en la figura 33 podemos apreciar como el sujeto 1 en el modo sin DDA (18.42 huesos) y el sujeto 4 en el modo con DDA (19 huesos) han tenido puntaciones más altas.

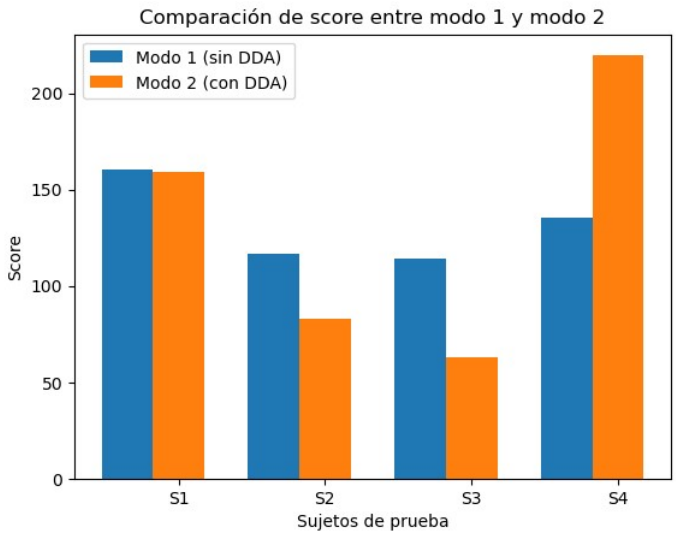


Figura 31: Promedio del score para los 4 sujetos de prueba, comparando ambos modos

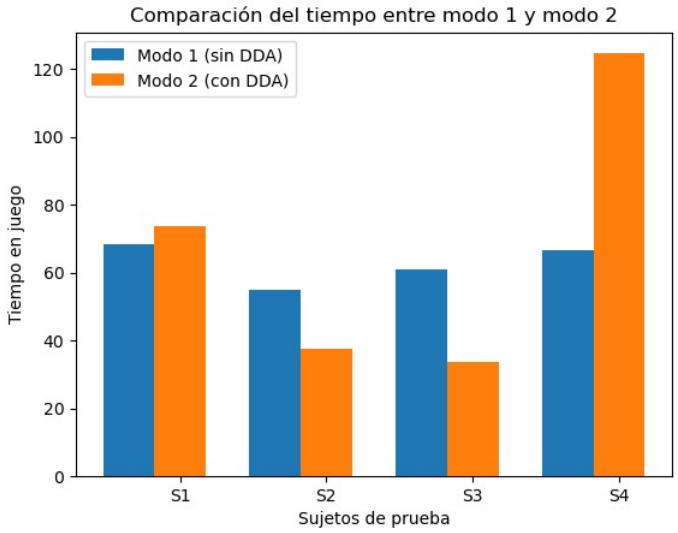


Figura 32: Promedio del tiempo de juego dentro el nivel para los 4 sujetos de prueba, comparando ambos modos

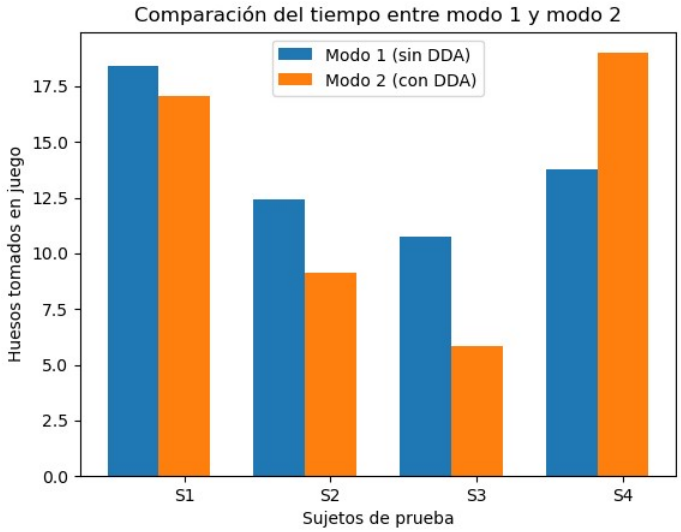


Figura 33: Promedio del número de huesos tomados para los 4 sujetos de prueba, comparando ambos modos

5.4. Cuestionario GEQ

Para determinar lo que en verdad sintió el jugador durante su sesión de juego. se utilizó un cuestionario llamado "Game Engagement Questionnaire" (G.E.Q.) respaldado por análisis clásico y de Rasch. Elaborado y revisado en [Brockmyer et al., 2009].

El cuestionario tiene como finalidad cuantificar el nivel de compromiso entre el jugador y el juego. Consta de 19 preguntas (Ver figura 34) Las cuales están divididas en grupos para estados de ánimo diferentes.

- Inmersión: El jugador tiene una experiencia de involucrarse en la experiencia del juego o se siente parte de.
- Presencia: El jugador está en un estado normal de conciencia y reconoce estar en un entorno virtual.
- Flujo: EL jugador pasa por una experiencia de disfrute cuando existe un equilibrio entre habilidad y desafío.
- Absorción: A diferencia de la inmersión y la presencia, y el flujo, estar en un estado de absorción psicológica induce un estado alterado donde la conciencia es menos accesible.

Los cuales corresponden con un 5.3 % a la inmersion, 15.8 % presencia, 26.3 % a la absorción y 52.6 % de flujo. (Ver figura 35).

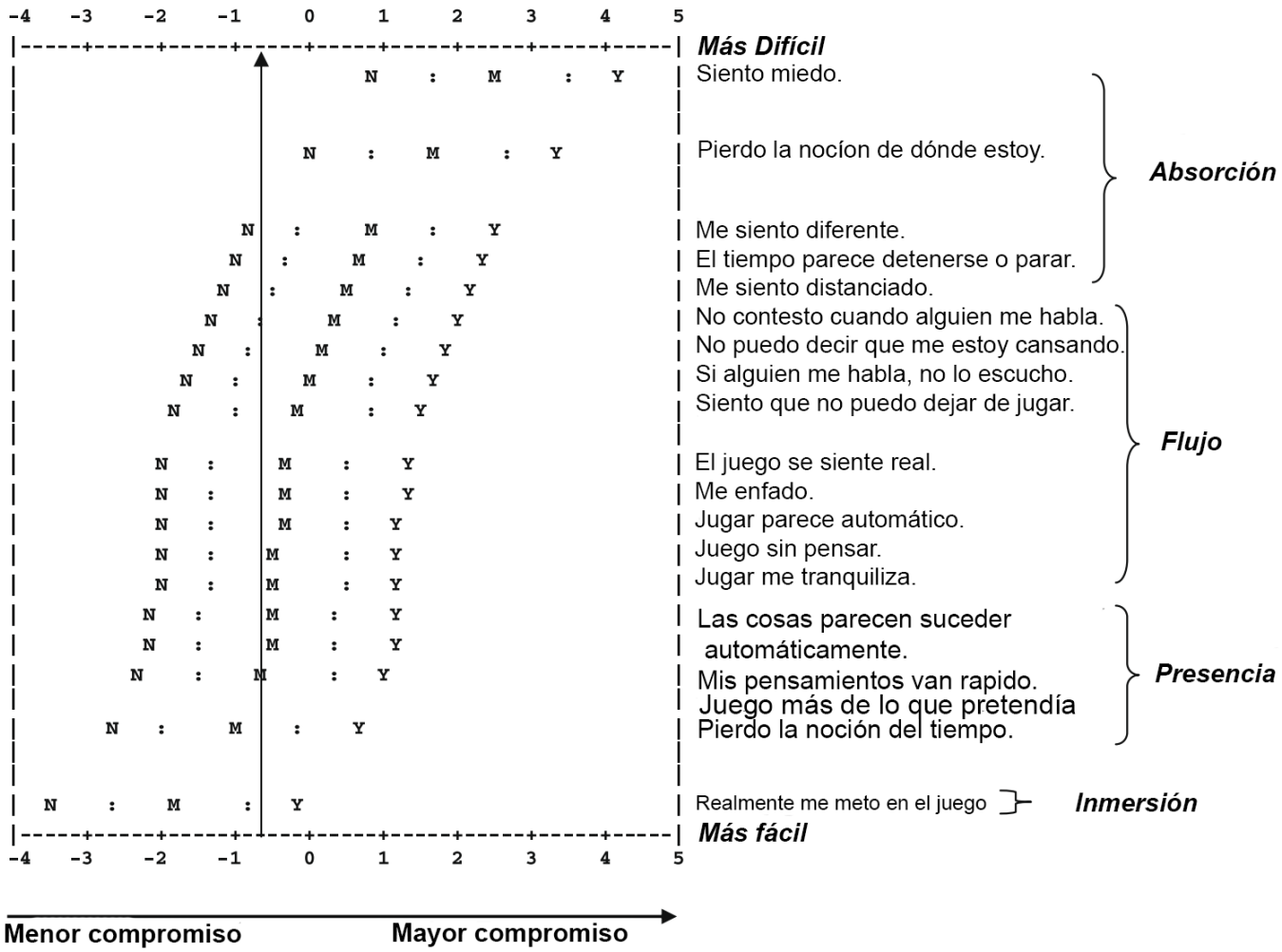


Figura 34: Preguntas del cuestionario con sus estados de ánimo

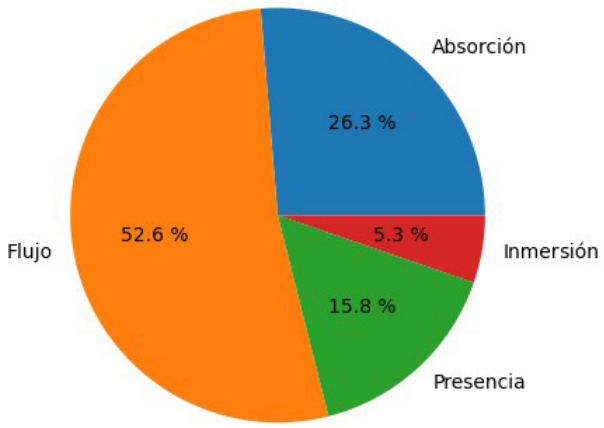


Figura 35: Gráfica de pastel de los estados de ánimo del cuestionario

Se realizó el mismo cuestionario 2 veces por cada sujeto, debido a que jugaba los 2 modos. Después de cada sesión se les pedía que contestaran el cuestionario. Los resultados obtenidos (Ver figura 36) indican que existe mayor inmersión sobre los otros 3 estados. También se logra apreciar que existe menor presencia en el modo que contiene DDA. En el flujo no existe tanta diferencia y por último la absorción en el modo 1 es mayor.

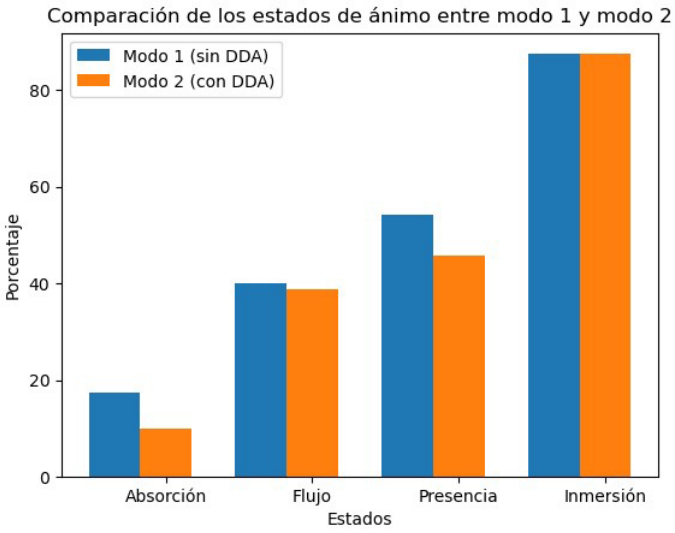


Figura 36: Gráfica de los resultados con cada estado de ánimo

A su vez se empleó la escala propuesta por los creadores del cuestionario, donde se muestra que preguntas contienen más peso que otras (Ver figura 34). En los anexos D y E se encuentran los resultados obtenidos para el modo de juego 1 y 2 sumando los valores individuales de la pregunta (Dependiendo si el resultado fue un "Sí (Y)", un "No (N).º un "Más o menos (M)"). Los resultados aplicando esta escala, muestran un nivel de compromiso de -0.329411765 para el modo 1 y de -0.470588235 para el modo 2 como se aprecia en la figura 37. Podemos apreciar como la diferencia es sutil y sin embargo se logra apreciar en la gráfica.

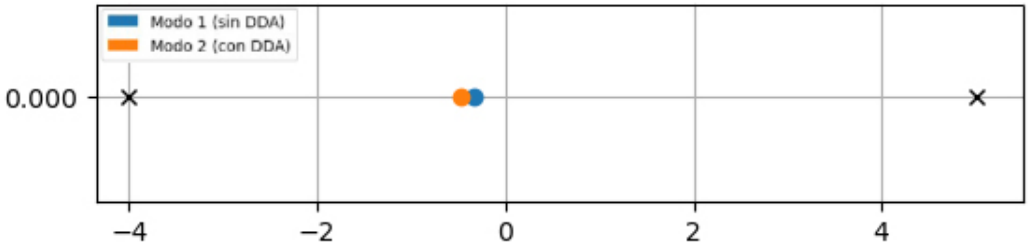


Figura 37: Escala donde -4 es menos compromiso y 5 es mayor compromiso

5.5. Análisis estadístico

Las siguientes pruebas se realizaron con la finalidad de determinar si existían diferencias significativas entre ambos modos de juego, donde se empleó un valor de $\alpha = 0.05$ como intervalo de confianza:

- Shapiro-Wilk. Se utilizó para determinar si las variables registradas seguían una distribución normal ($p >= 0.05$) o no ($p <0.05$).
- T de Student. Se utilizó para determinar si existe diferencia significativa ($p >=0.05$) o no ($p <0.05$).

Se realizaron dichas pruebas, en la variable de *score*, sin embargo a su vez se aplicaron en las variables de *tiempo* y *huesos tomados*. Con las cuales se forman las variables de *score*. Por lo que se realizaron dichas pruebas en 3 variables en total. A continuación se muestra el código en R para las pruebas, las variables son las siguientes:

- grupo1S: contiene los score de los 4 jugadores en orden del modo 1 (Ver código 1).
- grupo2S: contiene los score de los 4 jugadores en orden del modo 2 (Ver código 1).
- grupo1T: contiene la cantidad de tiempo tomada de los 4 jugadores en orden del modo 1 (Ver código 2).
- grupo2T: contiene la cantidad de tiempo tomada de los 4 jugadores en orden del modo 2 (Ver código 2).
- grupo1H: contiene la cantidad de huesos tomada de los 4 jugadores en orden del modo 1 (Ver código 3).
- grupo2H: contiene la cantidad de huesos tomada de los 4 jugadores en orden del modo 2 (Ver código 3).
- grupo1E: contiene la la puntación obtenida en el cuestionario CEQ tomada de los 4 jugadores en orden del modo 1 (Ver código 4).
- grupo2E: contiene la puntación obtenida en el cuestionario CEQ tomada de los 4 jugadores en orden del modo 2 (Ver código 4).

La prueba Shapiro Wilk reveló que las variables recolectadas seguían una distribución normal para después aplicar la prueba T student.

```
> grupo1S<- c(160.42857, 116.8333, 114.5454, 135.3333)
> shapiro.test(grupo1S)

  Shapiro-Wilk normality test

data:  grupo1S
W = 0.88486, p-value = 0.3598

> grupo2S<- c(159.125, 83.25, 62.9333, 219.5714)
> shapiro.test(grupo2S)

  Shapiro-Wilk normality test

data:  grupo2S
W = 0.93032, p-value = 0.5963

> t.test(grupo1S, grupo2S, paired = TRUE, conf.level =0.95)

  Paired t-test

data:  grupo1S and grupo2S
t = 0.018764, df = 3, p-value = 0.9862
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -95.29690  96.42734
sample estimates:
mean of the differences
              0.5652175
```

Listing 1: Código para análisis estadístico para la variable score de ambos modos

La prueba t student reveló que no existe diferencia significativa entre ambos modos de juego en lo que concierne al *score* (Ver código 1). Reportando un valor p =0.9862. Para las otras dos variables se repitieron las pruebas.

```
> grupo1T<- c(68.28571429, 54.75, 60.90909091, 66.44444444)
> shapiro.test(grupo1T)

  Shapiro-Wilk normality test

data:  grupo1T
W = 0.93662, p-value = 0.6339

> grupo2T<- c(73.75, 37.625, 33.6, 124.5714286)
> shapiro.test(grupo2T)

  Shapiro-Wilk normality test

data:  grupo2T
W = 0.87982, p-value = 0.3379

> t.test(grupo1T, grupo2T, paired = TRUE, conf.level =0.95)

  Paired t-test

data:  grupo1T and grupo2T
t = -0.25138, df = 3, p-value = 0.8178
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -65.42240  55.84381
sample estimates:
mean of the differences
             -4.789295
```

Listing 2: Código para análisis estadístico para la variable tiempo de ambos modos

```
> grupo1H<- c(18.42857143, 12.41666667, 10.72727273, 13.77777778)
> shapiro.test(grupo1H)

  Shapiro-Wilk normality test

data:  grupo1H
W = 0.92946, p-value = 0.5912

> grupo2H<- c(17.075, 9.125, 5.866666667, 19)
> shapiro.test(grupo2H)

  Shapiro-Wilk normality test

data:  grupo2H
W = 0.90534, p-value = 0.458

> t.test(grupo1H, grupo2H, paired = TRUE, conf.level =0.95)

  Paired t-test

data:  grupo1H and grupo2H
t = 0.48306, df = 3, p-value = 0.6621
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -5.984333  8.126144
sample estimates:
mean of the differences
               1.070905
```

Listing 3: Código para análisis estadístico para la variable huesos de ambos modos

Se determinó que para ambas variables (*Tiempo* y *Huesos*) no existe diferencia significativa tampoco. La variable de *Tiempo* en juego tuvo un valor $p = 0.8178$ (Ver código 2). Mientras que la variable *Huesos* tomados obtuvo un valor p =0.6621 (Ver código 3).

Se utilizaron los resultados obtenidos del cuestionario CEQ de la sección anterior, para realizar también análisis estadístico con los resultados promediados de los sujetos de prueba. Se obtuvieron los siguientes resultados (Ver código 4). Donde obtuvimos un valor p =0.9428 en la prueba de Shapiro Wilk lo cual demuestra que sigue una distribución normal.

```
> grupo1E<- c(17.5, 40,54.16667 ,87.5)
> shapiro.test(grupo1E)

  Shapiro-Wilk normality test

data:  grupo1E
W = 0.98722, p-value = 0.9428

> grupo2E<- c(10, 38.75, 45.8333, 87.5)
> shapiro.test(grupo2E)

  Shapiro-Wilk normality test

data:  grupo2E
W = 0.96545, p-value = 0.8131
```

Listing 4: Código para análisis estadístico para los resultados del cuestionario CEQ de ambos modos

Posteriormente se aplicó la prueba T de Student para determinar si existía diferencia significativa entre estos 2 grupos de resultados. Se obtuvo un valor $p = 0.1363$ (Ver código 5) por lo que no existe diferencia significativa.

```
> t.test(grupo1E, grupo2E, paired = TRUE, conf.level =0.95)

Paired t-test

data:  grupo1E and grupo2E
t = 2.0078, df = 3, p-value = 0.1383
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -2.498696 11.040381
sample estimates:
mean of the differences
               4.270843
```

Listing 5: Código para análisis estadístico para la variable emoción de ambos modos

Capítulo 6:

Conclusiones y trabajo futuro

6. Capítulo 6: Conclusiones y trabajo futuro

6.1. Conclusiones

El objetivo de este trabajo era determinar si existía diferencia significativa en aplicar un método de DDA con lógica difusa y uno que no lo tuviera. Para determinarlo se llevaron a cabo pruebas con 4 sujetos. Posteriormente se aplicó la prueba estadística de Shapiro-Wilk para determinar si las variables a analizar seguían una distribución normal. Una vez determinada la distribución, posteriormente se empleó t de Student en las 3 variables (*score, tiempo y huesos*) para determinar si existían diferencias significativas entre ambos modos.
Los resultados muestran que no existe diferencias significativas en términos de score y emociones entre el modo con DDA y el modo sin DDA.

El resultado obtenido, por las métricas que los autores de GEQ sugieren (Ver figura 34). Nos indica el peso que tiene cada una de las preguntas y a su vez sus 3 posibles respuestas. Una vez utilizada esta escala obtuvimos los siguientes resultados. El modo sin DDA tiene un nivel mayor de compromiso (-0.329411765) comparado con el modo con DDA (-0.470588235) como se aprecia en la figura 37.

Se promediaron y graficaron los resultados obtenidos del cuestionario para ambos modos y se muestran en la figura 36. De acuerdo con estos resultados, representa que la inmersión en el juego estuvieron presente (87.5 pts para ambos modos), y la presencia decayó en el modo con DDA, pasó de 54.1666 pts a 45.8333 pts, haciendo el juego entretenido. Los sujetos de prueba afirman que el videojuego tiene potencial para ser más entretenido.

Sin embargo se realizaron pruebas estadísticas en los resultados demostrando que no existen diferencias significativas entre los modos de juego basándonos en los estados de ánimo (Absorción, Flujo, Presencia, Inmersión) de los jugadores. No obstante, se tendría que reevaluar algunas preguntas en el cuestionario que no coinciden con nuestro tipo de videojuego como “Siento miedo” o “El juego se siente real” ya que en nuestro videojuego no hay intención alguna de hacer sentir miedo o hacerlo realista.

6.2. Limitaciones

A lo largo de esta investigación se buscó determinar si existía una diferencia significativa en términos de score, entre ambos modos de juego. Sin embargo esta investigación ha tenido ciertas limitaciones comenzando por los sujetos de prueba. Los participantes de la prueba se han limitado a 4 debido a la pandemia de covid-19. A su vez estos no tienen necesidad de rehabilitación en su muñeca. Por lo que tienen un mayor ROM al jugar.

6.3. Trabajo futuro

Para solventar las limitaciones de esta investigación se propone lo siguiente:

- Involucrar a expertos en rehabilitación, para que den su retroalimentación para el juego serio.
- Optimizar la jugabilidad del videojuego, con los comentarios de los sujetos de prueba.
- Explorar un nuevo método para aumentar la participación de DDA como quizá adaptar la velocidad en respuesta del comportamiento del jugador.
- Realizar experimentos con sujetos con necesidad de rehabilitación en el área de la muñeca.
- Mejorar el algoritmo de reconocimiento de ángulos.
- Añadir un sensor como el oculus rift para agregar más inmersión al juego.
- Añadir varios sensores como de respiración y actividad electro dérmica para obtener información biométrica del sujeto de prueba.
- Agregar retroalimentación sonora al colisionar, o tomar recompensas en el videojuego.

Anexos

Anexo A: código de captura de movimiento y ángulo

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


public class Movimiento : MonoBehaviour
{
    public float Horizontal, Vertical, RotationV, RotationH;
    public float MaxLeft, MaxRight, MaxBack, MaxFor;
    public float MaxUp, MaxDown;
    public float MaxRtLeft, MaxRtRight, MaxRtUp, MaxRtDown;
    public float Speed,SpeedR;
    public double angle;
    public float pitch, yaw;
    Vector2 vectorP, vectorM, newxAxisVector, correct;
    Vector3  vectorF;
    public double anguloF, anguloD;
    Controller leapController = new Controller();
    LeapProvider proveedor; /*It supplis images and leap hands*/


    float yBorrar;
    internal Transform tr;
    float ppz,ex;
    float wx,wz;
    float timer = 0.0f;

    // Start is called before the first frame update
    void Start()
    {
        tr = transform;
        proveedor = FindObjectOfType<LeapProvider>() as LeapProvider;

    }

    // Update is called once per frame
    void Update()
    {
        timer += Time.deltaTime; //contador
        Frame frame = leapController.Frame(); // controller is a Controller object


        if (timer <= 0.25)
        {
            //Debug.Log("TIMER" + timer);
            for (int h = 0; h < frame.Hands.Count; h++)
            {
                Hand leapHand = frame.Hands[h]; //basandose en la palma de la mano
                List<Finger> allFingers = leapHand.Fingers;
                foreach (Finger dedo in allFingers)
                {
                    //Cordenada de las falanges
                    Bone bone = dedo.Bone(Bone.BoneType.TYPE_PROXIMAL);
                    Vector falange = bone.PrevJoint;
                    ppz = falange.z;
                    vectorF = new Vector3(falange.x, falange.y, falange.z);
                    vectorF.Normalize();
                    Vector direction = dedo.Direction;
                }

                pitch = leapHand.Direction.Pitch;
                yaw = leapHand.Direction.Yaw;
                //Cordenada de la muneca y palma
```

```
67                  Leap.Vector muneca = leapHand.WristPosition;
68                  Leap.Vector palma = leapHand.PalmPosition;
69
70
71                  vectorM = new Vector2(muneca.y, muneca.z);
72                  vectorP = new Vector2(palma.y, palma.z);
73                  vectorM.Normalize();
74                  vectorP.Normalize(); //xzy
75
76                  //Cordenada del codo
77                  Arm brazo = frame.Hands[0].Arm;
78                  ex = brazo.ElbowPosition.x;
79
80                  anguloF = (Vector2.Angle(vectorM, vectorP) ) * pitch * 10; //Ya esta en
    grados, no en radianes 43 Math.Abs(
81
82
83
84              }
85
86              for (int h = 0; h < frame.Hands.Count; h++)
87              {
88                  Hand leapHand = frame.Hands[h];
89
90                  Vector handXBasis = leapHand.PalmNormal.Cross(leapHand.Direction).Normalized
    ;
91                  Vector handYBasis = -leapHand.PalmNormal;
92                  Vector handZBasis = -leapHand.Direction;
93                  Vector handOrigin = leapHand.PalmPosition;
94                  Matrix handTransform = new Matrix(handXBasis, handYBasis, handZBasis,
    handOrigin);
95                  handTransform = handTransform.RigidInverse();
96
97                  for (int f = 0; f < leapHand.Fingers.Count; f++)
98                  {
99                      Finger leapFinger = leapHand.Fingers[f];
100                     Vector transformedPosition = handTransform.TransformPoint(leapFinger.
    TipPosition);
101                     //Vector transformedDirection = handTransform.Transform Direction(
    leapFinger.Direction);
102                     correct = new Vector2(-transformedPosition[0], -transformedPosition[2]);
    //Se multiplica por -1 para cambiar la direccion
103
    // Do something with the transformed fingers
104                     Vector xAxisVector = Vector.XAxis;
105                     Vector newxAxisVector0 = handTransform.TransformPoint(xAxisVector);
106                     newxAxisVector = new Vector2(newxAxisVector0[0], newxAxisVector0[2]);
107                     newxAxisVector.Normalize();
108                     correct.Normalize();
109                     anguloD = (Vector2.Angle(newxAxisVector, correct) - 123)*yaw; //Ya esta
    en grados, no en radianes 60 Math.Abs(
110                 }
111             }
112         }
113         timer = 0.0f;
114
115
116         Horizontal += yaw * Speed * Time.deltaTime;
117         Horizontal = Mathf.Clamp(Horizontal, MaxLeft, MaxRight);
118
119         Vertical += pitch * Speed * Time.deltaTime;
120         Vertical = Mathf.Clamp(Vertical, MaxDown, MaxUp);
121
122         RotationH += yaw * SpeedR * Time.deltaTime;
123         RotationV += pitch * SpeedR * Time.deltaTime;
124
125         if (anguloD < 2 && anguloD > -2)
126         {
```

```
127             Quaternion target = Quaternion.Euler(0, 180, 0);
128             transform.rotation = Quaternion.Slerp(transform.rotation, target, Time.deltaTime
    );
129         }
130 
131         if (anguloF < 2 && anguloF > -2)
132         {
133             Quaternion target = Quaternion.Euler(0, 180, 0);
134             transform.rotation = Quaternion.Slerp(transform.rotation, target, Time.deltaTime)
    ;
135         }
136 
137 
138         tr.position = new Vector3(Horizontal, Vertical, 0);
139         tr.Rotate(Vector3.forward * yaw * SpeedR * Time.deltaTime* 8);
140         tr.Rotate(Vector3.right * pitch * SpeedR * Time.deltaTime* 10);
141     }
142 }
```

Anexo B: código generador de nivel con DDA

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


public class Generador : MonoBehaviour
{
    public Movimiento Script;
    public Score score;
    public double fA, dA, r1, a1, b1, c1, d1, r2, a2, b2, c2, d2, z1, z2, z3, z4;
    public Pool ItemsPool;
    public Pool SafeItemsPool;

    Queue<Transform> elements;/*Como una list apero solo puede tomar por los extremos, lo mas cerca y lo m[as lejos que el jugador puede ver*/
    public float co=0.0f,timeInGame=0.0f;
    public int Quantity = 25, SafeQuantity = 3,timI;
    public Vector3 direction;
    public Vector3 offset;
    public float Displace = 15f;
    public float Speed = 6;
    private float speed;
    public float Increment;
    public double deficienteF, mediaF, buenaF, deficienteDR, mediaDR, buenaDR,deficienteDC, mediaDC, buenaDC, deficienteE, mediaE, buenaE;
    public GameObject hueso, huesoxn, huesoxp, huesoyn, huesoyp, huesoPrueba;
    public Text pointsText;


    Transform tr;
    Vector3 originalPos;
    int moved = 0;
    float currentDisplace;
    public double areaT, centT, areaTS1, centTS1, areaTS2, centTS2, areaT2, centT2, areaTS3, centTS3, areaTS4, centTS4, areaT3, centT3, areaTS5, centTS5, areaTS6, centTS6, areaT4, centT4, areaTS7, centTS7, areaTS8, centTS8;

    private void Awake()
    {

        tr = transform;
        ItemsPool.Initialize();
        SafeItemsPool.Initialize();
        originalPos = tr.position;
        elements = new Queue<Transform>();
        //Debug.Log("Screen Width : " + Screen.width);
        //Generate();
    }

    public void Clean()
    {
        while (elements.Any())
        {
            elements.Dequeue().gameObject.SetActive(false); /*Descativa objetos del generador con el metodo clean*/
        }
    }


    public void Generate()
    {
        tr.position = originalPos;
```

```
62          speed = Speed;
63          currentDisplace = 0;
64          moved = 0;
65          timI = 0;
66          timeInGame = 0.0f;
67          score.Cantidad = 0;
68
69          for (int i = 0; i < Quantity; i++)
70          {
71              var elemntTransform = i < SafeQuantity ? SafeItemsPool.GetRandom() : ItemsPool.
    GetRandom(); /*condicional safe */
72              elemntTransform.position = offset - direction * Displace * i;
73              elemntTransform.gameObject.SetActive(true);
74              elements.Enqueue(elemntTransform);
75          }
76      }
77
78      private void Update()
79      {
80          //Se obtienen angulos desde el otro script
81
82          fA = Script.anguloF;
83          dA = Script.anguloD;
84
85          timeInGame+= Time.deltaTime;
86          timI = (int) Math.Round(timeInGame);
87
88          Vector3 tmpPos = Camera.main.WorldToScreenPoint(transform.position);
89
90
91          if (fA > 0)
92          {
93              //EXTENSION Y+
94              deficienteE = MembershipDegreeSemI(60, 64, 58, fA);
95              //Debug.Log("Deficiente: " + deficienteE);
96
97              areaTS7 = areaS1(437, 466.13, 582.65, deficienteE);
98              centTS7 = (582.65 + 437) / 2;
99
100             mediaE = MembershipDegreeComp(60, 64, 66, 70, fA);
101             //Debug.Log("Media: " + mediaE);
102
103             areaT4 = area(466.13, 582.65, 728.35, 844.87, mediaE);
104             centT4 = (844.87 + 466.13) / 2;
105
106             buenaE = MembershipDegreeSem2(66, 70, 72, fA);
107             //Debug.Log("Buena: " + buenaE);
108
109             areaTS8 = areaS1(874, 844.87, 728.35, buenaF);
110             centTS8 = (728.35 + 874) / 2;
111
112             z4 = ((areaTS7 * centTS7) + (areaT4 * centT4) + (areaTS8 * centTS8)) / (areaTS7
    + areaT4 + areaTS8);
113             //Debug.Log("EXTENSION " + z4);
114         }
115
116         if (fA < 0)
117         {
118             //FLEXION Y-
119             deficienteF = MembershipDegreeSemI(60, 68, 56, -fA);
120
121             //SemiTrap1
122             areaTS1 = areaS1(0, 36.416, 182.08, deficienteF);
123             centTS1 = (0 + 182.08) / 2;
124
125             mediaF = MembershipDegreeComp(60, 68, 72, 80, -fA);
126
127             areaT = area(36.416, 182.08, 255, 400.6, mediaF);
```

```
128             centT = (400.6 + 36.416) / 2;
129
130             buenaF = MembershipDegreeSem2(72, 80, 84, -fA);
131
132             //SemiTrap2
133             areaTS2 = areaS1(437, 400.6, 255, buenaF);
134             centTS2 = (437 + 255) / 2;
135
136             z1 = ((areaTS1 * centTS1) + (areaT * centT) + (areaTS2 * centTS2)) / (areaTS1 +
    areaT + areaTS2);
137
138         }
139
140         if (dA > 0)
141         {
142             //DESVIACION RADIAL X+
143             deficienteDR = MembershipDegreeSemI(15, 19, 13, dA);
144
145             areaTS3 = areaS1(581, 629.416, 823.08, deficienteDR);
146             centTS3 = (823.08 + 581) / 2;
147
148             mediaDR = MembershipDegreeComp(15, 19, 21, 25, dA);
149
150
151             areaT2 = area(629.416, 823.08, 919.1, 1113.6, mediaDR);
152             centT2 = (1113.6 + 629.416) / 2;
153
154             buenaDR = MembershipDegreeSem2(21, 25, 27, dA);
155
156
157             areaTS4 = areaS1(1162, 1113.6, 919.916, buenaDR);
158             centTS4 = (1162 + 919.916) / 2;
159
160             z2 = ((areaTS3 * centTS3) + (areaT2 * centT2) + (areaTS4 * centTS4)) / (areaTS3
    + areaT2 + areaTS4);
161
162         }
163
164         if (dA < 0)
165         {
166             //DESVIACION CUBITAL X-
167             deficienteDC = MembershipDegreeSemI(28, 32, 26, -dA);
168
169             areaTS5 = areaS1(0, 48.16, 242.08, deficienteDC);
170             centTS5 = (242.08 + 0) / 2;
171
172             mediaDC = MembershipDegreeComp(28, 32, 34, 38, -dA);
173
174             areaT3 = area(48.416, 242.08, 339, 532.6, mediaDC);
175             centT3 = (532.6 + 48.416) / 2;
176
177             buenaDC = MembershipDegreeSem2(34, 38, 40, -dA);
178
179             areaTS6 = areaS1(581, 532.6, 339, buenaDC);
180             centTS6 = (581 + 339) / 2;
181
182             z3 = ((areaTS5 * centTS5) + (areaT3 * centT3) + (areaTS6 * centTS6)) / (areaTS1
    + areaT3 + areaTS6);
183         }
184         // Modificacion prefab para gregar collider
185
186         speed += Increment;
187         tr.position += direction * speed * Time.deltaTime;
188         currentDisplace = Mathf.Abs(Vector3.Distance(tr.position, originalPos));
189
190         var timesToInfinite = currentDisplace / Displace;/*Podemos exceder las secciones del
     escenario y nos saldr[iamos cuantas celdas debemos desnciolar */
191
```

```
192         if (timesToInfinite > moved + 2)
193                 {
194                     ToInfinite();
195                 }
196         if (huesoxn != null)
197         {
198             huesoxn.transform.Translate(Vector3.right * 15 * Time.deltaTime);
199         }
200         if (huesoxp != null)
201         {
202             huesoxp.transform.Translate(Vector3.right * 15 * Time.deltaTime);
203         }
204         if (huesoyn != null)
205         {
206             huesoyn.transform.Translate(Vector3.right * 15 * Time.deltaTime);
207         }
208         if (huesoyp != null)
209         {
210             huesoyp.transform.Translate(Vector3.right * 15 * Time.deltaTime);
211         }
212             co += Time.deltaTime;
213             //Debug.Log("C:" + co);
214 
215 
216     }
217     public void ToInfinite()
218     {
219         var last = elements.LastOrDefault(); /*La ultima posicion donde debo poner el sig
    elemento */
220         var tel = elements.Dequeue();
221         tel.gameObject.SetActive(false);
222         double v1, v2, v3, v4;
223         float f1,f2,f3,f4;
224 
225         var elemntTransform = ItemsPool.GetRandom();
226         elemntTransform.position = last.position - direction * Displace;
227         elemntTransform.gameObject.SetActive(true);
228         elements.Enqueue(elemntTransform);
229 
230         //Para instanciar en la posici[on correcta se aplica la formula
231 
232          v1 = (z1 * 5f) / 437f; //Y-
233          v2 = ((z2 * 20f) / 1162f)-10f; //X+ el -10 es para evitar divsion entre 0
234          v3 = ((z3 * 10f) / 581f)-10f; //X-
235          v4 = (z4 * 10f) / 874f; //Y+
236 
237         f1 = (float)v1;
238         f2 = (float)v2;
239         f3 = (float)v3;
240         f4 = (float)v4;
241 
242 
243         if (co >= 22 || co==0)
244         {
245             Debug.Log("Generando huesos");
246             huesoyn = Instantiate(hueso, new Vector3((f3-0.63f), f1, 220 + (1 * 5)),
    Quaternion.identity) as GameObject;
247             huesoyn.transform.Rotate(0, 90.0f, 0, Space.Self);
248 
249             huesoxp = Instantiate(hueso, new Vector3((f2 - 0.63f), f4, 220 + (2 * 5)),
    Quaternion.identity) as GameObject;
250             huesoxp.transform.Rotate(0, 90.0f, 0, Space.Self);
251 
252             huesoxn = Instantiate(hueso, new Vector3((f2 - 0.63f), f1, 220 + (3 * 5)),
    Quaternion.identity) as GameObject;
253             huesoxn.transform.Rotate(0, 90.0f, 0, Space.Self);
254 
```

```
255             huesoyp = Instantiate(hueso, new Vector3((f3 - 0.63f), f4, 220 + (4 * 5)),
    Quaternion.identity) as GameObject;
256             huesoyp.transform.Rotate(0, 90.0f, 0, Space.Self);
257             co = 1;
258         }
259         moved++;
260     }
261     public static double MembershipDegreeComp(double a, double b, double c, double d, double
     angle)
262     {
263         double r;
264 
265         if (angle < a)
266         {
267             r = 0;
268             return r;
269         }
270         if (a <= angle && angle < b)
271         {
272             r = (angle - a) / (b - a);
273             return r;
274         }
275         if (b <= angle && angle < c)
276         {
277             r = 1;
278             return r;
279         }
280         if (c <= angle && angle < d)
281         {
282             r = (d - angle) / (d - c);
283             return r;
284         }
285         if (d <= angle)
286         {
287             r = 0;
288             return r;
289         }
290         return r = 2;
291     }
292     public static double MembershipDegreeSemI(double a, double b,double c, double angle)
293     {
294         double r;
295         if (angle <= a)
296         {
297             r = 0;
298             return r;
299         }
300         if (a <= angle && angle < b)
301         {
302             r = (a - angle) / (b - a);
303             return r;
304         }
305         if (b <= angle)
306         {
307             r = 1;
308             return r;
309         }
310         return r = 2;
311     }
312     public static double MembershipDegreeSem2(double a, double b, double c, double angle)
313     {
314         double r;
315         if (angle <= a)
316         {
317             r = 1;
318             return r;
319         }
320         if (a <= angle && angle < b)
```

```
321         {
322             r = (angle - a) / (b - a);
323             return r;
324         }
325         if (b <= angle)
326         {
327             r = 0;
328             return r;
329         }
330         return r = 2;
331     }
332     public static double area(double a, double b, double c, double d,double md)
333     {
334         double areaT,bx;
335         bx = ( (d-a)+(c-b) ) * (1 - md);
336         areaT = ((d - a) + (bx))*(md) / 2;
337         return areaT;
338     }
339 
340 
341 
342     public static double areaS1(double a, double b, double c, double md)
343     {
344         double areaTS1,bx;
345         bx = ((c - (a-c)) + (b - (a-b))) * (1 - md);
346         areaTS1 = ((c - (a-c)) + (bx))*(md) / 2;
347         return areaTS1;
348     }
349 
350 
351 }
```

Anexo C: código generador de nivel sin DDA

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


public class Gen2 : MonoBehaviour
{
    public Pool ItemsPoolN;
    public Pool SafeItemsPoolN;
    public Score score;

    Queue<Transform> elementsN;/*Como una list apero solo puede tomar por los extremos, lo mas cerca y lo m[as lejos que el jugador puede ver*/
    public int QuantityN = 25;
    public int SafeQuantityN = 3, timI;
    public Vector3 directionN;
    public Vector3 offsetN;
    public float DisplaceN = 10f;
    public float SpeedN = 6, timeInGame=0.0f;
    private float speedN;
    public float IncrementN;

    Transform tr;
    Vector3 orginialPosN;
    int moved2 = 0;
    float currentDisplaceN;


    private void Awake()
    {
        tr = transform;
        ItemsPoolN.Initialize();
        SafeItemsPoolN.Initialize();
        orginialPosN = tr.position;
        elementsN = new Queue<Transform>();
    }

    public void Clean2()
    {
        while (elementsN.Any())
        {
            elementsN.Dequeue().gameObject.SetActive(false); /*Descativa objetos del generador con el metodo clean*/
        }
    }


    public void Generate2()
    {
        tr.position = orginialPosN;
        speedN = SpeedN;
        currentDisplaceN = 0;
        moved2 = 0;
        timI = 0;
        timeInGame = 0.0f;
        score.Cantidad = 0;

        for (int i = 0; i < QuantityN; i++)
        {
            var elemntTransform = i < SafeQuantityN ? SafeItemsPoolN.GetRandom() : ItemsPoolN.GetRandom(); /*condicional safe */
            elemntTransform.position = offsetN - directionN * DisplaceN * i;
```

```
64              elemntTransform.gameObject.SetActive(true);
65              elementsN.Enqueue(elemntTransform);
66          }
67      }
68
69      private void Update()
70      {
71          timeInGame += Time.deltaTime;
72          timI = (int)Math.Round(timeInGame);
73          Vector3 tmpPosN = Camera.main.WorldToScreenPoint(transform.position);
74          speedN += IncrementN;
75          tr.position += (directionN * speedN * Time.deltaTime);
76          currentDisplaceN = Mathf.Abs(Vector3.Distance(tr.position, orginialPosN)); //Mathf.
    Abs()
77          var timesToInfinite2 = currentDisplaceN / DisplaceN;/*POdemos exceder las secciones
    del escenario y nos saldr[iamos cuantas celdas debemos desnciolar */
78
79          if (timesToInfinite2 > moved2 + 2)
80                  {
81                     ToInfinite2();
82                   }
83
84
85
86      }
87      public void ToInfinite2()
88      {
89          var last = elementsN.LastOrDefault(); /*La ultima posicion donde debo poner el sig
    elemento */
90          var tel = elementsN.Dequeue();
91          tel.gameObject.SetActive(false);
92
93          var elemntTransform = ItemsPoolN.GetRandom();
94          elemntTransform.position = last.position - directionN * DisplaceN;
95          elemntTransform.gameObject.SetActive(true);
96          elementsN.Enqueue(elemntTransform);
97
98
99          moved2++;
100     }
101 }
```

Anexo D: cuadro resultados del modo 1 con su escala

	Pierdo la noción del tiempo	Las cosas parecen suceder automáticamente	Me siento diferente	Siento miedo	El juego se siente real	Si alguien me habla no lo escucho	Me enfado	El tiempo parece detenerse o pararse	Me siento distanciado	No contesto cuando alguien me habla	No puedo decir que me estoy cansando	Jugar parece automático	Mis pensamientos van rápido	Pierdo la noción de dónde estoy	Juego sin pensar en como jugar	Jugar me tranquiliza	Juego más dédo que pretendía	Realmente me meto en el juego	Siento que no puedo dejar de jugar	
Y	0.8	2.2	2.6	0	2.6	3.2	0	0	2.4	2	0	1.1	2.2	0	0	0	2	-0.6	1.4	
N	-4.8	-2.1	-1.2	3.6	0	-3.2	-5.7	-1.8	-2.4	-3.6	-2.8	-4.2	-2.1	0.6	-4.2	0	-2.2	0	-3.6	
M	-0.8	-0.5	1	0	-0.6	0	-0.3	1	0	0	0.4	-0.5	-0.5	1.8	-1	-2	-0.6	-1.8	-0.2	
SUM	-1.2	-0.1	0.6	0.9	0.5	0	-1.5	-0.2	-2.22045E-16	-0.4	-0.6	-0.9	-0.1	0.6	-1.3	-0.5	-0.2	-0.6	-0.6	**-0.3294117765**

Anexo E: cuadro resultados del modo 2 con su escala

	Pierdo la noción del tiempo	Las cosas parecen suceder autonáticamente	Me siento diferente	Siento miedo	El juego se siente real	Si alguien me habla no lo escucho	Me enfado	El tiempo parece detenerse o pararse	Me siento distanciado	No contesto cuando alguien me habla	No puedo decir que me estoy cansando	Jugar parece automático	Mis pensamientos van rápido	Pierdo la noción de dónde estoy	Juego sin pensar en como jugar	Jugar me tranquiliza	Juego más de lo que pretendía	Realmente me meto en el juego	Siento que no puedo dejar de jugar	
Y	0.8	0	2.6	0	2.6	0	1.3	0	0	2	1.8	1.1	1.1	0	1.1	0	2	-0.6	1.4	
N	-2.4	-4.2	-1.8	3.6	0	-6.4	-3.8	-1.8	-3.2	-3.6	-2.8	-4.2	0	0.6	-6.3	0	-2.2	0	-3.6	
M	-1.6	-1	0	0	-0.6	0	-0.3	1	0	0	0.2	-0.5	-1.5	1.8	0	-2	-0.6	-1.8	-0.2	
SUM	-0.8	-1.3	0.2	0.9	0.5	-1.6	-0.7	-0.2	-0.8	-0.4	-0.2	-0.9	-0.1	0.6	-1.3	-0.5	-0.2	-0.6	-0.6	**-0.4705882353**

Referencias

[Andrade et al., 2014] Andrade, K. d. O., Fernandes, G., Caurin, G. A., Siqueira, A. A., Romero, R. A., and Pereira, R. d. L. (2014). Dynamic player modelling in serious games applied to rehabilitation robotics. In *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol*, pages 211–216.

[Andrade et al., 2016] Andrade, K. d. O., Pasqual, T. B., Caurin, G. A. P., and Crocomo, M. K. (2016). Dynamic difficulty adjustment with evolutionary algorithm in games for rehabilitation robotics. In *2016 IEEE International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8.

[Bouchard et al., 2012] Bouchard, B., Imbeault, F., Bouzouane, A., and Menelas, B.-A. J. (2012). Developing serious games specifically adapted to people suffering from alzheimer. In *Serious Games Development and Applications: Third International Conference, SGDA 2012, Bremen, Germany, September 26-29, 2012. Proceedings 3*, pages 243–254. Springer.

[Brockmyer et al., 2009] Brockmyer, J. H., Fox, C. M., Curtiss, K. A., McBroom, E., Burkhart, K. M., and Pidruzny, J. N. (2009). The development of the game engagement questionnaire: A measure of engagement in video game-playing. *Journal of experimental social psychology*, 45(4):624–634.

[Cabonell-Sánchez, 2020] Cabonell-Sánchez, X. (2020). La industria de los videojuegos contra el trastorno de juego por internet: el partido del siglo. *Aloma: revista de psicologia, ciències de l'educació i de l'esport Blanquerna*, 38(1):39–48.

[Choi et al., 2011] Choi, Y., Gordon, J., Park, H., and Schweighofer, N. (2011). Feasibility of the adaptive and automatic presentation of tasks (adapt) system for rehabilitation of upper extremity function post-stroke. *Journal of neuroengineering and rehabilitation*, 8(1):1–12.

[Costa et al., 2017] Costa, J., Neto, J., Alves, R., Escudeiro, P., and Escudeiro, N. (2017). Neurocognitive stimulation game: Serious game for neurocognitive stimulation and assessment. In *Serious Games, Interaction and Simulation: 6th International Conference, SGAMES 2016, Porto, Portugal, June 16-17, 2016, Revised Selected Papers 6*, pages 74–81. Springer.

[Csikszentmihalyi and Csikzentmihaly, 1990] Csikszentmihalyi, M. and Csikzentmihaly, M. (1990). *Flow: The psychology of optimal experience*, volume 1990. Harper & Row New York.

[da Cruz et al., 2020] da Cruz, L. C., Sierra-Franco, C. A., Silva-Calpa, G. F. M., and Raposo, A. B. (2020). A self-adaptive serious game for eye-hand coordination training. In *International Conference on Human-Computer Interaction*, pages 385–397. Springer.

[Di Loreto et al., 2012] Di Loreto, I., Gouaïch, A., and Hocine, N. (2012). Mixed reality serious games for post-stroke rehabilitation. In *Therapeutic Serious Games and Pervasive Computing*.

[El-Habr et al., 2019] El-Habr, C., Garcia, X., Paliyawan, P., and Thawonmas, R. (2019). Runner: A 2d platform game for physical health promotion. *SoftwareX*, 10:100329.

[Gouaïch et al., 2012] Gouaïch, A., Hocine, N., Van Dokkum, L., and Mottet, D. (2012). Digital-pheromone based difficulty adaptation in post-stroke therapeutic games. In *Proceedings of the 2nd ACM SIGHIT International Health Informatics Symposium*, IHI '12, page 5–12, New York, NY, USA. Association for Computing Machinery.

[Grubišić et al., 2015] Grubišić, I., Skala Kavanagh, H., and Grazio, S. (2015). Novel approaches in hand rehabilitation. *Periodicum biologorum*, 117(1):139–145.

[Hocine and Gouaïch, 2011] Hocine, N. and Gouaïch, A. (2011). Therapeutic games' difficulty adaptation: An approach based on player's ability and motivation. In *2011 16th International Conference on Computer Games (CGAMES)*, pages 257–261. IEEE.

[Hocine et al., 2014] Hocine, N., Gouaich, A., and Cerri, S. A. (2014). Dynamic difficulty adaptation in serious games for motor rehabilitation. In *International Conference on Serious Games*, pages 115–128. Springer.

[Hocine et al., 2015] Hocine, N., Gouaïch, A., Cerri, S. A., Mottet, D., Froger, J., and Laffont, I. (2015). Adaptation in serious games for upper-limb rehabilitation: an approach to improve training outcomes. *User Modeling and User-Adapted Interaction*, 25(1):65–98.

[Hocine and Gouaïch, 2011] Hocine, N. and Gouaïch, A. (2011). Therapeutic games' difficulty adaptation: An approach based on player's ability and motivation. In *2011 16th International Conference on Computer Games (CGAMES)*, pages 257–261.

[Hocine et al., 2011] Hocine, N., Gouaïch, A., Di Loreto, I., and Joab, M. (2011). Motivation based difficulty adaptation for therapeutic games. In *2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8.

[Huygelier et al., 2017] Huygelier, H., van Ee, R., Gillebert, C., and Vanden Abeele, V. (2017). Developing adaptive mental health games using player-patient modeling. In *CHI PLay 2017 Workshop Games for the Assessment and Treatment of Mental Health, Date: 2017/10/15-2017/10/15, Location: Amsterdam, The Netherlands*.

[Imbeault et al., 2011] Imbeault, F., Bouchard, B., and Bouzouane, A. (2011). Serious games in cognitive training for alzheimer's patients. In *2011 IEEE 1st International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8. IEEE.

[Ines et al., 2011] Ines, D. L., Abdelkader, G., and Hocine, N. (2011). Mixed reality serious games for post-stroke rehabilitation. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, pages 530–537.

[Irving Herrera Luna, 2019] Irving Herrera Luna, E. J. R. R. (2019). Fusión de características geométricas y señales electromiográficas para la identificación off-line de movimientos de muñeca utilizados en rehabilitación.

[Knobel et al., 2021] Knobel, S. E. J., Kaufmann, B. C., Gerber, S. M., Urwyler, P., Cazzoli, D., Müri, R. M., Nef, T., and Nyffeler, T. (2021). Development of a search task using immersive virtual reality: Proof-of-concept study. *JMIR Serious Games*, 9(3):e29182.

[Koster, 2014] Koster, R. (2014). A theory of fun for game design. a theory of fun for game design.

[Lazaridis et al., 2021] Lazaridis, L., Papatsimouli, M., Kollias, K.-F., Sarigiannidis, P., and Fragulis, G. F. (2021). Hitboxes: A survey about collision detection in video games. In *International Conference on Human-Computer Interaction*, pages 314–326. Springer.

[Lohse et al., 2013] Lohse, K., Shirzad, N., Verster, A., Hodges, N., and Van der Loos, H. M. (2013). Video games and rehabilitation: using design principles to enhance engagement in physical therapy. *Journal of Neurologic Physical Therapy*, 37(4):166–175.

[Nagle et al., 2015] Nagle, A., Riener, R., and Wolf, P. (2015). High user control in game design elements increases compliance and in-game performance in a memory training game. *Frontiers in psychology*, 6:1774.

[Norkin and White, 2016] Norkin, C. C. and White, D. J. (2016). *Measurement of joint motion: a guide to goniometry*. FA Davis.

[Ozkul et al., 2019] Ozkul, F., Palaska, Y., Masazade, E., and Erol-Barkana, D. (2019). Exploring dynamic difficulty adjustment mechanism for rehabilitation tasks using physiological measures and subjective ratings. *IET Signal Processing*, 13(3):378–386.

[Page et al., 2021] Page, M. J., Moher, D., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., et al. (2021). Prisma 2020 explanation and elaboration: updated guidance and exemplars for reporting systematic reviews. *bmj*, 372.

[Parsons and Reinebold, 2012] Parsons, T. D. and Reinebold, J. L. (2012). Adaptive virtual environments for neuropsychological assessment in serious games. *IEEE Transactions on Consumer Electronics*, 58(2):197–204.

[Pezzera et al., 2019] Pezzera, M., Tironi, A., Essenziale, J., Mainetti, R., and Borghese, N. A. (2019). Approaches for increasing patient's engagement and motivation in exer-games-based autonomous telerehabilitation. In *2019 IEEE 7th international conference on serious games and applications for health (SeGAH)*, pages 1–8. IEEE.

[Pinto et al., 2018] Pinto, J. F., Carvalho, H. R., Chambel, G. R. R., Ramiro, J., and Goncalves, A. (2018). Adaptive gameplay and difficulty adjustment in a gamified upper-limb rehabilitation. In *2018 IEEE 6th International Conference on Serious Games and Applications for Health (SeGAH)*, pages 1–8.

[Ponce-Cruz et al., 2016] Ponce-Cruz, P., Molina, A., and MacCleery, B. (2016). *Fuzzy Logic Type 1 and Type 2 Based on LabVIEW™ FPGA*. Springer.

[Ross, 2005] Ross, T. J. (2005). *Fuzzy logic with engineering applications*. John Wiley & Sons.

[Sekhavat, 2017] Sekhavat, Y. A. (2017). Mprl: Multiple-periodic reinforcement learning for difficulty adjustment in rehabilitation games. In *2017 IEEE 5th international conference on serious games and applications for health (SeGAH)*, pages 1–7. IEEE.

[Siegel and Smeddinck, 2012] Siegel, S. and Smeddinck, J. (2012). Adaptive difficulty with dynamic range of motion adjustments in exergames for parkinson's disease patients. In *International Conference on Entertainment Computing*, pages 429–432. Springer.

[Smeddinck et al., 2013] Smeddinck, J. D., Siegel, S., and Herrlich, M. (2013). Adaptive difficulty in exergames for parkinson's disease patients. In *Graphics Interface*, pages 141–148.

[Tageldeen et al., 2017] Tageldeen, M. K., Elamvazuthi, I., Perumal, N., and Ganesan, T. (2017). A virtual reality based serious games for rehabilitation of arm. In *2017 IEEE 3rd International Symposium in Robotics and Manufacturing Automation (ROMA)*, pages 1–6. IEEE.

[Törnquist, 2022] Törnquist, E. (2022). Design and implementation of an endless runner as an exergame.

[Unity, 2022] Unity (2022). Unity.

[Valencia Lemos et al., 2019] Valencia Lemos, Y. M., Majin Erazo, J. J., Guzmán Villamarín, D. E., and Londoño Prieto, J. (2019). Dynamic difficulty adjustment in virtual reality applications for upper limb rehabilitation. *Ingeniería y Desarrollo*, 37(2):173–191.

[Verhulst et al., 2015] Verhulst, A., Yamaguchi, T., and Richard, P. (2015). Physiological-based dynamic difficulty adaptation in a theragame for children with cerebral palsy. In *PhyCS*, pages 164–171.

[Wang and Tseng, 2013] Wang, J.-Y. and Tseng, Y.-R. (2013). Dynamic difficulty adjustment by fuzzy rules using in a neural network controlled game. In *2013 Ninth International Conference on Natural Computation (ICNC)*, pages 277–281. IEEE.

[Weichert et al., 2013] Weichert, F., Bachmann, D., Rudak, B., and Fisseler, D. (2013). Analysis of the accuracy and robustness of the leap motion controller. *Sensors*, 13(5):6380–6393.

[Zárate-Ramírez and Espinosa-Gutiérrez, 2013] Zárate-Ramírez, J. and Espinosa-Gutiérrez, A. (2013). ¿ cuánto valen las lesiones de la mano? *Acta Ortopédica Mexicana*, 27(5):345–349.

[Zheng et al., 1998] Zheng, J., Chan, K., and Gibson, I. (1998). Virtual reality. *Ieee Potentials*, 17(2):20–23.

[Zohaib, 2018] Zohaib, M. (2018). Dynamic difficulty adjustment (dda) in computer games: A review. *Advances in Human-Computer Interaction*, 2018.

[Zyda, 2005] Zyda, M. (2005). From visual simulation to virtual reality to games. *Computer*, 38(9):25–32.