UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE CUENCA

CARRERA DE INGENIERÍA ELECTRÓNICA

TRABAJO DE TITULACIÓN PREVIO
A LA OBTENCIÓN DEL TÍTULO DE
INGENIERO ELECTRÓNICO

PROYECTO TECNICO:

"DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN DISPOSITIVO INALÁMBRICO PARA ESTIMULAR EJERCICIOS DE REHABILITACIÓN EN PACIENTES CON PARKINSON"

AUTOR:

FABIOLA JESCENIA CHACHA CAJAMARCA

TUTOR:

ING. JUAN PABLO BERMEO

CUENCA – ECUADOR

2017

CESION DE DERECHOS DE AUTOR

Yo, Fabiola Jescenia Chacha Cajamarca con documento de identificación N° 0301672903, manifiesto mi voluntad y cedo a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que soy autora del trabajo de titulación: **"DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN DISPOSITIVO INALÁMBRICO PARA ESTIMULAR EJERCICIOS DE REHABILITACIÓN EN PACIENTES CON PARKINSON"**, mismo que ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en mi condición de autor me reservo los derechos morales de la obra antes citada. En concordancia, suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

Cuenca, noviembre del 2017

Fabiola Jescenia Chacha Cajamarca

CI:0301672903

AUTOR

CERTIFICACIÓN

Yo, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN DISPOSITIVO INALÁMBRICO PARA ESTIMULAR EJERCICIOS DE REHABILITACIÓN EN PACIENTES CON PARKINSON"**, realizado por Fabiola Jescenia Chacha Cajamarca, obteniendo el **Proyecto Técnico con enfoque investigativo** que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana.

Cuenca, noviembre del 2017

Ing. Juan Pablo Bermeo Moyano

CI: 0102645884

TUTOR DEL TRABAJO DE TITULACIÓN

DECLARATORIA DE RESPONSABILIDAD

Yo, Fabiola Jescenia Chacha Cajamarca con número de cédula CI.0301672903, autora del trabajo de titulación **"DISEÑO, DESARROLLO E IMPLEMENTACIÓN DE UN DISPOSITIVO INALÁMBRICO PARA ESTIMULAR EJERCICIOS DE REHABILITACIÓN EN PACIENTES CON PARKINSON"** certifico que el total contenido del **Proyecto Técnico con enfoque investigativo** es de mi exclusiva responsabilidad y autoría

Cuenca, noviembre del 2017

Fabiola Jescenia Chacha Cajamarca

CI:0301672903

AUTOR

AGRADECIMIENTOS

Agradezco primeramente a mi amado Jesús, por siempre estar a mi lado y ser mi luz, por darme fuerza, amor, sabiduría, salud y la oportunidad de hacer este sueño realidad.

A mi papi Oswaldo y mi mami Lupe, por su amor, su esfuerzo y todo el apoyo incondicional que me brindan día a día.

A mi esposo Carlos, por el apoyo brindado durante la realización de este proyecto.

A mis abuelitos Papito, Mamita y Mama, mis hermanos Freddy, Gaby y Thalía, mis tías queridas Mamita Diana, Ñaña Ana, Tia Rosita, Tia Karina, mis tíos y mis primos, porque siempre me demuestran su cariño y han sabido estar presente en todos los momentos de mi vida.

Un agradecimiento especial a mi tutor de tesis, Ing. Juan Pablo Bermeo y a mi cotutora Dra. Mónica Karel Huerta, que con su paciencia, guía y orientación contribuyeron en el desarrollo y culminación de este proyecto.

Fabiola Jescenia Chacha Cajamarca

DEDICATORIAS

A mis amados padres Oswaldo y Lupe que siempre están a mi lado, por todo el esfuerzo y sacrificio realizado para cumplir mi meta, los amo y les dedico este trabajo.
A mi niña Natasha, el amor de mi vida, mi princesa.

Fabiola Jescenia Chacha Cajamarca

ÍNDICE GENERAL

AGRADECIMIENTOS I
DEDICATORIAS II
ÍNDICE GENERAL III
ÍNDICE DE FIGURAS VI
ÍNDICE DE TABLAS IX
RESUMEN X
INTRODUCCIÓN XI
ANTECEDENTES DEL PROBLEMA DE ESTUDIO XII
JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES) XIV
OBJETIVOS XVI
OBJETIVO GENERAL XVI
OBJETIVOS ESPECIFICOS XVI
CAPÍTULO 1 : FUNDAMENTACIÓN TEÓRICA 1
1.1 CARACTERISTICAS DE LA ENFERMEDAD DE PARKINSON 1
1.1.1 DIAGNOSTICO 1
1.1.2 TRATAMIENTO 2
1.2 TERAPIA DE REHABILITACION 3
1.2.1 TÉCNICAS DE FISIOTERAPIA PARA EP 3
1.2.2 TERAPIA OCUPACIONAL 6
1.2.3 LOGOPEDIA 6
1.2.4 TÉCNICAS INNOVADORAS DE REHABILITACION 6
1.2.5 EXERGAMING EN REHABILITACION 7
1.3 TECNOLOGIAS PARA EXERGAMING 8
1.3.1 HADWARE Y SOFTWARE LIBRE 8
1.3.2 MOTORES DE VIDEOJUEGOS 8
1.3.3 PLATAFORMA ANDROID 13
1.3.4 ARDUINO 14
1.3.5 BLUETOOTH HC-05 15
1.3.6 ACELERÓMETROS 16
1.4 SISTEMAS DE JUEGOS COMERCIALES 16
1.4.1 NINTENDO WII 17
1.4.2 KINECT 18
1.4.3 TABLETS 22
1.5 INCIDENCIA DE LA ENFERMEDAD DE PARKINSON EN EL AZUAY 23
CAPÍTULO 2 : MARCO METODOLÓGICO 27

2.1 DISEÑO DEL SISTEMA 27
2.2 DISEÑO DEL DISPOSITIVO INALAMBRICO 28
2.2.1 DISEÑO DEL CIRCUITO IMPRESO 29
2.2.2 DISEÑO DE LA ESTRUCTURA PROTECTORA DEL CIRCUITO 30
2.3 ETAPA DE DETECCION Y ADQUISICION DE DATOS 32
2.4 ETAPA DE TRANSMISION DE DATOS 33
2.5 DISEÑO, DESARROLLO Y PROGRAMACION DEL VIDEOJUEGO 34
2.5.1 MINI JUEGO PLANEGAME 34
2.5.2 MINI JUEGO BALLGAME 45
2.5.3 INTERFAZ DE USUARIO 49
2.5.4 GUARDAR DATOS 55
2.6 DIAGRAMA DE FLUJO DEL SISTEMA 59
2.7 DIAGRAMAS DE CLASES 61
2.7.1 DIAGRAMA DE CLASES DE PLANEGAME 61
2.7.2 DIAGRAMA DE CLASES DE BALLGAME 63
2.7.3 DIAGRAMA DE CLASES DE LA RECEPCION DE DATOS 64
2.7.4 DIAGRAMA DE CLASES PARA GUARDAR DATOS 65
2.7.5 DIAGRAMA DE CLASES DE LA INTERFAZ DE USUARIO 66
CAPÍTULO 3 : ANÁLISIS DE RESULTADOS 67
3.1 DATOS OBTENIDOS EN PACIENTES CON EP USANDO UN CELULAR SAMSUNG 68
3.1.1 MINI JUEGO PLANEGAME 68
3.1.2 MINI JUEGO BALLGAME 70
3.2 DATOS OBTENIDOS EN PACIENTES SANOS USANDO UN CELULAR SAMSUNG 71
3.2.1 MINI JUEGO PLANEGAME 71
3.2.2 MINI JUEGO BALLGAME 73
3.3 DATOS OBTENIDOS USANDO UNA TABLET 74
3.4 VALIDACION DEL SISTEMA 79
3.5 COMPARACION CON LA TERAPIA CONVENCIONAL 82
3.5.1 PRUEBAS DE VALIDACIÓN 83
3.6 GRADO DE SATISFACCIÓN DEL SISTEMA PROPUESTO 85
3.7 ANÁLISIS DE RESULTADOS 86
CAPÍTULO 4 : CONCLUSIONES Y RECOMENDACIONES 91
REFERENCIAS BIBLIOGRÁFICAS 93
APÉNDICES 104
APÉNDICE A: DATOS DE LOS PARTICIPANTES SANOS 104

APÉNDICE B: DIMENSIONES DEL PROTECTOR PLÁSTICO PARA EL DISPOSITIVO..... 105
APÉNDICE C: DIMENSIONES DE LA MUÑEQUERA..... 107
APÉNDICE D: PROGRAMACIÓN DEL MINI JUEGO PLANEGAME..... 108
APÉNDICE E: PROGRAMACIÓN DEL MINI JUEGO BALLGAME..... 122
APÉNDICE F: PROGRAMACIÓN DE LA INTERFAZ DE USUARIO 126
APÉNDICE G: PROGRAMACIÓN DE LA RECEPCION DE DATOS..... 136
APENDICE H: PRESUPUESTO……..137

ÍNDICE DE FIGURAS

FIGURA 1.1 NUEVA ESCENA VACÍA, CON LOS OBJETOS 3D PREDETERMINADOS, UNA CÁMARA Y UN DIRECTIONAL LIGHT. 11
FIGURA 1.2 CUATRO DIFERENTES GAMEOBJECTS, UN PERSONAJE ANIMADO, UNA LUZ, UN ÁRBOL Y UNA FUENTE DE AUDIO 11
FIGURA 1.3. EJEMPLO DE UN CHARACTER CONTROLLER EN UNITY 12
FIGURA 1.4. EJEMPLO DE INTERFAZ DE USUARIO EN UNITY 13
FIGURA 1.5. ARDUINO PRO MICRO 14
FIGURA 1.6. MODULO BLUETOOTH 15
FIGURA 1.7. ACELERÓMETRO ADXL335 16
FIGURA 1.8. CONSOLA DE NINTENDO WII 18
FIGURA 1.9. VIDEOJUEGO 20.000 FUGAS PARA LA REHABILITACIÓN USANDO KINECT 19
FIGURA 1.10. ENTORNO GRAFICO DE SISTEMA DE REHABILITACIÓN DOCTORKINECT 20
FIGURA 1.11. VIDEOJUEGO DESARROLLADO EN KINECT PARA LA REHABILITACIÓN 21
FIGURA 1.12. VIDEOJUEGO DESARROLLADO EN KINECT PARA LA REHABILITACIÓN 22
FIGURA 1.13 REPRESENTACIÓN DEL NÚMERO DE PERSONAS CON EP EN RELACIÓN CON LA EDAD 25
FIGURA 2.1 DIAGRAMA DE BLOQUES DEL SISTEMA PROPUESTO 27
FIGURA 2.2. DIAGRAMA DE FUNCIONAMIENTO DEL SISTEMA 28
FIGURA 2.3. DIAGRAMA DE CONEXIÓN DE LOS ELEMENTOS QUE CONFORMAN EL DISPOSITIVO 29
FIGURA 2.4 DIAGRAMA DE PISTAS DE LA PLACA REALIZADA EN PROTEUS 8.6 29
FIGURA 2.5 PLACA IMPRESA PARA EL DISPOSITIVO 30
FIGURA 2.6 DISEÑO DEL PROTECTOR DEL DISPOSITIVO REALIZADO EN 3DMAX 30
FIGURA 2.7 VISTA SUPERIOR DEL PROTECTOR PARA EL DISPOSITIVO REALIZADO EN 3DMAX 31
FIGURA 2.8 COLOCACIÓN DE LA PLACA IMPRESA DENTRO DE LA ESTRUCTURA PROTECTORA 31
FIGURA 2.9 MUÑEQUERA DISEÑADA. A) PARTE POSTERIOR. B) PARTE SUPERIOR 32
FIGURA 2.10 COLOCACIÓN DEL DISPOSITIVO PARA ADQUISICIÓN DE DATOS. 33
FIGURA 2.11 POSICIÓN DEL DISPOSITIVO PARA LA TRANSMISIÒN DE DATOS EN BASE AL MOVIMIENTO DE LOS EJES X, Y DEL ACELERÓMETRO ADXL335 34
FIGURA 2.12 PANTALLA DEL PRIMER NIVEL DEL MINI JUEGO PLANEGAME EN UNITY 36
FIGURA 2.13 PANTALLA DEL SEGUNDO NIVEL DEL MINI JUEGO PLANEGAME EN UNITY 36
FIGURA 2.14 PANTALLA DEL TERCER NIVEL DEL MINI JUEGO PLANEGAME EN UNITY 37
FIGURA 2.15 ESCENA DEL NIVEL UNO DE PLANEGAME EN EL EDITOR DE UNITY 38
FIGURA 2.16 ESCENA DEL NIVEL DOS DE PLANEGAME EN EL EDITOR DE UNITY 38
FIGURA 2.17 ESCENA DEL NIVEL TRES DE PLANEGAME EN EL EDITOR DE UNITY 39
FIGURA 2.18 OBJETIVOS DE PLANEGAME DENTRO DE LA ARQUITECTURA DE JUEGO 40
FIGURA 2.19 COMPONENTES USADOS PARA LOS OBJETIVOS DEL JUEGO. 40
FIGURA 2.20 ELEMENTOS QUE CONFORMAN LAS ESCENAS, EJEMPLO DE LA ESCENA 2 41
FIGURA 2.21 COMPONENTES EN LA VENTANA INSPECTOR DE LA META. 42
FIGURA 2.22 ÍTEMS DENTRO DE LA ESCENA DEL JUEGO DE PLANEGAME 43
FIGURA 2.23 COMPONENTES QUE CONFORMAN EL TARRO DE GASOLINA. 43

FIGURA 2.24 PANTALLA DE JUEGO DEL PRIMER NIVEL DE BALLGAME 46
FIGURA 2.25 COMPONENTES DE LA ESCENA DE BALLGAME DEL PRIMER NIVEL 46
FIGURA 2.26 PANTALLA DE JUEGO DEL NIVEL 2 DE BALLGAME 47
FIGURA 2.27 COMPONENTES DE LA ESCENA DEL NIVEL 2 DE BALLGAME 47
FIGURA 2.28 COMPONENTES DE CADA OBJETIVO EN BALLGAME 48
FIGURA 2.29 COMPONENTES DE LA PELOTA EN BALLGAME 49
FIGURA 2.30 MENÚ DE CONEXIÓN CON EL DISPOSITIVO INALÁMBRICO 50
FIGURA 2.31 MENÚ DE INICIO DE JUEGO 51
FIGURA 2.32 MENÚ DE INICIO DE JUEGO 51
FIGURA 2.33 MENÚ DE LOS PARÁMETROS DE JUEGO INGRESADOS POR EL TERAPEUTA 52
FIGURA 2.34 MENÚ DE SELECCIÓN DE MINI JUEGOS 52
FIGURA 2.35. MENÚ DE RESULTADOS. 53
FIGURA 2.36 MENÚ OPCIONES DEL JUEGO 53
FIGURA 2.37 PANTALLA DE CARGA PARA CADA NIVEL. 54
FIGURA 2.38 MENÚ PAUSA DURANTE EL JUEGO 54
FIGURA 2.39 PANTALLA PARA GUARDAR LA PUNTUACIÓN. 55
FIGURA 2.40 PANEL DE LA TABLA DE PUNTUACIÓN 56
FIGURA 2.41 PANEL DE INICIO DE SESIÓN 57
FIGURA 2.42 PANEL DE REGISTRO. 57
FIGURA 2.43 PANEL PARA ENVIAR CORREO ELECTRÓNICO 58
FIGURA 2.44 PANEL CUANDO EL JUGADOR GANA UN NIVEL 58
FIGURA 2.45 DIAGRAMA DE FLUJO DEL VIDEOJUEGO 60
FIGURA 2.46 DIAGRAMA DE CLASES PARA LOS EVENTOS DEL JUEGO PLANEGAME 62
FIGURA 2.47 DIAGRAMA DE CLASES PARA LOS EVENTOS DEL JUEGO BALLGAME 63
FIGURA 2.48 DIAGRAMA DE CLASES IMPLEMENTADO PARA LA RECEPCION DE DATOS 64
FIGURA 2.49 DIAGRAMA DE CLASES IMPLEMENTADO PARA ALMACENAR LOS DATOS. 65
FIGURA 2.50 DIAGRAMA DE CLASES DE LA INTERFAZ DE USUARIO PARA LOS MINI JUEGOS 66
FIGURA 3.1 PARTICIPANTE CON EP DURANTE LAS PRUEBAS DEL SISTEMA 68
FIGURA 3.2 PARTICIPANTE SANA DURANTE LAS PRUEBAS DEL SISTEMA. 68
FIGURA 3.3 DIAGRAMA DE CAJA Y BIGOTES DE TIEMPO OBTENIDO DURANTE EL MINI JUEGO PLANEGAME EN PACIENTES CON EP, USANDO LA MANO DERECHA E IZQUIERDA 69
FIGURA 3.4 DIAGRAMA DE CAJA Y BIGOTES DE PUNTAJE OBTENIDO DURANTE EL MINI JUEGO PLANEGAME EN PACIENTES CON EP, USANDO LA MANO DERECHA E IZQUIERDA 69
FIGURA 3.5 DIAGRAMA DE CAJA Y BIGOTES DE TIEMPO OBTENIDO DURANTE EL MINI JUEGO BALLGAME EN PACIENTES CON EP, USANDO LA MANO DERECHA E IZQUIERDA 70
FIGURA 3.6 DIAGRAMA DE CAJA Y BIGOTES DE PUNTAJE OBTENIDO DURANTE EL MINI JUEGO BALLGAME EN PACIENTES CON EP, USANDO LA MANO DERECHA E IZQUIERDA 71
FIGURA 3.7 DIAGRAMA DE CAJA Y BIGOTES DE TIEMPO OBTENIDO DURANTE EL MINI JUEGO PLANEGAME EN PACIENTES SANOS, USANDO LA MANO DERECHA E IZQUIERDA 72

FIGURA 3.8 DIAGRAMA DE CAJA Y BIGOTES DE PUNTAJE OBTENIDO DURANTE EL MINI JUEGO PLANEGAME EN PACIENTES SANOS USANDO LA MANO DERECHA E IZQUIERDA 72
FIGURA 3.9 DIAGRAMA DE CAJA Y BIGOTES DE TIEMPO OBTENIDO DURANTE EL MINI JUEGO BALLGAME EN PACIENTES SANOS, USANDO LA MANO DERECHA E IZQUIERDA 73
FIGURA 3.10 DIAGRAMA DE CAJA Y BIGOTES DE PUNTAJE OBTENIDO DURANTE EL MINI JUEGO BALLGAME EN PACIENTES SANOS, USANDO LA MANO DERECHA 74
FIGURA 3.11 RESULTADOS DE TIEMPO OBTENIDO EN PACIENTES CON EP DURANTE EL MINI JUEGO PLANEGAME, CELULAR VS TABLET. 75
FIGURA 3.12 RESULTADOS DE PUNTAJE OBTENIDO EN PACIENTES CON EP DURANTE EL MINI JUEGO PLANEGAME, CELULAR VS TABLET. 75
FIGURA 3.13 RESULTADOS DE TIEMPO OBTENIDO EN PACIENTES CON EP DURANTE EL MINI JUEGO BALLGAME, CELULAR VS TABLET. 76
FIGURA 3.14 RESULTADOS DE PUNTAJE OBTENIDO EN PACIENTES CON EP DURANTE EL MINI JUEGO BALLGAME, CELULAR VS TABLET. 76
FIGURA 3.15 RESULTADOS DE TIEMPO OBTENIDO EN PACIENTES SANOS DURANTE EL MINI JUEGO PLANEGAME, CELULAR VS TABLET. 77
FIGURA 3.16 RESULTADOS DE PUNTAJE OBTENIDO EN PACIENTES SANOS DURANTE EL MINI JUEGO PLANEGAME, CELULAR VS TABLET. 77
FIGURA 3.17 RESULTADOS DE TIEMPO OBTENIDO EN PACIENTES SANOS DURANTE EL MINI JUEGO BALLGAME, CELULAR VS TABLET. 78
FIGURA 3.18 RESULTADOS DE PUNTAJE OBTENIDO EN PACIENTES SANOS DURANTE EL MINI JUEGO BALLGAME, CELULAR VS TABLET. 79
FIGURA 3.19 RESPUESTA EN RELACIÓN CON LA PREGUNTA ¿SINTIÓ CANSANCIO AL FINALIZAR LOS MINI JUEGOS? 80
FIGURA 3.20 RESPUESTA EN RELACIÓN CON LA PREGUNTA ¿LE FUE FÁCIL JUGAR? 80
FIGURA 3.21 RESPUESTA EN RELACIÓN CON LA PREGUNTA ¿SE DIVIRTIÓ CON LOS MINI JUEGOS? 81
FIGURA 3.22 RESPUESTA EN RELACIÓN CON LA PREGUNTA ¿CUÁL MINI JUEGO LE AGRADO MÁS? 82
FIGURA 3.23 MODELO DE PRUEBA DE MOTRICIDAD REALIZADA AL FINALIZAR LAS TERAPIAS DE REHABILITACIÓN. 82
FIGURA 3.24 RESULTADOS OBTENIDOS DE LA PRUEBA DE MOTRICIDAD APLICADA AL FINALIZAR CADA TERAPIA DE REHABILITACIÓN 84
FIGURA 3.25 MODELO DE ENCUESTA REALIZADA AL FINALIZAR LAS PRUEBAS DE LAS TERAPIAS EN LOS PARTICIPANTES 85
FIGURA 3.26 RESULTADOS OBTENIDOS DE LA PREGUNTA: "QUE TANTO LE AGRADO LA TERAPIA DE REHABILITACIÓN USANDO VIDEOJUEGOS" 85
FIGURA 3.27 RESULTADOS OBTENIDOS DE LA PREGUNTA: "LE GUSTARÍA QUE APLIQUE LA TERAPIA DE REHABILITACIÓN SON EL SISTEMA DE VIDEOJUEGOS EN SU TERAPIA CONVENCIONAL" 86
FIGURA B.1 VISTA FRONTAL DE LA ESTRUCTURA PROTECTORA……105
FIGURA B.2 VISTA LATERAL DERECHA E IZQUIERDA DE LA ESTRUCTURA PROTECTORA……...106
FIGURA B.3 VISTA SUPERIOR DE LA ESTRUCTURA PROTECTORA……107

ÍNDICE DE TABLAS

TABLA 1.1 EJERCICIOS DE MOVILIDAD GENERAL LOS MIEMBROS SUPERIORES E INFERIORES EN PACIENTES CON LA ENFERMEDAD DE PARKINSON..... 5
TABLA 1.2. CARACTERÍSTICAS TÉCNICAS DE ARDUINO PRO MICRO 15
TABLA 1.3. CARACTERÍSTICAS TÉCNICAS DE BLUETOOTH HC-05..... 15
TABLA 1.4. INCIDENCIA DE LA ENFERMEDAD DE PARKINSON EN LA PROVINCIA DEL AZUAY DURANTE EL AÑO 2015..... 24
TABLA 1.5. INCIDENCIA DE LA ENFERMEDAD DE PARKINSON EN LA PROVINCIA DEL AZUAY DURANTE EL AÑO 2016..... 24
TABLA 1.6. COMPARACIÓN DE LOS CASOS DE INCIDENCIA DE EP DURANTE LOS AÑOS 2015-2016..... 26
TABLA 3.1 CARACTERÍSTICAS DE LOS PACIENTES CON EP QUE 67
TABLA 3.2 PREGUNTAS QUE CONFORMAN LA ENCUESTA PARA LA..... 79
TABLA 3.3 RESULTADOS DEL NIVEL DE ACEPTACIÓN DE LOS MINI JUEGOS 81
TABLA 3.4 RESULTADOS DE LA PRUEBA EN LOS PARTICIPANTES QUE..... 83
TABLA 3.5 RESULTADOS DE LA PRUEBA DE MOTRICIDAD EN LOS PARTICIPANTES..... 84
TABLA 3.6 VARIACIÓN EN FUNCIÓN DEL TIEMPO EN PACIENTES CON EP. PLANEGAME 86
TABLA 3.7 VARIACIÓN EN FUNCIÓN DEL PUNTAJE EN PACIENTES CON EP. PLANEGAME..... 87
TABLA 3.8 VARIACIÓN EN FUNCIÓN DEL TIEMPO EN PACIENTES CON EP. BALLGAME 87
TABLA 3.9 VARIACIÓN EN FUNCIÓN DEL PUNTAJE EN PACIENTES CON EP. BALLGAME 87
TABLA 3.10 VARIACIÓN EN FUNCIÓN DEL TIEMPO EN PACIENTES SANOS. PLANEGAME..... 88
TABLA 3.11 VARIACIÓN EN FUNCIÓN DEL PUNTAJE EN PACIENTES SANOS. PLANEGAME..... 88
TABLA 3.12 VARIACIÓN EN FUNCIÓN DEL PUNTAJE EN PACIENTES SANOS. BALLGAME 89
TABLA 3.13 VARIACIÓN EN FUNCIÓN DEL TIEMPO EN PACIENTES SANOS. BALLGAME..... 89
TABLA 3.14. DATOS OBTENIDOS DE LOS PACIENTES CON EP VS. PACIENTES SANOS 89
TABLA A.1 DATOS DE LOS PACIENTES SANOS……104.

RESUMEN

El presente proyecto está enfocado en la rehabilitación de pacientes con EP, mediante el uso de videojuegos que responden a los movimientos del jugador. Para conseguir dicho objetivo, se inicia con la revisión del estado del arte, con el fin de delimitar las características del sistema. Posteriormente, se diseña e implementa el sistema utilizando Unity y Arduino. Finalmente se realizan las pruebas de validación del sistema con pacientes de la EP y sanos.

En primer lugar, se realiza una revisión del marco teórico de la Enfermedad de Parkinson, la rehabilitación y los videojuegos presentes en la actualidad, analizando ventajas y desventajas, de los diferentes sistemas existentes en el mercado.

En segundo lugar, se definen las características que debe cumplir el juego, como por ejemplo las tecnologías de transmisión, metodologías a implementar, equipos y software que se utilizaran para el diseño.

A continuación, se realiza el diseño del videojuego, se usa el software Unity, el mismo que permite exportar el videojuego a la plataforma Android, el sistema consiste en la adquisición de datos del paciente usando un módulo Bluetooth y un acelerómetro, cuyos datos intervienen en el juego y son enviados de manera inalámbrica al dispositivo móvil, para el proyecto fueron un celular y una Tablet. La aplicación “Rehabilitación” contiene los videojuegos: PlaneGame con tres niveles y BallGame con dos niveles de participación.

Finalmente, se procede con la validación del sistema y el análisis de resultados, por lo que, se realizan pruebas en la Universidad del Adulto Mayor usando un celular y una Tablet.

En conclusión, los resultados al finalizar las sesiones de rehabilitación demuestran que el sistema desarrollado mejora los movimientos motores de las extremidades superiores, lo que favorece para elevar la calidad de vida de los pacientes.

INTRODUCCIÓN

La enfermedad de Parkinson es un trastorno que afecta mayormente a mujeres y hombres de la edad adulta provocando que pierdan progresivamente sus capacidades funcionales y habilidades motoras, por tal motivo en la actualidad existen varias tecnologías que permiten que las personas con enfermedades neurodegenerativas recuperen o mejoren de alguna manera su condición física, mental y social.

Estas tecnologías sirven para la rehabilitación, pero incluyen equipos costosos en centros especializados, en donde el paciente debe asistir a veces diariamente para realizar sus actividades, lo que provoca que pierda el interés o sienta cansancio. Por tal motivo, existe otro tipo de rehabilitación denominada Exergame que consisten en ejercicios basados en juegos para brindar mayor facilidad de ejercitarse al paciente.

Los adultos mayores de hoy en día desconocen la tecnología en teléfonos móviles como conexión wifi, Blueotooth, consolas, etc., razón por la cual un tratamiento de rehabilitación basado en nuevas tecnologías debe poseer ciertas características especiales para que el adulto mayor se sienta cómodo y más aún si posee una enfermedad neurodegenerativa, ya que existen estudios que demuestran que los videojuegos mejoran las capacidades funcionales de los adultos con problemas degenerativos, pero estos videojuegos deben ser adaptables para ellos.

Debido a esto, en el presente proyecto se ha diseñado un dispositivo inalámbrico que permita estimular la rehabilitación en los pacientes con Parkinson, por medio de dos mini juegos que le brindaran entretenimiento y a la vez permitirá cierto grado de mejora en la movilidad de las extremidades superiores para facilitar la realización de las tareas cotidianas.

Este proyecto consta de cuatro capítulos, el primer capítulo aborda un sustento teórico sobre la enfermedad de Parkinson, el diagnóstico, tratamiento e incidencia en la Provincia del Azuay además de los tipos de Rehabilitación existentes y las tecnologías usadas para el sistema. Mientras que en el capítulo dos se realiza el diseño y la implementación del dispositivo y el videojuego. En el tercer capítulo se analizan los resultados de las pruebas realizadas en los pacientes con EP y sanos. Finalmente, en el cuarto capítulo se presentan las conclusiones y recomendaciones para el proyecto.

ANTECEDENTES DEL PROBLEMA DE ESTUDIO

La Enfermedad de Parkinson es un trastorno del movimiento cuyos síntomas comienzan lentamente y a medida que empeoran ocasionan dificultades para caminar o hacer labores simples. Para el control de los síntomas se establecen ciertos tratamientos médicos y otros basados en la rehabilitación, como son los Exergames que consisten en actividades de fuerza, equilibrio y flexibilidad para que los pacientes se ejerciten mientras juegan videojuegos integrando esfuerzo físico o movimientos sedentarios en respuesta a las demandas del juego de manera divertida, entretenida y se puede utilizar con pacientes de cualquier edad. [1], [2]. Por esta razón, en [3] se ha demostrado que los juegos contribuyen a aumentar la motivación en las sesiones de rehabilitación, ya que se consideran formas muy atractivas de entretenimiento interactivo y esta es la razón clave por la cual se han propuesto el uso de juegos para ayudar en el diseño de la rehabilitación.

Debido a esto, se han desarrollado estudios que determinan la importancia que tienen los Exergames para la prevención y la rehabilitación de pacientes con enfermedades neurodegenerativas [4] realizando entrenamientos de equilibrio basado en videojuegos de Nintendo Wii para mejorar el balance de las personas que se recuperan de una lesión de la médula espinal, una lesión cerebral o un accidente cerebrovascular [5]–[7].

En el tratamiento de la EP se han usado consolas de Nintendo Wii para demostrar que mediante su uso las personas pudieron estar de pie en una pierna por más tiempo, obteniendo un mejor equilibrio bípedo con los ojos abiertos y un mejor rendimiento en el cronometrado además de evaluar la satisfacción del paciente. [8], [9]. En [10] se ha diseñado un videojuego para mejorar la postura y el equilibrio usando Microsoft Kinect, el juego consta de 12 niveles de dos minutos cada uno con una complejidad creciente en el cual se pretende mejorar la flexión de las extremidades superiores para alcanzar frutas virtuales, mientras que en [11] se ha creado dos videojuegos teniendo en cuenta ejercicios de fisioterapia para mejorar la velocidad de los pacientes la gama de movimiento bilateral y reducir la rigidez, usando ambos brazos simultáneamente desde una posición sentada mediante la plataforma de Nintendo Wii. En [12] se presenta un videojuego basado en Microsoft Kinect que trabaja en la parte motora de las extremidades superiores y cognitiva de pacientes

con EP, cuyo juego posee 3 niveles y consiste en atrapar balones de colores a cierta velocidad.

De la misma manera, en [13] se presenta un juego basado en Kinect en el que el usuario hace estallar globos cayendo al azar a lo largo de cuatro postes verticales, para esto debe usar sus brazos y piernas. A medida que avanza a niveles más altos, la velocidad y el número de los globos aumentan de manera que caen con más frecuencia, permitiendo aumentar la movilidad general de los miembros superiores e inferiores. Otro caso, se da en [11], en donde se ha desarrollado un juego de computadora para rehabilitar el control postural de las personas con EP, para ello se usó el Kinect con un grupo de 9 personas. Al concluir las pruebas los pacientes indicaron seguridad en el juego y la aceptación del mismo para aplicarlo al hogar.

Finalmente, una aplicación es WuppDi que consiste en un minijuego para mejorar tres aspectos principales: el movimiento, la coordinación del movimiento y la mejora de la concentración en pacientes con EP. Posee una interfaz sencilla para el uso de personas de edad mayor, reservadas acerca de la tecnología y su funcionamiento consiste en el uso de una cámara web que se utiliza para detectar la entrada de movimiento. [14]

JUSTIFICACIÓN (IMPORTANCIA Y ALCANCES)

La enfermedad de Parkinson (EP) es un trastorno degenerativo, crónico y progresivo del sistema nervioso central que afecta la capacidad del cerebro para controlar los movimientos. Esta enfermedad se caracteriza por los trastornos del movimiento, tales como rigidez muscular, temblores, lentitud del movimiento (bradicinesia) y, en caso extremo, una pérdida de movimiento físico [15]–[17], ya que comúnmente los pacientes con EP tienden a ser inactivos [18] y a través de la inactividad, pueden surgir trastornos secundarios, que incluyen disminución de la capacidad aeróbica, disminución de la función muscular, disminución de la movilidad de las articulaciones y disminución de la calidad ósea. Las acciones como alcanzar, agarrar, manipular y reemplazar objetos a menudo causan problemas al realizar actividades como vestirse y comer provocando la reducción de la velocidad en las articulaciones.[19]

Estos síntomas dificultan las actividades cotidianas comunes, las relaciones sociales de los pacientes y reducen la calidad de vida. Dada esta situación, la prescripción de ejercicio físico se ha postulado como un tratamiento no farmacológico de bajo coste en la EP, ya que retrasa el deterioro de la función motora, mejora la cognición y prolonga la independencia funcional, considerándose una estrategia rehabilitadora de gran utilidad. [20], [21]

Los pacientes con EP generalmente tienen que asistir a rehabilitación clínica a largo plazo para notar mejoras en los trastornos, lo que implica a menudo utilizar equipos en las sesiones que representan un alto costo económico con un esquema repetible de ejercicios prescritos que a la vez permite que los pacientes pueden perder el interés y no terminar el tratamiento [13], [22].

Por esta razón, en [3] se ha demostrado que los juegos contribuyen a aumentar la motivación en las sesiones de rehabilitación, ya que se consideran formas muy atractivas de entretenimiento interactivo y esta es la razón clave por la cual se han propuesto el uso de juegos para ayudar en el diseño de la rehabilitación atractiva.

Teniendo estas consideraciones, en el presente proyecto se realiza dos mini juegos para estimular la rehabilitación de los pacientes con EP en las extremidades superiores mediante el uso de un dispositivo inalámbrico evitando usar equipos

costosos, palancas de Nintendo o dispositivos difíciles de manipular para pacientes de edad adulta, finalmente los ejercicios de rehabilitación están basados en ejercicios de fisioterapia de pronosupinación y flexo extensión del brazo, codo y mano lo que permitirá mejorar los síntomas motores de los pacientes, reducir la rigidez muscular, mejorar la movilidad en las extremidades superiores, disminuir el temblor ya que se realiza movimientos en los brazos, y por último la distracción del paciente con los videojuegos.

El proyecto propuesto beneficiará a adultos mayores con y sin la EP que necesiten rehabilitación para sus extremidades superiores, ya que no disponen de sistemas que utilicen Exergaming como parte de la rehabilitación, siendo la rehabilitación convencional dentro de la sociedad, la única elección en las terapias para los pacientes con EP.

OBJETIVOS

OBJETIVO GENERAL

- Diseñar, Desarrollar e Implementar un sistema inalámbrico de bajo costo para la rehabilitación en pacientes con Parkinson.

OBJETIVOS ESPECIFICOS

- Estudiar y analizar las características de la Enfermedad de Parkinson (EP) y las estadísticas de incidencia de la enfermedad en la provincia del Azuay.
- Identificar y estudiar las diferentes técnicas de rehabilitación en pacientes con la EP y los videojuegos existentes para la rehabilitación de enfermedades neurodegenerativas.
- Diseñar y construir el dispositivo inalámbrico de bajo costo para estimular la rehabilitación de las extremidades superiores de pacientes que sufren la EP.
- Diseñar y Desarrollar un videojuego para dispositivos móviles que estimule la rehabilitación de las extremidades superiores de pacientes que sufren la EP
- Realizar pruebas y evaluar el funcionamiento en laboratorio y en pacientes del sistema desarrollado.
- Realizar un manual del sistema y un artículo científico.

Capítulo 1 : FUNDAMENTACIÓN TEÓRICA

En este capítulo se presenta una descripción general de la Enfermedad de Parkinson, el diagnóstico, tratamiento, incidencia y las técnicas de rehabilitación para los pacientes que poseen EP. De la misma manera se aborda toda la información teórica de los componentes y tecnologías usadas para el desarrollo del proyecto

1.1 CARACTERISTICAS DE LA ENFERMEDAD DE PARKINSON

La enfermedad de Parkinson (EP) es una enfermedad neurodegenerativa y progresiva descrita por primera vez en 1817, [23] con una incidencia mayor en el hombre que en la mujer [24], [25] y cuya prevalencia aumenta con la edad, afectando a más del 1% de la población mayor de 60 años y 4% a la edad de 80 años [26], [27] y presenta manifestaciones clínicas motoras y no motoras, [13], [28]. Las principales características motoras se pueden agrupar bajo el acrónimo TRAP que incluye: temblor en reposo, rigidez, akinesia o bradicinesia e inestabilidad postural [29]–[31] mientras que signos no motores se presentan a menudo tales como disfunciones cognitivas, trastornos del sueño y disautonomía.[18]

Cabe destacar que esta enfermedad se distingue por la pérdida de las neuronas pigmentadas que contienen dopamina en la porción compacta de la sustancia negra y la presencia de inclusiones citoplásmicas conocidas como cuerpos de Lewy que contienen proteínas incluyendo sinucleina y ubiquitina que perjudican el funcionamiento óptimo de las neuronas [13], [30], [32]. A pesar de esto, estudios demuestran que el origen de la enfermedad se debe a factores genéticos y ambientales, [24], [33], ya que se ha descubierto 18 cromosomas genéticos con la forma esporádica de la enfermedad [34], [35] y aproximadamente, el 10% de los pacientes con manifestaciones clínicas de EP tienen una clara etiología familiar [31]

1.1.1 DIAGNOSTICO

El diagnóstico de EP es de vital importancia para el paciente ya que permite establecer de manera oportuna algún tratamiento, se basa clínicamente en reconocer la manifestación de los principales síntomas motores, temblor de reposo, rigidez,

inestabilidad postural y bradicinesia, durante las primeras etapas de la enfermedad [35]. Cuando el historial del paciente revela una progresión de los síntomas y una aceptable respuesta a la terapia con levodopa, es probable que el diagnóstico sea correcto, caso contrario un diagnóstico erróneo puede causar parkinsonismo neurodegenerativo o secundario debido a la existencia de un período en donde la degeneración está en curso siendo no evidente clínicamente, dando lugar a la presencia de síntomas no motores como anemia, ansiedad, apatía, fatiga, depresión, trastornos del sueño con movimiento oculares, que permiten determinar con anterioridad la presencia de la enfermedad [36]–[40]

Otro criterio que permite diagnosticar la EP está basado en las pruebas de olfato debido a que la gran mayoría de los pacientes con EP tienen una pérdida olfatoria severa al inicio de la enfermedad. [41], [42] por otra parte están los criterios que permiten identificar otros marcadores para la enfermedad como son: MDS-EP(Clinical Diagnostic Criteria for Parkinson's disease), PPMI (the Parkinson's Progression Marker Initiative), PDBP(the Parkinson's Disease Biomarkers Program) que utilizan un conjunto de datos comunes y datos públicamente disponibles para ayudar a identificar biomarcadores para la enfermedad de Parkinson.[36], [43]. Aparte existen escalas de calificación estandarizadas para la enfermedad de Parkinson como son las escalas de estadios de Hoehn-Yahr [44] y la Escala Unificada de la enfermedad de Parkinson de la Sociedad de Trastornos del Movimiento MDS-UPDR, la cual presenta un estudio más completo e integral de los aspectos clínicos relevantes para la evaluación del paciente, tanto para fines clínicos como de investigación.[45], [46]

Finalmente se consideran otras pruebas que permiten diagnosticar la EP, como es 7Tesla MRI que permite detectar cambios en la morfología nigral de EP [47], también la trasferencia de magnetización MRI puede detectar una reducción de la melanina y de lesiones estructurales que causan parkinsonismo secundario. [48]

1.1.2 TRATAMIENTO

El tratamiento más frecuente para la enfermedad de Parkinson consiste en utilizar medicamentos que compensen el déficit dopaminergico, en la actualidad la Levodopa sigue siendo el fármaco más eficaz que incrementa la concentración de

dopamina en los receptores del cerebro [49]–[51], el cual se prescribe al inicio de la enfermedad y con precaución en pacientes menores de 50 años. [52], [53]

A pesar de los beneficios que posee la Levodopa, esta no alivia efectivamente todos los signos cardinales asociadas con la EP y con el uso frecuente y con el avance de la enfermedad se presentan fluctuaciones motoras en el paciente. [49], [54]. Otra elección de fármaco para tratar la EP son los agonistas de la dopamina, pero se asocia con efectos más adversos que la levodopa, como son náuseas, somnolencia, mareos, alucinaciones, edemas y trastornos de control de los impulsos. [55]. Con el avance de la enfermedad se presentan mayores complicaciones sintomáticas que no responden a la medicación dopaminèrgica, por ende, la intervención quirúrgica basada en la implantación de electrodos en los ganglios del cerebro sigue siendo una herramienta importante para mejorar los síntomas motores, pero está reservada para una pequeña minoría de pacientes menores de 70 años. [54], [56]

1.2 TERAPIA DE REHABILITACION

Además del tratamiento médico y quirúrgico, se ha propuesto el ejercicio físico de diferente tipo e intensidad como un tratamiento para EP con el fin de maximizar las capacidades funcionales, mejorar la calidad de vida y minimizar los síntomas motores y no motores de los pacientes.[57]–[60], incluso en [61] se concluye que ejercicio físico reduce el estrés oxidativo crónico y estimula la biogénesis de las mitocondrias y la síntesis de neurotransmisores como la dopamina en los pacientes con EP.

La rehabilitación posee un enfoque multidisciplinario y heterogéneo que consiste en la fisioterapia, terapia ocupacional, terapia del habla y terapia cognitiva para pacientes con EP, sin embargo, la intervención de rehabilitación debe adaptarse a las condiciones individuales de los pacientes.[62].

1.2.1 TÉCNICAS DE FISIOTERAPIA PARA EP

La fisioterapia en personas afectadas por la EP se centra en seis áreas específicas esenciales: las transferencias, la postura, el funcionamiento de las extremidades superiores (alcanzar y agarrar), el equilibrio (y las caídas), la marcha y

la capacidad física, y la inactividad. También utilizan las estrategias de orientación atencional, estrategias cognitivas de movimiento y el ejercicio para mantener o aumentar la independencia, la seguridad y calidad de vida relacionada con la salud. [63]

Los tratamientos que incorporan ejercicios de fisioterapia basada en estiramiento, fortalecimiento muscular, terapia ocupacional y entrenamiento en cinta rodante mejoran aspectos específicos de la movilidad.[50], [62], [64]. En [65]–[67], se demuestra que el Tai-Chi mejora el equilibrio, control postural y la movilidad de los pacientes con EP permitiendo una disminución de caídas. Dentro de la Fisioterapia, intervienen algunas técnicas que son:

a) Terapia del movimiento o cinesiterapia

Esta técnica permite aumentar la elasticidad y flexibilidad de músculos y a prevenir retracciones que podrían causar deformidades, además de estimular la circulación sanguínea y mejorar la atrofia muscular. [68] Algunas intervenciones fisioterapéuticas incluyen:

- Ejercicios que mejoren la fuerza muscular.
- Ejercicios de coordinación y destreza.
- Ejercicios para disminuir la rigidez y eviten las contracturas por desuso de los músculos.
- Ejercicios para mejorar y mantener la movilidad articular.
- Ejercicios de estiramiento para que las estructuras musculares se mantengan lo más flexibles posible.
- Ejercicios posturales para intentar retrasar al máximo la tendencia de la inclinación del tronco y cabeza.
- Técnicas especiales para la marcha, con el fin de superar las alteraciones de la misma, como por ejemplo la técnica de Frenkell [69]

En la Tabla 1.1 se presenta algunos ejercicios específicos para mejorar las distintas alteraciones motoras en las extremidades superiores e inferiores que pueden presentarse en la enfermedad de Parkinson.

Tabla 1.1 Ejercicios de Movilidad General los Miembros Superiores e Inferiores en Pacientes con la Enfermedad de Parkinson
Fuente: [70]

Ejercicios de Movilidad General los Miembros Superiores e Inferiores en Pacientes con la Enfermedad de Parkinson	
Miembros Superiores	*Miembros Inferiores.*
Flexo-extensión, abducción-aducción, rotaciones de hombro.	Flexo-extensión, abducción y aducción, rotaciones de cadera.
Flexo-extensión, pronosupinación de antebrazo.	Flexo-extensión y rotaciones de rodilla.
Flexo-extensión, abducción-aducción, circunducción de muñeca	Flexo-extensión, inversión-eversión de tobillo.
Flexo-extensión de dedos.	Flexo-extensión de dedos.
Estiramientos: pectorales, bíceps, tríceps y músculos del antebrazo.	Estiramientos de cadena anterior y posterior del miembro inferior.
	Ejercicios de propiocepción de rodilla y tobillo

A continuación, se indican algunas recomendaciones para realizar los ejercicios:[68]

- Los ejercicios deben realizarse a un ritmo lento, progresivo y rítmico.
- Es recomendable efectuar los ejercicios a diario y siempre en situación “on”
- Se deben introducir pausas para evitar la fatiga.
- Antes de empezar cualquier ejercicio debemos rectificar la postura en la medida de lo posible.
- En caso de inestabilidad y riesgo de caídas, los ejercicios deben realizarse bajo la vigilancia de otra persona o bien teniendo cerca algún punto de apoyo seguro.

b) Hidroterapia

Los baños a 37º-38º tienen un efecto muy relajante sobre la musculatura disminuyendo la rigidez del paciente con enfermedad de Parkinson y mejorando la amplitud articular. [68]

c) Masaje

Esta técnica crea una dilatación de los vasos sanguíneos que favorece la nutrición celular, ayudando así a disminuir la tensión muscular y la ansiedad. El masaje relajante puede ayudar a disminuir la rigidez y la tensión muscular[68]

1.2.2 TERAPIA OCUPACIONAL

La terapia ocupacional está indicada para mejorar la autonomía en el autocuidado (comida, vestido y aseo), las actividades productivas y las de ocio. Cuando esto ya no es posible, en las fases avanzadas de la enfermedad el terapeuta ocupacional es responsable de enseñar al cuidador el manejo del paciente y de valorar e indicar las adaptaciones y ayudas técnicas necesarias en el domicilio.[71]

1.2.3 LOGOPEDIA

La terapia de logopedia tiene como objetivo mejorar la inteligibilidad del habla, la comunicación y el lenguaje mediante técnicas de tratamiento conductuales o ayudas instrumentales que incluyan ejercicios respiratorios, los ejercicios prosódicos, los métodos de tratamiento de la voz , y otros, como son aparatos de comunicación que pueden incluir: tableros alfabeto o para la estimulación, amplificadores de voz, sistemas de salida de voz digitalizada, mensajes de voz grabados, retroalimentación auditiva retardada, o dispositivos de retroalimentación por ordenador portátil.[63]

1.2.4 TÉCNICAS INNOVADORAS DE REHABILITACION

En la fisioterapia tradicional, la rehabilitación consiste en sesiones diarias de ejercicios realizados con un terapeuta supervisando al paciente en tiempo real [2] lo que conlleva a un aumento de cansancio, estrés y aburrimiento debido a la repetición de ejercicios, además de representar un costo mayor para los pacientes, por lo que se necesitan nuevas soluciones para administrar la rehabilitación. En este contexto, una gran cantidad de investigación se ha dirigido hacia los videojuegos provenientes desde consolas como utilizan Wii de Nintendo, Sony PlayStation Eye, el sensor de Microsoft Kinect, y los Exergames en particular.

Las imágenes motoras (IM) y la realidad virtual (RV) son dos enfoques que permiten obtener una rehabilitación motora y cognitiva para el tratamiento de los déficits de la marcha y el equilibrio en la EP [8], [72]–[74]. MI se describe como la representación mental de las acciones en ausencia de movimiento corporal [75] que

implica una variedad de componentes cognitivos, como la memoria, la excitación y la atención, mientras que RV es la interacción de la persona en el mundo real con un entorno virtual (EV), generado por un ordenador. [76], [77]. Estudios realizados en [8], [78] demuestran mejoras en la caminata en pacientes con problemas neurodegenerativos usando IM y RV, y en [77], [79] encontraron que el ejercicio realizado mediante una interfaz de realidad virtual mejora la función de los dedos, el pulgar y el rango general de movimiento.

1.2.5 EXERGAMING EN REHABILITACION

Exergaming es otra herramienta para prevenir o frenar el ritmo de las pérdidas funcionales y tiene un impacto positivo en la calidad de vida de los pacientes con EP, donde uno de sus beneficios es que se puede administrar en el hogar [80], [81] En base a un análisis realizado en [82] se indican seis estudios que muestran mejoras en las pruebas clínicas en EP después de una intervención exergaming mediante juegos de Nintendo Wii [9], [79], [83]–[85] cuyas pruebas fueron: balance de Berg, soporte de pierna única, prueba de alcance funcional, función motora , y pruebas en base a la escala UPDRS II . De igual manera en [86] mostraron que 14 sesiones de danza basada en exergaming resulta en un mejor equilibrio y reducción de los síntomas motores en las personas con EP y en [87] se usaron tres juegos XBOX Kinect ™ comerciales durante ocho semanas para obtener mejoras en la escala UPDRS (II y III).

Por otra parte, en [88] se ha diseñado un exergame que consiste en cinco mini juegos para rehabilitar los movimientos de la parte superior del cuerpo y la memoria. Los resultados mostraron que la mayoría de los participantes disfrutaron de la experiencia, sin embargo, se indica que algunos de los juegos eran demasiado complejos y que tenían dificultad con los controladores manuales, siendo no favorable para los temblores de los pacientes. Otro caso se da en [79] donde se encontró que pacientes con EP no mejoraron en dos juegos que requieren reacciones rápidas en respuesta a los objetivos virtuales.

A pesar de esto, los Exergames han demostrado ser una herramienta importante en el momento de la rehabilitación para pacientes con EP, sin embargo,

una rehabilitación basada en exergame debe contener cierto nivel de dificultad para el paciente y así evitar la frustración del mismo, además debe poseer seguridad en el momento de ejecutar los movimientos ya que el uso de plataformas elevadas puede provocar lesiones en los pacientes. [89]

Es importante tener otras consideraciones en el diseño de videojuegos para los pacientes con EP, razón por la cual en [88] se presenta ciertos criterios de diseño, entre ellos, es que el juego se debe adaptarse a las habilidades de los pacientes por lo que los movimientos no deben ser precisos ni lentos ya que podría intensificar algún tipo de dolor.[12]. De igual manera es necesario incorporar estímulos en el juego, recompensas por cada desafío [90], el juego debe ser simple, realista, con escenarios al aire libre, efectos de sonidos y con la tecnología adecuada para el paciente evitando manipular controladores o presionar botones además se debe proporcionar instrucciones del juego. [91]

1.3 TECNOLOGIAS PARA EXERGAMING

1.3.1 HADWARE Y SOFTWARE LIBRE

El software libre garantiza al usuario cuatro libertades: libertad de uso, de estudio y modificación, de distribución y redistribución de las versiones modificadas, sin tener que pedir ni pagar permisos al desarrollador o entidad específica. [92] Un programa es considerado libre si posee licencias de distribución como son GPL (General Public License) o LGPL, entre otras.[93]. Por otra parte el hadware libre son todos aquellos dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, en donde el usuario puede estudiarlo para entender su funcionamiento, modificarlo, reutilizarlo, mejorarlo y compartir dichos cambios. El objetivo del hardware libre es acercar la tecnología actual a la población de manera que involucre al usuario final para que entienda y participe en la creación de futuras tecnologías. [93][94]

1.3.2 MOTORES DE VIDEOJUEGOS

Los motores de videojuegos son sistemas software que permiten la creación y desarrollo de videojuegos, actuando como un software intermediario. Disponen de un

motor de renderizado para gráficos 2D y 3D, un motor físico para la detección de colisiones, animación, sonidos, inteligencia artificial, scripting, etc. [95]

A continuación, se describe de manera breve los cuatro motores más populares para el desarrollo de videojuegos

1.3.2.1 Unreal Engine:

Es un motor gráfico lanzado por Epic Games, sucesor del UDK, el mismo que permite desarrollar contenido para multitud de plataformas, Windows, Mac OS X, Android, iOS, etc. Tiene grandes capacidades gráficas incluyendo herramientas como la creación de luces dinámicas y un nuevo sistema de partículas que puede manejar hasta un millón de partículas en una misma escena. Requiere de bastante aprendizaje para un buen uso. El lenguaje que utiliza Unreal Engine 4 es C++, además con su descarga incluye acceso al código fuente del motor, permitiendo personalizar las funcionalidades que más convengan al desarrollador a su proyecto concreto. No obstante, esto mismo requiere un amplio conocimiento, tanto del lenguaje como del propio motor, siendo necesario pagar el 5% de las ventas.[95]

1.3.2.2 CryEngine:

Es un motor de juego muy potente con unas capacidades gráficas y de iluminación de última generación que requiere de mucho aprendizaje para usarlo de manera eficiente. Fue desarrollado por Crytek y está diseñado para ser usado en plataformas PC y consolas, incluyendo Playstation 4 y Xbox One. Su precio es de 8 euros al mes sin necesidad de aportar más dinero por las ganancias de los juegos creados. [95]

1.3.2.3 GameMaker:

Desarrollado por YoYo Games, éste es el motor más básico y enfocado a un público con escasos conocimientos de programación, con un corto periodo de aprendizaje. Es un motor cross-platform, es decir, el mismo código utilizado para desarrollar los juegos sirve para varias plataformas. No obstante, su versión gratuita sólo ofrece soporte para desarrollar para Windows. Si se desea crear contenido para todas las plataformas es necesario pagar los 559.99 dólares de su versión completa. Además, su versión gratuita es poco más que una demo del potencial del motor, pues

será necesario pasar por caja si se quiere realizar cualquier proyecto con un mínimo de complejidad. [95]

1.3.2.4 UNITY

Anteriormente llamado Unity3D, el renombrado Unity, desarrollado por Unity Technologies, es uno de los motores más utilizados entre los nuevos desarrolladores en la industria de los videojuegos, ya que está enfocado a juegos sencillos que no requieran de una gran complejidad. Aun así, también es el motor escogido por grandes empresas para sus pequeñas producciones y juegos que no necesitan de un gran potencial gráfico. [95]

Unity tiene soporte de compilación con diferentes tipos de plataformas como son Android, Xbox 360, PlayStation 3, Wii, Wii U, iPad y iPhone. Puede usarse junto con Blender, 3ds Max, Maya, Modo, ZBrush, Cinema 4D, Cheetah3D, Adobe Photoshop, Adobe Fireworks, entre otros para diseñar elementos del juego.

El lenguaje de programación implica C, Java o Boo, facilitando el desarrollo de scripts mediante el uso de MonoDevelope y posee un Asset Store en donde el usuario puede acceder a la colección de más de 4.400 paquetes de Assets en una amplia gama de categorías, incluyendo modelos 3D, texturas y materiales, sistemas de partículas, música y efectos de sonido, tutoriales y proyectos, paquetes de scripts, extensiones para el editor y servicios en línea. [96]

Además, la versión gratuita ofrece casi todas las funcionalidades completas del motor, reservando para la versión Pro, que se puede adquirir por 75 dólares al mes, o 1500 dólares si se prefiere pagar una única vez

Componentes de Unity

En breve se explican algunos conceptos que se necesita para crear una experiencia de juego única, increíble, y divertida. [96]

- ***Escena***

Las escenas contienen los objetos del juego y pueden ser usadas para crear un menú principal, niveles individuales, y cualquier otra cosa. En cada escena, se coloca el ambiente, obstáculos, y decoraciones, el diseño esencial y la construcción del juego. [97] En la figura 1.1 se observa una escena de juego en Unity

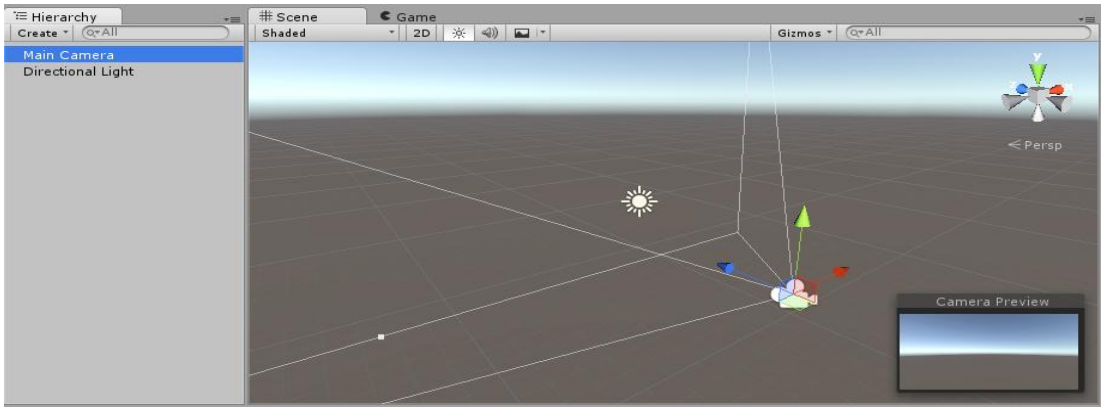


Figura 1.1 *Nueva escena vacía, con los objetos 3D predeterminados, una cámara y un directional light.*

Fuente: *Manual de Unity*[97]

- ***GameObjects***

Cada objeto en el juego es un GameObject, sin embargo, los GameObjects no hacen nada por sí mismos ya que necesitan propiedades especiales antes de que puedan volverse un personaje, un ambiente, o un efecto especial. [98]. La figura 1.2 presenta ejemplos de GameObjects en Unity.

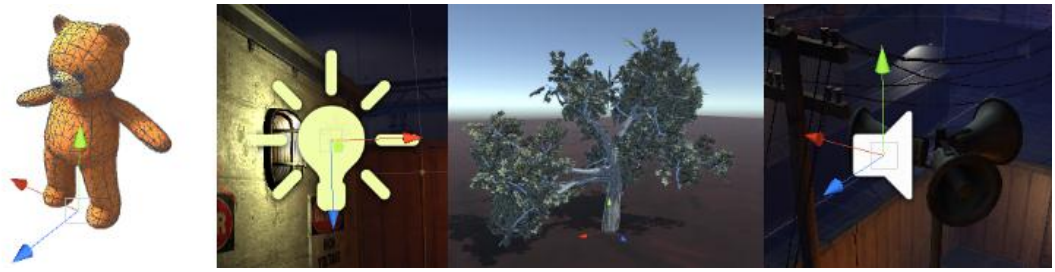


Figura 1.2 *Cuatro diferentes GameObjects, un personaje animado, una luz, un árbol y una fuente de audio*

Fuente: *Manual de Unity*[98]

- ***Colliders***

Los componentes Collider definen la forma de un objeto para los propósitos de colisiones físicas. Un collider, el cual es invisible, necesita no estar con la misma

forma exacta que el mesh del objeto, siendo estos los colliders más simples en 3D: Box Collider, Sphere Collider y Capsule Collider.

- ***Character Controllers (Controladores de Personaje)***

En físicas 3D, la aceleración de personajes y el movimiento pueden ser creados usando un Character Controller. Este componente le da al personaje un simple collider (colisionador) en forma de capsula que siempre está en posición vertical, tal como se muestra en el ejemplo de la Figura 1.3.[99].

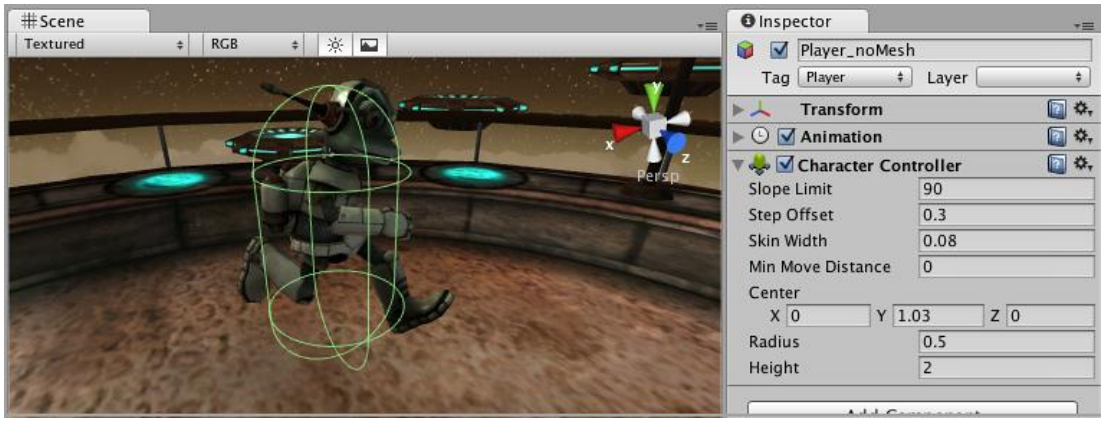


Figura 1.3. *Ejemplo de un Character Controller en Unity*
Fuente: *Manual de Unity*[99]

- ***Triggers***

Un collider configurado como Trigger, utilizando la propiedad Is Trigger, no se comporta como un objeto sólido y simplemente le permitirá a otros colliders pasar a través de él. Cuando un collider entra su espacio, un trigger va a llamar la función OnTriggerEnter en los scripts del trigger del objeto. [100]

- ***Scripts***

Unity permite crear un propio componente para cada comportamiento de los GameObject utilizando scripts, permitiendo activar/desactivar eventos del juego, modificar propiedades del Componente en el tiempo y responder al input del usuario de la forma deseada.

- ***Audio Source (Fuente de Audio)***

El sistema de audio de Unity es flexible y poderoso, puede importar la mayoría de formatos estándares de audio y tiene características sofisticadas para

reproducir sonidos en un espacio 3D. El Audio Source reproduce un Audio Clip en la escena. El audio source puede reproducir cualquier tipo de Audio Clip y puede ser configurado que reproduzca estos como 2D, 3D.[96]

- ***Interfaz de Usuario***

El sistema de Unity permite crear interfaces de usuario rápidas e intuitivas que trabaja con el componente principal, Canvas que es el área donde todos los elementos UI deben estar y es mostrado como un rectángulo en la Vista de Escena. La figura 1.4 muestra una vista general de la interfaz de usuario en Unity

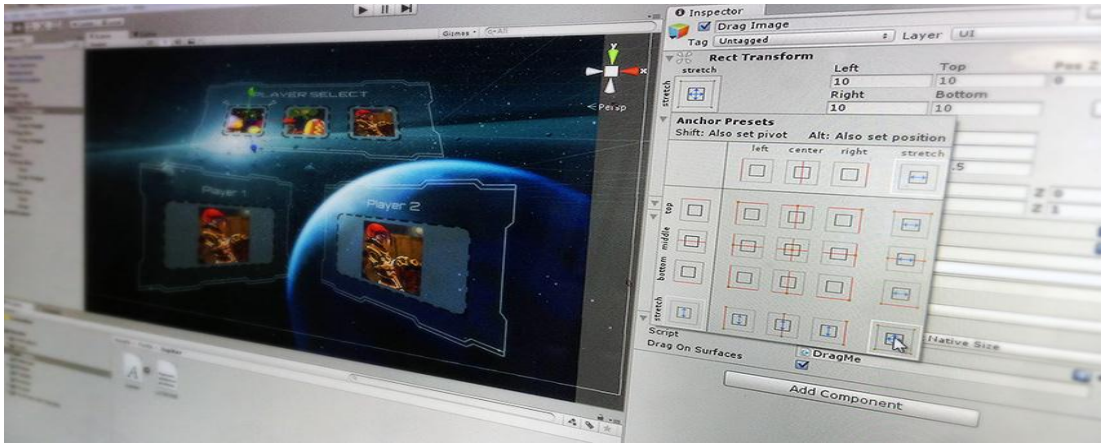


Figura 1.4. *Ejemplo de Interfaz de Usuario en Unity*
Fuente: *Manual de Unity*[96]

1.3.3 PLATAFORMA ANDROID

Android es un sistema operativo móvil basado en Linux, que permite desarrollar aplicaciones de manera completa, abierta y libre. La estructura del sistema operativo Android se compone de aplicaciones que se ejecutan en un framework Java, son orientadas a objetos sobre el núcleo de las bibliotecas de Java en una máquina virtual Dalvik con compilación en tiempo de ejecución [101]

1.3.3.1 Android SDK

El SDK de Android es el conjunto de herramientas de desarrollo software que permite programar aplicaciones para dicha plataforma. Este SDK incluye un depurador de código y un simulador de teléfono basado en QEMU (Quick Emulator). [102]

1.3.4 ARDUINO

Es una plataforma electrónica de código abierto basado en hardware y software que puede adquirir datos del entorno a través de sensores. Está dirigido a cualquier persona que hace proyectos interactivos ya que simplifica el proceso de operar con microcontroladores. Consiste en una placa económica con entradas y salidas, analógicas y digitales, cuyo entorno de programación sencillo y claro basado en Processing. [103]

1.3.4.1 Arduino Pro Micro

Es similar al Pro Mini con la diferencia que posee un ATmega32U4 a bordo, el cual permite agregar conectividad USB y acabar con la interfaz USB externa. Realiza todas las funciones Arduino con las que se está familiarizado con: 4 canales de 10-bit ADC, 5 pines PWM, 12 DIO, conexiones serie Rx y Tx y un regulador de tensión a bordo para que pueda aceptar una tensión de hasta 12V.[104]. En la figura 1.5 se indica el Arduino Pro Micro.

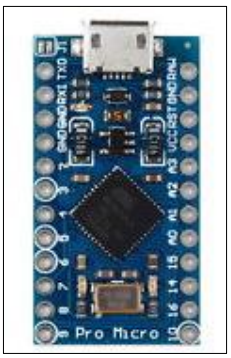


Figura 1.5. *Arduino Pro Micro*
Fuente: [104]

Mientras que en la Tabla 1.2 se presentan las principales características técnicas del Arduino Pro Micro usado para el desarrollo del dispositivo

Tabla 1.2. Características Técnicas de Arduino Pro Micro

Fuente: [104]

Características Técnicas del Arduino Pro Micro	
Pines Digitales I/O	12
Pines de Entrada Analógica	4
Voltaje de Operación	3.3-5.5V
Voltaje de Entrada	5-12V
Corriente máxima de salida	150mA
Microcontrolador	ATMega 32U4
Frecuencia Máxima	8MHz a 3V/ 16MHz a 5.5V
Dimensiones	3.31cm X 1.78 cm

1.3.5 BLUETOOTH HC-05

El Módulo Bluetooth HC-05 es un módulo Bluetooth SPP (Serial Port Protocol) fácil de usar, diseñado para la instalación de una conexión serial inalámbrica transparente. La figura 6 indica el modelo de un módulo Bluetooth.

Figura 1.6. *Modulo Bluetooth*
Fuente: [105]

En la Tabla 1.3 se presenta las características más importantes del módulo, siendo notable que este opera a partir de 3.3V facilitando el diseño del dispositivo a cierto valor de voltaje.

Tabla 1.3. Características Técnicas de Bluetooth HC-05
Fuente: Autor

Voltaje de alimentación	3.3VDC – 6VDC.
Voltaje de operación:	3.3VDC.
Baud rate por defecto	9600
Tamaño	4.4 cm x 1.6 cm x 0.7 cm
Corriente de operación	< 40 mA

1.3.6 ACELERÓMETROS

Este tipo de sensor transforma la aceleración en una señal eléctrica proporcional a la magnitud de dicha aceleración. En la actualidad existen los acelerómetros triaxiales que son capaces de registrar movimientos en diferentes planos del espacio, siendo más sensibles y capaces de registrar el tiempo que se emplea en diferentes actividades. [106]

1.3.6.1 Acelerómetro ADXL335

El ADXL335 es un acelerómetro pequeño, delgado, de baja potencia y completo de 3 ejes y utiliza un nivel de tensión de 3.3V. El producto mide la aceleración con un rango mínimo de escala de ± 3 g, en aplicaciones de inclinación, así como la aceleración dinámica resultante del movimiento, choque o vibración.[107]. En la figura 1.7 se puede apreciar el componente mencionado.

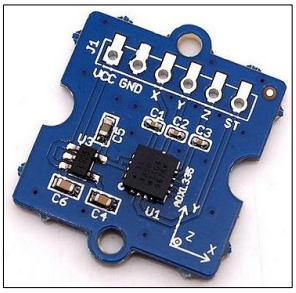


Figura 1.7. *Acelerómetro ADXL335*
Fuente: [108]

1.4 SISTEMAS DE JUEGOS COMERCIALES

Actualmente existen diferentes sistemas comerciales de videojuegos relativamente baratos que permiten la rehabilitación de enfermedades neurodegenerativas, entre ellas la enfermedad de Parkinson.

El hecho de utilizar interfaces y sistemas de juego ya disponibles en el mercado permite mayor facilidad de adquisición e instalación ya sea en el domicilio del paciente o en el centro de rehabilitación que, sin llegar a tener un coste elevado, garantizaría la captura de movimientos del cuerpo.[109]

A continuación, se detallan los principales sistemas de videojuegos comerciales.

1.4.1 NINTENDO WII

Utiliza el movimiento del jugador para controlar el juego a través de diversos dispositivos como el mando WiiPlus, que contiene un sensor de movimiento integrado y se comunica de manera inalámbrica mediante Bluetooth con la barra de sensores para ofrecer precisión. Incorpora un acelerómetro que permiten localizar en un espacio tridimensional al usuario, mientras que la interacción se realiza mediante gestos físicos y presión sobre los botones. Otro de los dispositivos es Wii Balance Board, un tablero a través del cual se controla el movimiento del jugador gracias a un sensor de presión [109]

Nintendo Wii fue diseñado para los usuarios sanos, sin embargo, se han realizado varios estudios [7], [9], [110], [111] que demuestran la efectividad de usar este sistema para la rehabilitación en pacientes con enfermedades neurodegenerativas incluyendo EP, en donde el terapeuta puede ser requerido para proporcionar estímulo e instrucciones y compensar las limitaciones de usabilidad de Wii. En la figura 1.8 se observa la consola de Nintendo Wii

Los videojuegos comerciales desarrollados usando este sistema consisten en: Wii Sports y Wii Fit proporcionando información audiovisual inmediata sobre el progreso. Los juegos del software Wii Sports adecuados para la rehabilitación incluyen: [112]

- Boxeo
- Bolos
- Tenis
- Golf
- Béisbol

Figura 1.8. *Consola de Nintendo Wii*
Fuente: [113]

1.4.2 KINECT

Microsoft Kinect incorpora una luz infrarroja y una cámara de vídeo para trazar un plano de la zona que se encuentra frente a ella, utiliza un algoritmo de decisión para determinar automáticamente referencias anatómicas del cuerpo, tales como articulaciones, en tiempo real. Tiene la capacidad de registrar el movimiento de todo el cuerpo, reconocimiento facial y de voz [109]

A continuación, se describe algunos videojuegos desarrollados para la rehabilitación neurodegenerativa usando Kinect

1.4.2.1 Paquete de la aventura de Kinect [114]

- **20.000 fugas:**

El jugador usa los brazos, las piernas y la cabeza para bloquear las fugas en la burbuja de vidrio para evitar que el agua se derrame en la burbuja. Este juego requiere la estabilidad del tronco, movimiento de los miembros superiores, miembros inferiores, tronco y cabeza. La figura 1.9 muestra una descripción grafica del videojuego.

Figura 1.9. *Videojuego 20.000 Fugas para la rehabilitación usando Kinect*
Fuente: [112]

- **Espacio Pop:**

El jugador debe dar un paso al costado, agacharse, soltarse y levantar los brazos para moverse en el espacio exterior y estallar burbujas. El jugador requiere la estabilidad del tronco y la fuerza adecuada en las extremidades superiores e inferiores para dedicarse a flotar. La exactitud del movimiento de la extremidad superior es necesaria ya que los objetivos son específicos.[112]

- **Bola de Rally:**

Con el uso de la extremidad superior y la cabeza, el jugador debe golpear una bola en movimiento que choca con una variedad de objetivos. El posicionamiento de los miembros superiores es importante para alcanzar un objetivo específico. [112]

1.4.2.2 Paquete de deporte de Kinect:

Dentro de este paquete se encuentran los juegos que atribuyen a deportes como Fútbol, voleibol, bolos, boxeo y mesa de tenis, los mismos que permiten el movimiento de las extremidades superiores e inferiores.

1.4.2.3 DoctorKinetic [115]

Tiene como objetivo mejorar la amplitud de movimiento, la fuerza, la coordinación, las actividades de la vida diaria y la cognición a través de juegos

divertidos y atractivos. Los juegos y ejercicios pueden ser modificados para acoplar los miembros superiores, los miembros inferiores o la espalda. El terapeuta puede prescribir ejercicios específicos que se incluirán en el programa y revisar el informe del desempeño del paciente. La figura 1.10 muestra el entorno de juego del sistema DoctorKinect [112]

Figura 1.10. *Entorno grafico de sistema de rehabilitación DoctorKinect*
Fuente: [112]

1.4.2.4 MIRA: Asistente de recuperación médica interactiva[116]

MIRA implica la participación de juegos dirigidos a la rehabilitación de la extremidad superior. El sistema se originó para ayudar a la rehabilitación de los impedimentos ortopédicos y se ha ramificado para implicar deficiencias neurológicas, centrada principalmente en la hemiparesia, tetraparesia, enfermedad de Parkinson, distonía, autismo y accidente cerebrovascular [112]

Los movimientos de los miembros superiores del hombro (flexión, abducción, rotación del plano frontal, rotación interna y externa, y levantamientos brazos bilaterales) se utilizan para alentar al jugador y permitir el logro de los objetivos del tratamiento. [112]. En la figura 1.11 se puede apreciar una gráfica del videojuego desarrollado por MIRA para la rehabilitación.

Figura 1.11. *Videojuego desarrollado en Kinect para la rehabilitación*
Fuente: [112]

1.4.2.5 SeeMe[117]

Otro sistema de rehabilitación es SeeMee cuyo objetivo permite mejorar la coordinación, equilibrio, fuerza muscular, amplitud de movimiento, tiempos de reacción y memoria. Los juegos se modifican fácilmente para adaptarse a las necesidades de tratamiento y se genera informes en tiempo real para permitir al terapeuta analizar los movimientos compensatorios, los ángulos relativos entre las extremidades, los tiempos de movimiento y el rango de movimientos. [112]

Es ideal para pacientes que han tenido un accidente cerebrovascular, enfermedad de Parkinson, síndromes extrapiramidales, operaciones de tumor cerebral, lesión de la médula espinal, atrofia muscular y una serie de problemas ortopédicos. [112]

1.4.2.6 Jintronix[118]

Es un sistema que utiliza ejercicios diseñados clínicamente para involucrar a los pacientes en su rehabilitación, el cual mide los resultados en tiempo real e informa al terapeuta para hacer ajustes a la terapia, así como desafiar al paciente a mejorar sus puntuaciones. El foco principal del sistema es para la rehabilitación de condiciones neurológicas tales como accidente cerebrovascular, Enfermedad de

Parkinson y TBI. La figura 1.12 representa a un paciente en su rehabilitación usando Jintronix mediante Kinect. [112]

Figura 1.12. *Videojuego desarrollado en Kinect para la rehabilitación*
Fuente: [112]

1.4.2.7 Play H4[119]

Play 4 Health es una plataforma de rehabilitación remota que usa Kinect para mejorar la eficacia del tratamiento de rehabilitación en fases iniciales y crónicas, siendo utilizado en diferentes niveles de asistencia, como hospitales, centros de salud y centros sociosanitarios, y lo que es más importante, puede ser utilizado en los hogares de los pacientes que implementan algún programa de rehabilitación.

La forma en que el sistema interactúa con el juego permite trabajar en diferentes aspectos físicos: [112]

- Movimiento de una o varias partes del cuerpo.
- Función motora fina y / o gruesa.
- Control de postura y equilibrio.
- Coordinación.

1.4.3 TABLETS

Es una computadora portátil con una pantalla táctil que combina los componentes de teclado, ratón, pantalla y altavoces separados en uno solo. Las tablets son portátiles y operadas por batería, lo que permite al paciente completar su rehabilitación remotamente desde la clínica generando cierta independencia de

operación. A continuación, se presentan algunos ejemplos de juegos disponibles que tienen un enfoque de rehabilitación, sin embargo, esta lista no es exhaustiva.[112]

ReHoblet:

Es un juego basado en Android que utiliza un avatar para navegar a través de un mundo, derrotar a los enemigos y recoger bonos, para ello, requiere movimientos gruesos del miembro superior para completar las acciones de elevación (arriba y abajo) y el transporte (izquierda y derecha). El terapeuta también puede iniciar sesión en un servidor web para rastrear los datos de su paciente, la configuración del juego y una lista de estadísticas de diferentes sesiones. Un estudio piloto [120] sobre cinco pacientes de edad adulta demostró un buen cumplimiento y compromiso con el juego.

Findex:

Es un juego basado en Android que utiliza actividades domésticas y relacionadas con el trabajo para involucrar al jugador en los ejercicios de tocar, arrastrar y estirar los dedos. El objetivo es mejorar el control de los dedos, el aislamiento, la coordinación, y el rango de movimiento. En [121] se encontró mejoras en la destreza de un pequeño grupo de pacientes con ictus.

Prehab:

Prehab es un juego diseñado para su uso en una tableta destinada a la rehabilitación de la parte superior del brazo a través de tareas como alcanzar y apuntar. Cada entorno es diferente, por lo que en [122] se demostró la preferencia por la interfaz submarina ya que era un nivel fácil y más adaptable al jugador.

1.5 INCIDENCIA DE LA ENFERMEDAD DE PARKINSON EN EL AZUAY

La incidencia de EP que se analiza a continuación se basa en información estadística del Ministerio de Salud Pública del Ecuador [123], la misma que contiene datos de morbilidad ambulatoria y mortalidad de la población en general, pero nuestro caso de estudio se centra en las personas de la provincia del Azuay que tienen la enfermedad.

Tabla 1.4. Incidencia de la Enfermedad de Parkinson en la Provincia del Azuay durante el año 2015
Fuente: [124]

Incidencia de la Enfermedad de Parkinson en la Provincia del Azuay -2015								
Morbilidad						Mortalidad		
Sexo		Edad (años)			Total	Sexo		Total
M	F	<50	50-65	>65		M	F	
68,25%	31,75%	4,76%	21,42%	73.80%	126	33.3%	66.6%	15

Como se observa en la Tabla 1.4 durante el año 2015, el mayor número de casos de EP sucede en la edad adulta a partir de los 65 años, mientras que el género masculino tiende a poseer con mayor frecuencia la enfermedad, sin embargo, el mayor número de defunciones es para el género femenino.

De la misma manera se obtienen los datos del año 2016, en donde se observa en la Tabla 1.5, un aumento en relación con el número de casos de EP, siendo evidente un mayor número de casos de mortalidad, mientras que en la Figura 1.13 se representa de manera gráfica la cantidad de personas con relación a la edad que poseen la Enfermedad.

Tabla 1.5. Incidencia de la Enfermedad de Parkinson en la Provincia del Azuay durante el año 2016
Fuente: [125]

Incidencia de la Enfermedad de Parkinson en la Provincia del Azuay -2016								
Morbilidad						Mortalidad		
Sexo		Edad (años)			Total	Sexo		Total
M	F	<50	50-65	>65		M	F	
63.22%	36.77%	6,62%	36,36%	57,02%	242	63.15%	36.84%	19

La figura 1.13 presenta una representación gráfica de la Incidencia de EP en la provincia del Azuay, teniendo en cuenta que cantidad de personas con relación a la edad poseen la enfermedad. Como es posible observar el 57,02% representa a personas mayores de 65 años que tienen EP, el 36,36% está entre los 50 y 65 años, mientras que el 6,62% de la población es menor a 50 años.

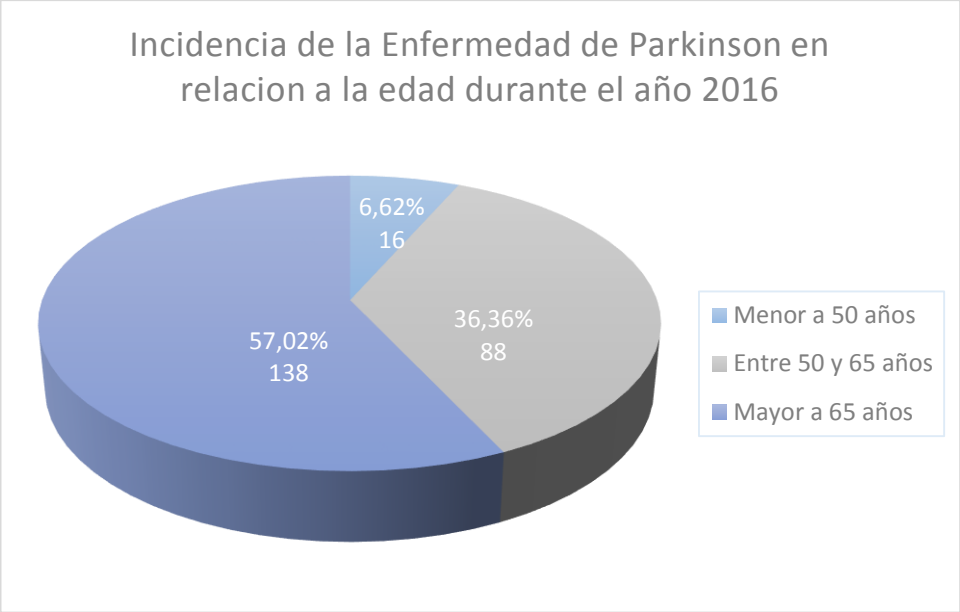
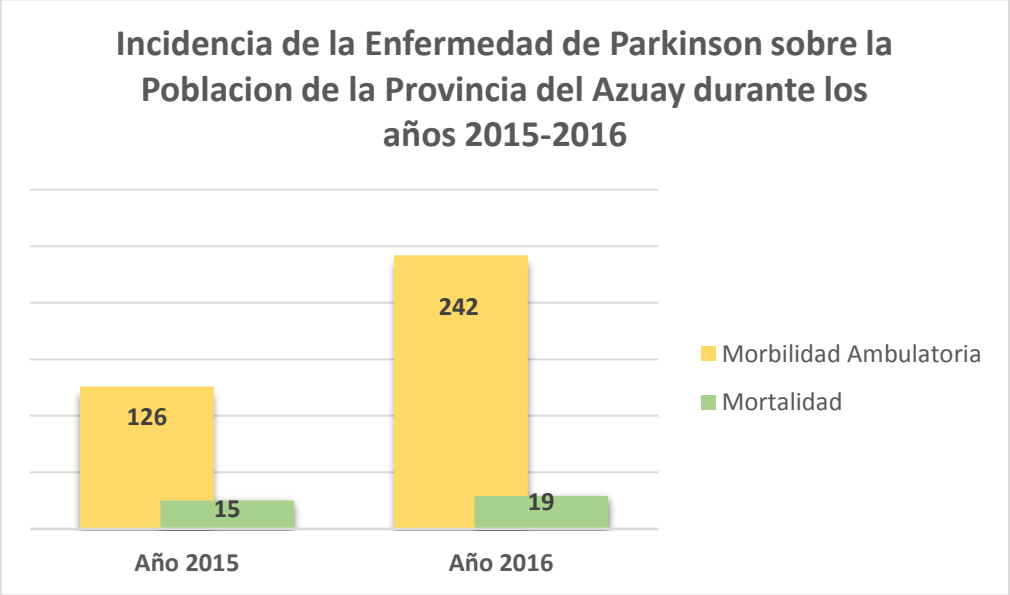


Figura 1.13 *Representación del número de personas con EP en relación con la edad*
Fuente: *Autor*

Teniendo en cuenta la incidencia de EP durante los años 2015-2016, en la Tabla 1.6 se presenta una comparación del número de personas con casos de morbilidad y mortalidad, en donde se puede observar que en el año 2017 el número de casos de morbilidad ambulatoria aumenta con 116 personas, mientras que el número de fallecidos aumenta a 4.

Tabla 1.6. *Comparación de los casos de Incidencia de EP durante los años 2015-2016*



El censo realizado en Ecuador en el año 2010 [126], indica que la provincia del Azuay cuenta con una población de 712.127 habitantes, dentro de la cual 375.083 son mujeres y 337.044 son hombres. Por lo que durante el año 2016 la Enfermedad de Parkinson represento el 0,33 ‰ de la población, es decir, 33.28 personas de cada 100mil habitantes de la provincia del Azuay poseen la Enfermedad.

Capítulo 2 : MARCO METODOLÓGICO

Este capítulo aborda todo el proceso de desarrollo del dispositivo, esquemas, diseños y programación que implica el sistema propuesto. Como primer punto se presenta el diseño del sistema, construcción del dispositivo y la programación para la respectiva comunicación Bluetooth, mientras que la segunda parte se realiza el diseño y programación del videojuego.

2.1 DISEÑO DEL SISTEMA

En la Figura 2.1 se presenta el diagrama de bloques del sistema, el cual consiste en cuatro etapas: Detección y Adquisición de datos, Transmisión de Datos, Mini Juegos y Análisis de Resultados.

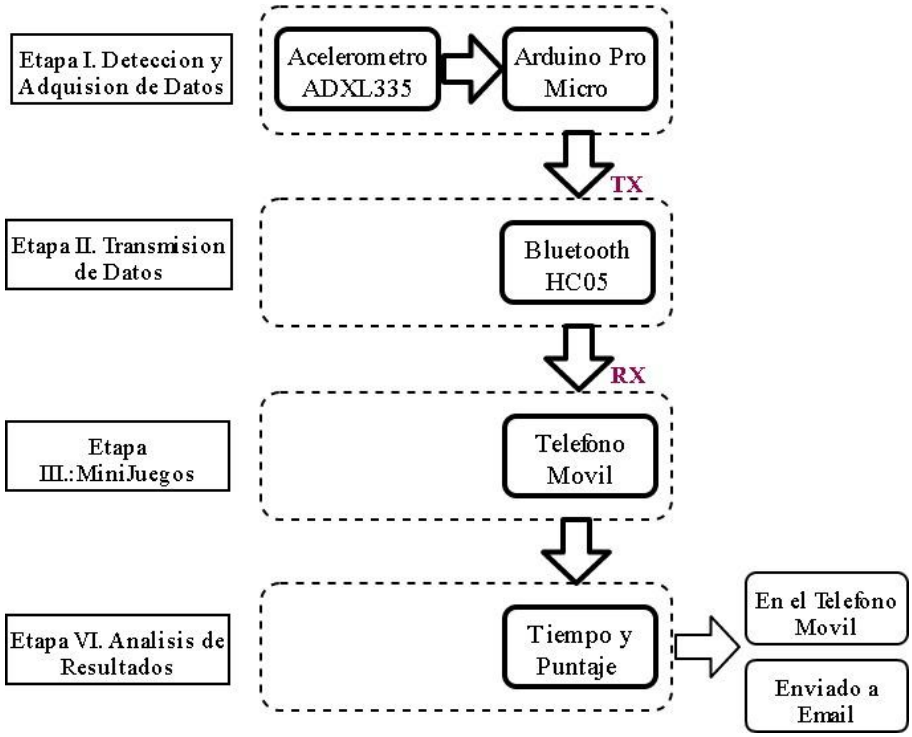


Figura 2.1 *Diagrama de Bloques del Sistema Propuesto*
Fuente: *Autor*

En base al diagrama expuesto anteriormente, en la Figura 2.2 se presenta el diagrama de funcionamiento del sistema, el cual muestra el proceso de rehabilitación propuesto para cada paciente.

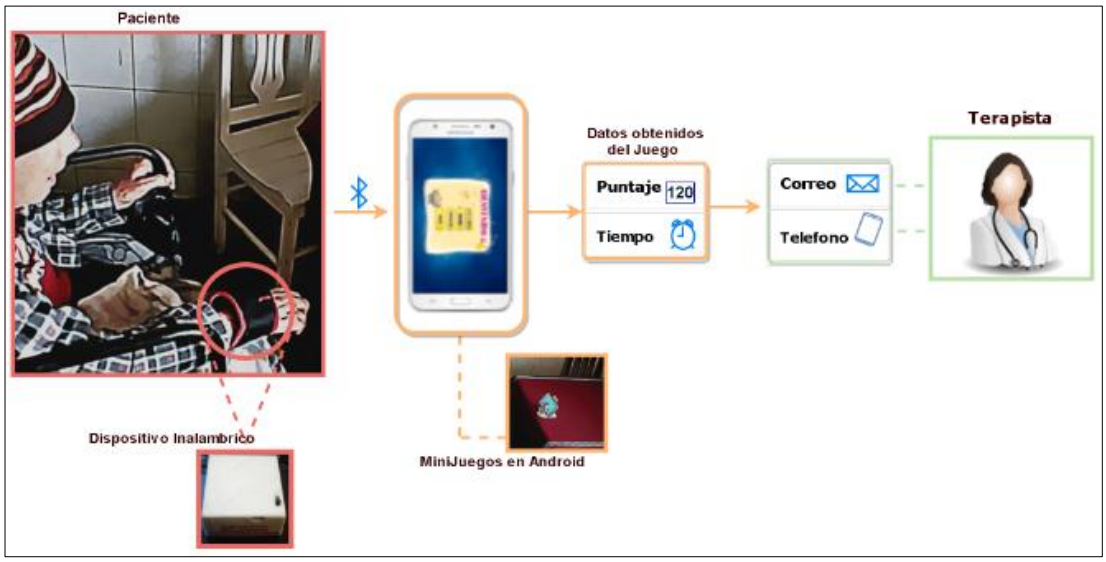


Figura 2.2. *Diagrama de Funcionamiento del Sistema*
Fuente: ***Autor***

2.2 DISEÑO DEL DISPOSITIVO INALAMBRICO

En este apartado se realiza el diseño del dispositivo, para ello se dispone del diagrama presentado en la Figura 2.3, el mismo que posee los elementos que intervienen en el dispositivo, como son: Modulo Bluetooth HC-05, Modulo Cargador de Batería TP4056, Batería LION de 3.7V, Arduino Pro Micro y un Acelerómetro ADXL335. Para el diseño del dispositivo, se tiene presente las características de cada elemento para evitar algún cortocircuito o daño durante su implementación.

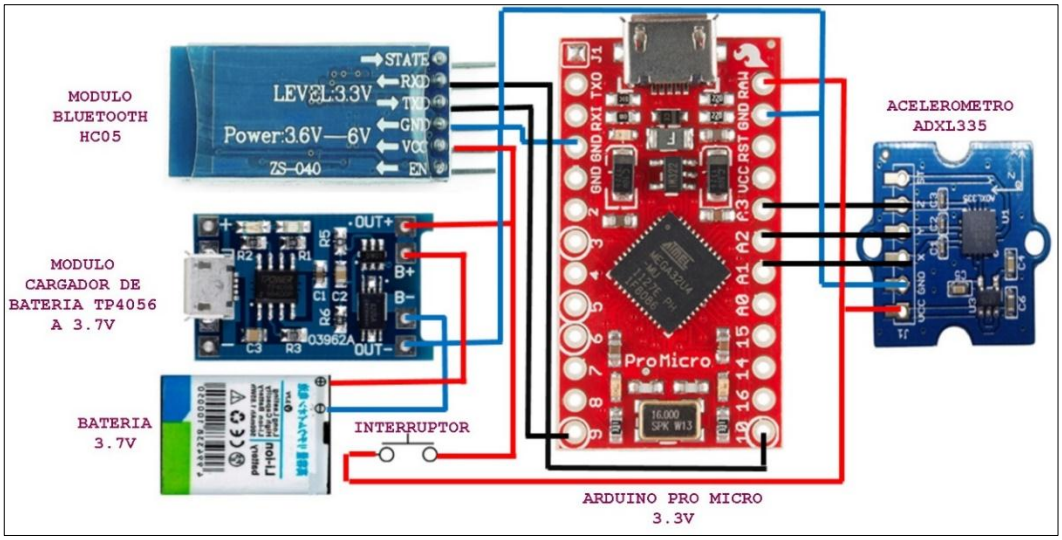


Figura 2.3. *Diagrama de Conexión de los Elementos que conforman el dispositivo*
Fuente: *Autor*

2.2.1 DISEÑO DEL CIRCUITO IMPRESO

El diseño del circuito impreso se lo realiza en Proteus 8.6, ya que es un software con una interfaz sencilla de operar. Una vez dispuesto del diseño se procede a elaborar la placa y luego a soldar cada componente de forma manual. La Figura 2.4 muestra el diseño realizado en Proteus junto con el diagrama de pistas de la placa.

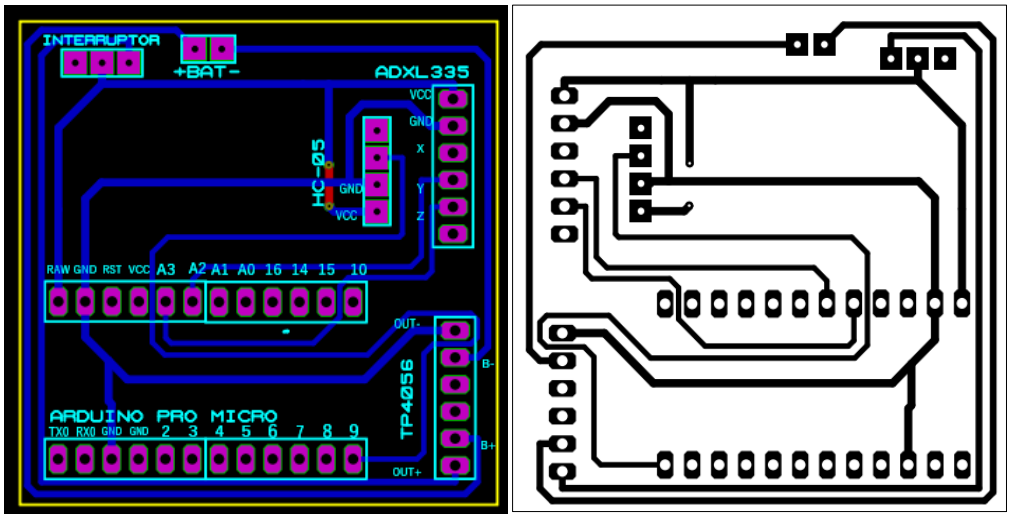


Figura 2.4 *Diagrama de Pistas de la Placa realizada en Proteus 8.6*
Fuente: *Autor*

En la Figura 2.5 se muestra la placa impresa la misma que posee un tamaño aproximado de 4.7cm x 4.9cm.

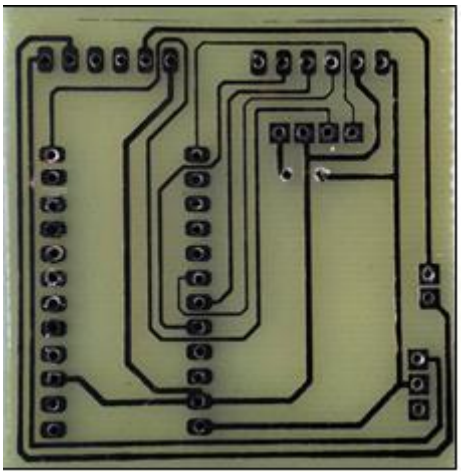


Figura 2.5 *Placa Impresa para el Dispositivo*
Fuente: *Autor*

2.2.2 DISEÑO DE LA ESTRUCTURA PROTECTORA DEL CIRCUITO

Teniendo el circuito impreso se realizó una estructura para colocar la placa y así otorgar mayor seguridad. La estructura se diseña en el software 3DMax y se imprime en una impresora 3D, debido a que no existen carcasas de la medida establecida. En las Figuras 2.6 y 2.7 se muestran el diseño de la carcasa en diferentes vistas

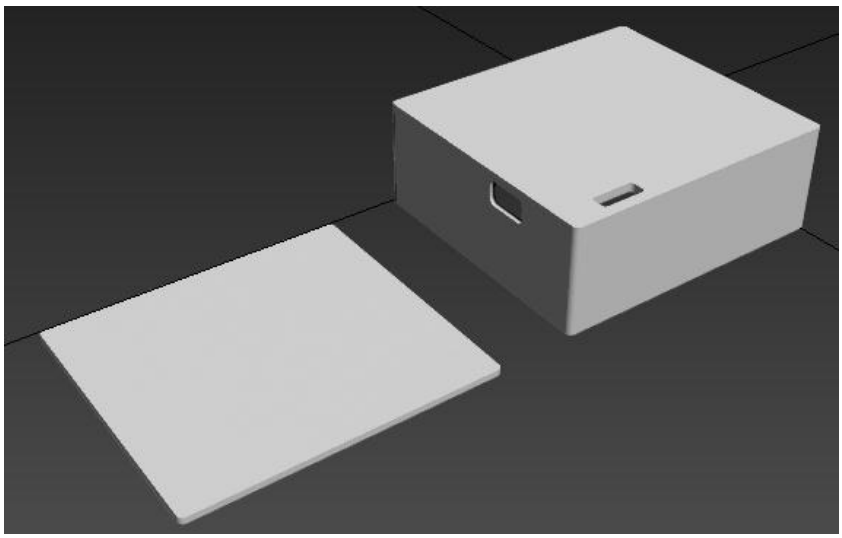


Figura 2.6 *Diseño del Protector del Dispositivo realizado en 3DMax*
Fuente: *Autor*

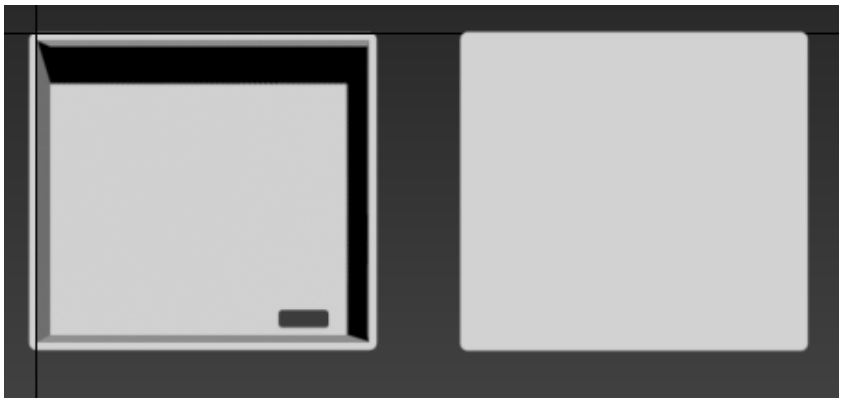


Figura 2.7 *Vista Superior del Protector para el dispositivo realizado en 3DMax*
Fuente: *Autor*

El protector impreso en 3D es de material PLA color blanco con tapa posterior color rojo, cuyas dimensiones son: 2.1cm x 4.7cm x 5.1cm, en donde se ha colocado en la parte delantera un adhesivo, entonces, teniendo la terminada la carcasa protectora se procede a colocar y tapar el circuito PCB, dando como resultado, la imagen que se puede observar en la Figura 2.8

Figura 2.8 *Colocación de la Placa Impresa dentro de la Estructura Protectora*
Fuente: *Autor*

De la misma manera para otorgar mayor protección y sobre todo comodidad al paciente, se ha diseñado una muñequera para colocar el dispositivo, la misma que se muestra en la Figura 2.9. El material que contiene es Nylon color negro debido a su elasticidad y durabilidad, posee bordes de color rojo para otorgar un toque de distinción, además en el interior se ha colocado algodón y en su extremo derecho se ubica la parte rugosa de dos pedazos de velcro, mientras que en la parte izquierda se coloca la parte lisa.

a)

b)

Figura 2.9 *Muñequera diseñada. a) Parte posterior. B) Parte superior*
Fuente: *Autor*

2.3 ETAPA DE DETECCION Y ADQUISICION DE DATOS

Esta etapa consiste en detectar los datos de cada participante por medio del acelerómetro colocado en la muñeca, por lo que cada participante deberá ejecutar dos tipos de movimientos*: pronosupinación y flexo extensión*, lo que implica que durante los movimientos de pronosupinación se trabajará con los datos del eje x del acelerómetro ADXL335, mientras que el movimiento de flexo extensión permite trabajar con datos del eje z. La Figura 2.10 presenta la etapa de adquisición de datos, en donde el dispositivo es ubicado en la muñeca del participante, cuyas fechas indican los movimientos que deberá ejecutar el paciente durante el videojuego.

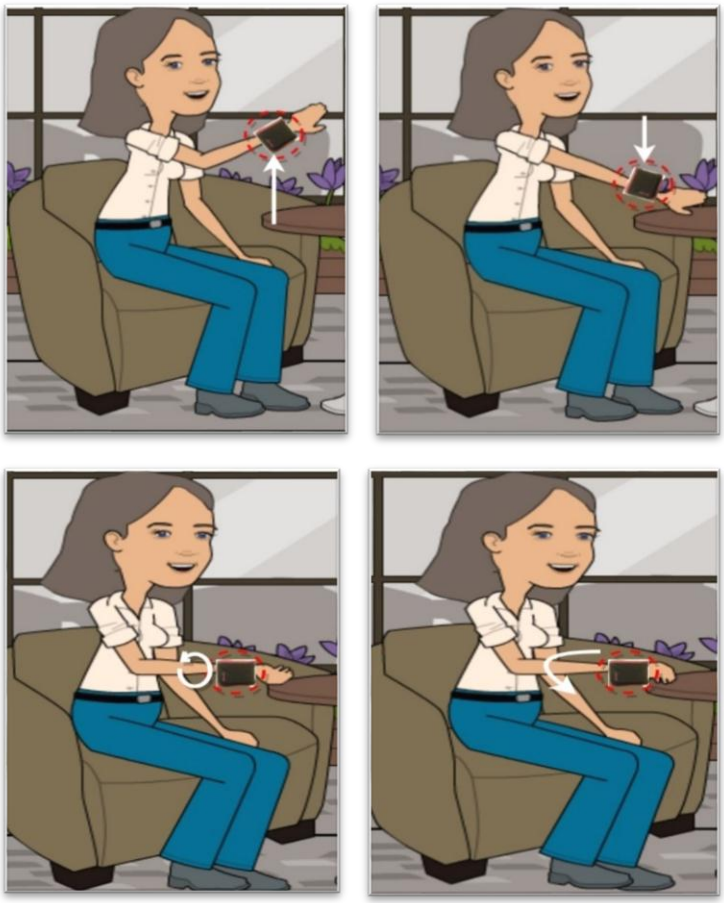


Figura 2.10 *Colocación del dispositivo para adquisición de datos.*
Fuente: *Autor*

La programación respectiva es realizada en Arduino usando el módulo Pro Micro de la marca SparkFun por lo que antes de iniciar, es necesario instalar las respectiva librería que esta disponibles en [104], de esta manera, se procede con la programación indicando las variables de los ejes del acelerómetro.

2.4 ETAPA DE TRANSMISION DE DATOS

Los datos adquiridos son calibrados de acuerdo con la posición del acelerómetro en donde los valores obtenidos están entre 317 a 450, luego estos datos son normalizados de -1 a 1 para enviar al software *Unity* usando el módulo Bluetooth HC05. La figura 2.11 muestra la posición del dispositivo para la transmisiòn de datos.

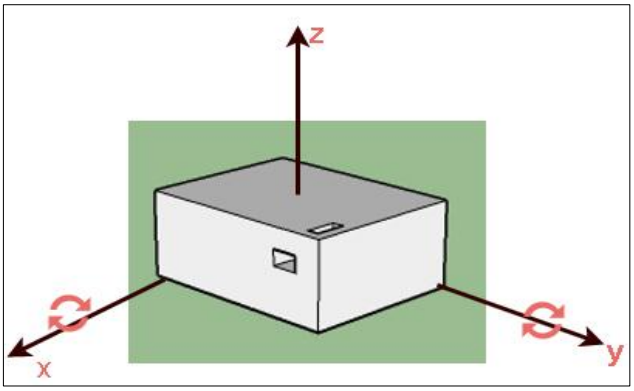


Figura 2.11 *Posición del dispositivo para la transmisiòn de datos en base al movimiento de los ejes x, y del acelerómetro ADXL335*
Fuente: *Autor*

Arduino no posee la condición de hacer un mapeo de valores float, sin embargo, se ha utilizado la función matemática disponible en [127], de esta manera los valores son mapeados y enviados al proceso de recepcion de datos.

2.5 DISEÑO, DESARROLLO Y PROGRAMACION DEL VIDEOJUEGO

En este apartado se presenta el desarrollo del videojuego para los pacientes con EP, el mismo que consiste en dos mini juegos: *PlaneGame* y *BallGame.* Para ello se usa el software Unity ya que permite exportar a plataformas Android.

2.5.1 MINI JUEGO PLANEGAME

2.5.1.1 Descripción

La idea para el diseño se basó en simular escenas reales en las que cada participante se sienta identificado, además de tener presente ciertas condiciones de diseño de videojuegos para la rehabilitación presentadas en el capítulo uno.

El juego *PlaneGame* consiste en tres escenas en donde el participante debe guiar al avión hacia una meta sin tocar los elementos del terreno para evitar perder el juego, por lo que en el trayecto existen flechas indicadoras que cambian de color y otra flecha que se mueve con el avión, las mismas que indican la meta y muestran la posición de los objetivos respectivamente.

Los niveles de dificultad consisten en el incremento del número de objetivos y el aumento de la velocidad del avión. Los objetivos que otorgan puntaje se ubican a lo largo del camino y consisten en figuras geométricas diseñados en 3DMax, como

anillos, triángulos y cuadrados. Por otra parte, el enfoque del juego consiste en otros elementos que son importantes para la distracción de los participantes, como se indican a continuación:

- *Música,* la música de PlaneGame está basada en la preferencia de la mayoría de adultos de edad mayor que eligen música de tipo romántica relajante, por lo que posee temas de la época de los 80, autores como Sandro, Juan Gabriel, etc.
- *Ambiente,* el mini juego PlaneGame incluye tres niveles, el primer nivel es una zona montañosa. El segundo nivel es el centro de una ciudad y el tercer nivel, el espacio.
- *Recompensas,* cada nivel de juego posee estímulos, como regalos, mensajes de apoyo y sonidos, para que el participante se anime a continuar el juego.
- *Controles,* el avión ejerce dos movimientos hacia la derecha e izquierda, por lo que el participante deberá ejecutar el movimiento de pronosupinación de antebrazo, usando el dispositivo inalámbrico desarrollado anteriormente.

También se ha creado un mini mapa para tener una referencia del recorrido de juego, además se ha diseñado una barra de energía que indica el nivel de gasolina del avión y se establece un marcador de puntos y el tiempo en el juego.

La Figura 2.12 presenta la vista frontal del primer nivel de PlaneGame, la misma que contiene los botones de pausa y salir, el marcador de tiempo y puntaje, la barra de energía y el mini mapa del juego.

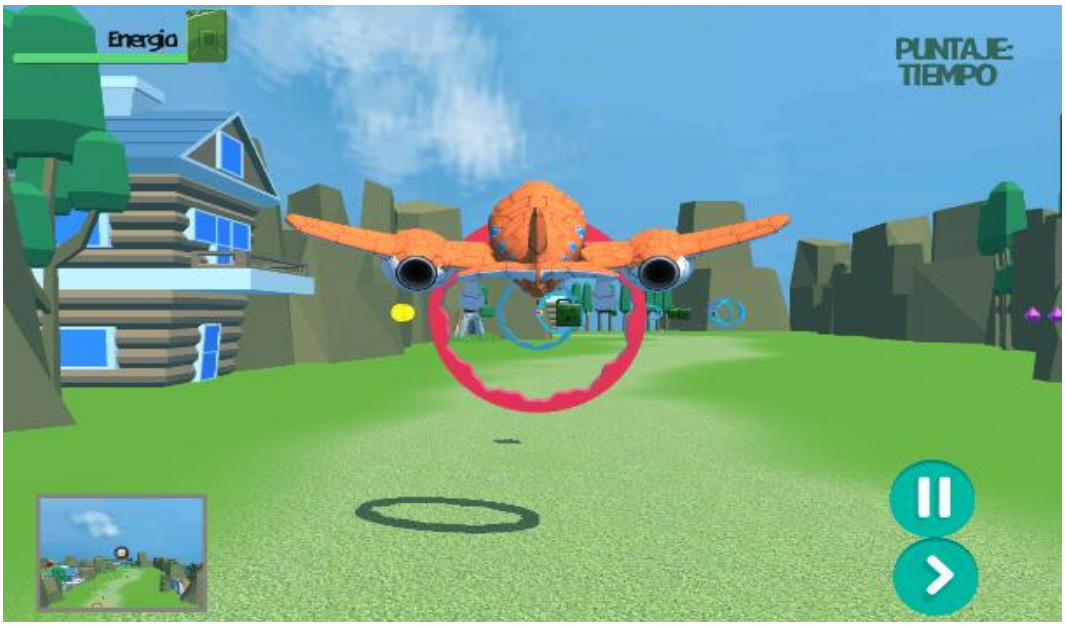


Figura 2.12 *Pantalla del Primer Nivel del mini juego PlaneGame en Unity*
Fuente: *Autor*

De la misma manera en la figura 2.13 se muestra la vista frontal del segundo nivel de PlaneGame, mientras que en la figura 2.15 se puede apreciar la pantalla del tercer nivel que incluye elementos del espacio exterior.

Figura 2.13 *Pantalla del Segundo Nivel del mini juego PlaneGame en Unity*
Fuente: ***Autor***

Figura 2.14 *Pantalla del Tercer Nivel del mini juego PlaneGame en Unity*
Fuente: ***Autor***

2.5.1.2 Diseño de Escenarios

Cada nivel de *PlaneGame* se denomina escenario, y todos los elementos que conforma la escena son conocidos como GameObjects en donde la mayoría son obtenidos del Asset Store de Unity[128], así que el primer nivel hace relación a un escenario montañoso, en donde se ha creado un terreno con un material color verde en cuyo camino se ha colocado una textura color crema. Posee montañas rocosas, arboles, casas de campo y los objetivos están a lo largo del terreno, mientras que la meta posee un plano de estrellas al final del camino, tal como se observa en la Figura 2.15

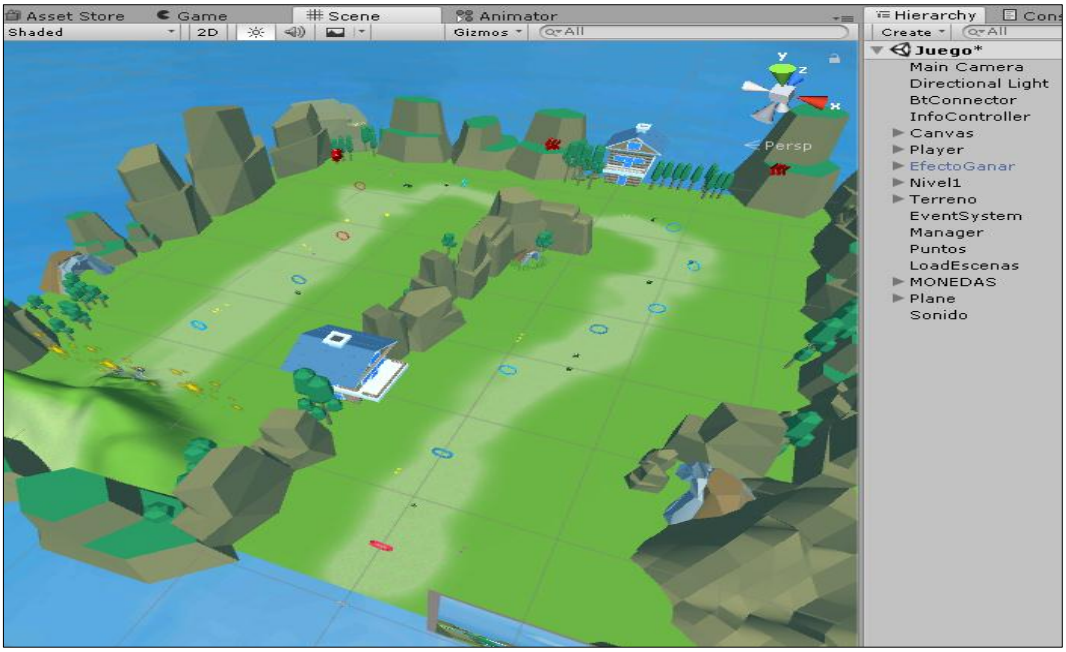


Figura 2.15 *Escena del Nivel Uno de PlaneGame en el Editor de Unity*
Fuente: *Autor*

El nivel dos tiene como nombre de escena JuegoNivel2 y contiene un terreno con material de carretera, un rio, edificios, casas, tiendas. Tal como se observa en la figura 2.16.

Figura 2.16 *Escena del Nivel dos de PlaneGame en el Editor de Unity*
Fuente: *Autor*

En la figura 2.17 se indica el entorno virtual del nivel 3 de PlaneGame el cual se compone de elementos del espacio como: asteroides, planetas, galaxias y dos planos laterales que indican el camino que debe seguir el jugador.

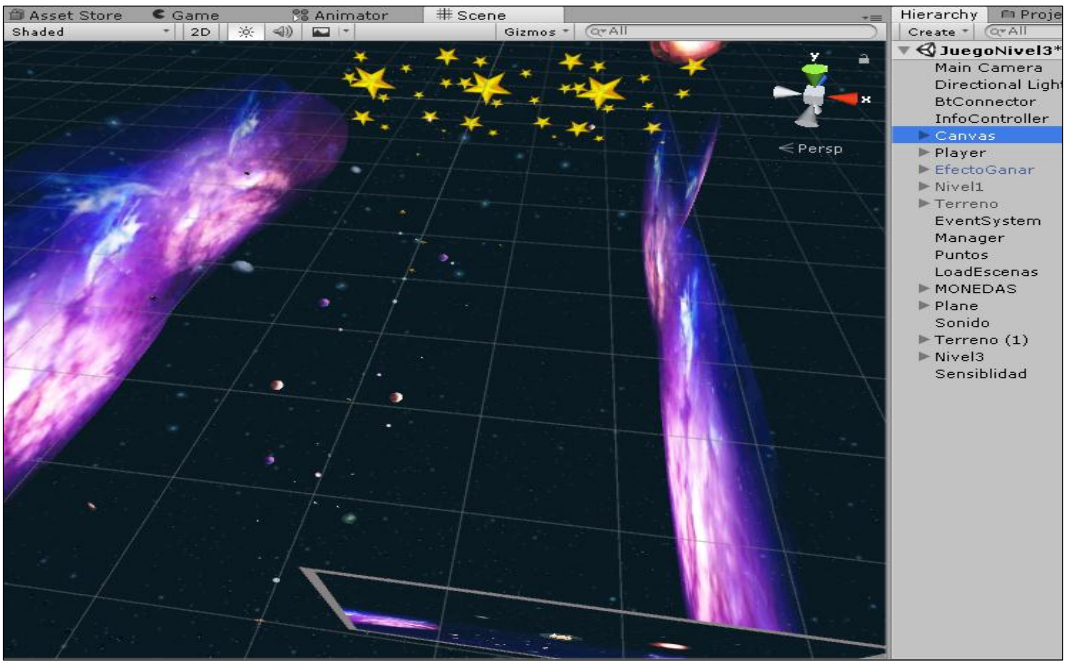


Figura 2.17 *Escena del Nivel tres de PlaneGame en el Editor de Unity*
Fuente: ***Autor***

2.5.1.3 Programación

Cada acción que se ejecuta en el juego debe tener su respectivo script, es decir su propia programación, la misma que usa lenguaje C-Sharp en Visual Studio.

Objetivos

Los objetivos del juego poseen el script *Anillos.cs* que permite que ciertos métodos se ejecuten durante el juego, cuando el avión atraviesa cada objetivo se activa un sonido mediante la función de Unity *AudioSource* y al instante anota puntos en el marcador controlado por el script *Puntos.cs.* De la misma manera, los objetivos están dentro de un GameObject denominado Nivel1, en donde el script *Objetivos.cs.* permite tener el conteo de los objetivos que son colisionados con el avión y así otorgar un cambio de color al siguiente anillo. La Figura 2.18 muestra todos los elementos mencionados anteriormente.

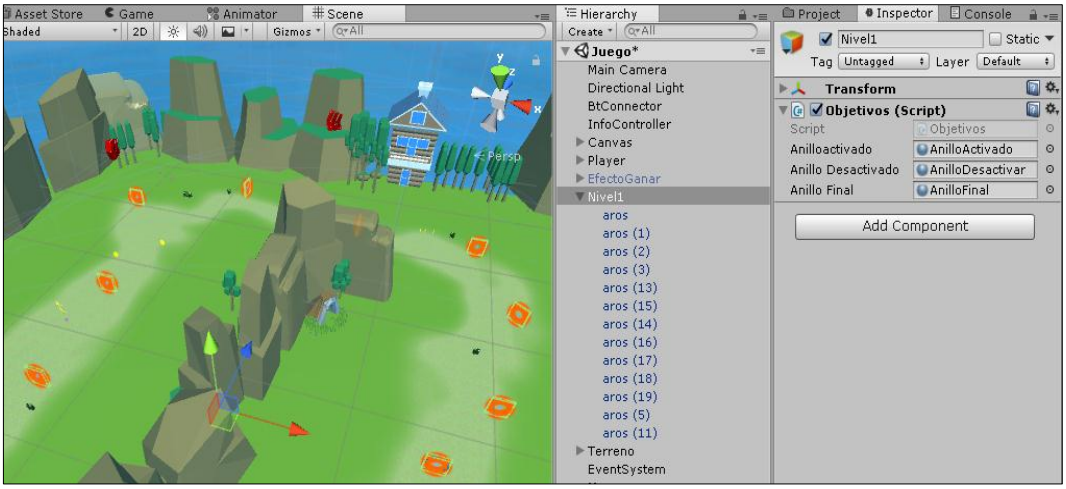


Figura 2.18 *Objetivos de PlaneGame dentro de la Arquitectura de juego*
Fuente: *Autor*

Además, cada objetivo tiene un Box Collider con la opción de trigger activado para que se ejecute la respectiva colisión, aparte se ha realizado una animación usando la herramienta de Unity *Animation* en donde el objetivo rota 360º. Finalmente, los objetivos se relacionan también con el Script *ObjetosMiniMapaJuego.cs* que permite que cada objetivo aparezca en el mini mapa. En la Figura 2.19 se muestra todos los componentes que poseen los objetivos del juego.

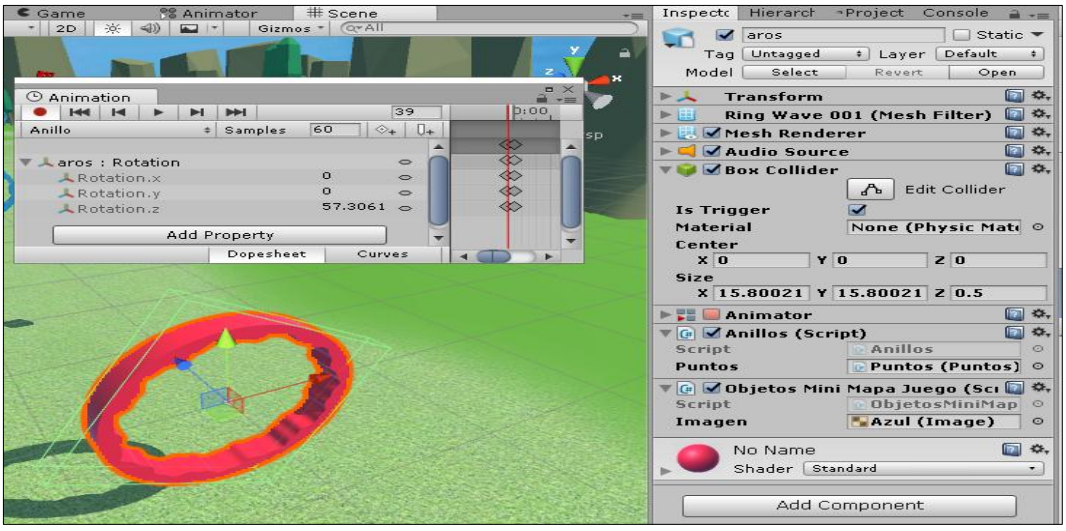


Figura 2.19 *Componentes usados para los objetivos del juego.*
Fuente: *Autor*

Elementos del Juego

Los elementos del juego son aquellos que conforman el terreno, por lo que cada uno de ellos dispone de un Box Collider y cada vez que el avión toque estos elementos, el juego termina. Véase la Figura 2.20

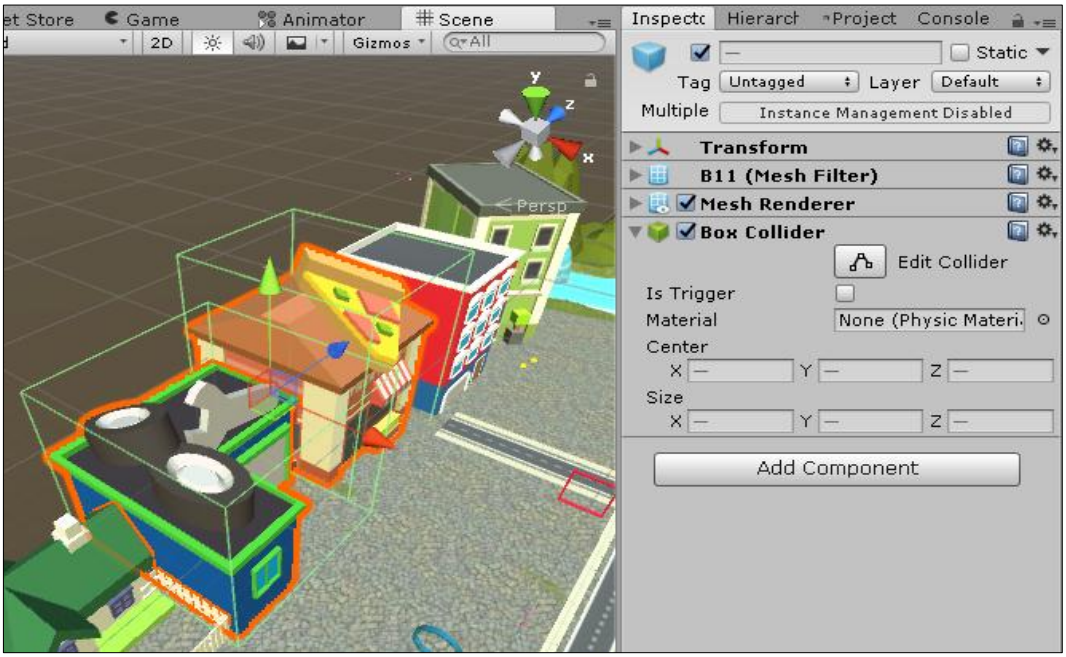


Figura 2.20 *Elementos que conforman las escenas, ejemplo de la Escena 2*
Fuente: *Autor*

Cámara Principal

La cámara principal permite visualizar el juego mientras el avión se está moviendo, en este GameObject se ha declarado la música de fondo para cada nivel usando AudioSource y activando la opción *Loop* y *Play Awake* para que la música este presente durante el juego sin interrupciones que no son declaradas, el script *PlayerCamara.cs* permite que la cámara siga al avión durante el recorrido, por lo que necesita conocer la ubicación del avión para crear un vector con la nueva posición.

Plano Ganar

El plano de estrellas que indican la meta está constituido por componentes Box Collider con la opción Trigger activada y algunas animaciones, ya que el avión cuando toque tal plano ganara el juego. Todo el componente denominado *Plano Ganar* dispone de un tag que se ha creado especialmente para ser utilizado en el

script del avión. La figura 2.21muestra los componentes que posee el GameObject PlanoGanar dentro de la Arquitectura de PlaneGame

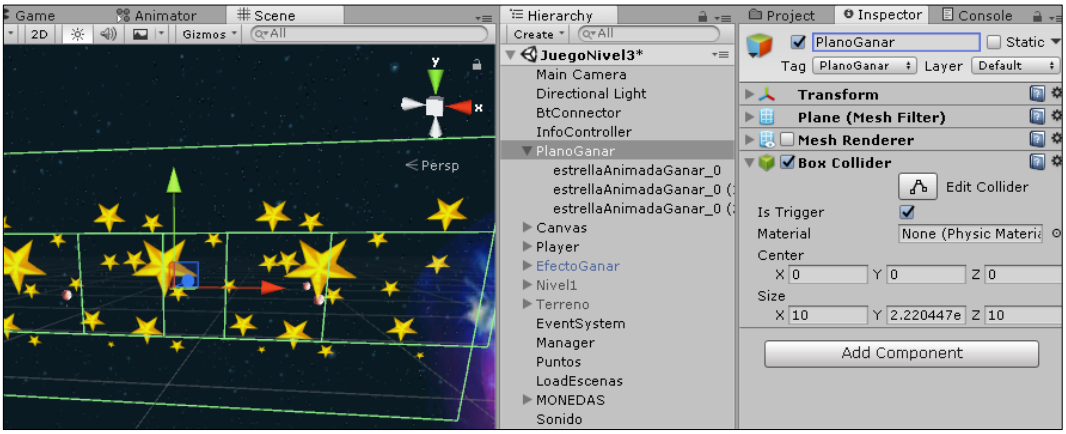


Figura 2.21 *Componentes en la ventana Inspector de la meta.*
Fuente: *Autor*

Mini Mapa

El mini mapa creado es para guiar al participante durante el juego, mostrando los objetivos y el camino para llegar a la meta. Para identificar al avión en el mini mapa se ha colocado una imagen de forma circular en el centro, mientras que cada objetivo posee un cuadrado de color azul.

Se crea un nuevo GameObject denominado Mini mapa que tiene el componente *RawImage* de Unity, que permite crear iconos o imágenes de manera interactiva los mismos que van ir variando de acuerdo con el movimiento del avión.

Ítems

Durante el trayecto de cada nivel se encuentran elementos que otorgan puntos extras en el juego, los cuales están compuestos por diamantes, monedas, estrellas y un tarro de gasolina para el avión. Cada ítem posee un Box Collider para que el avión colisione y se active un sonido. Véase la Figura 2.22

El script asociado a este componente se denomina *Tokens.cs*, en el cual se declaran todas las variables que permitan detectar que el avión toco los ítems y otorga 20 puntos extras al puntaje final.

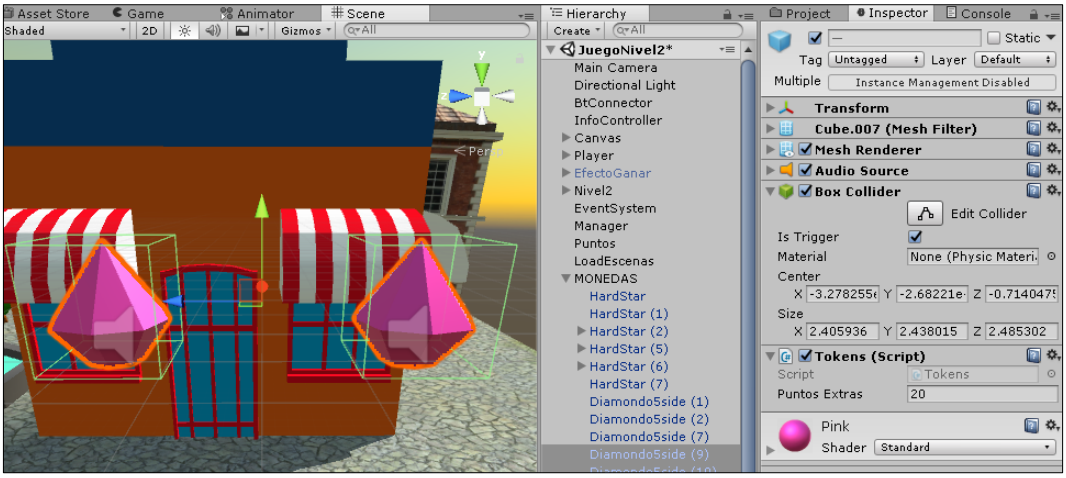


Figura 2.22 *Ítems dentro de la escena del juego de PlaneGame*
Fuente: *Autor*

Barra de Energía

La barra de energía indica la cantidad de gasolina que dispone el avión para continuar en el juego, y conforme avanza el tiempo de juego la barra va disminuyendo en un 2%, por lo que es necesario recoger tarros de gasolina. La barra de energía está formada por un Slider color verde.

El script que controla la barra de energía se encuentra en *PlayerMotor.cs* mientras que el script que permite recargar la barra de energía es *EnergiaGasolina.cs* en donde cada ítem que recoge aumenta en 30% a la barra de energía. En la Figura 2.23 se puede ver los componentes asociados a cada tarro de gasolina.

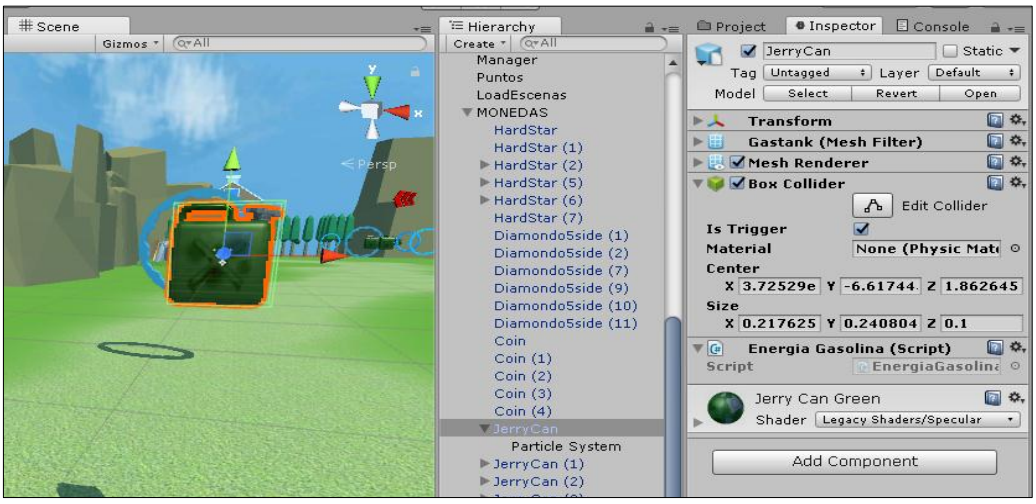


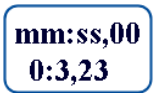
Figura 2.23 *Componentes que conforman el tarro de gasolina.*
Fuente: *Autor*

Flecha Indicadora

La flecha indicadora está localizada junto al avión y permite mostrar la ubicación de cada objetivo del juego, si el jugador alcanza un objetivo la flecha indica la dirección ya sea a la izquierda o a la derecha del siguiente objetivo. De la misma manera el script que permite realizar esta acción está asociado a *PlayerMotor.cs* el cual controla al avión.

Tiempo y Puntaje

El tiempo de juego tiene lugar en el script *Timer.cs*, en donde se declara una variable de tipo flotante que indicara el tiempo y utiliza la función *Time.time* de Unity que permite medir el tiempo del juego en segundos. Los valores de tipo flotante son transformados a string para ser mostrados en la pantalla del juego usando el método ToString, el cual devuelve una representación de cadena de la variable tipo float. El valor de tiempo es representado en minutos y fracciones de segundos teniendo la siguiente forma:



El GameObject de Puntaje se verá afectado cuando atraviese cada obstáculo otorgando 20 puntos en el marcador, el script que permite ese proceso es Puntos.cs.

2.5.1.4 Recepcion de Datos y Movimiento del Avión

Para la recepcion de datos se utiliza la interfaz Bluetooth del dispositivo móvil, sin embargo se utilizó una librería disponible en Asset Store de Unity [130] que permitirá la comunicación bluetooth desde el dispositivo inalámbrico y el Teléfono móvil, para ello se han modificado ciertos parámetros de tal librería para que se adapte a nuestro sistema.

Por lo que, se presenta la librería [130] que permite realizar la comunicación inalámbrica con el dispositivo y contiene los métodos principales como: conectar dispositivo, desconectar, mostrar dispositivos y el método leer. Este último permite

leer los datos enviados desde el módulo Bluetooth HC05 y los guarda en una variable.

Luego se ha creado un script Manager.cs que usa la variable creada anteriormente para separar y comprobar los valores enviados desde el Bluetooth HC-05, en nuestro caso se están enviado tres datos x, y, z. Finalmente se conserva el estado de la variable de tipo float usando el método floatParse para que la variable guarde la coma por lo que en este método se obtienen ya las variables que serán usadas en todos los scripts

Teniendo en consideración estos eventos, se procede a realizar la respectiva programación para el jugador, que en este caso es el avión. El script que controla el avión se denomina *PlayerMotor.cs*, en primer lugar, se ha declarado la variable de tipo float para invocar los datos del script *Manager.cs*, luego se selecciona solo el valor de x para permitir el movimiento dentro del juego.

También se declaran variables de velocidad hacia adelante y la velocidad de rotación en x, este script es uno de los más importantes ya que se han declarados varios eventos que suceden en el desarrollo del juego, como son: ganar, perder, sonidos, ítems, etc.

2.5.2 MINI JUEGO BALLGAME

2.5.2.1 Descripción

El juego *BallGame* contiene dos escenas en donde el participante deberá mover una pelota hacia arriba y hacia abajo evitando caer a los terrenos propuestos y chocar con bloques en la parte superior, para lograr alcanzar los objetivos, los mismos que se basan en figuras geométricas ubicados a lo largo del camino recto.

De igual manera que PlaneGame, el juego de la pelota BallGame contiene la misma preferencia de música, en donde el ambiente de juego se centra en dos niveles: El primer nivel es un entorno de lava cuya meta es llegar a un volcán, mientras que el segundo nivel es una zona de mar.

2.5.2.2 Diseño de Escenarios

El primer nivel posee eslabones creados con objetos 3D del editor de Unity y figuras geométricas distribuidas a lo largo de cada eslabón, mientras que la meta consiste en llegar a un volcán. La Figura 2.24 muestra la pantalla de la escena de juego y la Figura 2.25 muestra los componentes que conforman el mismo.

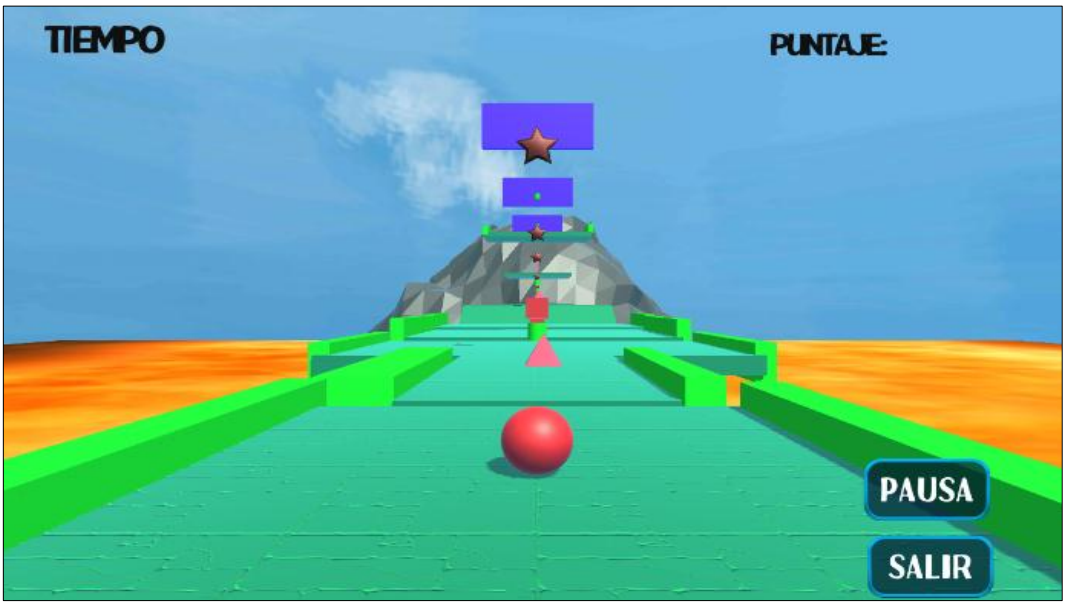


Figura 2.24 *Pantalla de Juego del primer nivel de BallGame*
Fuente: *Autor*

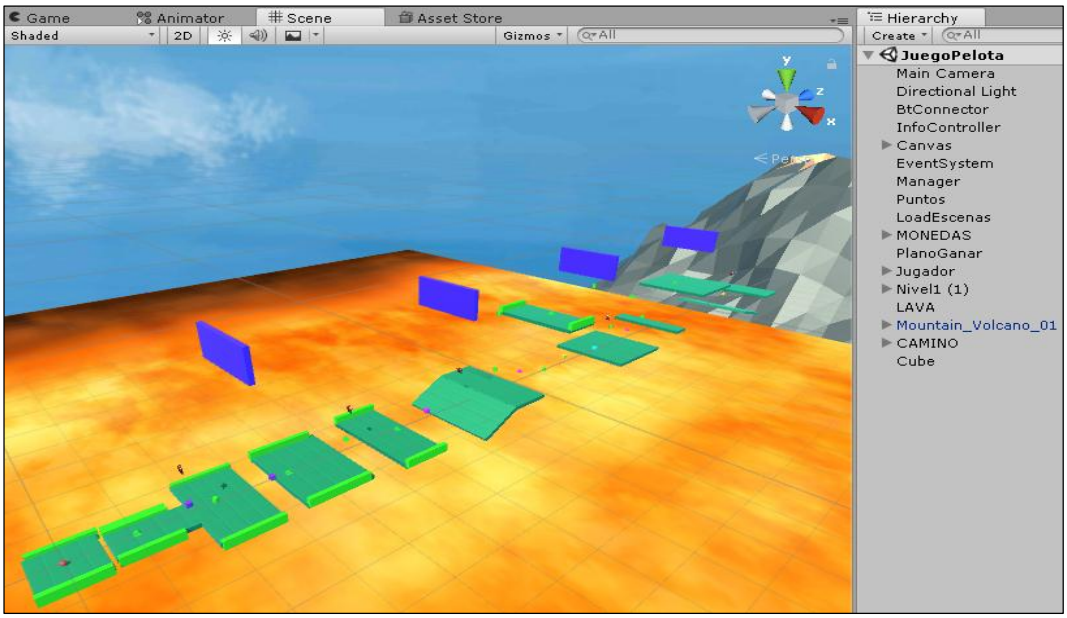


Figura 2.25 *Componentes de la escena de BallGame del primer nivel*
Fuente: *Autor*

De la misma manera, el segundo nivel posee eslabones con un ancho menor y mayor número de figuras geométricas y gemas distribuidas a lo largo de cada eslabón, mientras que la meta es una montaña rocosa. En las Figuras 2.26 y 2.27 se puede observar la pantalla de juego y el entorno.

Figura 2.26 *Pantalla de Juego del nivel 2 de BallGame*
Fuente: *Autor*

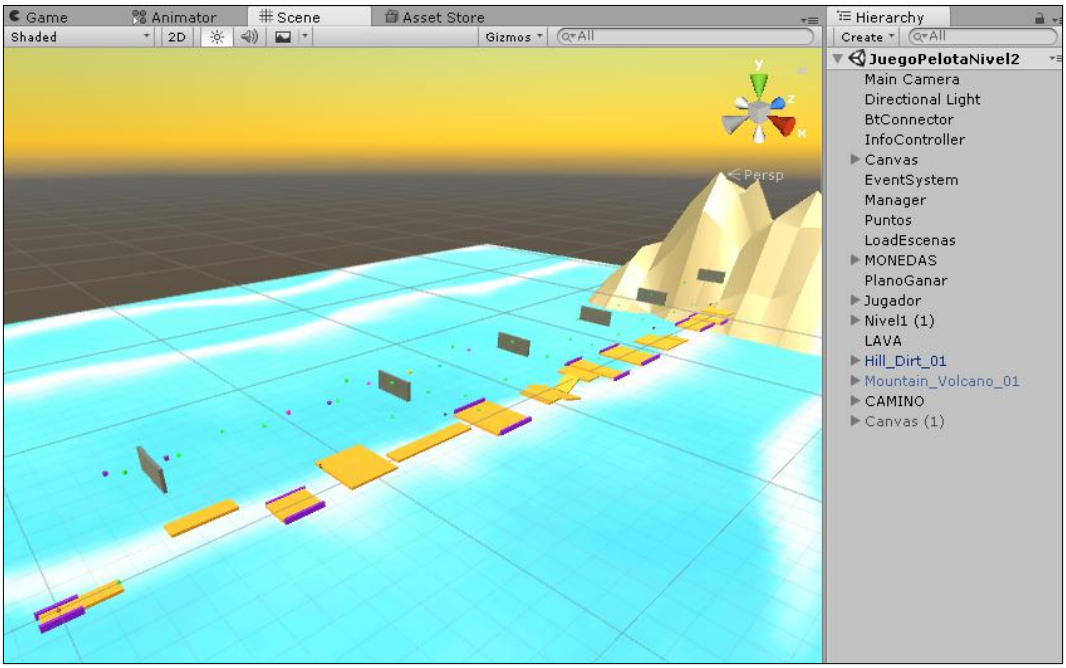


Figura 2.27 *Componentes de la escena del nivel 2 de BallGame*
Fuente: *Autor*

2.5.2.3 Programación

La programación de cada objeto del juego se realiza mediante lenguaje C-Sharp en Visual Studio.

Objetivos

Los objetivos que son las figuras geométricas y las gemas poseen un Box Collider que permite otorgar puntos cada vez que la pelota atraviese por ellos. Además, para reconocerlos en el juego, se ha creado un tag específico: Estrellaspelota y Collide, que serán llamadas en el script del jugador. La Figura 2.28 muestra los distintos componentes que conforman a los objetivos del juego

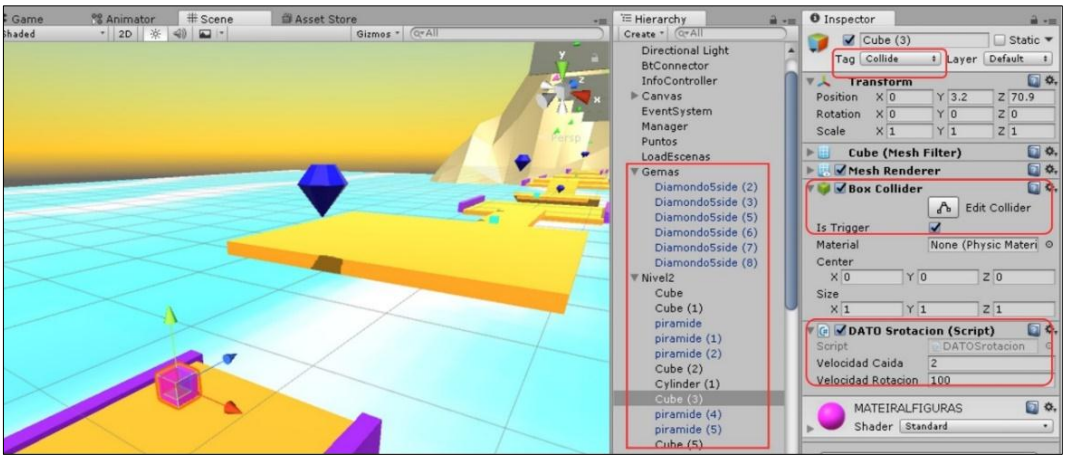


Figura 2.28 *Componentes de cada objetivo en BallGame*
Fuente: *Autor*

Cámara Principal

La cámara principal permite visualizar el juego cuando la pelota está en movimiento, por lo que en *Main Camera* se ha seleccionado la música de fondo para cada nivel usando AudioSource con la opción Loop y Play Awake además de añadir el Script que controla la cámara de la pelota.

2.5.2.4 Recepcion de Datos y Movimiento de la Pelota

Los datos que permiten el movimiento de la pelota parten del mismo proceso de recepcion para el avión, con la diferencia que en este caso se ha seleccionado la

variable y. Entonces se presenta el script *ControladorJugador.cs* que controla los diferentes eventos del juego como son: el método Ganar, Perder, MostrarPaneles, y las diferentes interacciones con los objetivos. Para el conteo de puntos, simplemente en el momento de interacción con los objetivos se realiza un conteo acumulativo, que se muestra en la pantalla de juego.

El jugador Pelota posee un componente denominado CharacterController que permite controlar al jugador para que realice las diferentes interacciones con los objetos del juego, cuando se cae, choca o recoge ítems. De igual manera se asigna un tag al a la pelota para identificarlo en las diferentes acciones del juego. Finalmente, el tiempo de juego es controlado de igual manera por el script *Timer.cs. y e*n la Figura 2.29 se muestra los componentes que posee el GameObject Jugador.

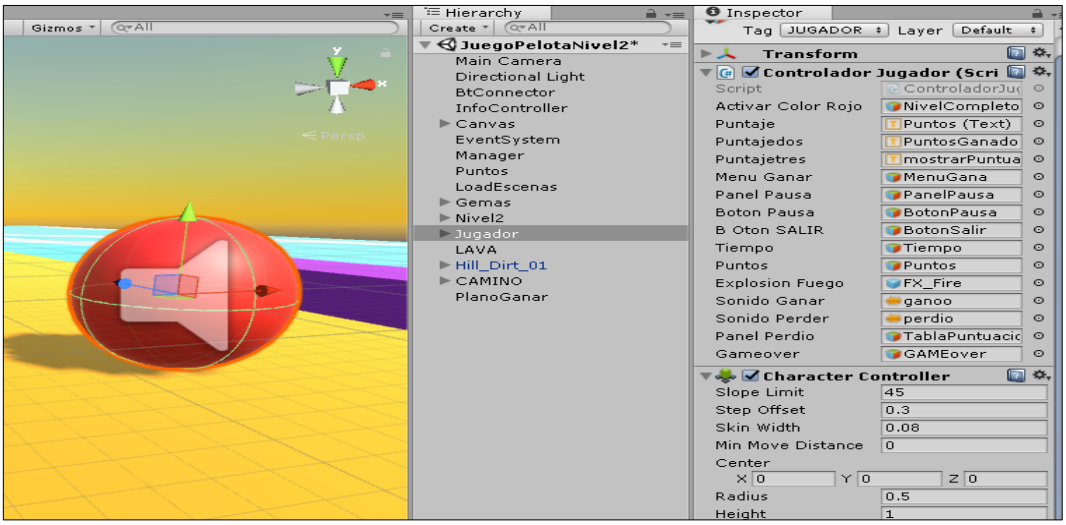


Figura 2.29 *Componentes de la pelota en BallGame*
Fuente: *Autor*

2.5.3 INTERFAZ DE USUARIO

La interfaz de usuario está compuesta por los entornos virtuales propios de cada juego que consisten en pantallas con menús, botones, imágenes, y algunos elementos que ayudan al usuario a acceder a distintas opciones y niveles del juego. La mayoría de botones fueron diseñados en Adobe Photoshop y otros obtenidos de forma gratuita de Internet.

Las pantallas utilizadas en el sistema se describen a continuación:

Menú de Conexión: Este menú está en localizado en la escena *BluetoothInfo* y permite enlazarse con el dispositivo inalámbrico, contiene cuatro botones y animaciones en los textos e imágenes. El botón Buscar localiza el dispositivo HC05, el botón Conectar permite que la conexión se efectué y no aparecerá mientras no haya pulsado el botón Buscar mientras que el botón Jugar se activa cuando el botón Conectar se haya pulsado y este dirige al menú de los mini juegos. Finalmente, el botón Salir permite dejar la aplicación. En la Figura 2.30 se muestra el menú de conexión

Figura 2.30 *Menú de Conexión con el dispositivo inalámbrico*
Fuente: *Autor*

Menú de Inicio: Está presente en la escena *BPanelJuego* incluye cinco botones que permiten iniciar sesión para aquellos participantes que deseen guardar los datos, el botón jugar, el botón parámetros terapéuticos, el botón opciones, el botón resultados de cada partida y la opción salir de la aplicación. La Figura 2.31 muestra este menú.

Figura 2.31 *Menú de Inicio de Juego*
Fuente: *Autor*

Menú de Inicio de Sesión: Esta pantalla contiene el inicio de sesión de cada paciente o participante, lo cual permitirá guardar sus datos de partida para visualizar en cualquier ocasión. Simplemente, debe ingresar un nombre de usuario o registrarse y dar clic en la opción iniciar para entrar en sesión y observar los datos de juego que se guarden, tal como se muestra en la Figura 2.32

Figura 2.32 Menú de Inicio de Juego
Fuente: *Autor*

Menú de Parámetros de Terapista Este menú se presenta para el terapeuta en donde deberá ingresar los tiempos para cada juego en general, lo que permitirá

tener una comparación a futuro de los tiempos ingresados por el terapeuta y los tiempos obtenidos por el jugador. Cabe recalcar que el tiempo ingresado debe estar en segundos, tal como se muestra en la Figura 2.33

Figura 2.33 *Menú de los Parámetros de Juego ingresados por el Terapeuta*
Fuente: *Autor*

Menú de Mini Juegos: Esta pantalla contiene la selección de nivel de cada mini juego, tal como se muestra en la Figura 2.34

Figura 2.34 *Menú de Selección de Mini Juegos*
Fuente: *Autor*

Menú de Resultados. Este menú permite conocer los resultados de cada sesión de juego, es decir existe la opción de crear un archivo .txt con los datos de

tiempo y puntaje de cada participante que haya iniciado una sesión de juego, y luego enviar por correo electrónico ya sea al terapeuta o al encargado de analizar tales resultados. Además, el botón Borrar Datos permite limpiar el archivo .txt cada vez que sea necesario. La figura 2.35 muestra la pantalla del menú de resultados.

Figura 2.35. *Menú de Resultados.*
Fuente: *Autor*

Menú de Opciones: Permite configurar el sonido y conocer las instrucciones del juego. Tal como se aprecia en la Figura 2.36

Figura 2.36 *Menú Opciones del Juego*
Fuente: *Autor*

Pantalla de carga: Muestra un texto de espera durante un tiempo corto mientras que el sistema inicializa a cada nivel. La inicialización del entorno virtual

requiere de tiempo para cargar en memoria todos los recursos de despliegue tales como los modelos tridimensionales, texturas y cualquier otro tipo de archivo requerido para crear al entorno. La figura 2.37 muestra esta descripción.

Figura 2.37 *Pantalla de carga para cada nivel.*
Fuente: *Autor*

Menú de pausa: Está presente en cada nivel del juego y permite al participante detener momentáneamente el flujo del juego y le proporciona las opciones para continuar, reiniciar o terminar la partida. Esta pantalla se muestra en la Figura 2.38

Figura 2.38 *Menú Pausa durante el juego*
Fuente: *Autor*

2.5.4 GUARDAR DATOS

El puntaje más alto obtenido por el participante se guarda en el teléfono mediante el script *LeaderBoard.cs*, el cual permite mostrar las 50 puntuaciones más altas con el nombre de jugador.

Cuando el jugador pierde la partida, se activa un panel con varias opciones de juego, la misma que se puede observar en la Figura 2.39, y contiene tres opciones, el botón Guardar permite almacenar los datos de esa partida en la base de datos del jugador que inicio sesión, el botón *Tabla de Puntuación* permite observar las puntuaciones más altas dentro del juego y el botón *Enviar correo* permitirá enviar los datos de juego obtenidos en ese nivel.

Mientras que en la Figura 2.40 se muestra el panel de las puntuaciones altas que se activa cuando se pulsa el botón antes indicado.

Figura 2.39 *Pantalla para Guardar la puntuación.*
Fuente: *Autor*

Figura 2.40 *Panel de la Tabla de Puntuación*
Fuente: *Autor*

Además de guardar las puntuaciones altas, como se indicó anteriormente cada participante que inicia una sesión de juego posee una base de datos, la cual permite guardar el valor de tiempo y puntaje de cada nivel ya que serán necesarios más para el análisis. La base de datos es obtenida del Asset Store de Unity [131] la cual fue modificada de acuerdo a las condiciones del juego.

El participante deberá tener una conexión a internet y luego tiene que ingresar un nombre de usuario y así continuar con la elección del nivel a jugar. Para obtener los datos de cada nivel, simplemente debe pulsar el botón Cargar Datos, que se encuentra en el panel de sesión. La Figura 2.41 muestra el panel de inicio de sesión, mientras que la Figura 2.42 indica la opción de registro para un nuevo participante.

Figura 2.41 Panel de Inicio de Sesión
Fuente: *Autor*

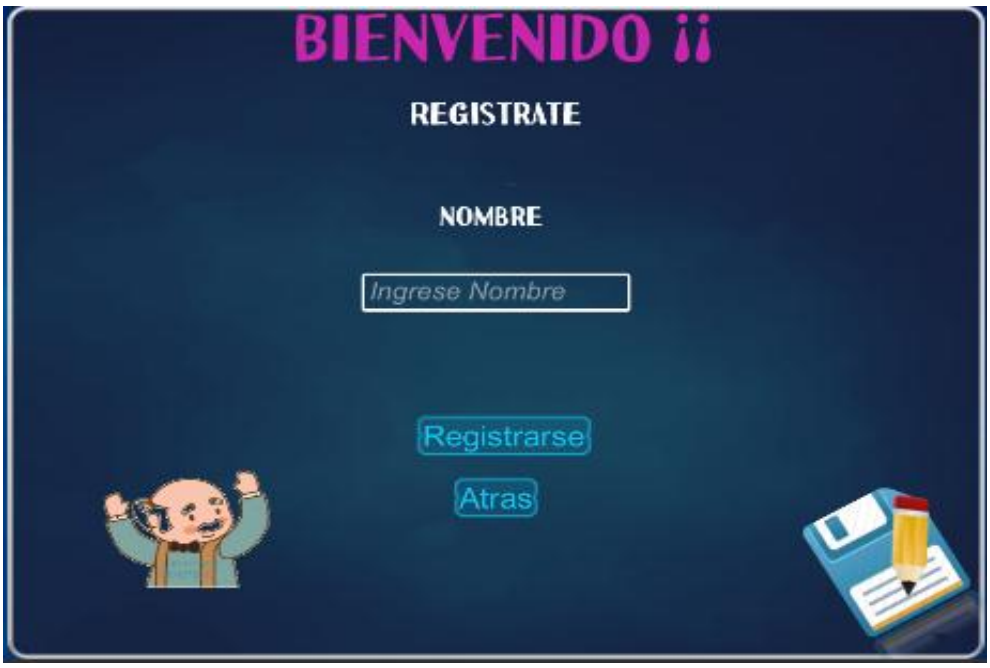


Figura 2.42 Panel de Registro.
Fuente: *Autor*

Otra opción para obtener los datos y su posterior análisis consiste en remitir el valor de puntaje y tiempo al correo electrónico ya sea del participante o de la persona a cargo, por lo que se usa el script *Email.cs*, el mismo que envía los correos desde fabi061992@gmail.com hacia un destinatario elegido. En la Figura 2.43 se muestra

el panel de Email, el cual está activo cuando gane o pierda la partida y funciona solo ingresando el correo al cual desea enviar los datos.

Figura 2.43 Panel para enviar Correo Electrónico
Fuente: *Autor*

Finalmente, en la Figura 2.44 se muestra el panel de opciones cuando el jugador gana la partida, entonces se tiene los dos botones para almacenar los datos.

Figura 2.44 Panel cuando el jugador gana un nivel
Fuente: *Autor*

2.6 DIAGRAMA DE FLUJO DEL SISTEMA

El diagrama de flujo del videojuego se presenta en la figura 2.45 en donde se puede observar que cada menú permite la ejecución de ciertas accione, teniendo en cuenta que el menú salir está directamente relacionado con el menú principal, permitiendo que otros menús accedan a este.

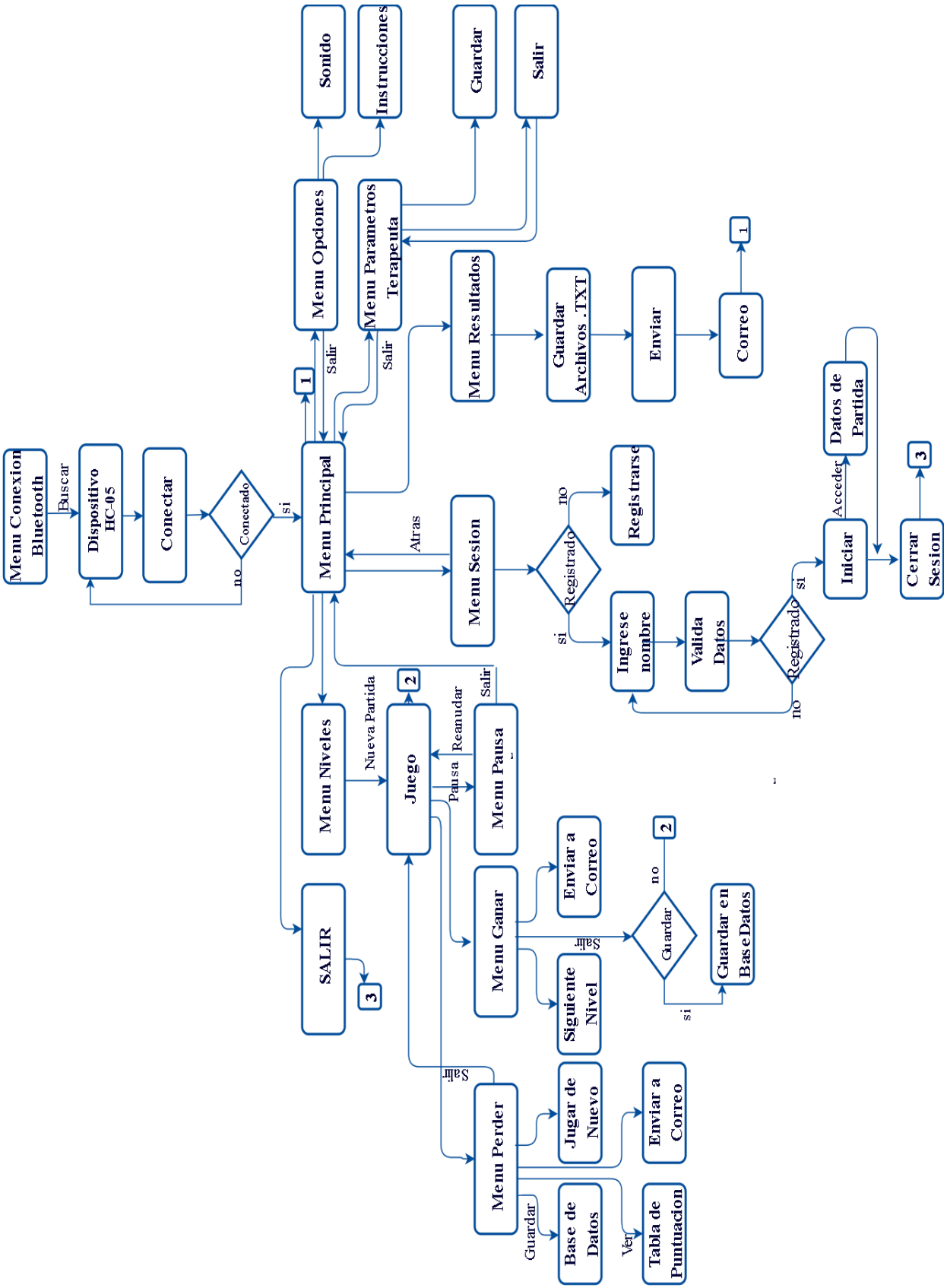


Figura 2.45 Diagrama de Flujo del Videojuego
Fuente: *Autor*

2.7 DIAGRAMAS DE CLASES

En este apartado se indican las diferentes clases que componen el juego, así como la interconexión que existe entre ellas. En Unity todas las clases que componen el juego heredan de la clase principal: MonoBehaviour, por ello se utiliza UML como lenguaje de modelado, el cual pretende mostrar a modo conceptual los distintos tipos de objetos que intervienen en cada nivel de los mini juegos.

2.7.1 DIAGRAMA DE CLASES DE PLANEGAME

A continuación, se desarrolla el diagrama de clases del mini juego *PlaneGame* con todos los elementos que intervienen en el mismo. El esquema que se presenta en la Figura 2.46 es para los tres niveles ya que poseen la misma lógica y mecánica, es decir, las variables de juego como puntaje, tiempo, barra de energía, ítems y anillos se van utilizando en el transcurso del videojuego.

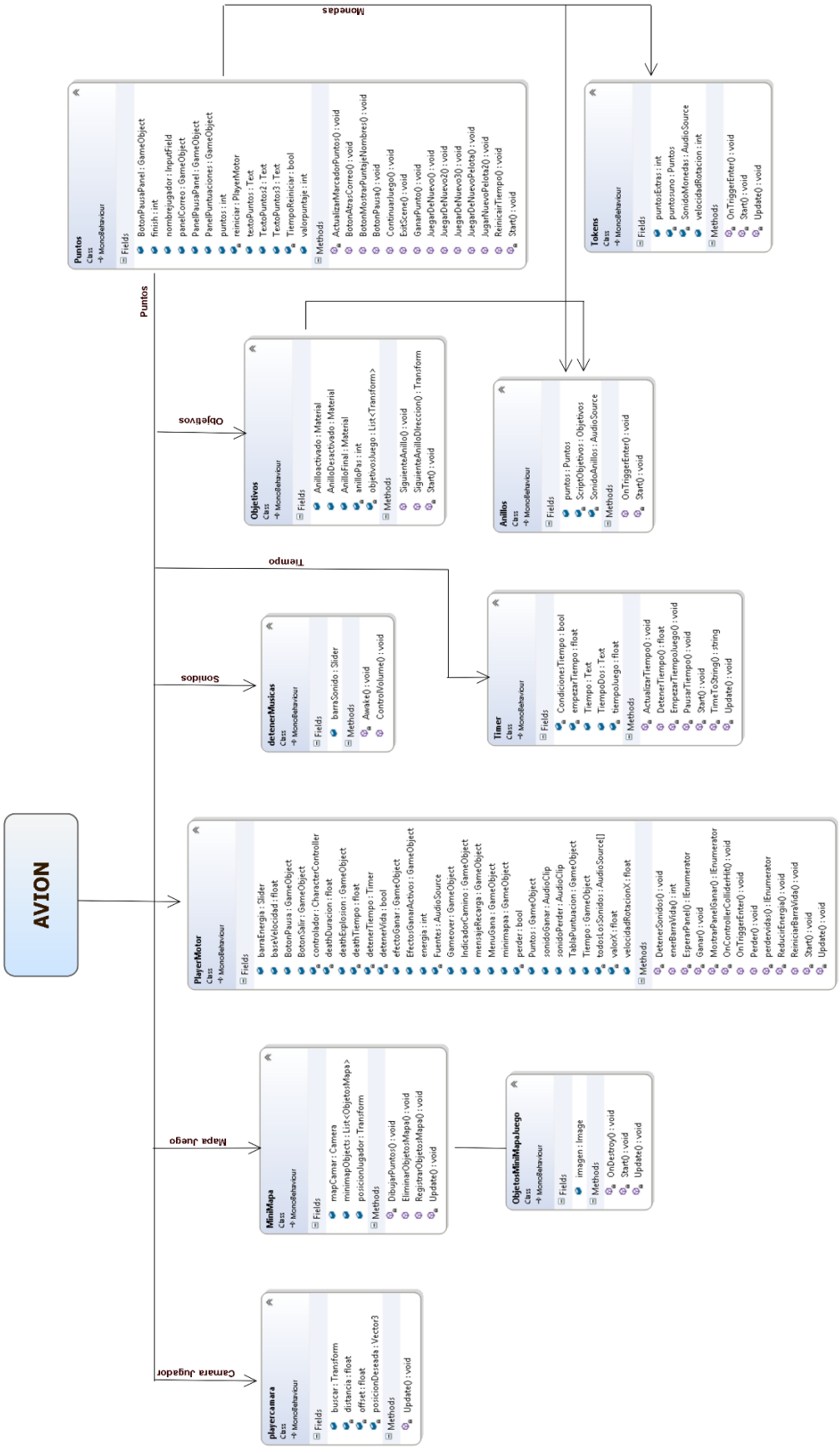


Figura 2.46 *Diagrama de Clases para los eventos del juego PlaneGame*
Fuente: *Autor*

2.7.2 DIAGRAMA DE CLASES DE BALLGAME

En la Figura 2.47 se puede observar el diagrama de clases del mini juego *BallGame* con todos los elementos que intervienen en el mismo. De la misma manera, el esquema que se presenta es para los dos niveles.

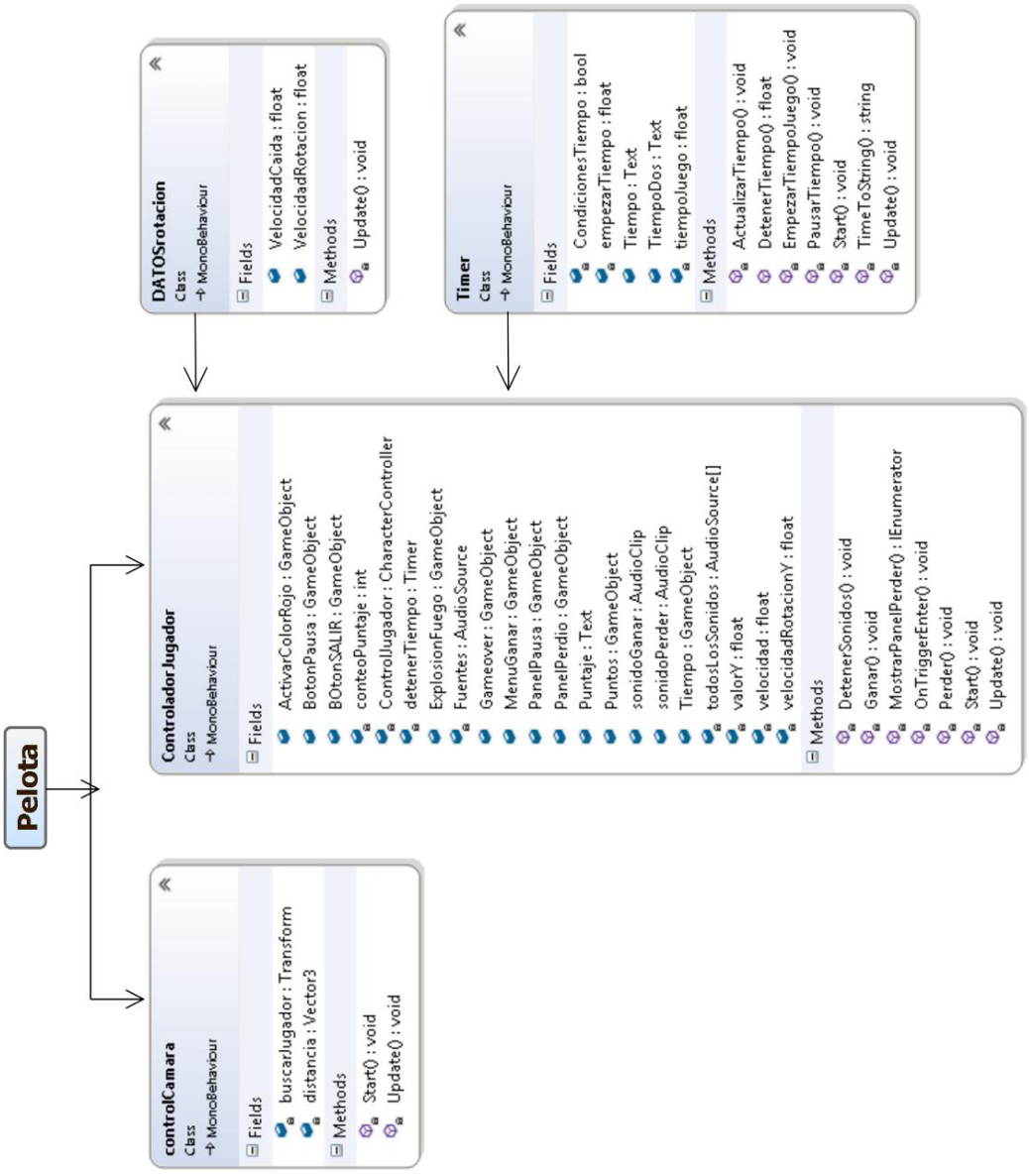


Figura 2.47 *Diagrama de Clases para los eventos del juego BallGame*
Fuente: *Autor*

2.7.3 DIAGRAMA DE CLASES DE LA RECEPCION DE DATOS

En la Figura 2.48 se muestra diagrama de clases implementado para la recepcion y procesado de datos provenientes desde el dispositivo inalámbrico.

Figura 2.48 *Diagrama de Clases implementado para la recepcion de datos*
Fuente: *Autor*

2.7.4 DIAGRAMA DE CLASES PARA GUARDAR DATOS

En la figura 2.49 se muestra el diagrama de clases para guardar los datos, los mismos que incluyen la base de datos, el puntaje más alto en el teléfono y la parte de enviar los datos al correo. Estos métodos se usan en general para todo el videojuego.

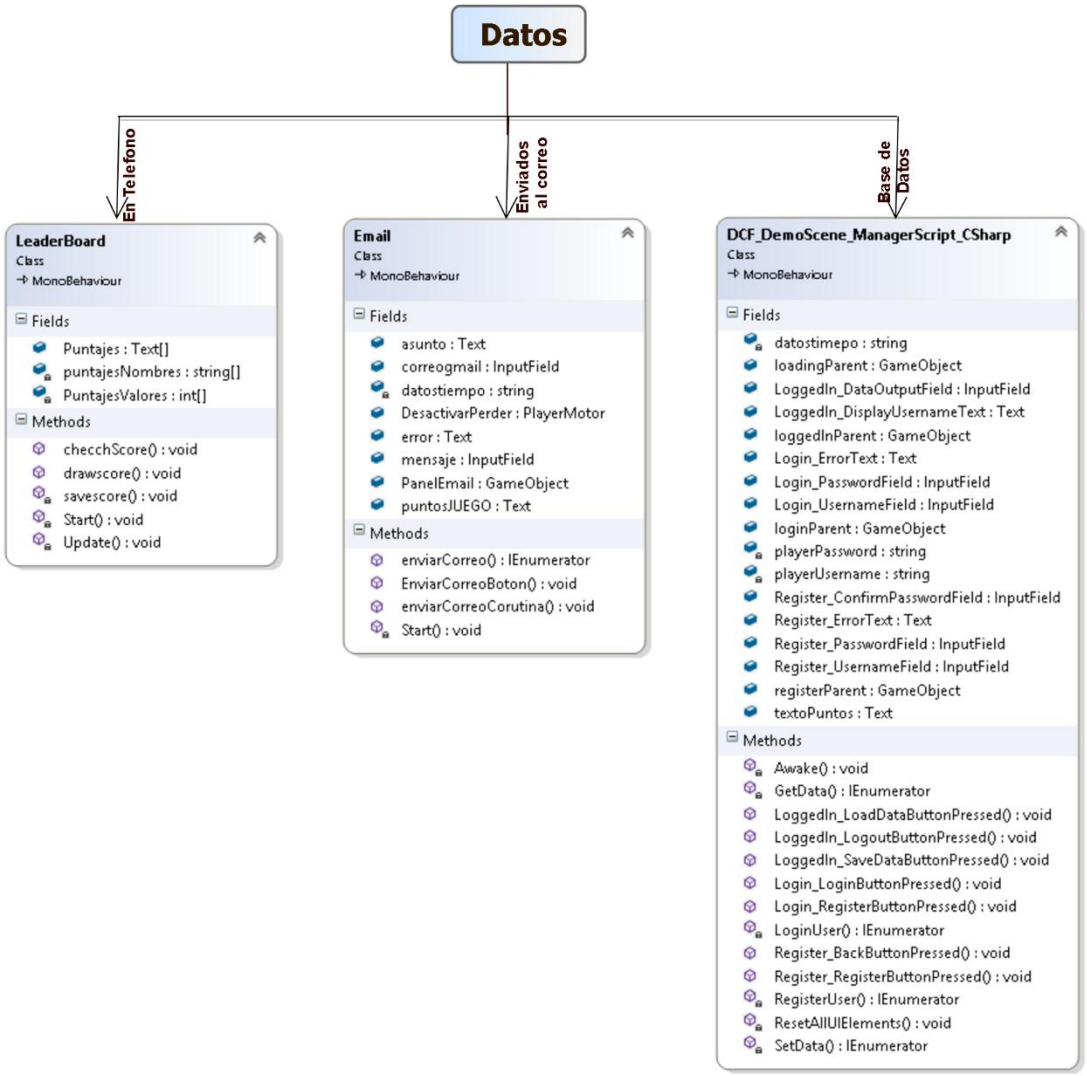


Figura 2.49 *Diagrama de Clases implementado para almacenar los datos.*
Fuente: *Autor*

2.7.5 DIAGRAMA DE CLASES DE LA INTERFAZ DE USUARIO

En la figura 2.50 se muestra el diagrama de clases implementados para la interfaz de usuario.

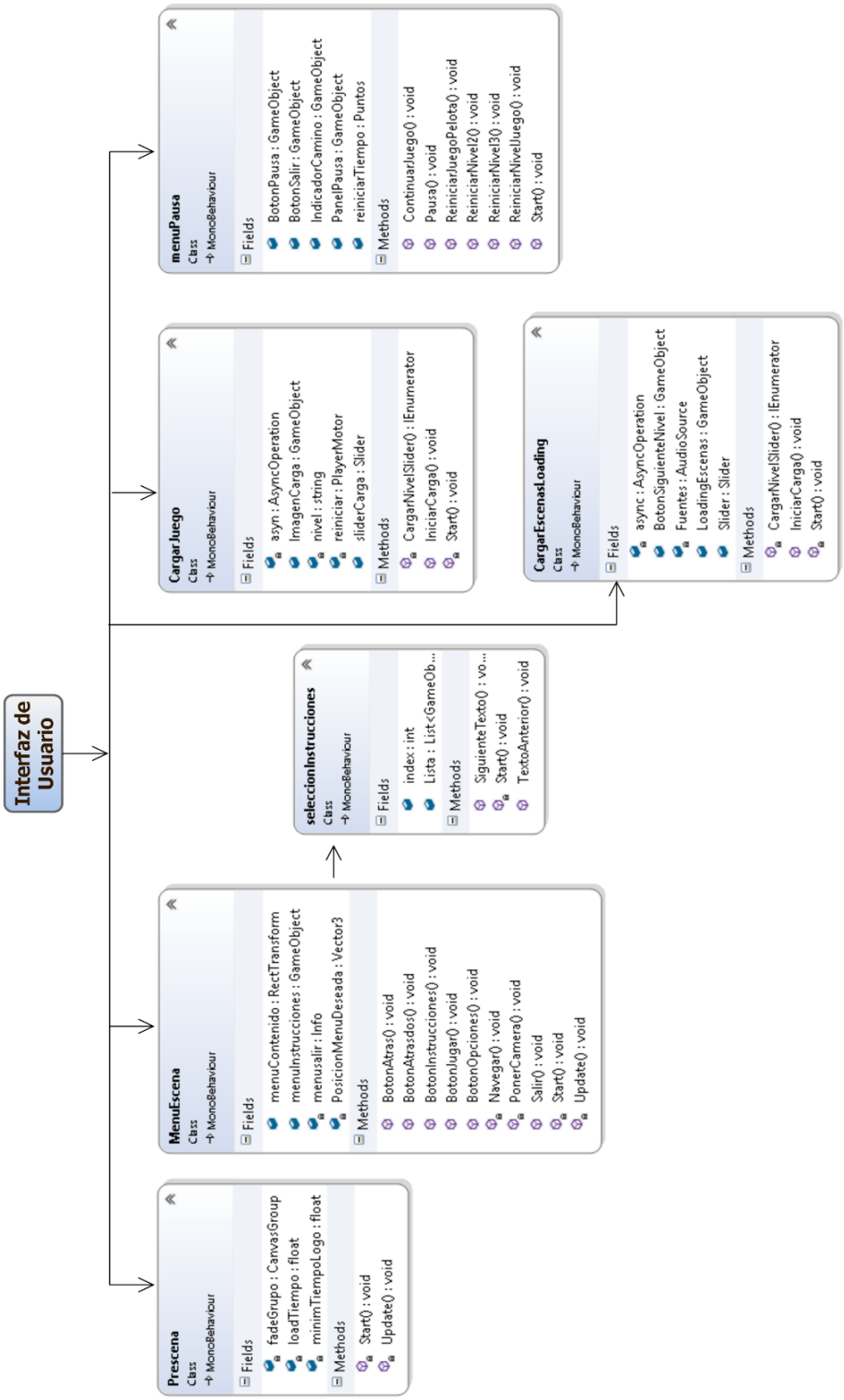


Figura 2.50 *Diagrama de Clases de la Interfaz de Usuario para los Mini juegos*
Fuente: *Autor*

Capítulo 3 : ANÁLISIS DE RESULTADOS

Este capítulo contiene el análisis de los resultados obtenidos durante las pruebas realizadas con el dispositivo desarrollado y el videojuego exportado a la plataforma Android tanto para un teléfono móvil de la marca Samsung cuanto para una Tablet Nexus.

Como primera parte se expone los resultados para las pruebas con el teléfono móvil por lo que se contó con 6 personas de edad adulta con la Enfermedad de Parkinson y 20 adultos mayores, en donde se realizaron cinco sesiones de juego para la mano derecha y cinco sesiones para la mano izquierda, con una duración de 8 minutos en cada mano, al finalizar el juego se obtienen los valores de puntaje y tiempo que fueron empleados para terminar cada nivel de los mini juegos. Además, se realiza una encuesta de validación del sistema.

La información de los pacientes con EP se muestra en la Tabla 3.1, cabe destacar que cada paciente estaba con medicación durante las pruebas. Mientras que en el Apéndice A, se puede observar los datos de género y edad de los 20 participantes adultos mayores sanos.

Tabla 3.1 Características de los Pacientes con EP que participaron en las pruebas
Fuente: Autor

Participante	Edad	Genero	Medicación
Paciente 1	82	Femenino	SI
Paciente 2	71	Masculino	SI
Paciente 3	83	Femenino	SI
Paciente 4	73	Femenino	SI
Paciente 5	85	Masculino	SI
Paciente 6	74	Femenino	SI

Cada participante fue ubicado en una silla a una distancia de 30cm con respecto a la posición del teléfono móvil y luego procedieron a jugar cada mini juego teniendo en cuenta las instrucciones de juego explicadas con anterioridad y los

movimientos de pronosupinación y flexo extensión a realizar. Por lo que, practicaron durante 10 minutos, alternando el dispositivo a la mano derecha e izquierda, es decir 20 minutos en cada mini juego. En las figuras 3.1 y 3.2 se puede observar a dos participantes durante las pruebas.

Figura 3.1 Participante con EP durante las pruebas del sistema
Fuente: *Autor*

Figura 3.2 *Participante sana durante las pruebas del sistema.*
Fuente: *Autor*

3.1 DATOS OBTENIDOS EN PACIENTES CON EP USANDO UN CELULAR SAMSUNG

Las gráficas que representan los datos obtenidos se basan en diagramas de caja y bigotes [132], lo que permite tener un promedio de todos los datos de las sesiones en cada mini juego para su comparación y posterior análisis.

3.1.1 MINI JUEGO PLANEGAME

A continuación, se obtienen los resultados de los pacientes con EP, durante las cinco sesiones teniendo en consideración el tiempo y puntaje obtenidos durante el mini juego PlaneGame, en la Figura 3.3 se presenta los datos de tiempo y en la Figura 3.4 los datos de puntaje.

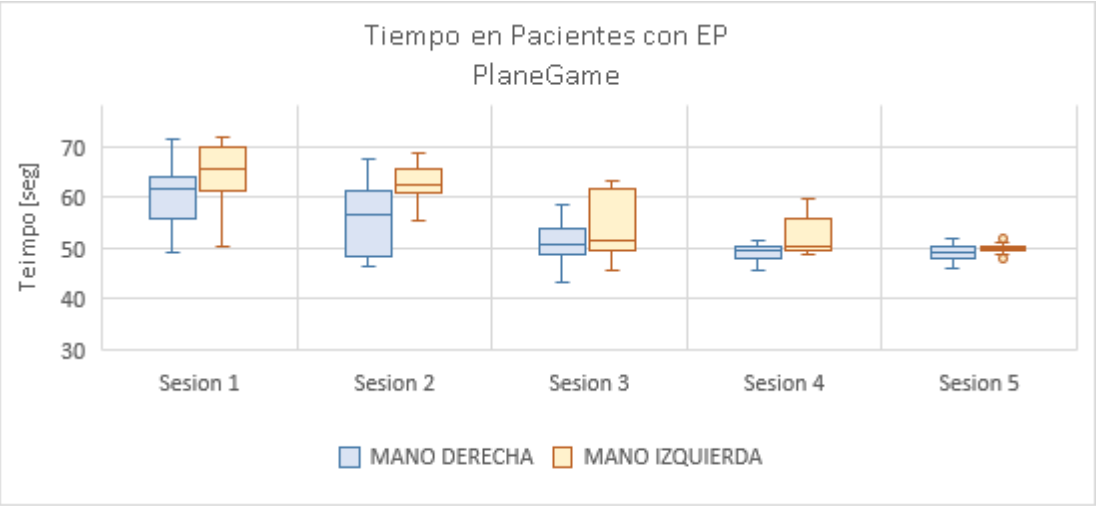


Figura 3.3 *Diagrama de Caja y Bigotes de Tiempo obtenido durante el Mini Juego PlaneGame en pacientes con EP, usando la mano derecha e izquierda*
Fuente: *Autor*

Los valores de tiempo obtenidos durante el mini juego PlaneGame en pacientes con EP varían desde la primera sesión, presentándose una cierta diferencia ya que en la primera sesión se obtiene una mediana alrededor de 61,88seg usando la mano derecha mientras que usando la mano izquierda la mediana es de 65,66seg, conforme avanzan las sesiones el tiempo va disminuyendo, tal como se observa en la sesión cinco en donde el valor de la mediana con la mano derecha es 49,33seg y con la mano izquierda 50,03 seg.

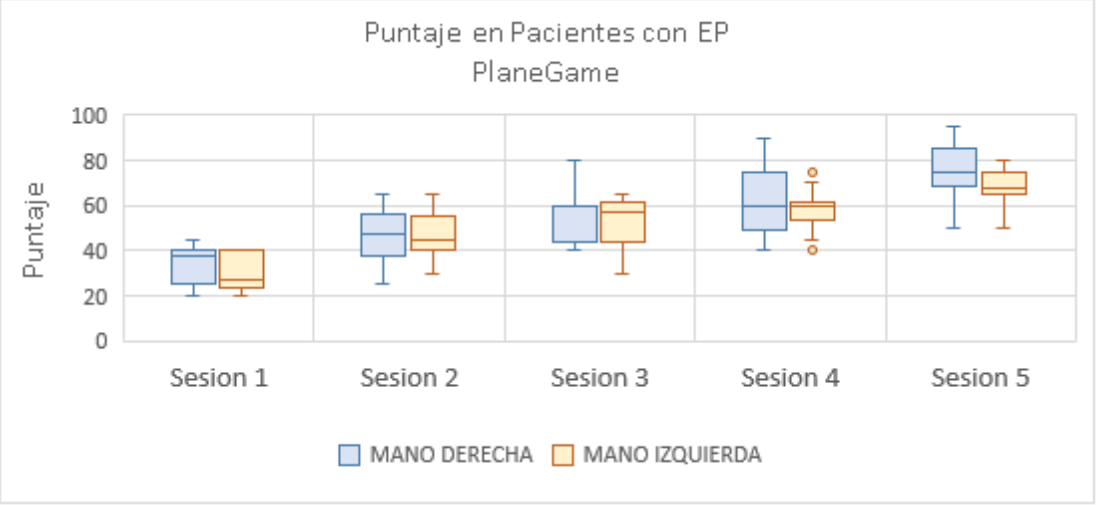


Figura 3.4 *Diagrama de Caja y Bigotes de Puntaje obtenido durante el Mini Juego PlaneGame en pacientes con EP, usando la mano derecha e izquierda*
Fuente: *Autor*

De acuerdo con los resultados mostrados en la figura 3.4, el puntaje al inicio del mini juego PlaneGame alcanza una mediana los 37,5 puntos usando la mano derecha y 27,5 puntos con la mano izquierda, sin embargo, conforme avanza las sesiones, el puntaje se incrementa, a tal punto que en la sesión cinco se obtiene un valor de mediana de 75 puntos en la mano derecha y 67,5 puntos usando la mano izquierda, siendo el puntaje máximo de 95 con la mano derecha y el mínimo de 50 puntos con las dos manos.

3.1.2 MINI JUEGO BALLGAME

De la misma manera, se muestran los datos del tiempo y puntaje obtenidos durante el mini juego *BallGame*, en la Figura 3.5 se presenta los datos de tiempo, mientras que la Figura 3.6 muestra los datos de puntaje.

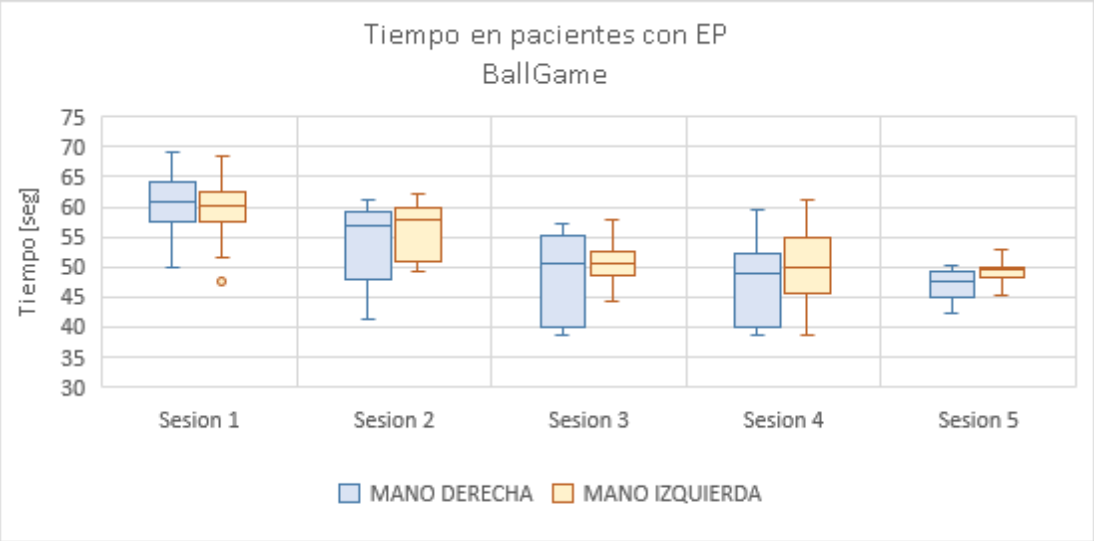


Figura 3.5 *Diagrama de Caja y Bigotes de Tiempo obtenido durante el Mini Juego BallGame en pacientes con EP, usando la mano derecha e izquierda*
Fuente: *Autor*

Como se observa en la Figura 3.5 el tiempo que se emplea para la primera sesión posee un valor máximo alrededor de 69,15 seg usando la mano derecha y 68,14 seg usando la mano izquierda con una mediana de 60,99 y 60,08 respectivamente. El tiempo va disminuyendo en cada sesión y al finalizar la sesión cinco se obtiene una mediana alrededor de 47,57seg con la mano derecha y 49,41 seg con la mano izquierda.

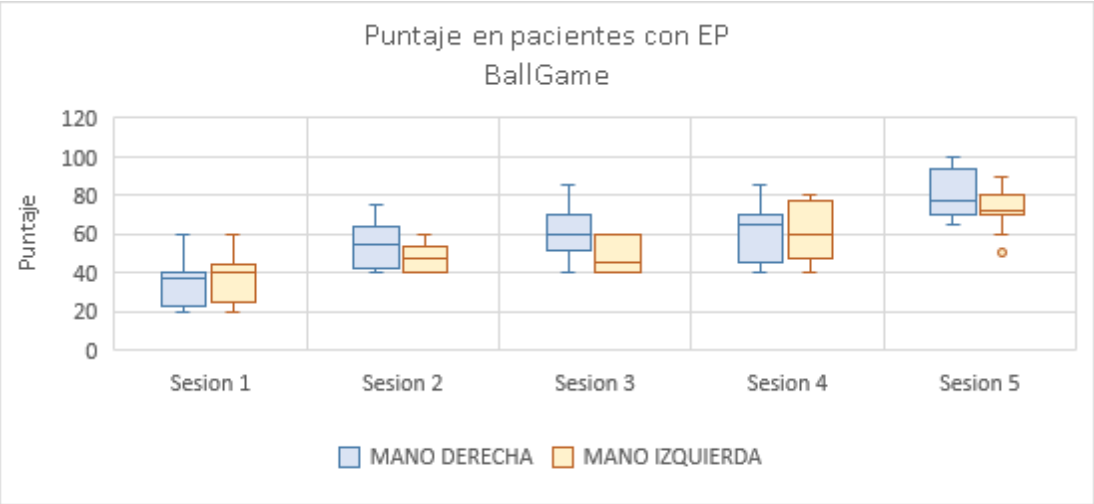


Figura 3.6 *Diagrama de Caja y Bigotes de Puntaje obtenido durante el Mini Juego BallGame en pacientes con EP, usando la mano derecha e izquierda*
Fuente: *Autor*

De acuerdo con la Figura 3.6 se puede establecer que el puntaje máximo durante el mini juego Ball en Pacientes con EP es 100 puntos practicando con la mano derecha mientras que con la mano izquierda se alcanzó un puntaje de 90. En la primera sesión los valores de mediana son 37,5 y 40 respectivamente siendo evidente que en cada sesión con la mano derecha el puntaje fue aumentando, ya que al finalizar la quinta sesión las medianas están alrededor de 77,5 y 72,5.

3.2 DATOS OBTENIDOS EN PACIENTES SANOS USANDO UN CELULAR SAMSUNG

3.2.1 MINI JUEGO PLANEGAME

En la Figuras 3.7 y 3.8 se muestran los datos de tiempo y puntaje obtenidos durante las cinco sesiones del mini juego PlaneGame usando la mano derecha e izquierda respectivamente.

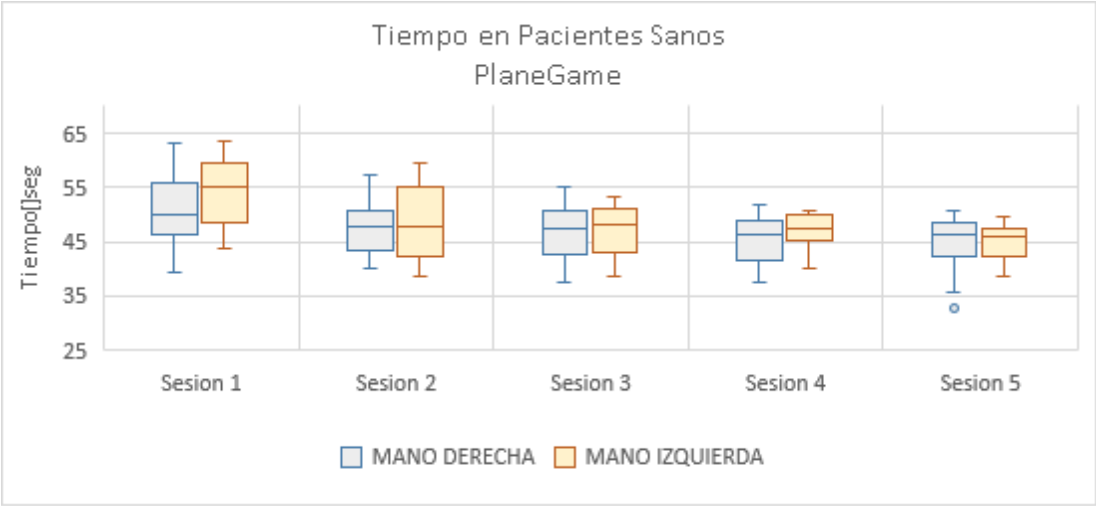


Figura 3.7 *Diagrama de Caja y Bigotes de Tiempo obtenido durante el Mini Juego PlaneGame en Pacientes Sanos, usando la mano derecha e izquierda*
Fuente: *Autor*

En la figura 3.7 se observa los datos de tiempo que emplearon los pacientes sanos durante el mini juego PlaneGame, en donde durante la primera sesión el tiempo máximo que se empleo fue de 63,6 seg con una mediana de 50,12 seg con la mano derecha y 55,04 seg con la mano izquierda. Como es notable conforme avanzan las sesiones, el tiempo disminuye, y durante la quinta sesión se obtiene un tiempo máximo de 50,61seg con una mediana alrededor de 46,13seg para la mano derecha y un tiempo máximo de 49,69 seg con un valor de mediana de 46,05 seg usando la mano izquierda.

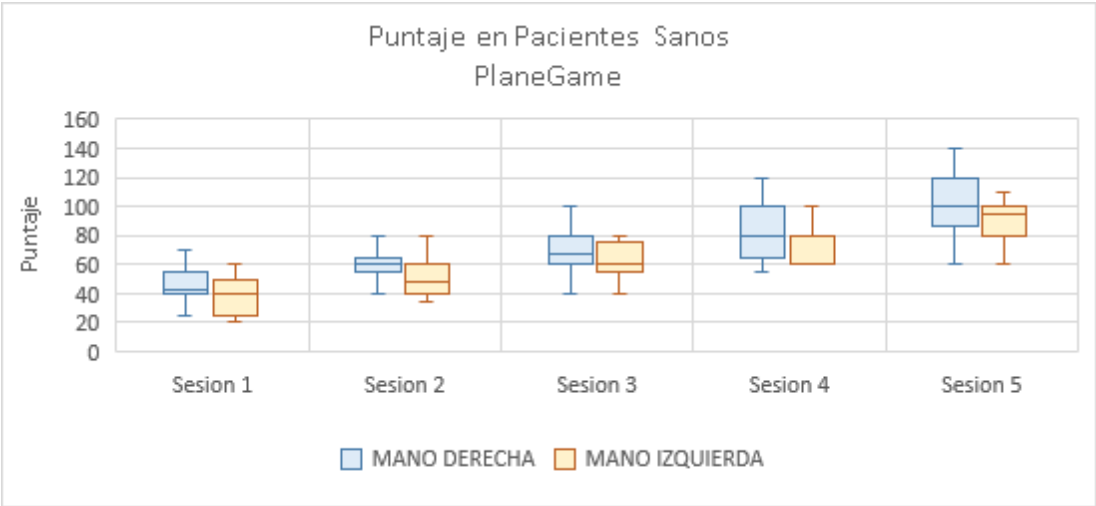


Figura 3.8 *Diagrama de Caja y Bigotes de Puntaje obtenido durante el Mini Juego PlaneGame en Pacientes Sanos usando la mano derecha e izquierda*
Fuente: *Autor*

Igualmente, el puntaje máximo alcanzado por los pacientes sanos en la primera sesión esta alrededor de 70 puntos con la mano derecha y 60 puntos con la mano izquierda con medianas alrededor de 42,5 puntos y 40 puntos respectivamente., y así existe un aumento de puntaje conforme avanza las sesiones hasta obtener un puntaje máximo en la quinta sesión de 140 puntos con una mediana de 100 puntos usando la mano derecha y 110 puntos con la mano izquierda con una mediana 95 puntos.

3.2.2 MINI JUEGO BALLGAME

En la Figuras 3.9-3.10 se muestran los datos obtenidos durante el mini juego BallGame

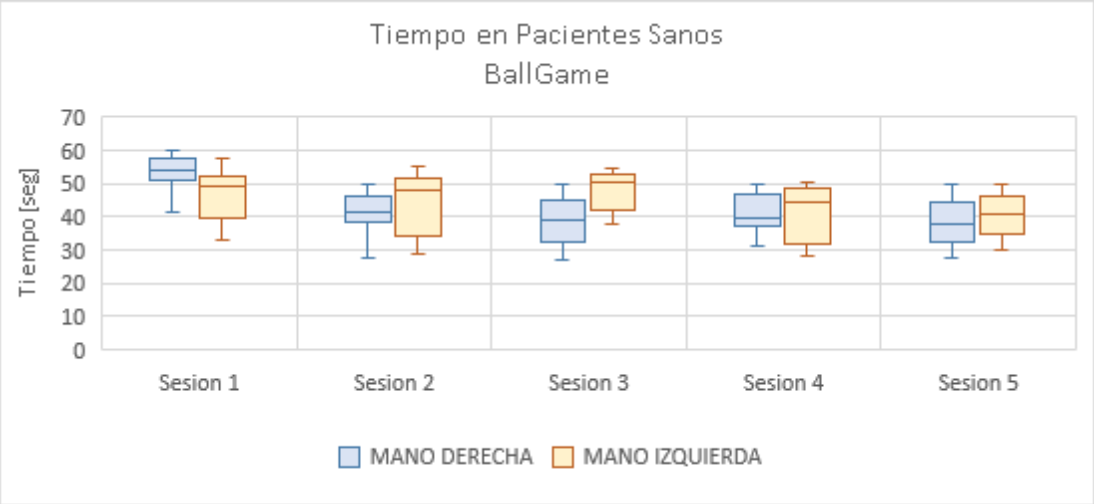


Figura 3.9 *Diagrama de Caja y Bigotes de Tiempo obtenido durante el Mini Juego BallGame en Pacientes Sanos, usando la mano derecha e izquierda*
Fuente: *Autor*

En la figura 3.9 se observa los valores de tiempo empleados durante las sesiones de juego de BallGame y se nota que los tiempos disminuyen conforme las sesiones transcurren, empezando con un valor de mediana alrededor de 54,10 seg usando la mano derecha y 49,04 usando la mano izquierda y obteniendo un tiempo máximo de 60 seg. Mientras que al finalizar la sesión cinco la mediana se establece en 37,83 seg para la mano derecha y 40,88 para la mano izquierda y un tiempo máximo de 50seg durante el juego.

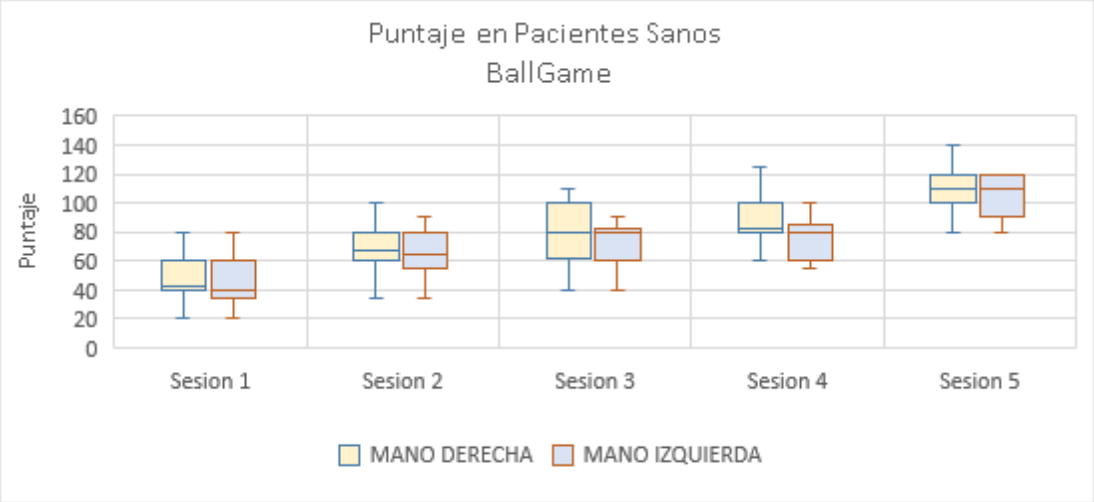


Figura 3.10 *Diagrama de Caja y Bigotes de Puntaje obtenido durante el Mini Juego BallGame en Pacientes Sanos, usando la mano derecha*
Fuente: *Autor*

Los valores de puntaje alcanzado durante las sesiones de BallGame en pacientes sanos indican que el puntaje en la primera sesión posee una mediana alrededor de 42,5 puntos usando la mano derecha y 40 puntos usando la mano izquierda cuyo valor máximo es 80 puntos para los dos casos, mientras que en la segunda sesión ya se nota la diferencia con el puntaje máximo obtenido de 100 puntos para la mano derecha y 90 puntos para la mano izquierda. Al finalizar la sesión cinco el puntaje la mediana es de 110 para las dos manos, siendo como máximo puntaje 140 puntos y como mínimo 80 puntos durante el juego.

3.3 DATOS OBTENIDOS USANDO UNA TABLET

En este apartado se presenta una comparación usando una Tablet de marca Nexus con relación a los datos obtenidos con el celular, se presenta la comparación en pacientes con EP y en pacientes Sanos.

Como se observa en la figura 3.11, con relación a los Pacientes con EP, se presenta una comparación de los datos obtenidos usando el celular y la Tablet, existiendo una cierta disminución de tiempo usando la Tablet, tanto para la mano derecha cuanto para la izquierda.

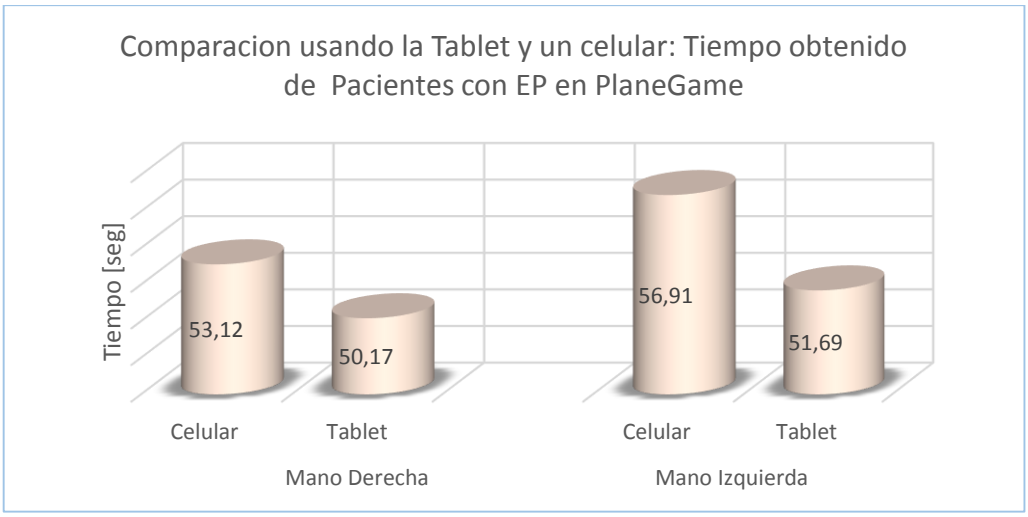


Figura 3.11 *Resultados de Tiempo obtenido en Pacientes con EP durante el mini juego PlaneGame, Celular vs Tablet.*
Fuente: *Autor*

De la misma manera en la Figura 3.12 se observa los resultados para el valor de puntaje presentado un aumento alrededor de 6,13 puntos usando la Tablet en la mano derecha y para la mano izquierda se indica un incremento de 5,77 puntos

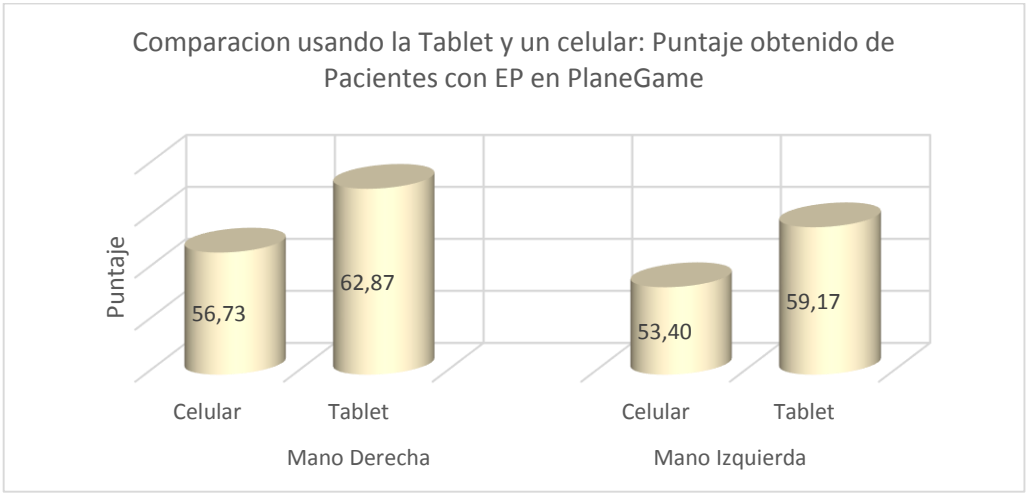


Figura 3.12 *Resultados de Puntaje obtenido en Pacientes con EP durante el mini juego PlaneGame, Celular vs Tablet.*
Fuente: *Autor*

Para el mini juego Ball se obtienen resultados diferentes en cuestion del tiempo, ya que se presenta cierto aumento de tiempo usando la Tablet alrededor de 3,06 seg para la mano derecha y 1,69 seg para la mano izquierda, tal como se muesta en la Figura 3.13

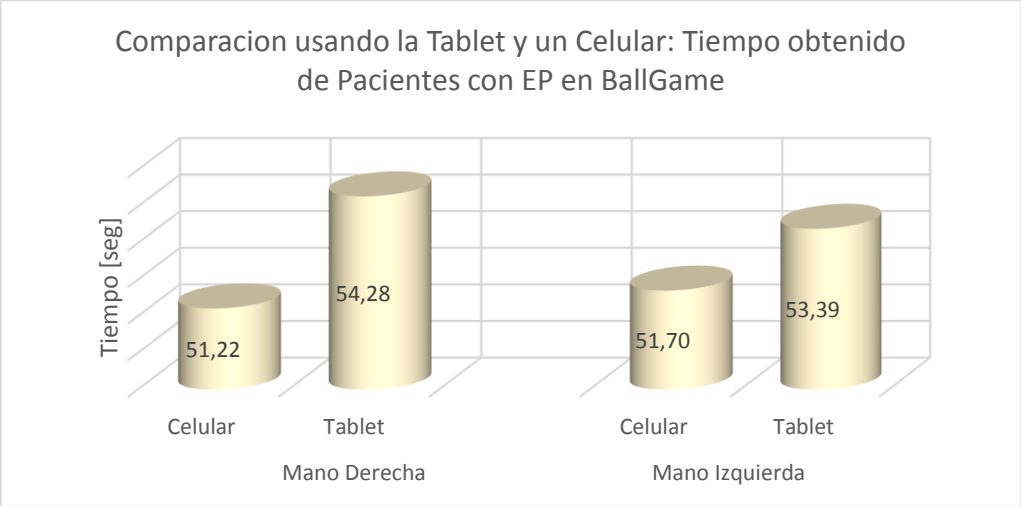


Figura 3.13 *Resultados de Tiempo obtenido en Pacientes con EP durante el mini juego BallGame, Celular vs Tablet.*
Fuente: ***Autor***

Mientras que, en la Figura 3.14 se muestran los resultados para el valor de puntaje, en donde se puede notar que usando la Tablet el puntaje aumenta para la mano derecha e izquierda, alrededor de 3,37 puntos y 5,39 puntos respectivamente.

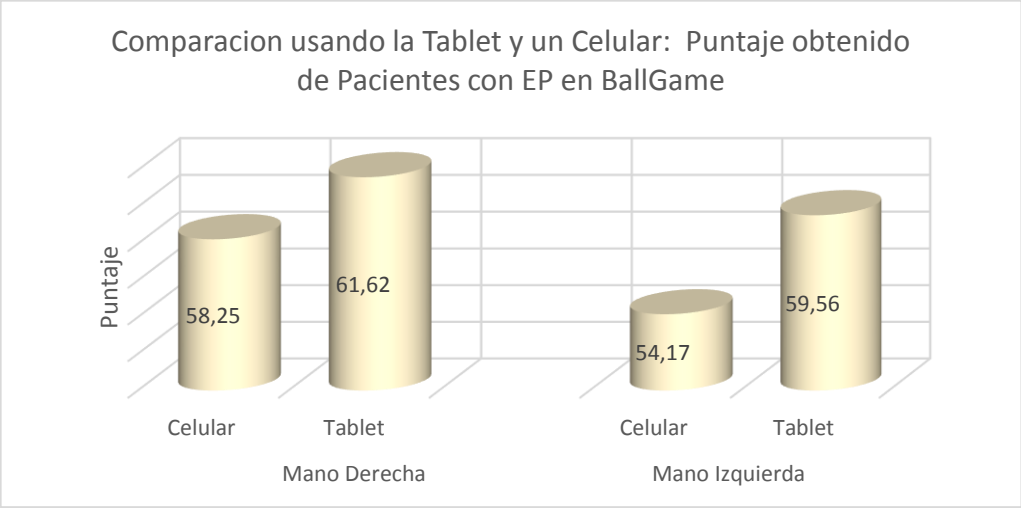


Figura 3.14 *Resultados de Puntaje obtenido en Pacientes con EP durante el mini juego BallGame, Celular vs Tablet.*
Fuente: ***Autor***

A continuación, se obtienen los resultados en los pacientes sanos, lo cual se muestra en las figuras 3.15 y 3.17 para los dos minijuegos respectivamente.

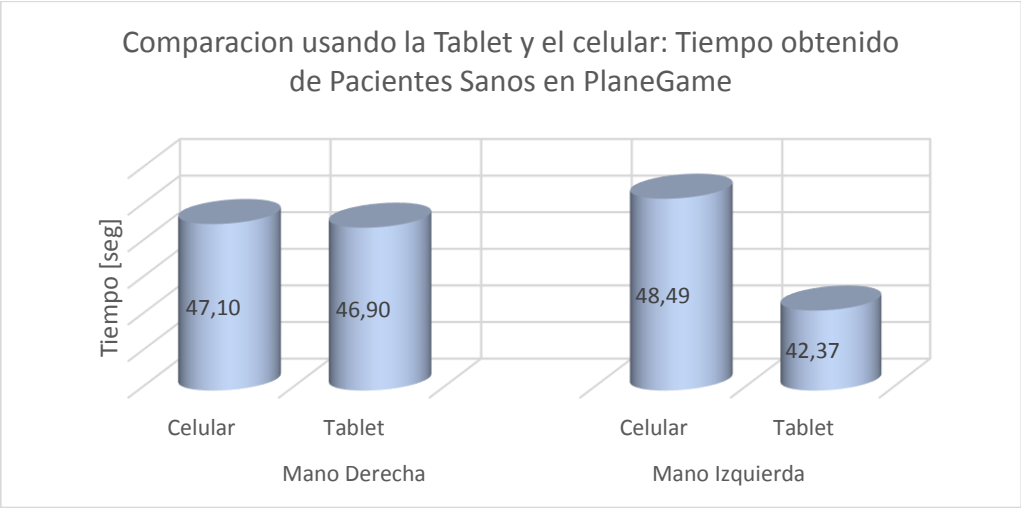


Figura 3.15 *Resultados de Tiempo obtenido en Pacientes Sanos durante el mini juego PlaneGame, Celular vs Tablet.*
Fuente: *Autor*

En la Figura 3.15 se observa los resultados obtenidos de los pacientes sanos durante el mini juego PlaneGame en donde el tiempo disminuye usando la Tablet tanto para la mano derecha de 0,2seg cuanto para la mano izquierda de 6,09 seg, siendo un valor notable en la mano izquierda.

De la misma manera se presenta los datos de puntaje para el mini juego PlaneGame en pacientes sanos, lo cual resulta un aumento en el puntaje usando la Tablet, que representa 8,55 puntos para la mano derecha y 1,66 puntos para la mano izquierda

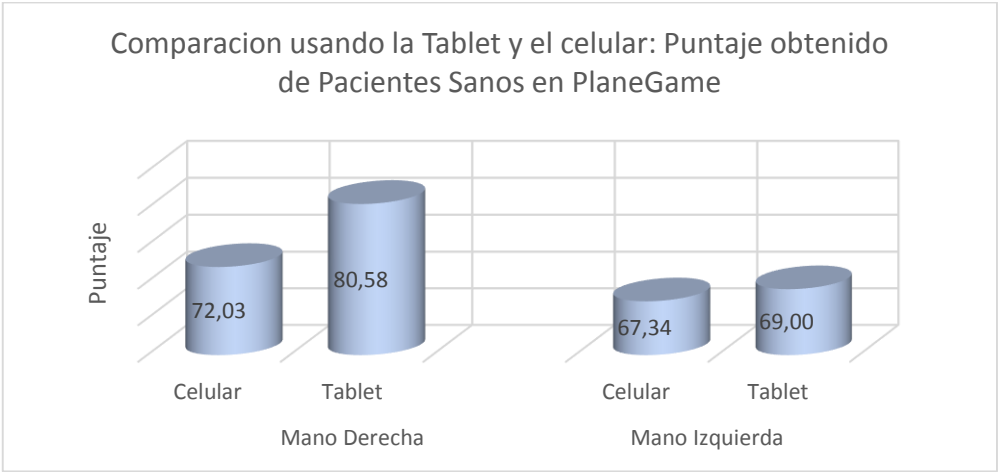


Figura 3.16 *Resultados de Puntaje obtenido en Pacientes Sanos durante el mini juego PlaneGame, Celular vs Tablet.*
Fuente: ***Autor***

En la figura 3.17 se representa los datos de tiempo durante el mini juego BallGame obtenidos usando el celular y la Tablet, lo cual indica que con la mano derecha el tiempo aumento 1,13seg usando la Tablet, mientras que con la mano izquierda el tiempo disminuyo 0,08seg.

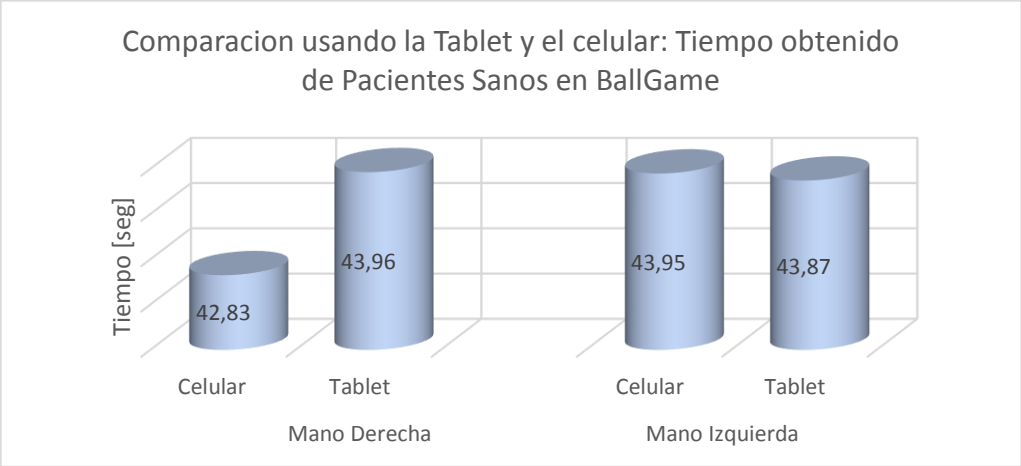


Figura 3.17 *Resultados de Tiempo obtenido en Pacientes Sanos durante el mini juego BallGame, Celular vs Tablet.*
Fuente: ***Autor***

Finalmente, se presenta en la Figura 3.18 la comparación en función del puntaje obtenido por lo pacientes sanos durante el mini juego BallGame, como se puede observar existe un aumento de puntaje usando la Tablet, con la mano derecha el puntaje aumento un 3,38 puntos y 1,79 puntos para la mano izquierda.

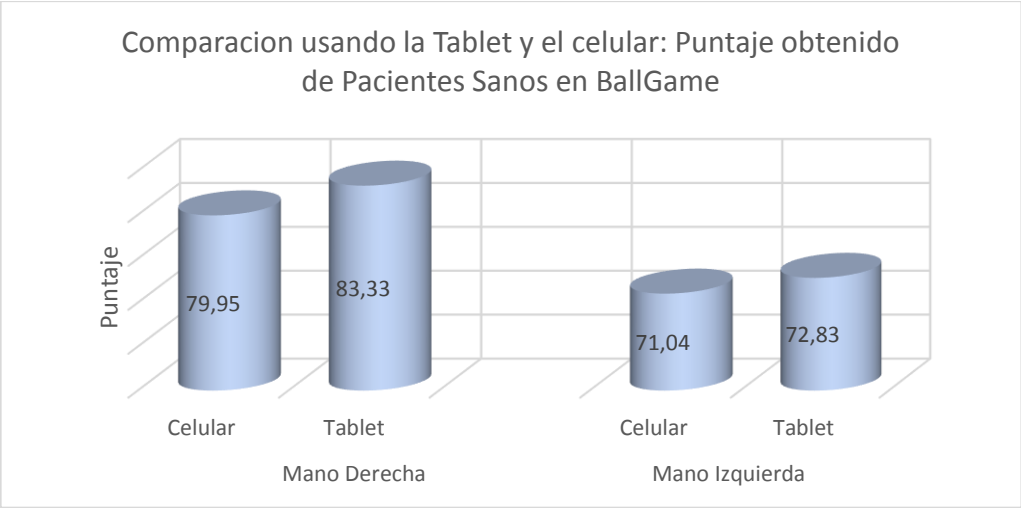


Figura 3.18 *Resultados de Puntaje obtenido en Pacientes Sanos durante el mini juego BallGame, Celular vs Tablet.*
Fuente: ***Autor***

3.4 VALIDACION DEL SISTEMA

Para la validación del videojuego y el dispositivo, se realizó una encuesta a las 26 personas que participaron de las pruebas, cuyas preguntas se muestran en la Tabla 3.2

Tabla 3.2 Preguntas que conforman la encuesta para la validación del Sistema
Fuente: *Autor*

Nº	**Pregunta**
P1	¿Sintió cansancio al finalizar los mini juegos?
P2	¿Le agrado el entorno grafico?
P3	¿Se sintió cómodo usando el dispositivo?
P4	¿Le agrado la música y sonidos?
P5	¿Se divirtió con los mini juegos?
P6	¿Le fue fácil jugar?
P7	¿Cuál mini juego le agrado más?

Los resultados obtenidos de la encuesta se muestran en las Figuras 3.19-3.22, sin embargo, están algunas preguntas cuyas respuestas no se muestran en la figura, esto se debe a que existió gran aceptación por parte de los participantes, indicando que les agrado mucho el videojuego, colores, dibujos y sobre todo la música, además

supieron decir que se divirtieron con ello ya que jamás habían experimentado algo así.

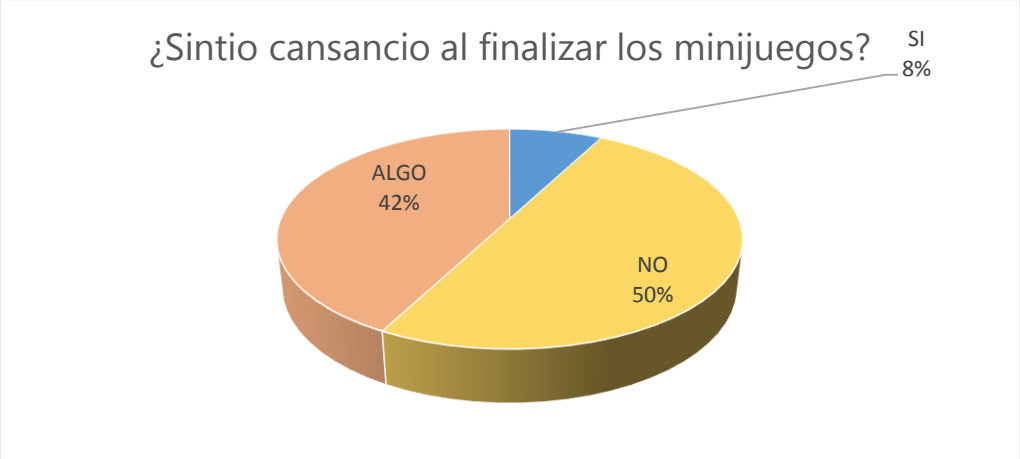


Figura 3.19 *Respuesta en relación con la pregunta ¿Sintió cansancio al finalizar los mini juegos?*
Fuente: *Autor*

Como se observa en la figura 3.19, el 50 % de los participantes indicaron que no sintieron ningún tipo de cansancio durante el juego, mientras que 42% indicaron tener algo de cansancio debido a la posición del brazo y el 8% indico si tener cansancio al finalizar las pruebas de juego.

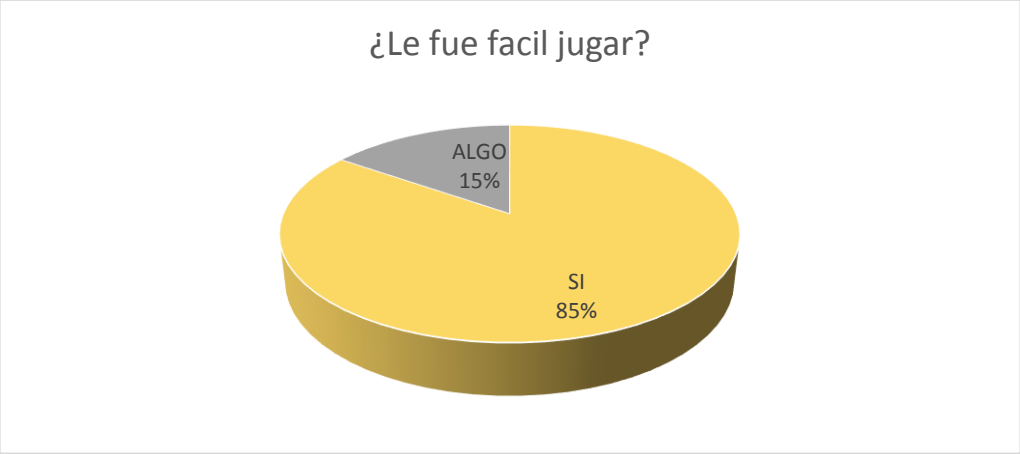


Figura 3.20 *Respuesta en relación con la pregunta ¿Le fue fácil jugar?*
Fuente: *Autor*

En la Figura 3.20 se nota una gran aceptación de los participantes con relación a la maniobrabilidad del dispositivo y por ende del videojuego, ya que el 85% indicaron que si les fue fácil jugar mientras que el 15% indicaron cierta dificultad durante el juego.

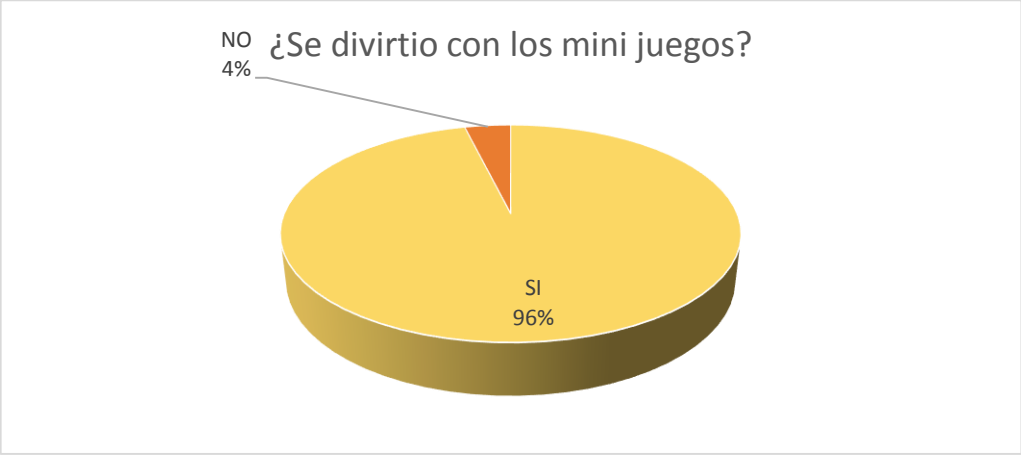


Figura 3.21 *Respuesta en relación con la pregunta ¿Se divirtió con los mini juegos*?
Fuente: *Autor*

La figura 3.21 representa los resultados de los participantes hacia la pregunta si se divirtieron con los mini juegos, en donde el 96% indican que si, mientras que el 4% responde que no se divirtieron

.

En la Tabla 3.3, se indica los resultados del nivel de aceptación de los mini juegos por parte del género masculino y femenino, en donde es notable que el 53.85% de los participantes indicaron que el mini juego BallGame fue de su mayor agrado, mientras que el 46.15% optaron por el mini juego PlaneGame. Esto también se representa en la Figura 3.22

Tabla 3.3 Resultados del nivel de aceptación de los mini juegos
Fuente: Autor

Resultados del Nivel de Aceptacion de los MiniJuegos			
	BallGame	**PlaneGame**	**Total**
Mujer	46,15%	15,38%	**61,54%**
Hombre	7,69%	30,77%	**38,46%**
Total	**53,85%**	**46,15%**	**100,00%**

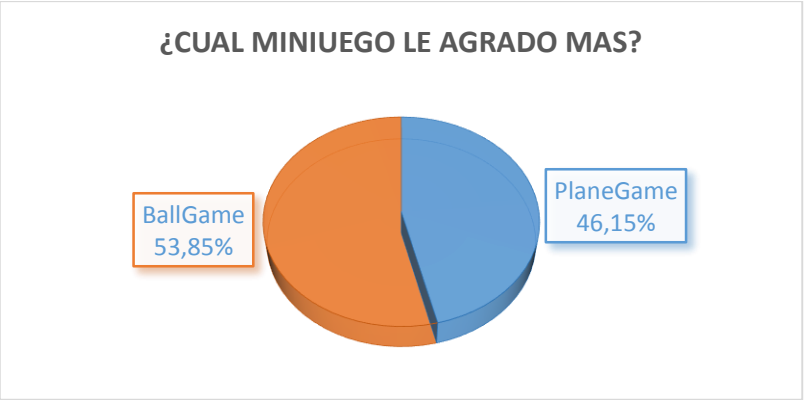


Figura 3.22 *Respuesta en relación con la pregunta ¿Cuál mini juego le agrado más?*
Fuente: *Autor*

3.5 COMPARACION CON LA TERAPIA CONVENCIONAL

En este apartado se realizó pruebas de comparación del sistema teniendo en consideración la terapia convencional que poseen los pacientes con EP. Para ello se dispone de los 5 pacientes con EP y 10 pacientes sanos.

Los 15 participantes tuvieron varias sesiones semanales de terapia ocupacional y dos sesiones diarias durante 10 días con el sistema de rehabilitación propuesto y al finalizar cada terapia se realizó una prueba de motricidad durante dos minutos en cada mano, que consiste en atrapar con pinzas objetos lo que permite trabajar la fuerza en dedos y mano, y al mismo tiempo la estabilidad en el codo y el hombro. Al finalizar la prueba se cuenta el número de elementos colocados para validar y comparar las dos terapias respectivamente. En la figura 3.23 se indica el tipo de prueba aplicada.

Figura 3.23 *Modelo de prueba de motricidad realizada al finalizar las terapias de rehabilitación.*
Fuente: *Autor*

3.5.1 PRUEBAS DE VALIDACIÓN

La prueba de motricidad mostrada en la figura 3.23 se realizó durante dos minutos en cada brazo, en donde cada participante debía atrapar el mayor número de pompones con el color indicado y colocarlos dentro del recipiente. Los resultados obtenidos de la prueba al finalizar la terapia convencional con los 15 participantes indican el número de pompones atrapados y se muestran en la Tabla 3.4

Tabla 3.4 Resultados de la Prueba en los participantes que realizaron terapia convencional.
Fuente: *Autor*

Terapia Convencional		
	Mano Derecha	Mano Izquierda
Participante 1	10	12
Participante 2	8	6
Participante 3	13	8
Participante 4	9	5
Participante 5	30	15
Participante 6	24	28
Participante 7	14	10
Participante 8	20	18
Participante 9	21	21
Participante 10	16	18
Participante 11	9	8
Participante 12	12	11
Participante 13	17	19
Promedio	15,62	13,77

De la misma manera en la Tabla 3.5 se muestran los resultados de la prueba cuando se finalizó los días de la terapia con el sistema propuesto.

Tabla 3.5 Resultados de la Prueba de motricidad en los participantes que realizaron la terapia basada en videojuegos
Fuente: *Autor*

Sistema Propuesto		
	Mano Derecha	Mano Izquierda
Participante 1	13	10
Participante 2	9	8
Participante 3	17	9
Participante 4	11	7
Participante 5	35	18
Participante 6	27	26
Participante 7	19	13
Participante 8	25	20
Participante 9	28	26
Participante 10	19	20
Participante 11	15	10
Participante 12	14	12
Participante 13	20	16
Promedio	19,38	15,00

Teniendo en cuenta los valores promedio obtenidos se realiza una comparación que se representa en la figura.3.24.

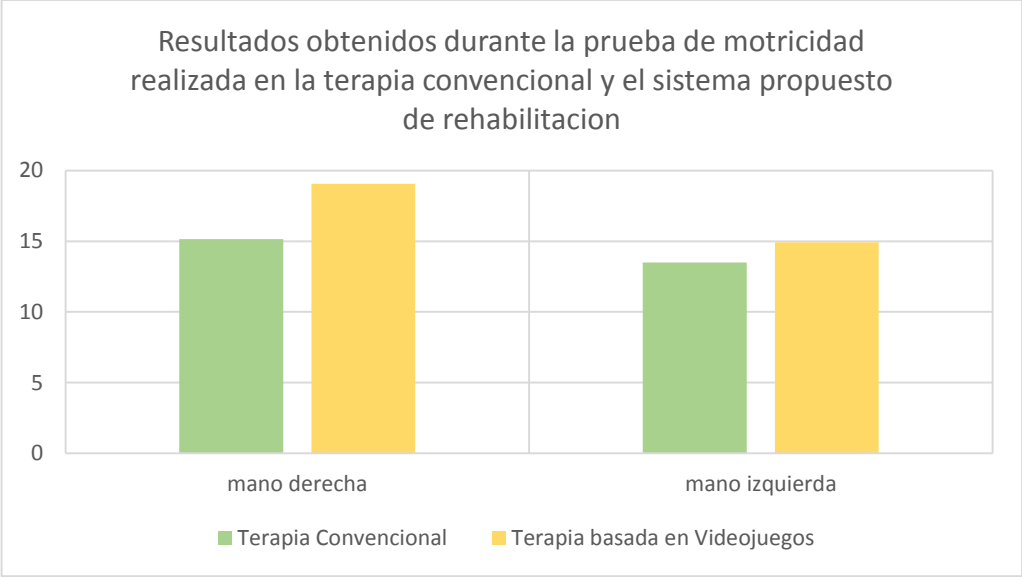


Figura 3.24 *Resultados obtenidos de la prueba de motricidad aplicada al finalizar cada terapia de rehabilitación*
Fuente: *Autor*

Como se observa en la figura 3.24, los participantes que utilizaron la terapia convencional obtuvieron un promedio de 15,62, mientras que los participantes que utilizaron el sistema propuesto de terapia basada en videojuegos obtuvieron un

promedio de 19,38 usando la mano derecha. Para la mano izquierda, se obtiene un promedio de 13,77 con la terapia convencional y 15 con el sistema propuesto, denotando cierta mejora para ambas manos.

3.6 GRADO DE SATISFACCIÓN DEL SISTEMA PROPUESTO

De la misma manera, al finalizar las pruebas en cada terapia se realizó una encuesta de grado de satisfacción, cuyo modelo se muestra en la figura 3.21 lo que otorgo los resultados mostrados en la figura 3.25.

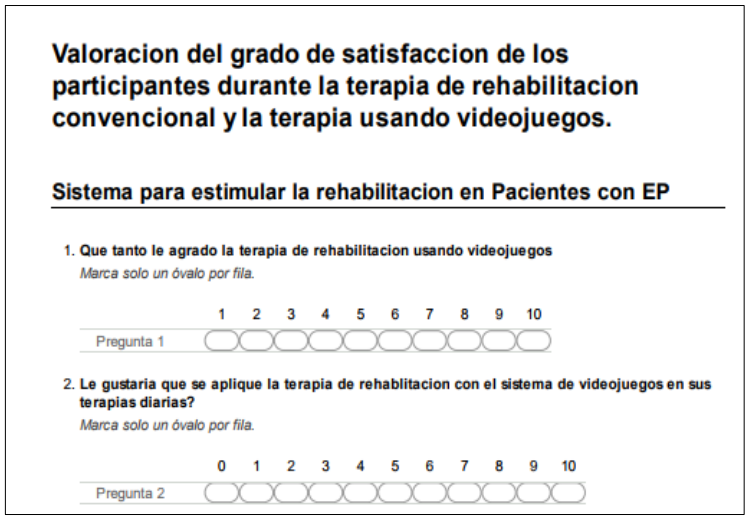


Figura 3.25 *Modelo de encuesta realizada al finalizar las pruebas de las terapias en los participantes*
Fuente: *Autor*

Los resultados que otorgo la encuesta se muestran en las figuras 3.26 y 3.27 para las dos preguntas realizadas

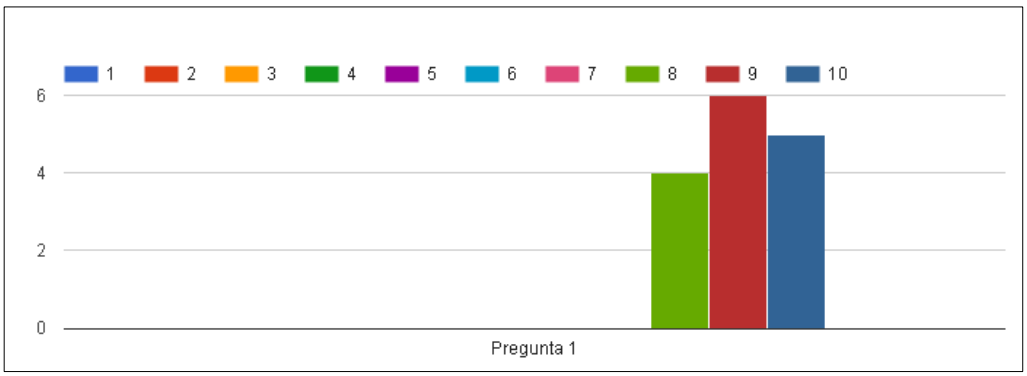


Figura 3.26 *Resultados obtenidos de la pregunta: "Que tanto le agrado la terapia de rehabilitación usando videojuegos"*
Fuente: *Autor*

De acuerdo con la Figura 3.26, el 26,66% de los participantes expresaron un grado de aceptación del sistema en 8, el 40% representa un grado 9, y el 33.33% tuvieron 10 grados de aceptación, lo que da una media ponderada de 9,07 sobre 10.

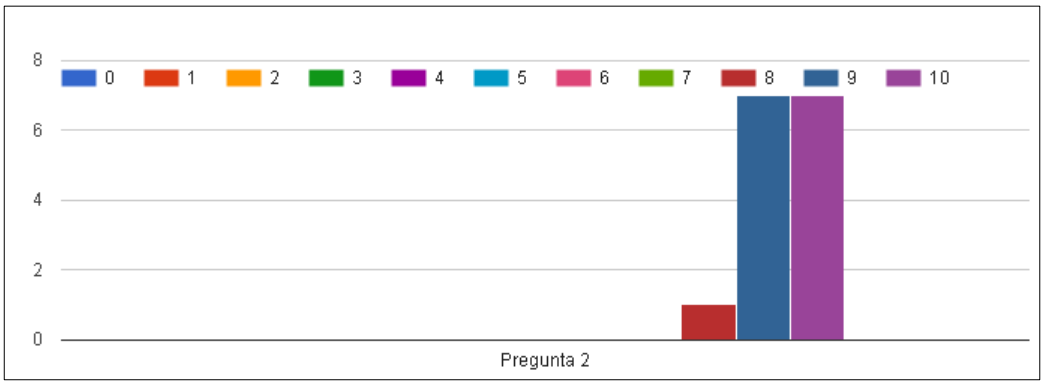


Figura 3.27 *Resultados obtenidos de la pregunta: "Le gustaría que aplique la terapia de rehabilitación son el sistema de videojuegos en su terapia convencional"*
Fuente: *Autor*

En relación con la pregunta dos del cuestionario acerca de si debería aplicar el sistema en las terapias diarias, el 93,33% indicaron el grado 9 y 10 de aceptación, mientras que el 6.66% optaron por el grado 8 de aceptación, teniendo una media ponderada de 9,40 sobre 10.

3.7 ANÁLISIS DE RESULTADOS

Teniendo en cuenta los resultados obtenidos con los pacientes con EP durante el mini juego PlaneGame al finalizar la quinta sesión se obtiene una disminución de tiempo del 20.3% en la mano derecha y 23.3% usando la mano izquierda, véase la Tabla 3.6.

Tabla 3.6 Variación en Función del Tiempo en Pacientes con EP. PlaneGame
Fuente: Autor

Tiempo en seg. Plane Game en pacientes con EP			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	61,88	49,33	-20,3%
Izquierda	65,66	50,33	-23,3%
Promedio	63,77	49,83	-21,9%

En relación con el puntaje hubo un aumento del 100% y 145,5% usando la mano derecha e izquierda respectivamente, esto se observa en la Tabla y 3.7.

Tabla 3.7 Variación en Función del Puntaje en Pacientes con EP. PlaneGame
Fuente: Autor

Puntaje de Plane Game en Pacientes con EP			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	37,5	75	100,0%
Izquierda	27,5	67,5	145,5%
Promedio	32,5	71,25	119,2%

Al finalizar el mini juego BallGame en Pacientes con EP, se obtuvo una disminución del tiempo de juego con relación a la primera sesión de 22,0% usando la mano derecha y de 17,8% usando la mano izquierda y 3.8. Véase la Tabla 3.8

Tabla 3.8 Variación en Función del Tiempo en Pacientes con EP. BallGame
Fuente: Autor

Tiempo [seg] de Ball Game en Pacientes con EP			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	60,99	47,57	-22,0%
Izquierda	60,08	49,41	-17,8%
Promedio	60,535	48,49	-19,9%

De la misma manera, la respuesta de acuerdo con la puntuación presenta mejoras ya que al finalizar la quinta sesión con respecto a la primera el puntaje máximo es de 100 puntos usando la mano derecha y 90 puntos con la mano izquierda, con un aumento en la mediana de 106,7% y 81,3% respectivamente. Véase la Tabla 3.9.

Tabla 3.9 Variación en Función del Puntaje en Pacientes con EP. BallGame
Fuente: Autor

Puntaje de Ball Game en Pacientes con EP			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	37,5	77,5	106,7%
Izquierda	40	72,5	81,3%
Promedio	38,75	75	93,5%

Por otra parte, los resultados se presentan para los pacientes sanos, es decir, que no poseen trastornos neurodegenerativos, por lo que los valores de puntaje y tiempo son mejores con relación a los pacientes con EP, sin embargo, durante las sesiones también existe un incremento de puntaje y disminución de tiempo de juego para pasar cada fase. Al concluir las pruebas con el mini juego PlaneGame se obtuvo una mejora con respecto a la primera sesión, en la disminución del tiempo de 8,0% con la mano derecha y 16,3% con la mano izquierda. Véase la Tabla 3.10.

Tabla 3.10 Variación en Función del Tiempo en Pacientes Sanos. PlaneGame
Fuente: Autor

Tiempo [seg] de Plane Game en Pacientes Sanos			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	50,12	46,13	-8,0%
Izquierda	55,04	46,05	-16,3%
Promedio	52,58	46,09	-12,3%

El puntaje aumenta un 135,3% con la mano derecha y 137,5% usando la mano izquierda, tal como se muestra en la Tabla 3.11

Tabla 3.11 Variación en Función del Puntaje en Pacientes Sanos. PlaneGame
Fuente: Autor

Puntaje de Plane Game en Pacientes Sanos			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	42,5	100	135,3%
Izquierda	40	95	137,5%
Promedio	41,25	97,5	136,4%

Con el mini juego Ball se obtienen resultados similares, ya que al concluir la quinta sesión el puntaje crece con relación a la primera sesión un 158,8% usando la mano derecha y 175,0% usando la mano izquierda, tal como se muestra en la Tabla 3.12

Tabla 3.12 Variación en función del Puntaje en Pacientes Sanos. BallGame
Fuente: Autor

Puntaje de Ball Game en Pacientes Sanos			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	42,5	110	158,8%
Izquierda	40	110	175,0%
Promedio	41,25	110	166,7%

Mientras que, el tiempo disminuye 30,1% y 16,6% respectivamente, obteniendo un promedio de 23,7% de disminución del tiempo de juego. Tal como se observa en la Tabla 3.13.

Tabla 3.13 Variación en función del Tiempo en Pacientes Sanos. BallGame
Fuente: Autor

Tiempo [seg] de Ball Game en Pacientes Sanos			
Mano	**Sesión 1**	**Sesión 5**	**Variación Porcentual**
Derecha	54,1	37,83	-30,1%
Izquierda	49,04	40,88	-16,6%
Promedio	51,57	39,355	-23,7%

En la tabla 3.14 se presenta los resultados obtenidos al finalizar la quinta sesión de pruebas para los pacientes con EP y sanos usando el celular, donde claramente se puede establecer que los pacientes sanos obtuvieron mayor puntaje y menor tiempo de juego en comparación con los pacientes con EP.

Tabla 3.14. Datos obtenidos de los Pacientes con EP vs. Pacientes Sanos

	PACIENTES CON EP				**PACIENTES SANOS**			
	MANO DERECHA		**MANO IZQUIERDA**		**MANO DERECHA**		**MANO IZQUIERDA**	
	TIEMPO	**PUNTAJE**	**TIEMPO**	**PUNTAJE**	**TIEMPO**	**PUNTAJE**	**TIEMPO**	**PUNTAJE**
MINIJUEGO BALLGAME	47,57	77,5	49,41	72,5	37,83	110	40,88	110
MINIJUEGO PLANEGAME	49,33	75	65,66	67,5	46,13	100	49,69	95

En relación con las pruebas realizadas con la Tablet, según los datos mostradas en las gráficas 3.11 a 3.17 es notable las mejoras en los resultados de tiempo y puntaje para los dos grupos de participantes, ya que la Tablet posee mayor tamaño en la pantalla lo que permite que cada participante pueda observar con más claridad el videojuego.

Finalmente, se realizaron las pruebas de motricidad denotando buenos resultados, en general hubo un incremento del 20,60 % en la mano derecha y en la mano izquierda se incrementó un 9,5%, tal como se pudo observar en la Figura 3.24, además los participantes expresaron que el sistema propuesto para la terapia tiene un alto nivel de aceptación. Ver Figura 3.26

Capítulo 4 : CONCLUSIONES Y RECOMENDACIONES

La enfermedad de Parkinson es un trastorno neurodegenerativo que afecta el sistema motor y sus efectos con el pasar de los años, es la ausencia de movimiento en las extremidades superiores e inferiores, interfiriendo de manera física, económica y social. El tratamiento más común que se efectúa consiste en medicamentos y terapias de rehabilitación, por lo que en el presente proyecto se revisaron diferentes terapias, entre las más importantes están: terapia ocupacional, fisioterapia y logopedia. En la terapia ocupacional intervienen actividades para mejorar la vida diaria de las personas, tales como comer y vestirse, por otra parte, la fisioterapia permite mejorar la función motora de las extremidades y músculos del cuerpo, mientras que la logopedia mejora el habla y la pronunciación.

Se analizaron los sistemas de rehabilitación modernos, donde el ejercicio basado en videojuegos y equipos tecnológicos (Exergaming) se consideró como la mejor opción, ya que brinda diversión a la hora de la rehabilitación y permite la mejora de los movimientos del cuerpo en general. Siendo su principal ventaja respecto a la terapia convencional, la misma que implica gastos económicos, consumo de tiempo y baja motivación provocando que el paciente comúnmente pierda el interés.

Una desventaja de los Exergames comerciales es que utilizan sistemas de juegos basados en Nintendo Wii o Kinect, que implica manipular consolas, controles o palancas, que resultan difíciles para los adultos mayores, que es la edad en la que se encuentran la mayoría de los pacientes con EP

El videojuego realizado utilizó el motor de Unity, debido a que posee la ventaja de exportar a varias plataformas, como Android. Además, posee más de dos lenguajes de programación y una tienda online con escenarios y objetos, lo que permitió una mayor familiarización con el software y el desarrollo de la aplicación.

El rendimiento de Unity se ve afectado conforme aumentan el tamaño de los elementos de juego, sobre todo si están en 3D, provocando costes en forma de calor.

Teniendo en consideración esto se realizó una optimización general del juego, eliminado elementos que ocupaban demasiada memoria que hacían lento el juego, mejorando la iluminación y el funcionamiento de scripts.

Con respecto al dispositivo, los elementos utilizados para su diseño poseen características de software libre y hardware de bajo costo facilitando el uso y la mejora del mismo por parte de otros usuarios. El hardware del dispositivo tiene un costo alrededor de 45$, que es un valor marginal con respecto al costo de equipos comerciales como Nintendo Wii (769$) y Xbox Kinect (499,99$)

Las pruebas realizadas con la Tablet demostraron cierta mejora con respecto a las pruebas realizadas con el celular en la disminución del tiempo y el aumento del puntaje durante el juego, conforme se pudo observar en las Figuras 3.11 a 3.18.

Durante las pruebas de validación la Tabla 3.2, demuestra que el mini juego BallGame agrado más al género femenino, mientras que el género masculino se inclinó hacia el mini juego PlaneGame. Según los resultados de la Figura 3.26 y la pregunta tres de la Tabla 3.4 permiten establecer que el videojuego tuvo un alto grado de satisfacción (9.06/10) y que el dispositivo se consideró bastante cómodo (100%).

De acuerdo con los resultados de las pruebas de motricidad realizadas en la UAM, los participantes obtuvieron mejor puntuación cuando terminaron la terapia con el sistema propuesto, que cuando utilizaron la terapia convencional.

Según la Figura 3.24, el sistema es de gran ayuda para la rehabilitación de los adultos mayores con EP, ya que se presentaron mejoras en la motricidad y movilidad de los brazos, codo y mano. Y a la vez, los participantes se entretuvieron ejerciendo movimientos en las extremidades superiores.

Al trabajar con Unity, es recomendable no incorporar demasiados objetos en los niveles ya que disminuye el rendimiento del juego, provocando que el dispositivo Android se caliente.

Para las primeras pruebas, se recomienda que exista una persona para que explique las indicaciones de juego a los pacientes con EP.

REFERENCIAS BIBLIOGRÁFICAS

[1] F. Li *et al.*, "Tai Chi and Postural Stability in Patients with Parkinson's Disease," *N. Engl. J. Med.*, vol. 366, no. 6, pp. 511–519, Feb. 2012.

[2] C. J. Winstein, P. S. Pohl, and R. Lewthwaite, "Effects of physical guidance and knowledge of results on motor learning: support for the guidance hypothesis.," *Res. Q. Exerc. Sport*, vol. 65, no. 4, pp. 316–323, Dec. 1994.

[3] P. Rego, P. M. Moreira, and L. P. Reis, "Serious Games for Rehabilitation A Survey and a Classification Towards a Taxonomy," *5th Iber. Conf. Inf. Syst. Technol.*, no. November 2015, pp. 1–6, 2010.

[4] M. Pirovano, E. Surer, R. Mainetti, P. Luca, and N. A. Borghese, "Exergaming and rehabilitation : A methodology for the design of effective and safe therapeutic exergames q," *Entertain. Comput.*, vol. 14, pp. 55–65, 2016.

[5] A. Kliem and J. Wiemeyer, "Comparison of a traditional and a Video Game Based Balance Training Program," *Comput. Sci. Sport*, vol. 9, no. January 2010, 2014.

[6] J. E. Deutsch, M. Borbely, J. Filler, K. Huhn, and P. Guarrera-Bowlby, "Use of a low-cost, commercially available gaming console (Wii) for rehabilitation of an adolescent with cerebral palsy.," *Phys. Ther.*, vol. 88, no. 10, pp. 1196–1207, Oct. 2008.

[7] L. Y. Joo *et al.*, "A feasibility study using interactive commercial off-the-shelf computer gaming in upper limb rehabilitation in patients after stroke," *J. Rehabil. Med.*, vol. 42, no. 5, pp. 437–441, 2010.

[8] A. Mirelman, I. Maidan, and J. E. Deutsch, "Virtual Reality and Motor Imagery : Promising Tools for Assessment and Therapy in Parkinson ' s disease," vol. 28, no. 11, pp. 1597–1608, 2013.

[9] G. Barry, B. Galna, and L. Rochester, "The role of exergaming in Parkinson ' s disease rehabilitation : a systematic review of the evidence," 2014.

[10] C. Aison, G. Davis, J. Milner, and E. Targum, "Appeal and interest of video game use among the elderly," *Harvard Grad. Sch. Educ. 2002), i-28. DOI= http//www. jrmilner. com*, 2002.

[11] B. Galna *et al.*, “Retraining function in people with Parkinson’s disease using the Microsoft kinect: game design and pilot testing.,” *J. Neuroeng. Rehabil.*, vol. 11, no. 1, p. 60, 2014.

[12] B. S. Lange *et al.*, “The potential of virtual reality and gaming to assist successful aging with disability,” *Phys. Med. Rehabil. Clin. N. Am.*, vol. 21, no. 2, pp. 339–356, 2010.

[13] I. Pachoulakis and N. Papadopoulos, “Exergames for Parkinson ’ s Disease Patients : the Balloon Goon game,” pp. 0–5, 2016.

[14] O. Assad *et al.*, “WuppDi ! – Supporting Physiotherapy of Parkinson ´ s Disease Patients via Motion-based Gaming,” no. January, 2011.

[15] S. Date, T. Tashiro, K. Nozaki, H. Nakamura, S. Sakoda, and S. Shimojo, “A grid-ready clinical database for Parkinson’s disease research and diagnosis,” *Proc. - IEEE Symp. Comput. Med. Syst.*, pp. 483–488, 2007.

[16] H. Mohammadi-Abdar, A. L. Ridgel, F. M. Discenzo, and K. A. Loparo, “Design and Development of a Smart Exercise Bike for Motor Rehabilitation in Individuals with Parkinson’s Disease,” *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 3, pp. 1650–1658, 2016.

[17] “Homepage | National Parkinson Foundation.” [Online]. Available: http://www.parkinson.org/. [Accessed: 28-Aug-2017].

[18] M. Spatola and C. Wider, “Parkinsonism and Related Disorders Genetics of Parkinson ’ s disease : the yield,” *Park. Realt. Disord.*, vol. 20, pp. S35–S38, 2014.

[19] E. Fertl, A. Doppelbauer, and E. Auff, “Physical activity and sports in patients suffering from Parkinson’s disease in comparison with healthy seniors.,” *J. Neural Transm. Park. Dis. Dement. Sect.*, vol. 5, no. 2, pp. 157–161, 1993.

[20] J. a Opara, W. Brola, M. Leonardi, and B. Błaszczyk, “Quality of life in Parkinson’s disease.,” *J. Med. Life*, vol. 5, no. 4, pp. 375–81, 2012.

[21] C. Ayán, J. M. Cancela, P. Rodríguez, P. Ríos, and N. Abal, “Mejora del equilibrio en los enfermos de Parkinson mediante el ejercicio calisténico-recreativo: un estudio piloto,” *Rehabilitación*, vol. 47, no. 1, pp. 22–26, 2013.

[22] R. Kizony, P. L. (Tamar) Weiss, M. Shahar, and D. Rand, “TheraGame: A home based virtual reality rehabilitation system,” *Int. J. Disabil. Hum. Dev.*, vol. 5, no. 3, pp. 209–214, 2006.

[23] J. Parkinson, “An Essay on the Shaking Palsy,” pp. 223–236, 2002.

[24] S. Gandhi and N. W. Wood, “Molecular pathogenesis of Parkinson ’ s disease,” vol. 14, no. 18, pp. 2749–2755, 2005.

[25] R. Savica, B. R. Grossardt, J. H. Bower, J. E. Ahlskog, and W. A. Rocca, “Risk factors for Parkinson’s disease may differ in men and women: an exploratory study.,” *Horm. Behav.*, vol. 63, no. 2, pp. 308–314, Feb. 2013.

[26] L. Hirsch and T. Steeves, “The Incidence of Parkinson ’ s Disease : A Systematic Review and Meta-Analysis,” pp. 292–300, 2016.

[27] L. M. L. de Lau and M. M. B. Breteler, “Epidemiology of Parkinson’s disease,” *Lancet. Neurol.*, vol. 5, no. 6, pp. 525–535, Jun. 2006.

[28] R. L. Nussbaum and C. E. Ellis, “Alzheimer’s disease and Parkinson’s disease,” *N. Engl. J. Med.*, vol. 348, no. 14, pp. 1356–1364, Apr. 2003.

[29] T. Hatano, S. Kubo, S. Sato, and N. Hattori, “Pathogenesis of familial Parkinson’s disease: new insights based on monogenic forms of Parkinson’s disease,” *J. Neurochem.*, vol. 111, no. 5, pp. 1075–1093, Dec. 2009.

[30] J. Jankovic, “Parkinson ’ s disease : clinical features and diagnosis,” no. 1957, pp. 368–376, 2008.

[31] Oms, “Trastornos neurológicos: desafíos para la salud pública.,” *Salud Ment.*, pp. 29–45, 2006.

[32] H. Cianci, “Parkinson ’ s Disease : Fitness Counts.”

[33] B. Thomas and M. F. Beal, “Parkinson’s disease,” *Hum. Mol. Genet.*, vol. 16 Spec No, pp. R183-194, Oct. 2007.

[34] T. Pringsheim, N. Jette, A. Frolkis, and T. D. L. Steeves, “The prevalence of Parkinson’s disease: a systematic review and meta-analysis,” *Mov. Disord. Off. J. Mov. Disord. Soc.*, vol. 29, no. 13, pp. 1583–1590, Nov. 2014.

[35] C. M. Lill, “Genetics of Parkinson’s disease,” *Mol. Cell. Probes*, vol. 30, no. 6, pp. 386–396, Dec. 2016.

[36] E. Tolosa, G. Wenning, and W. Poewe, “The diagnosis of Parkinson ’ s disease,” pp. 75–86, 2006.

[37] R. B. Postuma, D. Berg, M. Stern, W. Poewe, K. Marek, and I. Litvan, “MDS Clinical Diagnostic Criteria for Parkinson ’ s Disease,” vol. 30, no. 12, 2015.

[38] N. Review and I. Data, “When Does Parkinson Disease Start?,” vol. 67, no. 7, pp. 798–801, 2015.

[39] F. M. Skidmore *et al.*, “Apathy, depression, and motor symptoms have distinct and separable resting activity patterns in idiopathic Parkinson disease.,”

Neuroimage, vol. 81, pp. 484–495, Nov. 2013.

[40] A. B. Neikrug *et al.*, "Parkinson's disease and REM sleep behavior disorder result in increased non-motor symptoms.," *Sleep Med.*, vol. 15, no. 8, pp. 959–966, Aug. 2014.

[41] M. C. Chen, H. Yu, Z.-L. Huang, and J. Lu, "Rapid eye movement sleep behavior disorder," *Curr. Opin. Neurobiol.*, vol. 23, no. 5, pp. 793–798, Oct. 2013.

[42] C. Hawkes, "Olfaction in neurodegenerative disorder.," *Adv. Otorhinolaryngol.*, vol. 63, pp. 133–151, 2006.

[43] R. B. Postuma, J. F. Gagnon, and J. Montplaisir, "Clinical prediction of Parkinson ' s disease : planning for the age of neuroprotection," 2010.

[44] M. A. Nalls *et al.*, "Diagnosis of Parkinson ' s disease on the basis of clinical and genetic classifi cation : a population-based modelling study," vol. 4422, no. 15, pp. 1–8, 2015.

[45] M. M. Hoehn and M. D. Yahr, "Parkinsonism: onset, progression and mortality.," *Neurology*, vol. 17, no. 5, pp. 427–442, May 1967.

[46] A. Cervantes-arriaga, "La escala unificada de la enfermedad de Parkinson modificada por la Sociedad de Trastornos del Movimiento (MDS-UPDRS): aplicación clínica e investigación," vol. 19, no. 3, pp. 157–163, 2014.

[47] C. G. Goetz *et al.*, "Movement Disorder Society-sponsored revision of the Unified Parkinson's Disease Rating Scale (MDS-UPDRS): Process, format, and clinimetric testing plan," *Mov. Disord.*, vol. 22, no. 1, pp. 41–47, 2007.

[48] N. Tambasco *et al.*, "A magnetization transfer study of mild and advanced Parkinson's disease," *Eur. J. Neurol.*, vol. 18, no. 3, pp. 471–477, 2011.

[49] D. J. Brooks, "Parkinson's disease: Diagnosis," *Parkinsonism Relat. Disord.*, vol. 18, pp. S31–S33, Mar. 2017.

[50] L. Brichta, P. Greengard, and M. Flajolet, "Advances in the pharmacological treatment of Parkinson ' s disease : targeting neurotransmitter systems," *Trends Neurosci.*, no. Table 1, pp. 1–12, 2013.

[51] J. J. Ferreira, R. Katzenschlager, B. R. Bloem, U. Bonuccelli, D. Burn, and G. Deuschl, "Summary of the recommendations of the EFNS / MDS-ES review on therapeutic management of Parkinson ' s disease," pp. 5–15, 2013.

[52] P. S. Group, "Levodopa and the progression of Parkinson's disease," *N Engl J Med*, vol. 2004, no. 351, pp. 2498–2508, 2004.

[53] L. Defebvre and C. Moreau, “Traitements médical et chirurgical de la maladie de Parkinson Dossier thématique,” *Presse Med.*, 2017.

[54] S. Ku and G. A. Glass, “Age of Parkinson’s disease onset as a predictor for the development of dyskinesia,” *Mov. Disord.*, vol. 25, no. 9, pp. 1177–1182, 2010.

[55] P. F. Worth, “How to treat Parkinson’s disease in 2013.,” *Clin. Med.*, vol. 13, no. 1, pp. 93–96, Feb. 2013.

[56] A. Wagle Shukla and M. S. Okun, “Surgical Treatment of Parkinson’s Disease: Patients, Targets, Devices, and Approaches,” *Neurotherapeutics*, vol. 11, no. 1, pp. 47–59, 2014.

[57] C. J. Goodwin VA, Richards SH, Taylor RS, Taylor AH, “The effectiveness of exercise interventions for people with Parkinson’s disease: a systematic review and meta-analysis,” pp. 631–40, 2008.

[58] G. Frazzitta *et al.*, “Effectiveness of intensive inpatient rehabilitation treatment on disease progression in parkinsonian patients: a randomized controlled trial with 1-year follow-up,” *Neurorehabil. Neural Repair*, vol. 26, no. 2, pp. 144–150, 2012.

[59] G. Frazzitta *et al.*, “Multidisciplinary intensive rehabilitation treatment improves sleep quality in Parkinson ’ s disease,” pp. 4–11, 2015.

[60] N. M. van der Kolk and L. A. King, “Effects of exercise on mobility in people with Parkinson’s disease.,” *Mov. Disord.*, vol. 28, no. 11, pp. 1587–1596, Sep. 2013.

[61] G. Ebersbach, A. Ebersbach, F. Gandor, B. Wegner, and A. Kupsch, “Impact of Physical Exercise on Reaction Time in Patients With Parkinson ’ s Disease d Data From the Berlin BIG Study,” pp. 996–999, 2014.

[62] R. S. Monteiro-Junior *et al.*, “We need to move more: Neurobiological hypotheses of physical exercise as a treatment for Parkinson’s disease,” *Med. Hypotheses*, vol. 85, no. 5, pp. 537–541, Nov. 2015.

[63] “Guía de Práctica Clínica para el Manejo de Pacientes con Enfermedad de Parkinson,” 2014.

[64] D. Journal, “KNGF Guidelines Parkinson â€™ s disease,” vol. 114, no. 3, 2004.

[65] T. Herman, N. Giladi, and J. M. Hausdorff, “Treadmill training for the treatment of gait disturbances in people with Parkinson’s disease: a mini-

review," *J. Neural Transm.*, vol. 116, no. 3, pp. 307–318, 2009.

[66] Q. Gao *et al.*, "Effects of Tai Chi on balance and fall prevention in Parkinson's disease: a randomized controlled trial," *Clin. Rehabil.*, vol. 28, no. 8, pp. 748–753, 2014.

[67] S. G. Reich, W. J. Weiner, and R. F. Macko, "Randomized Clinical Trial of 3 Types of Physical Exercise for Patients With Parkinson Disease," vol. 70, no. 2, pp. 183–190, 2013.

[68] M. Villalobos, "Consejos para mejorar el estado físico de pacientes con enfermedad de parkinson," *Asoc. Enfermos Park.*, 2008.

[69] "Terapia Fisica en la Enfermedad de Parkinson." [Online]. Available: http://www.terapia-fisica.com/parkinson/. [Accessed: 28-Aug-2017].

[70] "Tratamiento fisioterápico en pacientes con Parkinson | eFisioterapia." [Online]. Available: https://www.efisioterapia.net/articulos/tratamiento-fisioterapico-pacientes-parkinson. [Accessed: 28-Aug-2017].

[71] A. F. C, T. M. C. M, and C. B. I, "Neurorehabilitation in Parkinson's disease," *Neurol Supl*, vol. 3, no. 7, pp. 22–29, 2007.

[72] I. Pachoulakis, N. Papadopoulos, and C. Spanaki, "Are Game Platforms suitable for Parkinson Disease patients ?," pp. 1–6.

[73] J. E. Deutsch, I. Maidan, and R. Dickstein, "Patient-centered integrated motor imagery delivered in the home with telerehabilitation to improve walking after stroke.," *Phys. Ther.*, vol. 92, no. 8, pp. 1065–1077, Aug. 2012.

[74] A. Mirelman, I. Maidan, T. Herman, J. E. Deutsch, N. Giladi, and J. M. Hausdorff, "Virtual reality for gait training: can it induce motor learning to enhance complex walking and reduce fall risk in patients with Parkinson's disease?," *J. Gerontol. A. Biol. Sci. Med. Sci.*, vol. 66, no. 2, pp. 234–240, Feb. 2011.

[75] E. Bigotte and S. Pires, "Executive Function Assessment in Parkinson ' s Disease Patients using Mobile Devices."

[76] A. Guillot and C. Collet, "Contribution from neurophysiological and psychological methods to the study of motor imagery.," *Brain Res. Brain Res. Rev.*, vol. 50, no. 2, pp. 387–397, Dec. 2005.

[77] G. Riva, "Applications of virtual environments in medicine.," *Methods Inf. Med.*, vol. 42, no. 5, pp. 524–534, 2003.

[78] E. D. de Bruin, D. Schoene, G. Pichierri, and S. T. Smith, "Use of virtual

reality technique for the training of motor control in the elderly. Some theoretical considerations.," *Z. Gerontol. Geriatr.*, vol. 43, no. 4, pp. 229–234, Aug. 2010.

[79] T. Esculier JF, Vaudrin J, Beriault P, Gagnon K, "Home-Based Balance Training Programme Using Wii Fit with Balance Board for Parkinson's Disease: A Pilot Study," *J Rehabil Med*, vol. 44:144–150, 2012.

[80] D. M. Harris, T. Rantalainen, M. Muthalib, L. Johnson, and W.-P. Teo, "Exergaming as a Viable Therapeutic Tool to Improve Static and Dynamic Balance among Older Adults and People with Idiopathic Parkinson's Disease: A Systematic Review and Meta-Analysis ," *Frontiers in Aging Neuroscience* , vol. 7. p. 167, 2015.

[81] C. G. Ribas, L. Alves da Silva, M. R. Correa, H. G. Teive, and S. Valderramas, "Effectiveness of exergaming in improving functional balance, fatigue and quality of life in Parkinson's disease: A pilot randomized controlled trial.," *Parkinsonism Relat. Disord.*, Feb. 2017.

[82] E. A. Keshner, "Virtual reality and physical rehabilitation: a new toy or a new research and rehabilitation tool?," *J Neuroeng Rehabil*, vol. 3, 2004.

[83] F. A. dos Santos Mendes *et al.*, "Motor learning, retention and transfer after virtual-reality-based training in Parkinson's disease–effect of motor and cognitive demands of games: a longitudinal, controlled clinical study," *Physiotherapy*, vol. 98, no. 3, pp. 217–223, 2012.

[84] J. E. Pompeu *et al.*, "Effect of Nintendo Wii™-based motor and cognitive training on activities of daily living in patients with Parkinson's disease: A randomised clinical trial," *Physiotherapy*, vol. 98, no. 3, pp. 196–204, 2012.

[85] P. V Mhatre *et al.*, "Wii Fit balance board playing improves balance and gait in Parkinson disease," *Pm&r*, vol. 5, no. 9, pp. 769–777, 2013.

[86] N. B. Herz, S. H. Mehta, K. D. Sethi, P. Jackson, P. Hall, and J. C. Morgan, "Nintendo Wii rehabilitation ('Wii-hab') provides benefits in Parkinson's disease," *Parkinsonism Relat. Disord.*, vol. 19, no. 11, pp. 1039–1042, 2013.

[87] A. Dos Santos, F. Pegollo, R. Alencar, R. Avanzi, and J. E. Pompeu, "A new tool for assessment and balance training of patients with Parkinson's disease based on low cost comercial Wii balance board," in *MOVEMENT DISORDERS*, 2012, vol. 27, pp. S301–S301.

[88] M. V Alvarez, P. M. Grogan, and M. Rodriguez, "Connecting with Kinect (R)

to improve motor and gait function in Parkinson's disease," in *MOVEMENT DISORDERS*, 2012, vol. 27, pp. S297–S297.

[89] O. Assad *et al.*, "Motion-based games for Parkinson's disease patients," in *International Conference on Entertainment Computing*, 2011, pp. 47–58.

[90] A. C. Macedo, R. Prada, P. A. Santos, J. J. Ferreira, and J. Domingos, "Serious Game for Motion Disorders Rehabilitation of Parkinson ' s Disease Patients."

[91] E. Flores, G. Tobon, E. Cavallaro, F. I. Cavallaro, J. C. Perry, and T. Keller, "Improving patient motivation in game development for motor deficit rehabilitation," in *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, 2008, pp. 381–384.

[92] M. J. M. Galindo, *Escaneando la informática*. Editorial UOC, 2010.

[93] Ó. T. Artero, *Arduino : curso práctico de formación*. RC Libros, 2013.

[94] S. USERS, *ELECTRÓNICA - Plataformas Arduino y Raspberry Pi: Plataformas Arduino y Raspberry Pi*. 2014.

[95] M. Á. Ramírez Luna, R. Favila Zavalza, and L. S. Valencia Castro, *Ambiente virtual para ubicación de aulas.* 2014.

[96] Unity, "Unity - Game Engine." [Online]. Available: https://unity3d.com/es/. [Accessed: 29-Aug-2017].

[97] Unity, "Unity - Manual: Escena." [Online]. Available: https://docs.unity3d.com/es/current/Manual/CreatingScenes.html. [Accessed: 29-Aug-2017].

[98] Unity, "Unity - Manual: GameObjects." [Online]. Available: https://docs.unity3d.com/es/current/Manual/GameObjects.html. [Accessed: 29-Aug-2017].

[99] Unity, "Unity - Manual: El Character Controller (Controlador del personaje)." [Online]. Available: https://docs.unity3d.com/es/current/Manual/class-CharacterController.html. [Accessed: 29-Aug-2017].

[100] Unity, "Unity - Manual: Colliders." [Online]. Available: https://docs.unity3d.com/es/current/Manual/CollidersOverview.html. [Accessed: 29-Aug-2017].

[101] Z. Cataldi and F. J. Lage, "TICs en Educación: Nuevas herramientas y nuevos paradigmas," in *VII Congreso de Tecnología en Educación y Educación en Tecnología*, 2012.

[102] M. Gargenta, D. T. Milano, N. Rapp, and M. Martin, *Hello , Android 3rd*, vol. 1163, no. 1–2. 2010.

[103] Abudeye, *Diseño e implementación de un módulo didáctico y desarrollo de aplicaciones electrónicas sobre plataforma Android*. 2015.

[104] "Pro Micro - 3.3V/8MHz - DEV-12587 - SparkFun Electronics." [Online]. Available: https://www.sparkfun.com/products/12587. [Accessed: 29-Aug-2017].

[105] M. . Villa, Suquilanda, T.V & Idrovo Torres, "Enlace, Diseño y Montaje de Enlace Inalambrico para transmision de datos utilizando tecnologia Bluetooth," 2013.

[106] E. P. Cobo, *Rehabilitación integral en el paciente con enfermedad pulmonar obstructiva crónica*. Editorial Médica Panamericana, 2007.

[107] E. Estudio, "Acelerómetro," *Electron. Estud. - Robot. y Mecatronica*, p. 1, 2013.

[108] "3-Axis Analog Accelerometer ADXL335." [Online]. Available: https://www.epictinker.com/3-Axis-Analog-Accelerometer-ADXL335-p/sen02241p.htm. [Accessed: 29-Aug-2017].

[109] M. Costa Marin, "Sistemas de juego comerciales como nuevas herramientas rehabilitación médica," Universitat Oberta de Catalunya.

[110] P. A. McNulty *et al.*, "The efficacy of Wii-based Movement Therapy for upper limb rehabilitation in the chronic poststroke period: a randomized controlled trial," *Int. J. Stroke*, vol. 10, no. 8, pp. 1253–1260, 2015.

[111] K. Thomson, A. Pollock, C. Bugge, and M. Brady, "Commercial Gaming Devices for Stroke Upper Limb Rehabilitation: A Systematic Review," *Int. J. Stroke*, vol. 9, no. 4, pp. 479–488, Mar. 2014.

[112] J. Sadaf, J. Scullion, A. Doyle, A. Low, and K. Coninx, "The effectiveness of gaming technology in neurological rehabilitation." [Online]. Available: https://www.physio-pedia.com/The_effectiveness_of_gaming_technology_in_neurological_rehabilitation. [Accessed: 02-Oct-2017].

[113] "Nintendo Wii Console White." [Online]. Available: https://www.newegg.com/Product/Product.aspx?Item=N82E16868100018. [Accessed: 02-Oct-2017].

[114] D. Levac, D. Espy, E. Fox, S. Pradhan, and J. E. Deutsch, "'Kinect-ing' With

Clinicians: A Knowledge Translation Resource to Support Decision Making About Video Game Use in Rehabilitation," *Phys. Ther.*, vol. 95, no. 3, pp. 426–440, Mar. 2015.

[115] "Doctor Kinetic." [Online]. Available: http://doctorkinetic.com/. [Accessed: 02-Oct-2017].

[116] "MIRA." [Online]. Available: http://www.mirarehab.com/. [Accessed: 02-Oct-2017].

[117] "Brontes Processing. SeeMe Rehabilitation." [Online]. Available: http://www.virtual-reality-rehabilitation.com/products/seeme/what-is-seeme. [Accessed: 02-Oct-2017].

[118] "Sense Your Progress Jintronix." [Online]. Available: http://www.jintronix.com/. [Accessed: 02-Oct-2017].

[119] "PLAY FOR HEALTH (P4H) v2.0." [Online]. Available: http://www.play4health.com/. [Accessed: 02-Oct-2017].

[120] L. Lopez-Samaniego and B. Garcia-Zapirain, "A Robot-Based Tool for Physical and Cognitive Rehabilitation of Elderly People Using Biofeedback," *Int. J. Environ. Res. Public Health*, vol. 13, no. 12, p. 1176, Dec. 2016.

[121] C. G. G. Carabeo, C. M. M. Dalida, E. M. Z. Padilla, and M. M. T. Rodrigo, "Stroke Patient Rehabilitation," *Simul. Gaming*, vol. 45, no. 2, pp. 151–166, Apr. 2014.

[122] N. Hocine, A. Gouaïch, S. A. Cerri, D. Mottet, J. Froger, and I. Laffont, "Adaptation in serious games for upper-limb rehabilitation: an approach to improve training outcomes," *User Model. User-adapt. Interact.*, vol. 25, no. 1, pp. 65–98, 2015.

[123] R. del E. Miniterio de Salud, "Informacion Estadisticas y Geografica de Salud." [Online]. Available: https://public.tableau.com/profile/publish/Perfildemorbilidadambulatoria2015.

[124] Ministerio de Salud Publica del Ecuador, "Perfil de morbilidad ambulatoria 2015 - Dirección Nacional de Estadística y Análisis de información de salud - DNEAIS | Tableau Public." [Online]. Available: https://public.tableau.com/profile/publish/Perfildemorbilidadambulatoria2015/Men#!/publish-confirm. [Accessed: 29-Aug-2017].

[125] Ministerio de Salud Publica del Ecuador, "Perfil de morbilidad ambulatoria 2016 - Dirección Nacional de Estadística y Análisis de información de salud -

DNEAIS | Tableau Public." [Online]. Available: https://public.tableau.com/profile/darwin5248#!/vizhome/Perfildemorbilidadambulatoria2016/Men?publish=yes. [Accessed: 29-Aug-2017].

[126] Instituto Nacional de Estadística y Censos, "Población y Demografía." [Online]. Available: http://www.ecuadorencifras.gob.ec/censo-de-poblacion-y-vivienda/. [Accessed: 05-Oct-2017].

[127] "Arduino-Map." [Online]. Available: https://www.arduino.cc/en/Reference/Map. [Accessed: 29-Aug-2017].

[128] Unity, "Asset Store." [Online]. Available: https://www.assetstore.unity3d.com/en/. [Accessed: 29-Aug-2017].

[130] "Android & Microcontrollers / Bluetooth - Asset Store." [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/16467. [Accessed: 29-Aug-2017].

[131] Unity, "Database Control (Free) - Asset Store." [Online]. Available: https://www.assetstore.unity3d.com/en/#!/content/41337. [Accessed: 29-Aug-2017].

[132] H. Llinás Solano and C. Rojas Álvarez, *Estadística descriptiva y distribuciones de probabilidad*. 2005.

APÉNDICES

APÉNDICE A: DATOS DE LOS PARTICIPANTES SANOS

En la Tabla A.1 se presenta las características de los participantes sanos de edad adulta que participaron en las pruebas de validación del sistema

Participante	Edad	Genero
Participante 1	67	Masculino
Participante 2	71	Masculino
Participante 3	73	Femenino
Participante 4	71	Femenino
Participante 5	71	Femenino
Participante 6	66	Femenino
Participante 7	58	Femenino
Participante 8	70	Femenino
Participante 9	65	Masculino
Participante 10	80	Femenino
Participante 11	82	Femenino
Participante 12	67	Masculino
Participante 13	63	Femenino
Participante 14	72	Masculino
Participante 15	85	Masculino
Participante 16	81	Femenino
Participante 17	70	Masculino
Participante 18	74	Femenino
Participante 19	62	Femenino
Participante 20	72	Masculino

Tabla A.1 *Datos de los Participantes Sanos*

Fuente: Autor

APÉNDICE B: DIMENSIONES DEL PROTECTOR PLÁSTICO PARA EL DISPOSITIVO

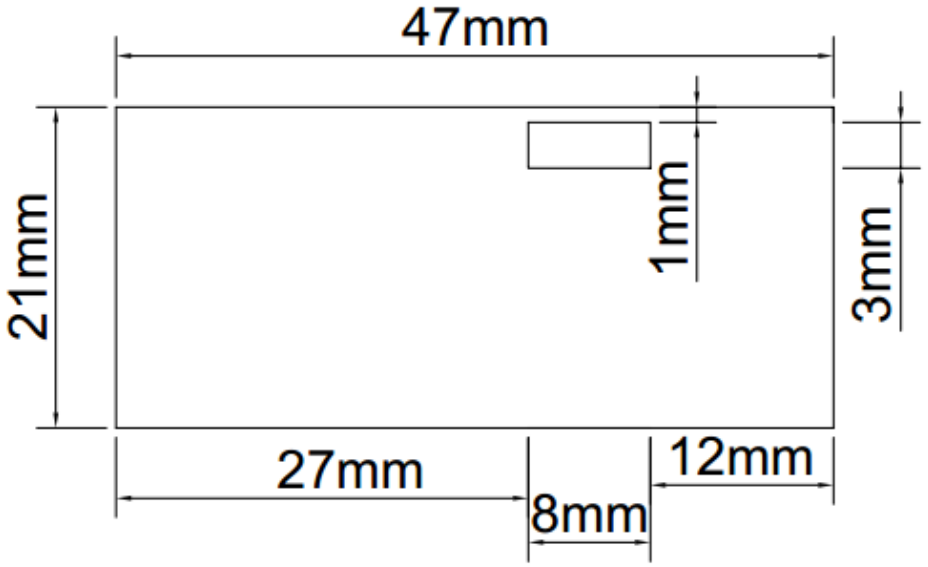


Figura B.1 *Vista Frontal de la Estructura protectora*

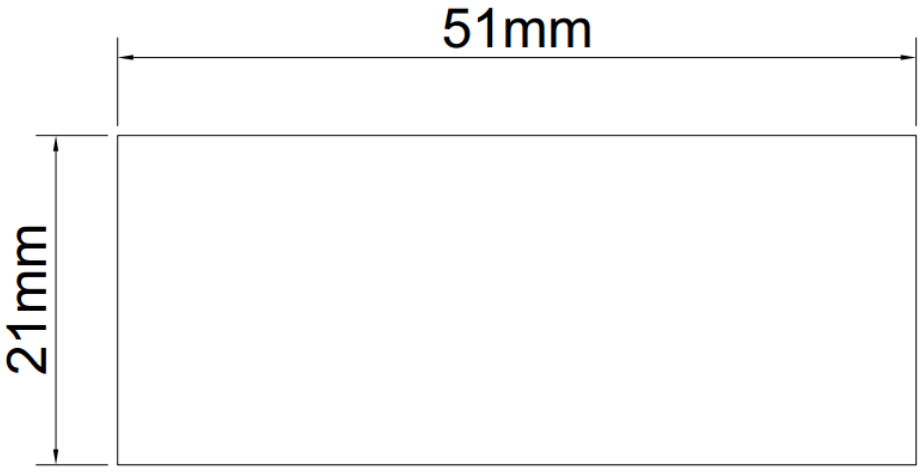


Figura B.2 *Vista Lateral Derecha e Izquierda de la Estructura protectora*

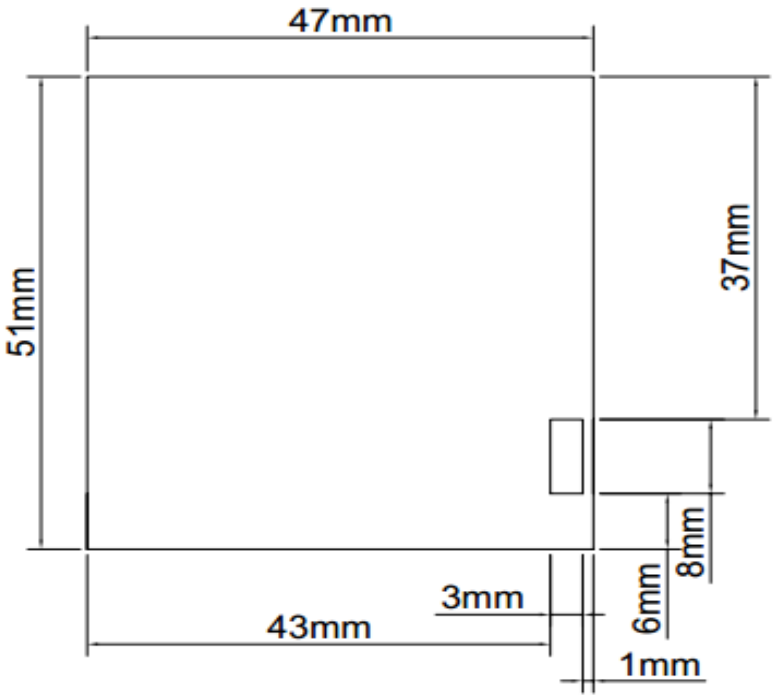


Figura B.3 *Vista Superior de la Estructura protectora*

APÉNDICE C: DIMENSIONES DE LA MUÑEQUERA

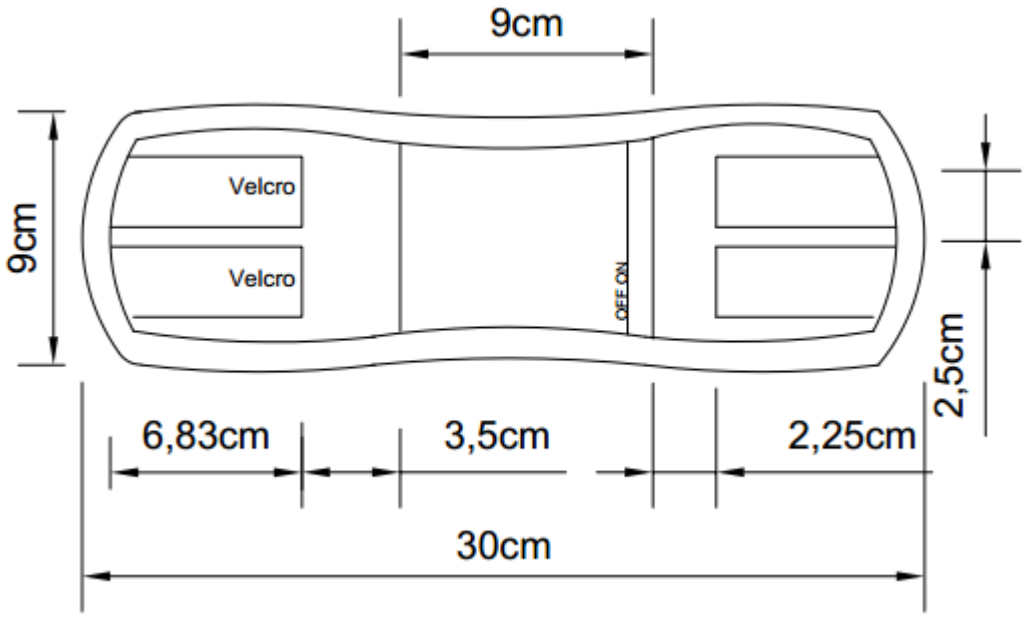


Figura C.1 *Vista Superior de la Muñequera*

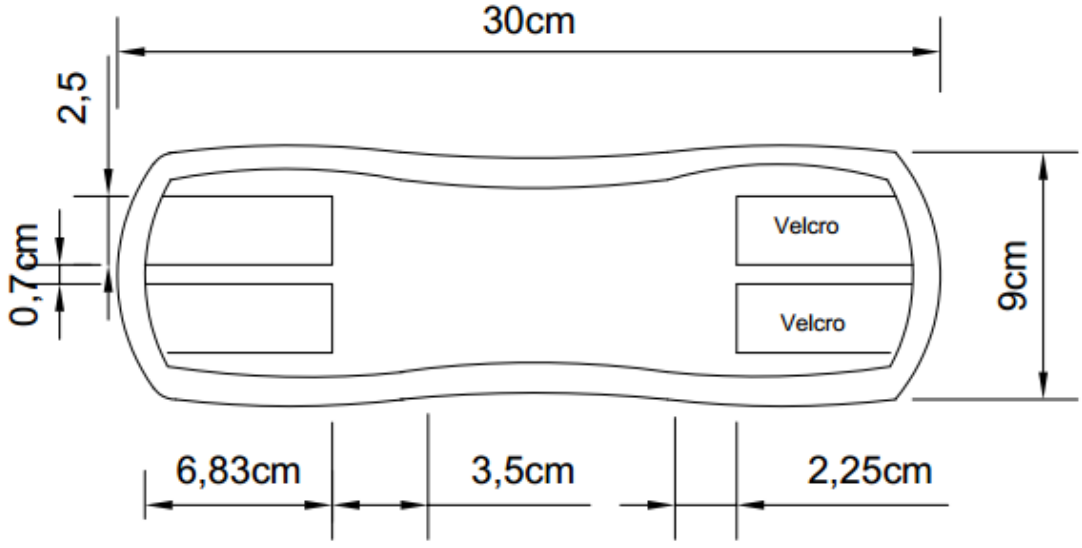


Figura C.2 *Vista Posterior de la Muñequera*

APÉNDICE D: PROGRAMACIÓN DEL MINI JUEGO PLANEGAME

SCRIPT PLAYERMOTOR.CS

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

public class PlayerMotor : MonoBehaviour
{

//DECLARACION DE VARIABLES
    Timer detenerTiempo; //variable llamada desde script Tiempo
    bool detenerVida;  //condicion de energia de vida
    public AudioClip sonidoGanar;  // variable de sonido de victoria
    public AudioClip sonidoPerder;
    private AudioSource Fuentes;
    private AudioSource[] todosLosSonidos;
    public GameObject mensajeRecarga;

    private CharacterController controlador;
    public float baseVelocidad ; //estaba 6
    public float velocidadRotacionX ;
    public GameObject minimapaa;

    private float valorX;

    public GameObject Tiempo;
    public GameObject Puntos;
    bool perder;

    private float deathTiempo;
    private float deathDuracion = 1;

    public GameObject efectoGanar;
    public GameObject deathExplosion;

    public GameObject Gameover;
    public GameObject MenuGana;
    public GameObject TablaPuntuacion; //ACTIVA MOSTRADOR DE PUNTOS
    public GameObject BotonPausa;
    public GameObject BotonSalir;
    public GameObject IndicadorCamino;
    public GameObject EfectosGanarActivos;
    public GameObject TODOenergia;


    private void Start()
    {

        controlador = GetComponent<CharacterController>();
        Fuentes = GetComponent<AudioSource>();
        detenerTiempo = GetComponent<Timer>();
// InvokeRepeating("ReducirEnergia", 1, 1);
 //    ReiniciarBarraVida();
    }
    IEnumerator EsperarPanel()
  {
```

```
        yield return new WaitForSeconds(2f);
        TablaPuntuacion.SetActive(true);
        Gameover.SetActive(false);
    }


    IEnumerator MostrarPanelGanar()
    {

        yield return new WaitForSeconds(2f);
        MenuGana.SetActive(true);
    }



    private void Update()
    {


   //     if (detenerVida)
     //   {
  //          ReiniciarBarraVida();
    //    }

        // si el jugador esta muerto
        if (deathTiempo != 0)
        {
            //esperar x segundos luego reiniciar el nivel
            if (Time.time - deathTiempo > deathDuracion)
            {
            //    Gameover.SetActive(true);
             //   StartCoroutine(EsperarPanel());
            }
            return;

            valorX = Manager.xfloat;

        Vector3 moverVector = transform.forward * baseVelocidad;

        //Conseguir direccion
        Vector3 yaw = valorX * transform.right * velocidadRotacionX *
Time.deltaTime;
        // Vector3 pitch = valorY * transform.up * velocidadRotacionY *
Time.deltaTime;
        Vector3 dir = yaw; //+ pitch;

        //limit al jugador de un loop

        float maxX = Quaternion.LookRotation(moverVector + dir).eulerAngles.x;

        if (maxX < 90 && maxX > 70 || maxX > 270 && maxX < 290)
        {
        }
        else
        {
            moverVector += dir;
            transform.rotation = Quaternion.LookRotation(moverVector);
        }

        controlador.Move(moverVector * Time.deltaTime);

    }
```

```
  private void OnControllerColliderHit(ControllerColliderHit hit)
  {
       Perder();

  }


  public void OnTriggerEnter(Collider other)
  {
      if (other.tag == "PlanoGanar")
      {
          Ganar();
      }
  }


  public void Perder()
  {

      StartCoroutine(EsperarPanel());
      DetenerSonidos();
      deathTiempo = Time.time;
      GameObject avion = Instantiate(deathExplosion) as GameObject;
      avion.transform.position = transform.position;
      Destroy(avion,2.5f);
      transform.GetChild(0).gameObject.SetActive(false);//destruir
      BotonPausa.SetActive(false);
      BotonSalir.SetActive(false);
      Fuentes.PlayOneShot(sonidoPerder);
      detenerTiempo.DetenerTiempo();
      Gameover.SetActive(false);
      IndicadorCamino.SetActive(false);
      minimapaa.SetActive(false);
      mensajeRecarga.SetActive(false);
     TODOenergia.SetActive(false);

  }
public void ReiniciarBarraVida()
  {

      energia = 100;
  }
 public int enerBarraVida()
  {
     detenerVida = false;
     return energia;
 }

  public Slider barraEnergia;
  public static int energia = 100;


 void ReducirEnergia()
  {
      energia = energia - 2;
      barraEnergia.value = energia;
      if (energia <= 0)
      {
          mensajeRecarga.SetActive(true);
```

```
            StartCoroutine(perdervidas());
            mensajeRecarga.SetActive(false);
        }
    }
    IEnumerator perdervidas()
    {

        yield return new WaitForSeconds(4f);
        Perder();
        mensajeRecarga.SetActive(false);
    }

    private void Ganar()
    {

        DetenerSonidos();
        Fuentes.PlayOneShot(sonidoGanar,3f);
        detenerTiempo.DetenerTiempo();
        EfectosGanarActivos.SetActive(true);
        GameObject avionGanado = Instantiate(efectoGanar) as GameObject;
        avionGanado.transform.position = transform.position;
        GameObject.Find("Player").GetComponent<PlayerMotor>().enabled = false;
        BotonPausa.SetActive(false);
        BotonSalir.SetActive(false);
        Tiempo.SetActive(false);
        Puntos.SetActive(false);
        IndicadorCamino.SetActive(false);
        minimapaa.SetActive(false);
        TODOenergia.SetActive(false);
        StartCoroutine(MostrarPanelGanar());

    }



  private  void DetenerSonidos()
    {
        todosLosSonidos = FindObjectsOfType(typeof(AudioSource)) as
AudioSource[];

        foreach (AudioSource audioS in todosLosSonidos)
        {
            audioS.Stop();
        }
    }
}
```

SCRIPT MINIMAPA.CS

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


public class ObjetosMapa
 //se crea la clase Objetos Mapa para declarar los componentes que el avion
invoca cuando quiera conocer el proximo objetivo
{
    public Image iconoObjetivos { get; set; }
```

```
// declara el prefab o componente creado para distignuir a cada obstaculo en
el minimapa que en este caso es el icono color azul
    public GameObject iconoAvion { get; set; }
 // se declara el gameobject para el icono del avion y asi distinguirlo en el
minimapa
}

public class MiniMapa : MonoBehaviour {
//clase  MiniMapa

    //declaracion de variables
    public Transform posicionJugador;
 // se declara la variable posicionJugador usando el componente Transform para
conocer las coordenadas de posicion del avion y ubicarlo en el centro del
minimapa.
    public Camera mapCamar;
 // se declara la camara por la cual se guiara el minimapa dentro del juego.
    public static List<ObjetosMapa> minimapObjects = new List<ObjetosMapa>();
// se declara los objetivos del juego que estaran presentes en el minimapa,
conformandoles en una lista radObjects que anteriormente se declaro en la
clase ObjetosMapa

    public static void RegistrarObjetosMapa(GameObject avion, Image objetivo)
//metodo que se invoca solo en esta clase para conocer los objetivos del juego
y registrarlos con el respectivo icono de jugador y objetivo.
    {
        Image imagen = Instantiate(objetivo);
//se usa la funcion Instantiate para crear el objetivo ya con la imagen
establecida.
        minimapObjects.Add(new ObjetosMapa(){ iconoAvion = avion,
iconoObjetivos = imagen });
 // se agrega los respectivos iconos al minimapa
    }

    public static void EliminarObjetosMapa(GameObject o)
//metodo creado cada vez que un objetivo es alcanzado, entonces eliminara tal
objetivo del minimapa
    {
        List<ObjetosMapa> newList = new List<ObjetosMapa>();
 //se declara una lista temporal de objetivos presentes en el minimapa
        for (int i=0; i< minimapObjects.Count; i++)
// se realiza un bucle para que busque cada objetivo
        {
            if (minimapObjects[i].iconoAvion == o)
 //funcion bucle if.. si cada objetivo esta siendo alcanzado se va eliminando
la imagen del minimapa usando la funcion
            {
                Destroy(minimapObjects[i].iconoObjetivos);
                continue;

            }
            else
 // si no pasara esta condicion y siguiera existiendo objetivos por alcanzar
                newList.Add(minimapObjects[i]);
 // se agrega cada imagen del objetivo faltante al minimapa
        }
        minimapObjects.RemoveRange(0, minimapObjects.Count);
 //devuelve la funcion eliminar objetivos.
        minimapObjects.AddRange(newList);
//se ubican la imagen de los objetivos en una nueva lista
    }
```

```
    void DibujarPuntos()
 //funcion que devuelve como resultado la grafica de los objetivos en el
minimapa
    {
        foreach (ObjetosMapa ro in minimapObjects)

   //bucle para indicar las variables que muestran a los objetivos presentes
en el minimapa
        {
            Vector2 posicionObjetivos = new
Vector2(ro.iconoAvion.transform.position.x,
ro.iconoAvion.transform.position.y);   // se crea un vector en 2d que toma las
coordenas en las que estara ubicado el icono de los objetivos en el minimapa.
            Vector2 posJug = new Vector2(posicionJugador.position.x,
posicionJugador.position.y);                    //se crea el vector 2d para
determinar la posicion del jugador.

            if (Vector2.Distance(posicionObjetivos, posJug) > 200)
// bucle que determina que los objetivos no se salgan o se dispersen por todo
el minimapa sino que tengan una distancia coherente
            {
                ro.iconoObjetivos.enabled = false;
//si el vector dado por la posicicon del jugador y los objetivos es mayor a
200 , entonces inabilitar el icono de objetivos que queden fuera del mapa.
                continue;
            }
            else
                ro.iconoObjetivos.enabled = true;
 // caso contrario habilitar iconos dentro del mapa.

            Vector3 screePos =
mapCamar.WorldToViewportPoint(ro.iconoAvion.transform.position);
// conocer la posicion de la camara durante el juego, esta linea usa el
comando Camara.WorldToViewportPoint

//Transforma la posición del espacio mundial en espacio de vista, teniendo
presente que se normaliza con respecto a la camara creada para el minimapa
            ro.iconoObjetivos.transform.SetParent(this.transform);

            RectTransform rt = this.GetComponent<RectTransform>();
// se crea la variable rt para ubica cada objetivo dentro del cuadro del
minimapa
            Vector3[] corners = new Vector3[4];
//se crea un vector esquinas
            rt.GetWorldCorners(corners);
            screePos.x = Mathf.Clamp( screePos.x * rt.rect.width +
corners[0].x,corners[0].x,corners[2].x);
            screePos.y = Mathf.Clamp( screePos.y * rt.rect.height+
corners[0].y, corners[0].y,corners[1].y);
            screePos.z = 0;
            ro.iconoObjetivos.transform.position = screePos;
        }
    }

    void Update()
    {
        DibujarPuntos();
    }

}
```

SCRIPT OBJETIVOS.CS

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

public class Objetivos : MonoBehaviour {

    private List<Transform> objetivosJuego = new List<Transform>();
//obtiene el numero de objetivos en el GameObjet Nivel
    public Material Anilloactivado;
//variable que invoca el material para el primer objetivo
    public Material AnilloDesactivado;
//variable declarada para invocar el material para el siguiente objetivo
    public Material AnilloFinal;
// variable que invoca al objetivo final
    private int anilloPas = 0;
//se inicializa en cero los objetivos.

    private void Start()
//funcion que se inicializa las variables se lo llama una sola vez en todo el
script.
    {

        FindObjectOfType<Manager>().objetivos=this;                          //
llama a una funcion del Script Manager para ubicar la flecha indicadora en
cada objetivo.

        foreach (Transform t in transform)
//funcion loop para recorrer todos los objetivos de la lista declarnaado la
variable t para invocar a cada elelmento
        {
            objetivosJuego.Add(t);                                          //
se agrega cada objetivo en la variable t
            t.GetComponent<MeshRenderer>().material = AnilloDesactivado;   //
se cambia el material para el objetivo que a sido atravesado.
        }

        if (objetivosJuego.Count == 0)                                      //
condicion aplciada cuando el numero de objetivos sea igual a cero, se
reinicia.
        {
       //mensaje a mostrar en la consola de Unity, comprobando que se llego al
ultimo objetivo
                return;
        }
        objetivosJuego[anilloPas].GetComponent<MeshRenderer>().material =
Anilloactivado;    //  cuando pase un objetivo, otorga el material al
siguiente objetivo.
    }

    public void SiguienteAnillo()
    {
         anilloPas++;
        if (anilloPas == objetivosJuego.Count)
        {
         return;
        }
            //si este es el antepenultimo dar al sigueiente anillo como el
anillo final
```

```
        if (anilloPas == objetivosJuego.Count - 1)
         {
         Debug.Log("pasoUltimo");
            objetivosJuego[anilloPas].GetComponent<MeshRenderer>().material =
AnilloFinal;

        }
          else
            objetivosJuego[anilloPas].GetComponent<MeshRenderer>().material =
Anilloactivado;
        Debug.Log("pasoEsteoNo");
    }

    public Transform SiguienteAnilloDIreccion()
    {
        return objetivosJuego[anilloPas];
    }



}
```

SCRIPT PUNTOS.CS

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

public class Puntos : MonoBehaviour {

    //DECLARACION DE VARIABLES
    public string currentLevel;
    public int valorpuntaje;
    public static int puntos = 0;
    public Text textoPuntos;
    public Text TextoPuntos2;
    public Text TextoPuntos3;
    public static int highScore = 0;
    public static int highScore2 = 0;
    public static int highScore3 = 0;
 //   public Text ola;

  //  PlayerMotor reiniciar;
    public static int finish;


    //DECLARACION DE METODOS


    void Start () {
        puntos = 0;

        ActualizarMarcadorPuntos();
     //   reiniciar = GetComponent<PlayerMotor>();
    }
  /*  private void Update()
    {
        ActualizarMarcadorPuntos();
```

```
    }*/

    void ActualizarMarcadorPuntos()
    {
        DontDestroyOnLoad(gameObject);
        textoPuntos.text = "Puntos:" + Puntos.puntos;
        TextoPuntos2.text = Puntos.puntos.ToString();
        TextoPuntos3.text = "Puntaje:" + puntos.ToString();

   //CONDICIONES PARA GUARDAR DATOS

        if (currentLevel == "Juego")
        {
            Debug.Log("este es el nivel uno");
            PlayerPrefs.SetInt("PUNTOSNIVEL1", Puntos.puntos);
            PlayerPrefs.Save();
            highScore = PlayerPrefs.GetInt("PUNTOSNIVEL1" );
       //     ola.text = highScore.ToString(); SI PONER
        }
        else

            if (currentLevel == "NivelDOS")
            {
                Debug.Log("este es el nivel dos");
                PlayerPrefs.SetInt("PUNTOSNIVEL2", Puntos.puntos);
                PlayerPrefs.Save();
                highScore2 = PlayerPrefs.GetInt("PUNTOSNIVEL2" );
            //    Debug.Log(highScore2);


            }
            else

                if (currentLevel == "JuegoNivel3")
                {
                    Debug.Log("este es el nivel tres");
                    PlayerPrefs.SetInt("PUNTOSNIVEL3", Puntos.puntos);
                    PlayerPrefs.Save();
                    highScore3 = PlayerPrefs.GetInt("PUNTOSNIVEL3");
                //    ola.text = highScore3.ToString(); SI PONER

                }

    }


/*   public void LoadScore()
    {
        highScore = PlayerPrefs.GetInt("highScore" + scene.name, 0);
        Debug.Log(highScore);
    }*/

    public void GanarPunto()
    {
        Puntos.puntos += valorpuntaje;
        ActualizarMarcadorPuntos();

    }
    public void ReiniciarPuntos()
    {
        puntos = 0;
       }}
```

SCRIPT TIMER.CS

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


public class Timer : MonoBehaviour {


    //DECLARACION DE VARIABLES
    public string currentLevel;
    public static string tiempo1 ;
    public static string tiempo2 ;
    public static string tiempo3 ;
    public static string tiempo4 ;
    public static string tiempo5 ;
    public static float valorT1;
    public static float valorT2;
    public static float valorT3;
    public static float valorT4;
    public static float valorT5;
  //  public Text ola;
    public Text TiempoDos;
    public Text Tiempo;
  //  public static string TiempoNivel1;
    private float empezarTiempo;
//   public Text mostrartiempo; //MOSTRAR TIEMPO DE GET
    bool CondicionesTiempo;
    private float tiempoJuego;




    void Start()
    {
        EmpezarTiempoJuego();
    }

    void Update()
    {

        if (CondicionesTiempo)
        {
            ActualizarTiempo();
        }
    }

    void ActualizarTiempo()
    {
        tiempoJuego = Time.time-empezarTiempo;
        Tiempo.text = TimeToString(tiempoJuego);
        TiempoDos.text = TimeToString(tiempoJuego);

        tiempo1= TimeToString(tiempoJuego);
        tiempo2 = TimeToString(tiempoJuego);
        tiempo3 = TimeToString(tiempoJuego);
        tiempo4 = TimeToString(tiempoJuego);
```

```
    tiempo5 = TimeToString(tiempoJuego);

    if (currentLevel == "Juego")
    {

        PlayerPrefs.SetFloat("TIEMPONIVEL1" , tiempoJuego);
        PlayerPrefs.Save();
        valorT1 = PlayerPrefs.GetFloat("TIEMPONIVEL1");

    }
    else
    {
        if (currentLevel == "NivelDOS")
        {

            PlayerPrefs.SetFloat("TIEMPONIVEL2", tiempoJuego);
            PlayerPrefs.Save();
            valorT2 = PlayerPrefs.GetFloat("TIEMPONIVEL2");
        }
else
        if (currentLevel == "JuegoNivel3")
        {

            PlayerPrefs.SetFloat("TIEMPONIVEL3", tiempoJuego);
            PlayerPrefs.Save();
            valorT3= PlayerPrefs.GetFloat("TIEMPONIVEL3");
        }

    else
              if (currentLevel == "JuegoPelota")
        {

            PlayerPrefs.SetFloat("TIEMPONIVELPELOTA", tiempoJuego);
            PlayerPrefs.Save();
            valorT4 = PlayerPrefs.GetFloat("TIEMPONIVELPELOTA" );

        }

        else
             if (currentLevel == "JuegoPelotaDos")
        {

            PlayerPrefs.SetFloat("TIEMPOPELOTA2",tiempoJuego);
            PlayerPrefs.Save();
          valorT5 = PlayerPrefs.GetFloat("TIEMPOPELOTA2");

        }
    }

}

public float DetenerTiempo()
{
    CondicionesTiempo = false;
    return tiempoJuego;
}

void PausarTiempo()
{
    CondicionesTiempo = true;
    empezarTiempo = Time.time - tiempoJuego;
}
```

```
 public void EmpezarTiempoJuego()
    {
       CondicionesTiempo = true;
       empezarTiempo = Time.time;

    }

    public void reiniciartiempo()
    {

        tiempoJuego = 0;
    }


    public string TimeToString(float t)
    {
        string minutes = ((int)t / 60).ToString();
        string seconds = (t % 60).ToString("f2");
        return minutes + ":" + seconds;
    }


}
```

SCRIPT ANILLOS.CS

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

public class Anillos : MonoBehaviour {

    private Objetivos ScriptObjetivos;          // crea una variable privada
"ScriptObjetivos" para llamar a un metodo de la clase Objetivos
    AudioSource SonidoAnillos;                  // invoca funcion de audio
mediante la creacion de la variable SonidoAnillos
    public Puntos puntos;                       // crea la variable publica
"puntos" para llamar a un metodo de la clase Puntos

    private void Start()                       //funcion que inicializa las
variables y se lo llama una sola vez en todo el script.
    {
        ScriptObjetivos = FindObjectOfType<Objetivos>();      // inicializa la
variable ScriptObjetivos buscando en la clase Objetivos
        SonidoAnillos = GetComponent<AudioSource>();          // inicializa el
componente de audio.
    }

    public void OnTriggerEnter(Collider other)            //metodo de colision
de objetos, cuando el avion colisione con el anillo pasara las siguientes
acciones:
    {
        ScriptObjetivos.SiguienteAnillo();              // se activa el
metodo siguiente anillo del script objetivos
        SonidoAnillos.Play();                           // suena un audio
        Destroy(gameObject, 1f);                        // esto permite que
el anillo desapareza al momento que el aviona atraviese por el
        transform.SetParent(null);                      // inicializa desde
cero el conteo de objetivos
```

```
        puntos.GanarPunto();                                // permite ganar
puntaje en el marcador usnado el metodo GanarPunto del script Puntos

    }

}
```

SCRIPT BARRA DE ENERGÍA.CS

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

public class BarraEnergia : MonoBehaviour {


    public Slider barraEnergia;
     public static int energia = 100;
    PlayerMotor perder;
    // Use this for initialization
    private void Awake()
    {
        perder = GetComponent<PlayerMotor>();

    }
    void Start () {
         InvokeRepeating("ReducirEnergia", 1, 1);

     }
     void ReducirEnergia()
     {
         energia = energia - 10;
         barraEnergia.value = energia;
         if (energia <= 0)
         {
            perder.Perder();
         }
     }

}
```

SCRIPT ENERGIAGASOLINA.CS

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

public class EnergiaGasolina : MonoBehaviour {

    private void OnTriggerEnter(Collider other)
    {

        if (other.tag == "JUGADOR")
        {

          //  PlayerMotor.energia += 25;
            Destroy(gameObject);
        }
    }
}
```

SCRIPT PLAYERCAMARA.CS

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

public class CamaraJugador : MonoBehaviour {

    public GameObject Jugador;
    private Vector3 distancia;
       // Use this for initialization
       void Start () {

        distancia = transform.position = Jugador.transform.position;
       }

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

        transform.position = Jugador.transform.position + distancia;
       }
}
```

SCRIPT DETENERMUSICA.CS

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

public class detenerMusicas : MonoBehaviour {
    private void Awake()
    {
        if (barraSonido) {
        AudioListener.volume = PlayerPrefs.GetFloat("VolumenCurva");
        barraSonido.value = AudioListener.volume;
        }
    }
    public Slider barraSonido;
    // Use this for initialization


    public void ControlVolume()
    {
        AudioListener.volume =  barraSonido.value;
        PlayerPrefs.SetFloat("VolumenCurva",AudioListener.volume);
    }
}
```

APÉNDICE E: PROGRAMACIÓN DEL MINI JUEGO BALLGAME

SCRIPT CONTROLJUGADOR.CS

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

public class ControladorJugador : MonoBehaviour {

    //DECLARACION DE VARIABLES
    public string NiveldeJuego;
    public static int puntaje1 = 0;
    public static int puntaje2 = 0;
  //  public Text ola;
    Timer detenerTiempo;
    public GameObject ActivarColorRojo;
    private float velocidad = 6.0f;
    private CharacterController ControlJugador;
    public static int conteoPuntaje = 0;
    public Text Puntaje;
    public Text Puntajedos;
    public Text Puntajetres;
    public GameObject MenuGanar;
    private float valorY;
    public GameObject PanelPausa;
    public GameObject BotonPausa;
    public GameObject BOtonSALIR;
    public GameObject Tiempo;
    public GameObject Puntos;
    public GameObject ExplosionFuego;
    private AudioSource Fuentes;
    private AudioSource[] todosLosSonidos;
    public AudioClip sonidoGanar;
    public AudioClip sonidoPerder;
    public GameObject PanelPerdio;
    public GameObject Gameover;
    void Start()
    {
        conteoPuntaje = 0;
        ControlJugador = GetComponent<CharacterController>();
        Fuentes = GetComponent<AudioSource>();
        detenerTiempo = GetComponent<Timer>();
    }

    private float velocidadRotacionY=80f;

    private void Update()
    {
        valorY = Manager.yfloat;
        Vector3 moverVector = transform.forward * velocidad;
        ControlJugador.Move((Vector3.forward*velocidad)* Time.deltaTime);

        Vector3 pitch = valorY * transform.up * Time.deltaTime *
velocidadRotacionY;
        Vector3 dir =  pitch;
       moverVector += dir;
          GetComponent<Rigidbody>().AddForce(moverVector*velocidad *
Time.deltaTime);

        float maxX = Quaternion.LookRotation(moverVector + dir).eulerAngles.x;
```

```
    if (maxX < 90 && maxX > 70 || maxX > 270 && maxX < 290)
    {
    }
    else
    {
        moverVector += dir;
        transform.rotation = Quaternion.LookRotation(moverVector);
    }

    ControlJugador.Move(moverVector * Time.deltaTime);

}

private void OnTriggerEnter(Collider other)
{

    //CONDICIONES PARA ADQUIRIR PUNTOS
    if (other.tag == "Collide")
    {
        Debug.Log("hola");
        conteoPuntaje += 20;
        other.gameObject.SetActive(false);//desaparece figuras
        Puntaje.text = "Puntos:" + conteoPuntaje;
        Puntajedos.text= "Puntos:" + conteoPuntaje;
        Puntajetres.text= "Puntaje "+" "+         conteoPuntaje;
    }
    //CONDICION PARA PERDER JUEGO

    if (other.tag == "LAVA")
    {
        Debug.Log("CAYO LAVA");
        conteoPuntaje += 0;
        Destroy(gameObject);
        Perder();
    }

    //CONDICION PARA GANAR JUEGO
    if (other.tag == "PlanoGanar")
    {
        Debug.Log("GANO");
        Destroy(gameObject);
        Ganar();
    }

    //CONDICIONES PARA BASE DE DATOS


    if (NiveldeJuego == "JuegoPelota")
    {

        PlayerPrefs.SetInt("PuntosNivel1Pelota" , conteoPuntaje);
        PlayerPrefs.Save();
        puntaje1 = PlayerPrefs.GetInt("PuntosNivel1Pelota");
      //     ola.text = puntaje1.ToString(); //SI PONER
                }
    else

        if (NiveldeJuego == "JuegoPelotaDos")
    {

        PlayerPrefs.SetInt("PuntosNivel2Pelota", conteoPuntaje);
```

```
            PlayerPrefs.Save();
            puntaje2= PlayerPrefs.GetInt("PuntosNivel2Pelota");
            //    ola.text = highScore3.ToString(); SI PONER
        }

    }

    public void ReiniciarPuntaje()
    {
        conteoPuntaje = 0;
    }

    IEnumerator MostrarPanelPerder()
    {
        yield return new WaitForSeconds(1.5f);
        PanelPerdio.SetActive(true);

    }

    //METODO PARA GANAR EL NIVEL
    private void Ganar()
    {
        BotonPausa.SetActive(false);
        BOtonSALIR.SetActive(false);
        Tiempo.SetActive(false);
        Puntos.SetActive(false);
        PanelPausa.SetActive(true);
        MenuGanar.SetActive(true);
        GameObject.Find("JUGADOR").GetComponent<ControladorJugador>().enabled
= false;
    }

    //METODO QUE SE ACTIVA CUANDO PIERDE EL NIVEL
    private void Perder()
    {

        DetenerSonidos();
        Fuentes.PlayOneShot(sonidoPerder, 3f);
        GameObject PELOTA = Instantiate(ExplosionFuego) as GameObject;
        PELOTA.transform.position = transform.position;
        transform.GetChild(0).gameObject.SetActive(false);//destruir
        BotonPausa.SetActive(false);
        BOtonSALIR.SetActive(false);
        detenerTiempo.DetenerTiempo();
        Gameover.SetActive(true);
        Tiempo.SetActive(false);
        Puntos.SetActive(false);
        StartCoroutine(MostrarPanelPerder());
        PanelPerdio.SetActive(true);
    }


    private void DetenerSonidos()
    {
        todosLosSonidos = FindObjectsOfType(typeof(AudioSource)) as
AudioSource[];

        foreach (AudioSource audioS in todosLosSonidos)
        {
            audioS.Stop();
        }
    }}
```

SCRIPT CONTROLCAMARA.CS

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

public class controlCamara : MonoBehaviour {
    //declaracion de variables
    private Transform buscarJugador;
    private Vector3 distancia;

       // inicializa cada variable
       void Start () {
       buscarJugador= GameObject.FindGameObjectWithTag("JUGADOR").transform;
        distancia = transform.position - buscarJugador.position;
       }
       // es llamado cada frame
       void Update () {
        transform.position = buscarJugador.position + distancia;
       }}
```

APÉNDICE F: PROGRAMACIÓN DE LA INTERFAZ DE USUARIO

SCRIPT DE LA PANTALLA CARGANDO

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

public class CargarEscenasLoading : MonoBehaviour {


   // public AudioClip sonidoCargando;
    public Slider Slider;
    public GameObject LoadingEscenas;
    public GameObject BotonSiguienteNivel;
    AsyncOperation async;

    private void Start()
    {

    }
    public void IniciarCarga(string nivel)
    {

        LoadingEscenas.SetActive(true);//activa imagen loading
        BotonSiguienteNivel.SetActive(false);
        StartCoroutine(CargarNivelSlider(nivel));
    }


    IEnumerator CargarNivelSlider(string nivel)
    {
       async = SceneManager.LoadSceneAsync(nivel);
        while (!async.isDone)
        {
            Slider.value = async.progress;
            yield return null;
        }
    }
}
```

SCRIPT PARA CARGAR EL JUEGO

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

public class CargarJuego : MonoBehaviour
{

    private string nivel;
    public GameObject ImagenCarga;
    public Slider sliderCarga;
    private AsyncOperation asyn;
```

```
    public void IniciarCarga(string nivel)
    {
        ImagenCarga.SetActive(true);//activa imagen loading
        StartCoroutine(CargarNivelSlider(nivel));
    }

    IEnumerator CargarNivelSlider(string nivel)
    {
        asyn = SceneManager.LoadSceneAsync(nivel) ;
        while (!asyn.isDone)
        {
            sliderCarga.value = asyn.progress;
            yield return null;
        }

   }
}
```

SCRIPT DE LA TABLA DE PUNTUACIÓN

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

public class LeaderBoard : MonoBehaviour {
    public Text[] Puntajes;

    int [] PuntajesValores;
    string[] puntajesNombres;

    private void Start() //método iniciar
    {
        MARCAR();

    }
    void OnDestroy() //métodos para mantener los datos entre escena
    {
        for (int x = 0; x < Puntajes.Length; x++)
        {
            PlayerPrefs.SetInt("PUNTAJESALTOS" + x, PuntajesValores[x]);
            PlayerPrefs.SetString("NOMBRES" + x, puntajesNombres[x]);
            PlayerPrefs.Save();
        }
    }

    void savescore() //método para guardar los puntajes y nombres
    {
        for (int x = 0; x < Puntajes.Length; x++)
        {
            PlayerPrefs.SetInt("PUNTAJESALTOS" + x, PuntajesValores[x]);
            PlayerPrefs.SetString("NOMBRES" + x, puntajesNombres [x]);
        }
    }

   public void drawscore() // método para mostrar en la tabla de puntuacion
    {
        for (int x = 0; x < Puntajes.Length; x++)
        {
```

```
          Puntajes [x].text= puntajesNombres[x]+" "+" "+ PuntajesValores
[x].ToString();
        }
    }

    public void checchScore(int _value, string Name) //método para actualizar
los datos cada vez que se desea ver en la tabla de puntuacion
    {
        for (int x = 0; x < Puntajes.Length; x++)
        {
            if (_value > PuntajesValores [x])
            {
                for(int y=Puntajes.Length -1; y>x ; y--)
                {
                    PuntajesValores[y] = PuntajesValores[y - 1];
                    puntajesNombres[y] =  puntajesNombres[y - 1];
                }
                PuntajesValores [x] = _value;
                puntajesNombres[x] = Name;

               // savescore();
                drawscore();
                break;
            }
        }
    }


    void MARCAR () {  //Guarda los datos usando PlayerPrefs dentro de Unity

        PuntajesValores = new int[Puntajes.Length];
        puntajesNombres = new string[Puntajes.Length];
        for (int x = 0; x < Puntajes.Length; x++)
        {
            PuntajesValores[x] = PlayerPrefs.GetInt("PUNTAJESALTOS" + x);
            puntajesNombres[x] = PlayerPrefs.GetString("NOMBRES" + x);
        }
         drawscore();
    }
}
```

SCRIPT MENUESCENA.CS

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using DatabaseControl;
using TechTweaking.Bluetooth;


public class MenuEscena : MonoBehaviour {

//Declaracion de objetos para los Menus
    public GameObject MenuEmail;
    public GameObject menuInstrucciones;
    public GameObject MenuIniciarSesion;
```

```
    public GameObject MenuIniciarSesionSQL;
    public GameObject MenuJuego;
    public GameObject MenuRegistro;
    public GameObject MenuBaseDatos;



    public RectTransform menuContenido; //panel de jugar

    private Vector3 PosicionMenuDeseada; //panel de opciones


    private void Start() {
        //    PonerCamera(Manager.Instance.menuF);


    }

    private void Update() //método de Actualizacion de Frames
    {
        menuContenido.anchoredPosition3D =
Vector3.Lerp(menuContenido.anchoredPosition3D, PosicionMenuDeseada, 0.1f);

    }

   private void PonerCamera(int menuIndex)
    {
      Navegar(menuIndex);
      menuContenido.anchoredPosition3D = PosicionMenuDeseada;
    }

    public void Navegar(int menuIndex) //Metodo para Navegar en los Menus
    {
    switch (menuIndex)
        {
           //0 es el caso del menu principal
           default:
           case 0:
           PosicionMenuDeseada = Vector3.zero;

              break;
            //1 es el caso del menu play o jugar
            case 1:
                PosicionMenuDeseada = Vector3.right*1280;

             break;
            // para las opciones q en este caso es el shop
            case 2:
             PosicionMenuDeseada = Vector3.left * 1280;

              break;
            case 3:
                PosicionMenuDeseada = Vector3.left * 2560;

                break;
            case 4:
                PosicionMenuDeseada = Vector3.up * -800;
                break;
            case 5:
                PosicionMenuDeseada = Vector3.down * -800;
                break;
        }
```

```
}


// Función para los botones
public void BotonJugar(){
    Navegar(1);
   }

public void BotonIRMENUBASEDATOS()
{

    MenuBaseDatos.SetActive(true);
    MenuJuego.SetActive(false);


}

public GameObject MenuIniciar;
public void irAabrirDatos()
{
    MenuIniciar.SetActive(true);
}


public void BotonParametrosTerapeuta()
{

    MenuIniciarSesion.SetActive(true);

}

public void BotonINICIARSESION()
{
    Navegar(4);

}
public void BotonSalirBaseDdatos()
{
    MenuJuego.SetActive(true);
    MenuBaseDatos.SetActive(false);
}

public void BotonOpciones() {
    Navegar(2);
    Debug.Log ("Boton OPCIONES");
   }

public void BotonInstrucciones()
{
      menuInstrucciones.SetActive(true);

}
public void BotonEmail()
{
    Navegar(5);
}

public void BotonAtras()
{
    Navegar(0);
    Debug.Log("Boton Atras");
```

```
    }

    public void BotonAtrasdos()
    {
        Navegar(2);
    }
    public void AtrasMenuSesion()
    {
        Navegar(0);
    }
    public void AtrasRegistro()
    {
        MenuRegistro.SetActive(false);
        MenuIniciarSesionSQL.SetActive(true);
    }

    IEnumerator Salirjuego()
    {
        Application.Quit();
        yield return new WaitForSeconds(.1f);

    }
    public void Salir()
   {

        StartCoroutine("Salirjuego");

    }
}
```

SCRIPT MENUGAMEOBJECTS

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

public class MenuGameObjects : MonoBehaviour {

    public static string NombreJugador;
  //  public InputField nombrejugador;//nombre jugador
    public GameObject PanelPuntuaciones;
//    public GameObject PanelPausaPanel;
    public GameObject TablaPuntuacion;
    public GameObject panelCorreo;
    public GameObject PanelPausa;


    public GameObject MenuGuardarDatosAntesdeSalir;
    public GameObject SalirMenuGana;
//    public GameObject MenuInserteNombre;


  public void IrGuardar()
    {
        MenuGuardarDatosAntesdeSalir.SetActive(true);
        SalirMenuGana.SetActive(false);
        TablaPuntuacion.SetActive(false);
        PanelPausa.SetActive(false);
    }
```

```
    public void Guardar()
    {

        MenuGuardarDatosAntesdeSalir.SetActive(false);
    }
    public void AtrasMenuGuardar()
    {
        MenuGuardarDatosAntesdeSalir.SetActive(true);

    }
    public void BotonAtrasCorreo()
    {
        panelCorreo.SetActive(false);
        TablaPuntuacion.SetActive(true);

    }
    public Text mostrarAlertaBluetooth;
    public void MostrarAlertasBluetooth()
    {

        mostrarAlertaBluetooth.text = Info.sms;
    }


    public void bOTONaTRASDETABLAPUNTUACION()
    {
        PanelPuntuaciones.SetActive(false);
        TablaPuntuacion.SetActive(true);
    }

    public void BotonMostrarPuntajeNombres()
    {
        NombreJugador= PlayerPrefs.GetString("NombreJugador");
        TablaPuntuacion.SetActive(false);
        PanelPuntuaciones.SetActive(true);
        GetComponent<LeaderBoard>().checchScore(Puntos.puntos, NombreJugador);
    }

}
```

SCRIPT MENUPAUSA.CS

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

public class menuPausa : MonoBehaviour {


    public GameObject BotonPausa;
    public GameObject BotonSalir;
    public GameObject BotonReiniciarNivelJuego;
    public GameObject PanelPausa;
    public GameObject IndicadorCamino;
    public Puntos reiniciarPuntos;
    public Timer reiniciarTiempo;
    public PlayerMotor reiniciar;
    private float empezarTiempo;
```

```
public void Start()
{
    ContinuarJuego();
    reiniciarPuntos = GetComponent<Puntos>();
    reiniciar = GetComponent<PlayerMotor>();
    reiniciarTiempo = GetComponent<Timer>();
   // reiniciar.ReiniciarBarraVida();
}

public void Pausa()
{
    PanelPausa.SetActive(true);
    BotonPausa.SetActive(false);
    BotonSalir.SetActive(false);
    IndicadorCamino.SetActive(false);
    Time.timeScale = 0;
}
public void ReiniciarNivelJuego()
{
    SceneManager.LoadScene("Juego");
    PanelPausa.SetActive(false);
    reiniciarTiempo.reiniciartiempo();
    reiniciarPuntos.ReiniciarPuntos();

    // reiniciar.ReiniciarBarraVida();

}
public void ReiniciarNivel2()
{
    PanelPausa.SetActive(false);
    SceneManager.LoadScene("JuegoNivel2");
    reiniciarTiempo.reiniciartiempo();
    reiniciarPuntos.ReiniciarPuntos();
    //  reiniciar.ReiniciarBarraVida();
}
public void ReiniciarNivel3()
{
    PanelPausa.SetActive(false);
    SceneManager.LoadScene("JuegoNivel3");
    reiniciarTiempo.reiniciartiempo();
    reiniciarPuntos.ReiniciarPuntos();
    //   reiniciar.ReiniciarBarraVida();
}

public void ReiniciarJuegoPelota()
{
    PanelPausa.SetActive(true);
    SceneManager.LoadScene("JuegoPelota");
    reiniciarTiempo.reiniciartiempo();
}

public void ReiniciarJuegoPelotaDos()
{
    PanelPausa.SetActive(true);
    SceneManager.LoadScene("JuegoPelotaDos");
    reiniciarTiempo.reiniciartiempo();
}

public void ContinuarJuego()
{
    PanelPausa.SetActive(false);
    BotonPausa.SetActive(true);
```

```
        BotonSalir.SetActive(true);
        Time.timeScale = 1;
    }
    public void SalirJuegos()
    {

    }
    public void ExitScene()
    {
        SceneManager.LoadScene("BPanelJuego");

    }
}
```

SCRIPT .TXT

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


public class TXT : MonoBehaviour
{

    public GameObject MenuEmail;
    public static string NOMBREJUGADOR;
    public static string tiempoterapeuta;
    public static string tiempocomparacion;
  //  public InputField IngreseDatosJuego;
    public InputField ingreseNombre;


    private string TIEMPOAVION;
    private string TIEMPOPELOTA;
    public static float TiempoTotalJUGADO1;
    public static float TiempoTotalJUGADO2;
    private string TIEMPOAVION2;
    private string TIEMPOPELOTA2;
    private string VALORACION1;
    private string VALORACION2;
    private string VALORACION3;
    private string VALORACION4;
    private string hi;


 /*   private void Update()
    {
        datos();
    }*/
    public  void datos()
    {
        TIEMPOAVION = SUMA.TiempoTotalTerapeuta1;
        TIEMPOPELOTA = SUMA.TiempoTotalTerapeuta2;
        TiempoTotalJUGADO1 = PlayerPrefs.GetFloat("SUMAJUEGOAVIONjugadores");
        TiempoTotalJUGADO2 = PlayerPrefs.GetFloat("SUMAJUEGOPELOTAjugadores");
        TIEMPOAVION2 = TiempoTotalJUGADO1.ToString();
        TIEMPOPELOTA2 = TiempoTotalJUGADO2.ToString();
        VALORACION1 = PlayerPrefs.GetString("MENSAJE1PLANE");
        VALORACION2 = PlayerPrefs.GetString("MENSAJE2PLANE");
```

```
        VALORACION3 = PlayerPrefs.GetString("MENSAJE1PELOTA");
        VALORACION4 = PlayerPrefs.GetString("MENSAJE2PELOTA");

    }

    public void WriteToFile()
    {
        NOMBREJUGADOR=PlayerPrefs.GetString("NombreJugador");

        MenuEmail.SetActive(true);
        // NOMBREJUGADOR = DBCONTROL.loginjUGADOR;
        string dateAndTimeVar = System.DateTime.Now.ToString("yyyy/MM/dd");

        string FILE_PATH = Application.persistentDataPath + "/DATOS.txt";

        if (File.Exists(FILE_PATH))
        {
            StreamWriter sr = new StreamWriter(FILE_PATH, true); //
System.IO.File.CreateText(FILE_PATH); //ESTE VALE

         sr.WriteLine(dateAndTimeVar);
         sr.Write("MiniJuego PlaneGame"+ "\n"); //datos de terapeuta que se
recogen
         sr.Write("Nombre de Paciente"+ "\t" + "Tiempo Terapeuta" + " "+
"Tiempo Realizado" +" "+ "Observacion \n");
         sr.WriteLine("----------------------");
         sr.Write(NOMBREJUGADOR+ " "+ TIEMPOAVION + " "+ TIEMPOAVION2 + " "+
VALORACION1 + " "+ VALORACION2 +"\n");
         sr.WriteLine("MiniJuego BallGame" + "\n");
         sr.WriteLine("Nombre de Paciente" +" "+ "Tiempo Terapeuta" + " "+
"Tiempo Realizado" + " "+ "Observacion" + "\n");
         sr.Write(NOMBREJUGADOR+ " " + TIEMPOPELOTA+ " "+ TIEMPOPELOTA2 + " "+
VALORACION3+ " "+ VALORACION4 + "\n");

            sr.Close();
        }
              else
                          {
            StreamWriter sr = System.IO.File.CreateText(FILE_PATH);

sr.Write("MiniJuego PlaneGame" + "\n"); //datos de terapeuta que se recogen
                        sr.Write("Nombre de Paciente" + "\t" + "Tiempo
Terapeuta" + " " + "Tiempo Realizado" + " " + "Observacion \n");
                        //  sr.WriteLine(ingreseNombre.text);
                        // sr.WriteLine(ingreseNombre.text+TIEMPOAVION +
TIEMPOAVION2+ VALORACION1+ VALORACION2);
                        sr.WriteLine("----------------------"); //datos de
terapeuta que se recogen
                        sr.WriteLine(NOMBREJUGADOR + " " + TIEMPOAVION + " " +
TIEMPOAVION2 + " " + VALORACION1 + " " + VALORACION2 + "\n");
                        sr.Write("MiniJuego BallGame" + "\n");// +
LoggedIn_DataInputFieldDos.text); //datos de terapeuta que se recogen
                        sr.Write("Nombre de Paciente" + " " + "Tiempo
Terapeuta" + " " + "Tiempo Realizado" + " " + "Observacion" + "\n");
                        sr.WriteLine(NOMBREJUGADOR + " " + TIEMPOPELOTA + " "
+ TIEMPOPELOTA2 + " " + VALORACION3 + " " + VALORACION4 + "\n");
//     sr.WriteLine(Login_UsernameField.text + "Juego PlaneGame en " +
TiempoTotalAvion); //DATOS REALIZADOS EN PLANE GAME

//    sr.WriteLine(Login_UsernameField.text + "Juego PlaneGame en " +
TiempoTotalAvion); //DATOS REALIZADOS EN PLANE GAME
            sr.Close();}}}
```

APÉNDICE G: PROGRAMACIÓN DE LA RECEPCION DE DATOS

SCRIPT MANAGER.CS

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

public class Manager : MonoBehaviour {
    public static Manager Instance { set; get; }
//Declaracion de variables
   public Transform flechitas;
    private Transform JugadorCamino;
    public Objetivos objetivos;
    public static float xfloat;
    public static float yfloat;
    public static float zfloat;
    private string datos;

    private void Awake()
  {
       //Permite que no se destruya los objectos al iniciar el juego
      DontDestroyOnLoad(gameObject);
      Instance = this; }

    private void Start()
    {
        JugadorCamino = FindObjectOfType<PlayerMotor>().transform;
    }

    private void Update()
    {

       //obtiene datos de aceleracion
        datos = Info.content;
        //se separa los datos con la "coma" como separador
        string[] vec3 = datos.Split(',');
        if (vec3[0] != "" && vec3[1] != "" )//&& vec3[2] != "") //Revisa si
todos los valores son recibidos
        {

             // transform.rotation = Quaternion.Slerp(transform.rotation,
Quaternion.Euler(0, rot.x, rot.y), Time.deltaTime* 3);
             //Se asigna los valores de aceleracion en variables separadas
           xfloat = float.Parse(vec3[0]);
          yfloat = float.Parse(vec3[1]);
        }

        if (objetivos != null)
        {
            Vector3 direccion =
JugadorCamino.InverseTransformPoint(objetivos.SiguienteAnilloDIreccion().posit
ion);
            float a = Mathf.Atan2(direccion.x, direccion.z) * Mathf.Rad2Deg;
            a += 180;
            flechitas.transform.localEulerAngles = new Vector3(0, 180, a);
        }
    }}
```

APENDICE H: PRESUPUESTO

COSTO DE ELABORACION DEL SISTEMA			
CANTIDAD	**DESCRIPCION**	**Valor Unitario**	**PRECIO**
1	Arduino Pro Micro	$15,00	$15,00
1	Bluetooth HC05	$10,00	$10,00
1	Bateria 3.7V	$5,00	$5,00
1	Acelerometro ADXL335	$10,00	$10,00
1	Apatador para carga bateria	$5,00	$5,00
1	Micro interruptor	$0,45	$0,45
1	Carcasa impresa 3D	$7,00	$7,00
1	PCB impreso	$25,00	$25,00
6 horas	Diseño de Muñequera	$5,00	$30,00
10 horas	Diseño de Carcasa Protectora	$5,00	$50,00
20 horas	Diseño de Dispositivo	$5,00	$100,00
200 horas	Programacion Interfaz de Usuario	$8,00	$1.600,00
300 horas	Programacion del Videojuego	$8,00	$2.400,00
100 horas	Programacion Arduino	$8,00	$800,00
5 horas	Soldado de componenetes y modulos	$5,00	$25,00
1	Materiales y Herramientas	$40,00	$40,00
		Total	**$5.122,45**