DESARROLLO DE VIDEOJUEGOS INTERACTIVOS PARA MEJORA DE CALIDAD DE PACIENTES

DEVELOPMENT OF INTERACTIVE VIDEO GAMES TO IMPROVE THE QUALITY OF PATIENT CARE

TRABAJO FIN DE GRADO
CURSO 2022-2023

AUTOR
MARIO QUINTAS MORCILLO

DIRECTOR

CARLOS GARCÍA SÁNCHEZ

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

DESARROLLO DE VIDEOJUEGOS INTERACTIVOS PARA MEJORA DE CALIDAD DE PACIENTES

DEVELOPMENT OF INTERACTIVE VIDEO GAMES TO IMPROVE THE QUALITY OF PATIENT CARE

TRABAJO DE FIN DE GRADO EN INGENIERÍA INFORMÁTICA

AUTOR
MARIO QUINTAS MORCILLO

DIRECTOR
CARLOS GARCÍA SÁNCHEZ

CONVOCATORIA: JUNIO 2023

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y ADMINISTRACIÓN Y DIRECCIÓN DE EMPRESAS
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

29 DE MAYO DE 2023

RESUMEN

Algunos pacientes después de pasar mucho tiempo en el hospital sin apenas moverse pierden gran cantidad de masa muscular. Actualmente, para intentar solucionar esto se les manda una tabla de ejercicios propuestos por el equipo de fisioterapia del hospital, pero aun así el período de recuperación de dicha masa muscular es bastante largo.

Para evitar que estos pacientes pierdan esta masa muscular se desarrollará un videojuego que permita al paciente de manera autónoma realizar una serie de ejercicios. Este videojuego tendrá como entrada un sistema de sensorización del paciente de forma que interaccione de forma interactiva con el propio videojuego.

Con esto lo que se espera conseguir es que los pacientes obtengan una rutina para evitar está pérdida de masa muscular en su paso por el hospital a la vez que los médicos tienen un lugar donde monitorizan todos los resultados del paciente para ver la posible mejora, o si por el contrario no la hay, pensar alternativas para ello.

Además, los médicos contarán con una interfaz en la que podrán observar los datos de las partidas de los pacientes y también tendrán la posibilidad de configurar la dificultad del videojuego mediante el número de repeticiones que tenga que realizar el paciente, para así conseguir una mejora al ir subiendo la dificultad del videojuego.

Palabras clave

Pacientes, videojuego, sensor, masa muscular, interfaz, rutina.

ABSTRACT

Some patients, after spending a long time in hospital with little or no movement, lose a large amount of muscle mass. Currently, to try to solve this, they are sent a table of exercises proposed by the hospital's physiotherapy team, but even so, the recovery period for this muscle mass is quite long.

In order to prevent these patients from losing this muscle mass, a video game will be developed that allows the patient to perform a series of exercises autonomously. This video game will have as an input a sensor system for the patient so that he or she interacts interactively with the video game itself.

With this, what is expected to be achieved is that patients get a routine to avoid the loss of muscle mass during their stay in hospital, while doctors have a place where they can monitor all the patient's results to see the possible improvement, or if there is no improvement, to think of alternatives.

In addition, the doctors will have an interface where they will be able to observe the data of the patients' games and they will also have the possibility of configuring the difficulty of the video game by means of the number of repetitions that the patient has to perform, in order to achieve an improvement by increasing the difficulty of the video game.

Keywords

Patients, video game, sensor, muscle mass, interface, routine.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

Capítulo 1 - Introducción

1.1 Motivación

Este trabajo tiene como motivación principal dar una solución para aquellas personas que se encuentren hospitalizadas para evitar que estas pierdan una gran cantidad de masa muscular debido a la inactividad.

La mayoría de las personas que se encuentran en un hospital, pierden mucha masa muscular, la cual luego se intenta recuperar mediante unos ejercicios propuestos por fisioterapias, pero esta recuperación suele ser de larga duración y a veces no resulta efectiva.

Sin embargo, a través de este trabajo se ha pensado desarrollar un videojuego interactivo para los pacientes, a través de sensores de movimiento, lo cual hará que los pacientes hagan determinados ejercicios físicos para jugarlo haciendo que a largo plazo no pierdan masa muscular evitando la inactividad.

Además, se conseguirá que todos los resultados del videojuego sean guardados en una base de datos, en la cual se podrán ver los avances de los pacientes y también poder personalizar para cada uno de ellos la dificultad de la partida que jueguen a través de una interfaz específica para los médicos.

En específico, en este proyecto se ha desarrollado un videojuego que trata de un simulador de tráfico. En este, el jugador controla cuando frena y cuando se pone en marcha el coche, y lo que tiene que evitar es saltarse los semáforos en rojo. El control del coche lo hará mediante un sensor, el cual tendrá en uno de los pies y tendrá que hacer el mismo movimiento que cuando una persona conduce y pulsa el acelerador y el freno.

Con este videojuego, lo que se espera conseguir es que las personas hospitalizadas sigan teniendo la posibilidad de conducir después de su paso por el hospital.

1.2 Objetivos

El objetivo principal de este proyecto es el desarrollo de un videojuego interactivo que mediante sensores de movimiento logré hacer que personas que se encuentren en terapia de rehabilitación a través de un movimiento físico, para jugar al videojuego, no pierdan gran cantidad de masa muscular debido a la inactividad en la que se encuentran la mayoría de los pacientes.

Para este objetivo, el desarrollo del proyecto se basa en cuatro capítulos:

- Desarrollo del videojuego interactivo de manera dinámica.
- Desarrollo del funcionamiento del sensor de movimiento con el videojuego.
- Desarrollo de una base de datos, donde almacenar los resultados y de donde coger la información de la partida.
- Desarrollo de una interfaz para los médicos, donde podrán ver los resultados y configurar próximas partidas.

1.3 Plan de trabajo

El plan de trabajo seguido a lo largo del presente curso para el desarrollo del proyecto se representa mediante un diagrama de Gantt:

Tareas	SEP	OCT	NOV	DIC	ENE	FEB	MAR	ABR	MAY
Presentación del proyecto									
Introducción Unity									
Desarrollo del videojuego									
Desarrollo base de datos									
Desarrollo interfaz médico									
Desarrollo sensor de movimiento									
Pruebas correcto funcionamiento									
Elaboración de la memoria									

Introduction

Motivation

The main motivation of this work is to provide a solution for people who are hospitalised in order to prevent them from losing a large amount of muscle mass due to inactivity.

Most people who are in hospital lose a lot of muscle mass, which they then try to recover by means of exercises proposed by physiotherapies, but this recovery is usually of long duration and sometimes it is not effective.

However, through this work we have thought of developing an interactive video game for patients, through movement sensors, which will make patients do certain physical exercises to play it so that in the long term they do not lose muscle mass, avoiding inactivity.

In addition, all the results of the video game will be saved in a database, in which the progress of the patients can be seen and the difficulty of the game can be personalised for each of them through a specific interface for the doctors.

Specifically, in this project, a video game has been developed that deals with a traffic simulator. In this game, the player controls when to brake and when to start the car, and the player has to avoid running red lights. The car is controlled by means of a sensor, which is placed in one of the feet and has to make the same movement as when a person drives and presses the accelerator and brake.

With this video game, it is hoped that hospitalised people will continue to be able to drive after they have been in hospital.

Objectives

The main objective of this project is the development of an interactive video game that, by means of movement sensors, will ensure that people

undergoing rehabilitation therapy do not lose a large amount of muscle mass due to the inactivity in which most patients find themselves, by means of physical movement to play the video game.

For this objective, the development of the project is based on four chapters:

- Development of the interactive video game in a dynamic way.

- Development of the functioning of the movement sensor with the video game.

- Development of a database, where to store the results and where to get the information of the game.

- Development of an interface for the doctors, where they will be able to see the results and configure future games.

Capítulo 2 - Estado del arte

2.1 Contexto de las terapias de rehabilitación

Actualmente, la gente que pasa mucho tiempo hospitalizada tiende a perder gran cantidad de masa muscular debido a su inactividad durante este tiempo.

Para remediar esta gran cantidad de pérdida muscular, a los pacientes se les manda hacer una serie de ejercicios físicos asignados por un fisioterapeuta para que poco a poco se vayan recuperando. En muchas ocasiones estas recuperaciones suelen ser de larga duración y a veces la pérdida de masa muscular es tan grande que puede llegar a ser irrecuperable.

Estos pacientes no disponen como tal de unas medidas preventivas para evitar esta perdida y siempre se hace a posteriori mediante una serie de ejercicios básicos los cuales deben hacer con un profesional delante para que lleve un control del avance que vaya experimentando el paciente.

Además, en este ámbito se ve muy poco la utilización de videojuegos para incitar al paciente a realizar estos ejercicios físicos, pero los pocos que llevan este método a la práctica usan hardwares implementados en consolas de marcas conocidas como puede ser la Wii de Nintendo o la Kinect de Microsoft, por lo que disponer de una de estas consolas se vuelve obligatorio para estos casos.

En base al uso de estas consolas se han hecho numerosos estudios como es el caso del trabajo realizado por Laurie A. Malone, Christen J. Mendonca y Yumi Kim, el cual nos habla de los beneficios que conlleva esta practica sobre la salud de los pacientes diciéndonos que "los estudios informaron mejoras significativas en los resultados de salud, específicamente en el equilibrio (n = 30/36), la movilidad (n = 24/27) y la aptitud cardiorrespiratoria (n = 6/8). También se observaron cambios positivos en las condiciones secundarias (n = 8/12), la actividad física (n = 3/4) y los resultados de calidad de vida (n = 8/16)." [1]

Las n hacen referencia a que, de la cantidad de los estudios realizados para cada área médica, cuantos reportaron una mejora significativa en los resultados.

Otro estudio que nos trata también de la mejora que pueden tener los pacientes en su salud incluyendo en su rutina de rehabilitación los videojuegos sería el estudio de Jason K Hsu, Richard Thibodeau, Stephanie J Wong, Daniel Zukiwsky, Sara Cecile y David M Walton, los cuales tratan de la mejora en la salud de los pacientes al incluir el juego de los bolos de la Nintendo Wii.

En el propio resumen de su estudio dicen lo siguiente, aclarando que cuando usan las siglas SG hacen referencia a los ejercicios estándares, es decir sin el uso del videojuego:

"Los resultados sugieren que los sujetos mejoraron en todos los resultados antes y después de la intervención, pero que sólo el disfrute de la actividad mostró una diferencia significativa entre los grupos SG y Wii. Entre los grupos SG y Wii. Los tamaños del efecto (d de Cohen) oscilaron entre pequeño (0,30 para molestias) y grande (1,77 para capacidad funcional)." [2]

Este fragmento del resumen del estudio sirve de gran ayuda para apoyarnos en la motivación principal que tiene este trabajo. Esta motivación no solo se basa en desarrollar un videojuego para que los pacientes hagan el movimiento para mejorar su salud, sino que también cabe destacar que se busca que el paciente disfrute de esta actividad para con ello, conseguir una mejora más significativa.

Centrándonos en los resultados de estos estudios para tener un motivo más para entender la importancia de implantar esta opción en la rehabilitación de los pacientes y por tanto la motivación del desarrollo de este trabajo, podemos observar que los resultados son muy positivos, y que esta mejora en la calidad de vida de los pacientes es una realidad.

Como nos dice el estudio nombrado anteriormente, los resultados obtenidos por este son los siguientes:

"De los 36 estudios que informaron resultados de equilibrio, el 75 % (n = 30/36) de los estudios informaron que al menos un resultado mejoró significativamente antes y después de la intervención. Se encontraron mejoras significativas en el equilibrio en función de varios tipos de evaluación, incluido el desempeño, el instrumento, la calificación del observador y el autoinforme. Para caminar/movilidad, 89% (n = 24/27)

de los estudios informaron al menos una mejora significativa después de la intervención según lo determinado por evaluaciones biomecánicas" [1].

Las n, como en el fragmento de estudio comentado anteriormente, hacen referencia a la cantidad de estudios que han reportado una mejora significativa de los estudios totales realizados para cada área médica.

Observamos que estos resultados son muy positivos y relevantes como para poder afirmar que el uso de un videojuego en una terapia de rehabilitación puede ser positivo para la salud del paciente, además de presentar grandes ventajas sobre aquellas terapias más convencionales.

Además, otro aspecto que nos motiva a la hora de hacer este trabajo es dar al paciente una retroalimentación visual del movimiento que este haciendo, obligándolo a hacerlo de manera correcta para que el juego responda y motivándole a hacerlo mejor.

Según otro estudio realizado sobre los beneficios de usar la Wii para terapias de rehabilitación, "el uso potencial de los juegos con la retroalimentación visual puede facilitar la reducción del tiempo que el terapeuta tiene que pasar con el paciente." [3]

Esto es otra motivación más, ya que esto nos indica que al tener los pacientes este tipo de rehabilitación con una retroalimentación visual como la de un videojuego, entienden más rápido el movimiento solicitado y hacer que lo ejecuten de una mejor manera.

2.2 Propuesta de innovación

El desarrollo de este Trabajo de Fin de Grado va orientado a dar una solución a estos pacientes que se hallen hospitalizados. Con este videojuego se busca evitar que estos pierdan tanta masa muscular en su paso por el hospital y también tener unos ejercicios para después de su salida de este.

Como hemos visto en el punto anterior, todos los estudios que tratan sobre la mejora que puede suponer incorporar la utilización de videojuegos en la rutina de los

pacientes con escasa movilidad física, tienen un punto específico en común, que es: ¿qué se usa para implantar estos videojuegos en las rutinas de los pacientes?

Todos ellos lo implantan usando consolas que son comercializadas con otro propósito como es la Wii de Nintendo, por lo que se ha querido enfocar en este punto a la hora de desarrollar este trabajo.

Lo que se busca es dar una alternativa a las consolas convencionales para este ámbito médico en concreto, desarrollando un producto específico para ello. Con este producto se consigue, entre otras cosas, evitar un gran desembolso económico como puede suponer la compra de una de estas consolas (el sensor usado para este trabajo cuesta apenas setenta euros) así como la facilidad en el uso de estos dispositivos. También lo que se lograría es evitar tener un hardware muy grande, como es una consola, conectado a una televisión para poder jugar al videojuego. Para jugar al videojuego desarrollado en este trabajo tan solo te haría falta un ordenador donde ejecutar el juego y el sensor de movimiento utilizado.

Además, lo que se consigue también desarrollando un producto específico para el ámbito médico, es hacer que todo el desarrollo que se haga de este será con fines específicamente médicos. Un ejemplo de esto es hacer que los médicos puedan tener un seguimiento de la evolución de los pacientes mediante la monitorización de los resultados y así evitar que los pacientes tengan que desarrollar sus ejercicios con un profesional delante facilitando que sus resultados se guarden de manera automática en una base de datos. Esto es un factor muy diferencial de los demás proyectos que tienen que ver con este tema, al utilizar en las terapias consolas comerciales, los resultados de las partidas no se guardaban en una base de datos que luego podía ser vista por el médico para llevar un seguimiento de los resultados del paciente.

Cabe destacar, que el desarrollo de este trabajo se hace con la utilización de un sensor en específico, del cual se hablará con mayor detalle en los siguientes capítulos, pero este proyecto se podría ajustar a cualquier otro sensor elegido.

Entonces, ¿por qué este ha sido el sensor elegido? El sensor que se usa para el desarrollo de este proyecto es un Nicla Sense ME, que es un sensor de movimiento

desarrollado por la compañía Bosch. La propia Universidad Complutense de Madrid tiene una Cátedra Extraordinaria con la marca Bosch con el siguiente objetivo:

"La Cátedra Extraordinaria Bosch-UCM consiste en impulsar la investigación y la transferencia tecnológica en el campo de Inteligencia Artificial aplicada al Internet de las Cosas mediante el desarrollo de soluciones innovadoras que demuestren el potencial de los recursos tecnológicos de Bosch, así como la divulgación de las posibilidades de dicha tecnología y los resultados obtenidos mediante distintas actividades de difusión y colaboración docente" [3]

Esto quiere decir que Bosch nos ha facilitado el sensor utilizado para el desarrollo de este Trabajo de Fin de Grado, además de presentar unas especificaciones ideales para este proyecto en concreto.

Por último, con esta propuesta también quiere conseguirse que los pacientes tengan una motivación para realizar estos ejercicios mandados, alejándonos de la monotonía de hacer simplemente los ejercicios. Como hemos comentado en el punto anterior, otros estudios [1], [2] confirman que el hacer que el paciente disfrute de las actividades que realiza ayudan de manera positiva a los resultados obtenidos.

Capítulo 3 - Arquitectura y relaciones de los componentes desarrollados

En este apartado se verá una imagen genérica de todo el proyecto y las conexiones que hay dentro de este, para luego poder entender de una mejor forma los siguientes epígrafes.

A continuación, se mostrará un diagrama de todos los componentes desarrollados en el trabajo y su relación entre ellos.

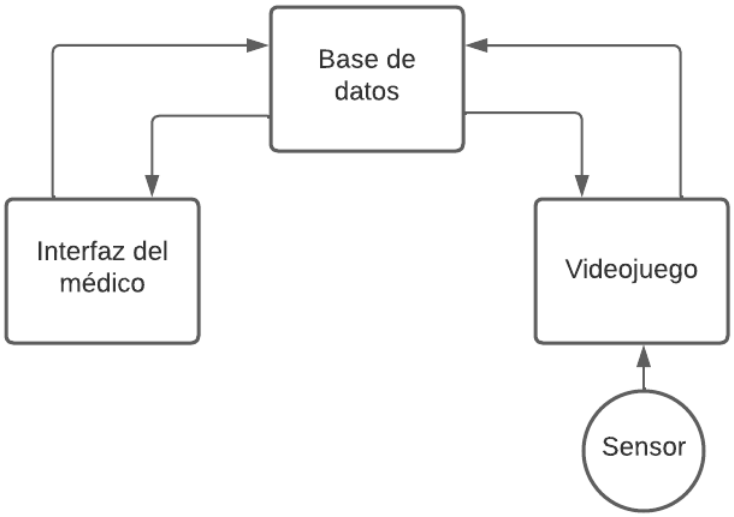

Figura 1: Diagrama componentes proyecto

Como observamos en el diagrama, el proyecto se basa en cuatro componentes los cuales forman los siguientes cuatro capítulos de la memoria, explicando con mayor detalle dichos componentes.

En primer lugar, tendríamos al sensor, el cual solo se relaciona con el videojuego notificando la posición de este para que el videojuego haga diferentes acciones dentro del juego, como arrancar o frenar el coche.

En segundo lugar, tendríamos el videojuego, el cual se relaciona con la base de datos en ambos sentidos. Esto quiere decir que lee y escribe de la base de datos, por ejemplo, el videojuego lee de la base de datos al iniciar sesión el paciente mirando si el

DNI existe y cuantos semáforos deben crearse por consiguiente en el videojuego. Por otro lado, un ejemplo de escritura sería cuando se guardan los datos de la partida jugada por el paciente.

En tercer lugar, tendríamos la interfaz del médico la cual se relaciona con la base de datos en ambas direcciones, es decir que lee y escribe de la base de datos. Por ejemplo, lee a la hora de mostrar a los pacientes registrados y escribe cuando un médico añade a un paciente a la base de datos.

Por último, tendríamos la base de datos, cuyas relaciones con los componentes de la interfaz del médico y con el videojuego ya se han explicado anteriormente.

Una vez entendido este esquema general de los componentes que forman el proyecto, será más sencillo entender los siguientes capítulos, donde se explicaran con mayor detalle estos componentes.

Capítulo 4 - Videojuego

La mayor parte de del desarrollo del trabajo trata sobre el videojuego. Entorno a este, gira todo el proyecto y por ello es al que más tiempo se le ha dedicado.

El videojuego es un simulador de tráfico, en el que el usuario controla un coche en tercera persona y sobre el que decide cuando se para y cuando se pone en marcha.

Figura 2: Videojuego desarrollado

Como observamos en la figura 2, este coche va por una carretera y conforme avanzas por esta, el usuario se ira encontrado con semáforos, los cuales tendrá que evitar saltarse parándose cuando se encuentre uno en rojo.

Conforme avanza el juego el usuario irá ganando puntos si pasa los semáforos de forma correcta o irá perdiendo si salta alguno en rojo. El objetivo es llegar con los máximo puntos al final de la carretera evitando saltarte semáforos en rojo.

4.1 Entorno de desarrollo del videojuego

Todo el desarrollo del videojuego se ha realizado en el programa Unity 3D. Según la página oficial de Unity:

> Unity 3D es la plataforma líder en el mundo para crear y operar contenido interactivo 3D en tiempo real (RT3D). Programadores de juegos, artistas, arquitectos, diseñadores automotrices y cineastas, entre otros, usan Unity para darle vida a los productos de su imaginación [4].

Además, es una herramienta muy popular debido a su facilidad de uso y su amplia gama de características. Unity 3D tiene una interfaz de usuario intuitiva y fácil de usar, lo que lo hace ideal tanto para principiantes como para desarrolladores experimentados. Los desarrolladores pueden utilizar el editor de Unity para crear y modificar objetos, terrenos, personajes y escenarios.

Unity 3D es compatible con varios lenguajes de programación, incluyendo C# y JavaScript, lo que significa que los desarrolladores pueden trabajar con el lenguaje de programación con el que se sientan más cómodos.

Otra característica útil de Unity 3D es su activo tienda, que contiene una amplia variedad de modelos 3D, texturas, música y efectos de sonido que los desarrolladores pueden utilizar en sus proyectos.

Unity ofrece a sus usuarios la opción de comprar recursos para complementar el desarrollo de sus proyectos. La mayoría de estos productos digitales son creados por desarrolladores independientes. Unity se encarga de evaluar cada uno de estos recursos y en caso de ser aprobados, se publican en la Unity Asset Store (Tienda de recursos de Unity) [5].

En resumen, Unity 3D es una herramienta que permite a los desarrolladores crear juegos y aplicaciones en 3D de alta calidad. Su facilidad de uso y compatibilidad con varias plataformas hacen que sea una opción popular entre los desarrolladores de juegos y aplicaciones en todo el mundo.

4.2 Desarrollo del videojuego

Como se ha dicho en el apartado anterior, el videojuego se ha desarrollado en la herramienta Unity 3D. En este epígrafe se contará con más detalle todo lo que forma el videojuego.

En primer lugar, para la creación del juego y darle un aspecto más profesional a este, se han usado diferentes paquetes descargados de la tienda de Unity, la cual se ha comentado en puntos anteriores.

Para el desarrollo del videojuego en cuanto a aspecto se refiere se han usado dos paquetes de la tienda de Unity.

Uno de ellos se ha usado para el aspecto del coche. El paquete que se decidió elegir se llama Simple Cars Pack [6]. Se seleccionó un coche que no fuese el que mejor gráficos presentase ya que se quería hacer el juego con una apariencia retro y no tan realista y fue por eso la decisión de este paquete.

El otro se usó para la creación de toda la ciudad. Para esto, se escogió el paquete Polygon City Pack [7], del cual se cogió todo lo que se ve en el juego que no es el coche. Se uso para la carretera y la acera, para los edificios que se ven a los lados de la carreta y para los semáforos. Todos estos aspectos siguen la línea temática que el coche presenta, alejándonos de unos gráficos realistas y acercándonos a unos de dibujos.

En cuanto al funcionamiento completo del juego, este se puede dividir en tres scripts, los cuales corresponden a aquellos objetos que van cambiando durante el juego y por tanto hay que programar su comportamiento.

En la siguiente figura se muestra un diagrama de las clases que se van a describir en este apartado para poder tener una visualización grafica de estas para un mejor entendimiento.

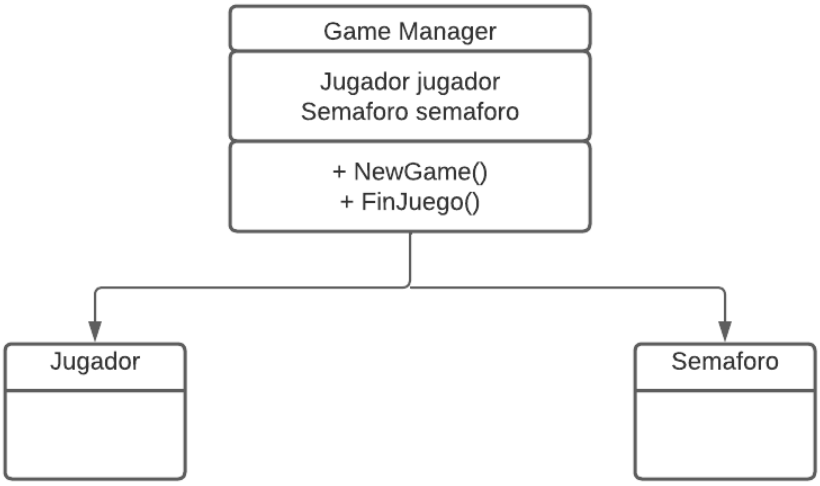

Figura 3: Diagrama de clases Videojuego

Los demás scripts que se encuentran dentro del videojuego hacen referencia a la conexión con el sensor de movimiento, los cuales se explicarán en el siguiente capítulo.

En primer lugar, tendríamos los semáforos, los cuales hay que programar cuando deben cambiar de luz. También nos encontraríamos con el coche sobre el que debemos programar cuando debe moverse y cuando debe parar además de todos sus atributos como son los puntos o la posición en la que se encuentra el propio vehículo.

La última parte que faltaría por comentar no es un objeto del juego que va cambiando como tal, sino que sería una clase que se encarga del funcionamiento global del videojuego. Esta clase es llama Game Manager y será la encargada de coordinar las diferentes acciones que se necesiten para un correcto funcionamiento del videojuego.

4.2.1 Semáforo

Esta clase será la encargada de definir todo el funcionamiento que hay detrás de un semáforo. Antes de describir el código que hay detrás de cada semáforo hay que ver aquellos componentes del juego que forman el semáforo.

El semáforo está formado, por el aspecto del paquete mencionado en el punto anterior, tres esferas que representan los tres colores del semáforo y una luz para hacer que se vea mejor la luz que está encendida. Además, este también tiene un box collider.

Un Box Collider es una primitiva básica de colisión en forma de cubo, sirve para controlar cuando otro objeto colisiona con este collider y hacer acciones determinadas a raíz de esta colisión [8].

Según la página oficial de Unity [8], los Box Colliders son útiles para cualquier objeto con forma cúbica, como por ejemplo una caja o un cofre, y también pueden usarse como piso, muro o rampa. La forma en cubo también es un elemento útil en un collider compuesto, al querer saber cuánta área ocupa en forma de cubo un objeto [9].

Esto lo usaremos para detectar cuando el jugador pasa por debajo del semáforo al estar atravesando esta área definida como box collider.

Sin entrar aún al detalle del código, la idea del funcionamiento del semáforo es la siguiente. Teniendo el objeto del paquete, poner tres esferas de color negro delante de cada luz del semáforo, dando la sensación de que estuviese apagado, por lo que cada esfera tendría una posición dentro del semáforo en función del color que vaya a representar, ya sea verde, ámbar o roja.

Una vez comience el juego una de las esferas cambiará de negro al color que represente en función de la posición en la que esté del semáforo mientras las otras se mantienen en negro, y cuando pase una cantidad de segundos cambiará la siguiente a su color volviendo a poner a negro la primera esfera. Además, para darle mayor visibilidad y sensación de semáforo, este tiene una luz que sale de las esferas que tendrá el mismo color que la esfera que no tenga el color negro puesto.

Analizando el código observamos que esta clase cuenta con cinco métodos además del Start y el Update. El primero de estos inicializa el semáforo a

verde y el segundo, llama continuamente a un método que se encargará del cambio de luz.

El método al que se llama en Update se llama CambioLuz. Este método comprobará la luz que está encendida en cada momento, cambiará el color de la luz, pondrá todas las esferas en color negro menos la que tenga que estar encendida y pondrá a falso aquel color del que venía, por ejemplo, si estamos en verde podrá a falso el indicador de rojo.

A continuación, se muestra un diagrama de estados de los tres estados en los que puede encontrarse un semáforo y el tiempo que se mantiene en cada estado hasta pasar al siguiente.

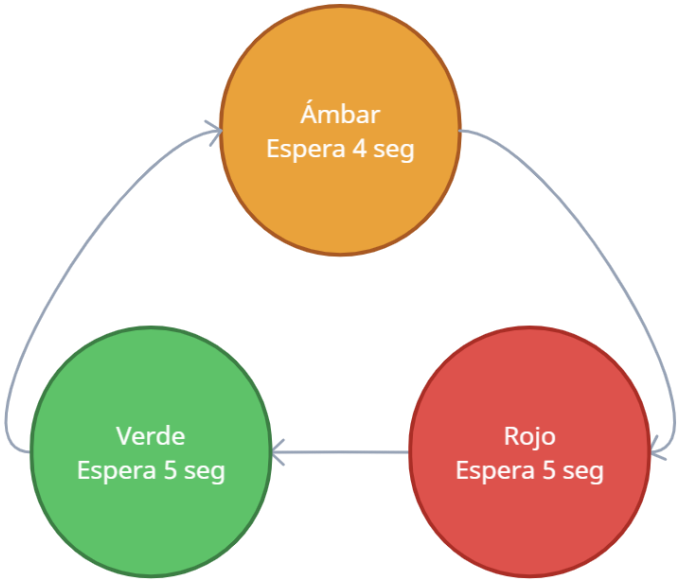

Figura 4: Diagrama de estados semáforos

También se muestra el código usado, donde vemos que tenemos el estado del semáforo como booleanos:

```
1. private void CambioLuz()
2.     {
3.         if (verde == true)
4.         {
5.             Luz.transform.position = posVerde.position;
6.             Luz.GetComponent<Light>().color = Color.green;
7.             StartCoroutine(luzVerde());
```

```
8.              EsferaVerde.GetComponent<Renderer>().material.SetColor("_Color", Color.green);
9.                          EsferaAmarilla.GetComponent<Renderer>().material.SetColor("_Color",
Color.black);
10.             EsferaRoja.GetComponent<Renderer>().material.SetColor("_Color", Color.black);
11.             rojo = false;
12.         }
13.
14.
15.         if (amarillo == true)
16.         {
17.             //Luz.transform.position = posAmarillo.position;
18.             Luz.GetComponent<Light>().color = Color.yellow;
19.             StartCoroutine(luzAmarillo());
20.             EsferaVerde.GetComponent<Renderer>().material.SetColor("_Color", Color.black);
21.                         EsferaAmarilla.GetComponent<Renderer>().material.SetColor("_Color",
Color.yellow);
22.             EsferaRoja.GetComponent<Renderer>().material.SetColor("_Color", Color.black);
23.             verde = false;
24.         }
25.
26.         if (rojo == true)
27.         {
28.             // Luz.transform.position = posRojo.position;
29.             Luz.GetComponent<Light>().color = Color.red;
30.             StartCoroutine(luzRojo());
31.             EsferaVerde.GetComponent<Renderer>().material.SetColor("_Color", Color.black);
32.                         EsferaAmarilla.GetComponent<Renderer>().material.SetColor("_Color",
Color.black);
33.             EsferaRoja.GetComponent<Renderer>().material.SetColor("_Color", Color.red);
34.             amarillo = false;
35.         }
36.     }
37.
```

Como se muestra en el código de CambioLuz, cuando un estado está a true llama a un método con StartCoroutine. Este método se usa para llamar a una corrutina para que el programa no siga hasta que dicha corrutina haya finalizado de ejecutarse. "Las corrutinas son una forma de realizar una operación a lo largo del tiempo en lugar de instantáneamente" [8].

Esto se usa para que el semáforo este los segundos que se indican en cada estado antes de cambiarse al siguiente. En mi juego está programado para que esté cinco segundos en las luces verde y roja y cuatro segundos en ámbar. Después de esperar eso segundos ponemos a verdaderos el siguiente estado al que debe pasar.

Para terminar con la clase semáforos, esta tiene un método que viene definido en una librería de Unity que es el OntriggerEnter. "OnTriggerEnter se llama cuando el Collider other ingresa al disparador." [8] .

Este uso del box collider mencionado al principio del epígrafe nos indicará cuando un jugador pasa por un semáforo. En este método comprobaremos si un objeto cuya etiqueta sea la del jugador ha entrado en nuestro box collider, y en función del estado del semáforo llamaremos al GameManager para indicar si el jugador ha pasado el semáforo en rojo o no.

A continuación, se muestra el código de esta función para que se vea en el código reflejada la explicación:

```
1. private void OnTriggerEnter(Collider other)
2.     {
3.         if (other.CompareTag("Player"))
4.         {
5.             if (rojo == true)
6.             {
7.                 FindObjectOfType<GameManager>().SemaforoSaltado();
8.             }
9.             else{
10.                FindObjectOfType<GameManager>().SemaforoNoSaltado();
11.            }
12.        }
13.    }
```

4.2.2 Jugador

En esta clase se define toda la lógica encargada del funcionamiento del coche del videojuego, el cual será controlado por el jugador.

Este objeto en Unity estará formado por el objeto cogido del paquete descargado de la tienda, la cámara principal, la cual se mete dentro del jugador para que se mueva en concordancia con este, y un objeto puntos, el cual será un texto que se muestre en la parte superior central de la pantalla mostrando los puntos que va obteniendo el jugador a medida que avanza el juego.

Entrando más en detalle en el código, esta clase tiene cinco métodos. De estos cinco métodos, tres son métodos que vienen con Unity, ya sea de manera predeterminada o en alguna librería.

Uno sería el Start que viene definido en Unity, en el cual solo se inicializa la posición inicial del coche para tenerla guardada para reiniciarle a dicha posición una vez acabe el juego.

Otro de los métodos que ya viene definido de forma predeterminada en Unity sería el Update, el cual se encargará de llamar al método que se encargará de mover el coche; y también se encargará de llamar a la función que controlará si el juego a llegado a su fin. Por último, este método también se encarga de mostrar los puntos en la pantalla una vez comience el juego.

El último de los tres métodos comentados sería Awake, el cual hará que cuando despierte el objeto por primera vez este se cree en la posición inicial cargada previamente en el Start. Según la página oficial de Unity Documentation:

Awake se usa para inicializar cualquier variable o estado del juego antes de que comience el juego. Se llama a Awake solo una vez durante la vigencia de la instancia del script. Awake se llama después de que todos los objetos se inicializan para que pueda hablar con seguridad con otros objetos o consultarlos. [8].

El coche cuando acaba una partida su estado es puesto en falso, para que se llame al método Awake para despertarlo al cargar una partida nueva. Este método lo inicializará para que aparezca en la posición adecuada.

A continuación, se muestran los tres métodos comentados del código de Unity para entender mejor lo explicado anteriormente:

```
1. private void Awake()
2.     {
3.         this.posicionInicial = this.transform.position;
4.     }
5.
6.     void Start()
7.     {
8.         this.transform.position = this.posicionInicial;
9.
10.    }
11.
12.    void Update()
13.    {
14.
15.        Movimiento();
16.
17.        if (FinCamino(posicionFinal.z))
18.        {
19.            FindObjectOfType<GameManager>().FinJuego();
20.        }
21.
22.        if (gameManager.listoInicio == 1)
23.        {
24.            gameManager.numSemaforosTexto.text = "Puntos: " + gameManager.puntos;
25.        }
26.
27.    }
28.
```

En esta clase también nos encontramos con el método Movimiento el cuál, como bien indica su nombre, se encargará del movimiento del coche. En primer lugar, deberá de ver si el juego ha comenzado, es decir si el usuario ha iniciado sesión. Después comprobará si el sensor se ha movido lo mínimo establecido y si el estado anterior es 0, lo que significa que acaban de bajar el sensor y no que aún no ha subido a su posición inicial.

Si se cumplen todos estos condicionantes se podrá a uno el estado actual para que se sepa que se acaba de mover el sensor y se comprobará si el coche estaba en movimiento o parado para que haga la acción contraria, es decir si estaba parado que se ponga en marcha y viceversa.

Por último, si no se cumplían los primeros condicionantes y se registra que es sensor ha vuelto a su posición inicial, se pondrá el estado actual a cero indicando que se ha vuelto a esta posición.

A continuación, se muestra el código del método movimiento implementado en el videojuego:

```
1. private void Movimiento()
2.     {
3.
4.         if (gameManager.mover() == 1 && gameManager.listoInicio == 1 && estadoAnterior ==
0)
5.         {
6.             estadoAnterior = 1;
7.             semaforos = FindObjectOfType<Semaforo>();
8.             if (estaMoviendo)
9.             {
10.
11.                 //transform.Translate(0, 0, 0);
12.                 estaMoviendo = false;
13.             }
14.             else if(!estaMoviendo)
15.             {
16.
17.                 //transform.Translate(Vector3.forward * Time.deltaTime * 5);
18.                 estaMoviendo = true;
19.             }
20.
21.
22.         }
23.         else if(gameManager.mover()==0)
24.         {
25.             estadoAnterior = 0;
26.         }
27.
28.         if (!estaMoviendo)
29.         {
```

```
30.             transform.Translate(0, 0, 0);
31.         }
32.
33.         if (estaMoviendo)
34.         {
35.             transform.Translate(Vector3.forward * Time.deltaTime * 5);
36.         }
37.
38.     }
39.
```

Por último, en esta clase nos encontramos el método booleano de FinCamino, el cual irá comprobando la posición del coche con la posición marcado como final, para que en cuanto el coche llegue a dicho punto el juego termine. Dicha posición marcada como final es calculada en la clase Game Manager, por lo que se explicará en el siguiente punto.

4.2.3 Game Manager

Esta clase es la encargada de coordinar el juego de una manera correcta haciendo que los diferentes componentes de este se conecten y todo funcione bien.

Game Manager es el encargado de hacer que el usuario se conecte con sus credenciales para poder cargar la partida y guardar los datos de esta cuando acabe. Se encarga también de la creación e inicialización de la partida cuando esta comienza, es el encargado de llevar la puntuación del juego a medida que este avanza y de finalizarlo cuando llega a su fin. Por último, se encarga también de traducir las coordenadas del sensor de movimiento para poder determinar cuando se ha movido lo suficiente como para hacer una acción sobre el coche.

El videojuego implementado en este trabajo es un videojuego dinámico, es decir que este se va creando en función de unos parámetros introducidos para hacer más largas o cortas las partidas. Esto se hace de esta forma, ya que los médicos podrán personalizar la duración de las partidas para cada paciente. Esta personalización se hará a través de la interfaz del médico, que se explicará

más adelante en el "Capitulo 7- Interfaz del médico", desde donde este podrá poner cuantos semáforos quiere que tenga la partida en concreto que este personalizando.

Entrando al detalle del código, en primer lugar, tenemos dos IEnumerator los cuales serán llamados como corrutina desde otros métodos. Estos IEnumerator sirven para realizar la conexión de Unity con la base de datos que se explicará en los puntos siguientes, más específicamente en el "Capítulo 6- Base de datos".

Para empezar, tenemos el método getNumSemaforos, el cual como indica su nombre nos devolverá los semáforos con los que se creará la partida para un determinado usuario. Para obtener estos semáforos se pasará una consulta a la base de datos por medio de una URL indicándole el DNI introducido por el usuario en el login del videojuego.

Esta conexión a través de la URL con la base de datos se hace usando UnityWebRequest. La descripción de este objeto según la página oficial de Unity es:

UnityWebRequest es un remplazo para el objeto WWW original de Unity. Este proporciona un sistema modular para componer peticiones HTTP y manejar respuestas HTTP. El objetivo principal del sistema de UnityWebRequest es permitirles a los juegos de Unity interactuar con backends Web modernos. También soporta características de alta demanda tal como solicitudes HTTP fragmentadas, operaciones de streaming POST/PUT y un control completo sobre encabezados HTTP y verbs. [10].

Con este UnityWebRequest obtenemos los números de semáforos, los cuales si son cero es porque el DNI introducido por el usuario no existe. Si este numero es diferente de 0 entonces se cargará un nuevo juego con este número de semáforos.

En el siguiente código se ve reflejada la explicación:

```
1. IEnumerator getNumSemaforos(string DNI)
2.     {
3.         string URLCompletaSemaforos = numSemaforosURL + "DNI=" + DNI;
4.         UnityWebRequest hs_get = UnityWebRequest.Get(URLCompletaSemaforos);
```

```
        yield return hs_get.SendWebRequest();
        if (hs_get.error != null)
        {
            this.numSemaforosTexto.text = "Ha habido un error";
        }
        else
        {
            string datos = hs_get.downloadHandler.text;
            try
            {
                int numSemaforosParsed = Int32.Parse(datos);
                if (numSemaforosParsed == 0)
                {
                    this.numSemaforosTexto.text = "No se ha encontrado el DNI en la base de datos";
                }
                else
                {
                    NewGame(numSemaforosParsed);
                }

            }
            catch (FormatException)
            {
                Debug.Log("Error when parsing the semaphore number");
            }
        }
    }

```

Después de esta función, tenemos otro IEnumerator el cual sirve para guardar los datos de la partida cuando se haya finalizado esta. El método es similar al de la obtención del número de semáforos solo que esta vez la URL sirve para guardar datos en la base de datos insertando en la URL los datos que se tienen que proporcionar a la consulta como son el DNI y los puntos obtenidos en la partida. Además, tenemos diferentes controles por si los datos son incorrectos que nos avisen por pantalla de que no se han podido guardar los datos de la partida.

A continuación, podemos ver el código del método explicado:

```
1. IEnumerator guardarDatosPartida(string DNI, int puntos)
2.     {
3.         // Incluimos los puntos en el path de la URL
4.          string URLCompletaSetScore = insertFinalScoreURL + "DNI=" + DNI + "&PUNTOS=" +
puntos;
5.         UnityWebRequest hs_get = UnityWebRequest.Get(URLCompletaSetScore);
6.         yield return hs_get.SendWebRequest();
7.         if (hs_get.error != null)
8.         {
9.             this.numSemaforosTexto.text = hs_get.error;
10.         }
11.         else
12.         {
13.             if (Int32.TryParse(hs_get.downloadHandler.text, out int numSemaforosParsed))
14.             {
15.                 Debug.Log("El número es: " + numSemaforosParsed);
16.             }
17.             else
18.             {
19.                     Debug.Log("La cadena de texto no puede ser convertida en un número
entero.");
20.             }
21.             //int numSemaforosParsed = Int32.Parse(hs_get.downloadHandler.text);
22.             if (numSemaforosParsed == 0)
23.             {
24.                 this.numSemaforosTexto.text = "Error guardando en la base de datos";
25.             }
26.             else
27.             {
28.                 this.numSemaforosTexto.text = "Guardado en la base de datos con exito";
29.             }
30.         }
31.     }
32.
```

En esta clase también tenemos un método llamado *NewGame*. Esta clase es llamada una vez el DNI del usuario haya sido introducido, por lo que desactivara el botón y el texto que teníamos a modo de login.

Después de esto y con el número de semáforos ya obtenido este método se encargará de crear las instancias de las ciudades y de estos semáforos, poniéndolos a una distancia determinada entre ellos. Se ha programado de tal

forma que en una ciudad entran seis semáforos por lo que en función de eso también se calcula cuantas ciudades deberían de crearse.

Además, al final de este método se determinará la posición final del juego mediante la posición del último semáforo creado.

La forma en la que este método ha sido implementado en el código es la siguiente:

```
1. private void NewGame(int numSem)
2.     {
3.         //Debug.Log(numSem);
4.         this.listoInicio = 1; /* Marcamos que el juego está listo para ser comenzado */
5.         inputDNI.SetActive(false); /* Quitamos el campo de texto de DNI en la UI*/
6.         buttonDNI.SetActive(false); /*Quitamos el boton de enviar el DNI*/
7.
8.         int num_semaforos = numSem;
9.         int num_ciudades;
10.        SetPuntos(0);
11.        this.jugador.gameObject.SetActive(true);
12.        for (int i = 0; i < num_semaforos - 1; i++)
13.        {
14.
15.            Instantiate(semaforos);
16.              semaforos.transform.position = new Vector3(semaforos.transform.position.x,
semaforos.transform.position.y,
17.                  semaforos.transform.position.z+40);
18.        }
19.        num_ciudades = (num_semaforos) / 6;
20.        for (int j = 0; j < num_ciudades; j++)
21.        {
22.
23.            Instantiate(ciudad);
24.                  ciudad.transform.position = new Vector3(ciudad.transform.position.x,
ciudad.transform.position.y,
25.                  ciudad.transform.position.z + 250);
26.        }
27.
28.        jugador.posicionFinal = semaforos.transform.position + new Vector3(0, 0, 10);
29.        //Debug.Log(jugador.posicionFinal);
30.
31.    }
32.
```

También tenemos un método *FinJuego* que, llamada a la corrutina de guardar los datos de la partida en la base de datos, pone en falso al jugador para que cuando se despierte de nuevo se cree en la posición adecuada y resetea la escena para que todo se reinicie.

Para terminar, en esta clase tenemos dos métodos usados para mover el coche en función de las coordenadas que nos lleguen del sensor de movimiento.

El primero de los métodos sería obtenerCoordenadas el cual lee un string de la clase LecturaArchivo, de la parte del sensor de movimiento, y lo transforma a tres números, uno para cada componente de las coordenadas al ser unas coordenadas (x, y, z).

El segundo de estos métodos es mover y lo que hará será leer la variable z de las coordenadas obtenidas por el método anterior y determinar si el jugador ha movido suficiente el sensor como para hacer que el coche se mueva o pare. En este caso hemos determinado que el valor que debe de tener la z es de -2500 que es un movimiento razonable y lo más parecido al que debes de hacer cuando estas conduciendo.

La implementación de estos dos métodos se muestra a continuación:

```
1. public void obtenerCoordenadas()
2.     {
3.
4.         coordenadas = lectura.lastLine;
5.         coordenadas = coordenadas.Trim('(', ')');
6.         string[] tokens = coordenadas.Split(',');
7.
8.
9.         num1 = float.Parse(tokens[0]);
10.        num2 = float.Parse(tokens[1]);
11.        num3 = float.Parse(tokens[2]);
12.        num1 = num1 / 10;
13.        num2 = num2 / 10;
14.        num3 = num3 / 10;
15.
16.    }
```

```
17.
18.     public int mover()
19.     {
20.         if (num3 < -2500)
21.         {
22.             mov = 1;
23.             return mov;
24.         }
25.         else
26.         {
27.             mov = 0;
28.             return mov;
29.         }
30.     }
31.
```

Capítulo 5 - Sensor de movimiento

El sensor movimiento es un apartado importante en el trabajo, ya que este es una pieza fundamental para jugar al videojuego y por tanto para el trabajo en general.

El sensor utilizado para el trabajo es un Nicla Sense ME [11]. Este sensor es una placa del tamaño aproximado de una moneda de dos euros que combina cuatro sensores.

Según la tienda oficial de Arduino donde se vende este sensor Nicla Sense ME, nos dice que:

Diseñado para analizar fácilmente el movimiento y el entorno circundante, de ahí la "M" y la "E" en el nombre, mide la rotación, la aceleración, la presión, la humedad, la temperatura, la calidad del aire y los niveles de CO2 mediante la introducción de sensores Bosch Sensortec completamente nuevos en el mercado. [11].

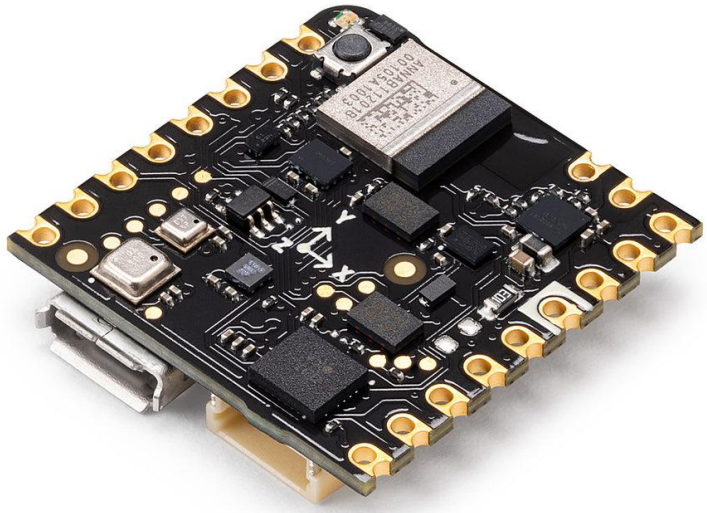

Figura 5: Nicla Sense ME

5.1 Tecnologías aplicadas al sensor de movimiento

Para este sensor de movimiento las tecnologías usadas han sido Arduino, Python y BLE (Bluetooth Low Energy), la cual es un bluetooth que funciona en frecuencias de 2,4 GHz y a través de la cual puedes conectarte a periféricos de bajo consumo de energía.

En primer lugar, según la página oficial de [12], Arduino se basa en una placa electrónica con un microprocesador, el cual puede programarse. Esto sirve para establecer conexiones entre sensores y actuadores de manera muy sencilla.

A través de Arduino el usuario puede programar que quiere en concreto que haga una placa determinada, conectándola al ordenador a través de un USB para que Arduino la reconozca y para darle alimentación.

En este caso, el Arduino que se le instaló al sensor Nicla Sense ME es el proporcionado por la página oficial de Arduino [13].

A través de este programa las variables se inicializan y se establecen las conexiones con los diferentes sensores que conforman el Nicla Sense ME, haciendo que retransmita la señal de estos sensores recogiéndolos en unas variables internas, de las cuales en nuestro caso concreto se recogerá la aceleración que se imprimirá en un fichero, el cual después será leído por Unity.

La segunda de las tecnologías aplicadas al Nicla Sense ME es Python. Este lenguaje se utiliza para describir un código el cual trata de hacer la conexión con el sensor y una vez establecida la conexión a través de una señal Bluetooth, le solicita la aceleración para grabarla en un fichero el cual genera el propio programa. Esta conexión la hace a través de la dirección de la placa la cual se pone de forma manual en el programa.

Hay que destacar el funcionamiento de un método que se llama en el bucle infinito del main. Esta función es la encargada de obtener la aceleración del sensor dada en coordenadas (x, y, z), abrir el fichero donde copia el valor de la aceleración obtenido, cerrar el fichero y esperar 0,5 segundos. Esta espera se hace, ya que el fichero tiene que ser accedido por Unity mediante una clase, después descrita. Para evitar fallos de concurrencia debe de dejar cierto tiempo sin usar el fichero, ya que este no puede ser abierto por dos procesos diferentes al mismo tiempo.

Este fichero, para evitar que contenga mucha información y pueda ser demasiado pesado, cada vez que es abierto por Python para escribir la aceleración lo que hace es sobrescribir la línea, por lo que lo único que se guarda en el fichero es la última aceleración obtenida del sensor.

A continuación, se muestra el código de la función específica descrita para un mejor entendimiento:

```
1. def notification_handler(sender, data: bytearray):
2.       """Simple notification handler which prints the data received and writes it to a file."""
3.     accel = struct.unpack('<3f',data)
4.     #print(accel)
5.     with open("C:/Unity/Projects/movimiento coche/Assets/Scripts/pythonScript/datos.txt", "w") as file:
6.         file.write(str(accel) + "\n")
7.         file.flush()
8.         file.close()
9.     time.sleep(0.5)
10.
```

5.2 Desarrollo en Unity

En este apartado se explicará como se consiguen los datos de la aceleración del sensor en Unity. Esta conexión se logra a partir de dos clases de Unity.

A continuación, podemos observar un diagrama que muestra la jerarquía de clases y como se relacionan estas:

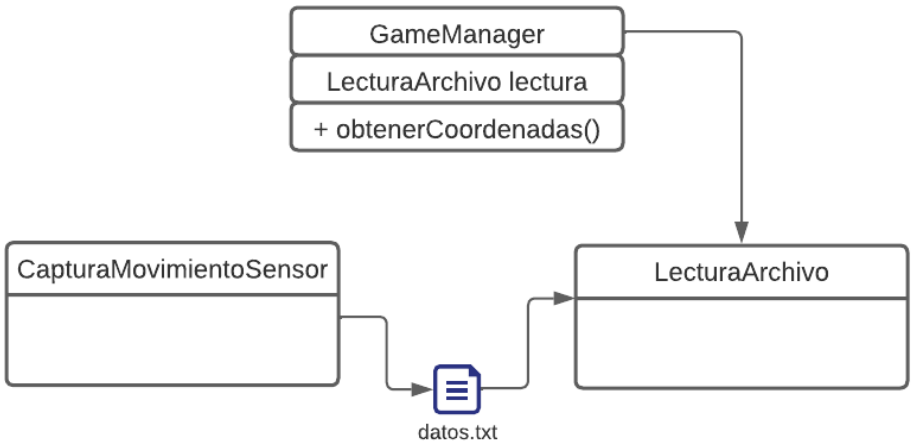

Figura 6: Diagrama de clases Sensor

La primera clase, Captura Movimiento Sensor se encarga de ejecutar el script de Python ya mencionado anteriormente que establecía la conexión con el sensor y creaba un fichero donde escribía la aceleración del sensor.

La segunda clase, Lectura Archivo, se usa para la lectura del archivo creado por el código de Python y en el cual estará la última aceleración pasada por el sensor, y por tanto la posición de este.

5.2.1 Captura Movimiento Sensor

Esta clase sirve para ejecutar el código de Python desde Unity, ya que necesitamos que se haga la conexión con el sensor cuando se empiece a jugar al videojuego.

Para ejecutar el código de Python hace falta hacerlo mediante un proceso en Unity. En esta clase se abrirá un proceso el cual se encargué de ejecutar el código de Python en segundo plano, evitando que se abra una ventana emergente de la ejecución y haciendo que la información no salga por consola.

Además de este proceso, para que se logré ejecutar el código de Python mientras se está jugando al videojuego hace falta usar un hilo para la ejecución de este en segundo plano.

Este hilo sirve para poder ejecutar el script de Python, el cual es un bucle infinito al tener que estar recibiendo, todo el rato, información del sensor, a la vez que se está ejecutando el videojuego.

El código que ha sido desarrollado para lo descrito en este apartado es el siguiente:

```
1. public class CapturaMovimientoSensor : MonoBehaviour
2. {
3.
4.     private Process process;
5.     private Thread thread;
6.
7.     void Start()
8.     {
```

```
        // Crear hilo para ejecutar el script de Python
        thread = new Thread(new ThreadStart(ExecutePythonScript));
        thread.Start();

    }

    void OnDestroy()
    {
        // Finalizar proceso si existe
        if (process != null && !process.HasExited)
        {
            process.Kill();
        }

        // Finalizar hilo si existe
        if (thread != null && thread.IsAlive)
        {
            thread.Abort();
        }
    }

    void ExecutePythonScript()
   {
            // Crear proceso
            process = new Process();
            process.StartInfo.FileName = "python";
          process.StartInfo.Arguments = "Assets/Scripts/pythonScript/niclaBLE_client.py";
            process.StartInfo.UseShellExecute = false;
            process.StartInfo.RedirectStandardOutput = true;
            process.StartInfo.CreateNoWindow = true;
            process.Start();
            process.WaitForExit();
        }
}

```

5.2.2 Lectura Archivo

Esta clase es la que se encarga de la lectura del archivo en el que el código de Python va copiando las coordenadas de la posición del sensor de movimiento.

Esta clase esta compuesta por un método Start que se ejecuta en la creación del objeto al que vaya asociado, y lo que hace es un ciclo infinito en el cual accede al fichero creado por el ejecutable de Python y del cual extrae la ultima línea escrita. Además, en este método se controla mediante una excepción si al intentar leer el archivo, este esta siendo abierto por otro proceso, en cuyo caso lo que hará será volver a probar en 0,1 segundos, al esperar este rato en cada vuelta del bucle para reducir el uso de recursos del sistema.

Por último, esta clase tiene un método OnDestroy el cuál leerá la última línea del archivo en el caso en el que el objeto al que ha sido asignado se destruya.

Para un mejor entendimiento de la explicación, a continuación, se mostrará el código del método Start explicado:

```
1. private IEnumerator Start()
2.     {
3.         while (true)
4.         {
5.             try
6.             {
7.               using (var stream = new FileStream(filePath, FileMode.Open, FileAccess.Read,
FileShare.ReadWrite))
8.                 using (var reader = new StreamReader(stream))
9.                 {
10.                    while (!reader.EndOfStream)
11.                    {
12.                        lastLine = reader.ReadLine();
13.                    }
14.                }
15.            }
16.            catch (IOException)
17.            {
18.
19.            }
20.
21.            yield return new WaitForSeconds(0.1f);
22.        }
23.    }
24.
```

Capítulo 6 - Base de datos

La base de datos servirá como intermediario entre el médico y el paciente, es decir servirá de unión entre la interfaz del médico y el videojuego del paciente.

Esta se usará para guardar los datos de todos los pacientes que jueguen al videojuego y los resultados de las partidas que hayan jugado para lograr tener un seguimiento de estos. Además, en esta también se guardará la información de las partidas personalizadas para los pacientes por los médicos.

6.1 Tecnologías usadas en la base de datos

El lenguaje usado para el desarrollo de la base de datos es SQL. Según la página oficial de Microsoft, SQL es un lenguaje de computación que se usa para trabajar con conjuntos de datos y las relaciones entre ellos. SQL se caracteriza por su fácil entendimiento a diferencia que otros lenguajes de computación, además de ser un estándar internacional. [14].

En cuanto a la visualización de la base de datos desarrollada se ha usado phpMyAdmin.

"PhpMyAdmin es una aplicación web escrita en PHP y contiene -como la mayoría de las aplicaciones web- código cliente XHTML, CSS y JavaScript. Proporciona una interfaz web completa para administrar bases de datos MySQL y es ampliamente reconocida como la aplicación líder en este campo." [15].

6.2 Desarrollo de la base de datos

La base de datos ha sido desarrollada en PhpMyAdmin y cuenta con cuatro tablas. La relación entre estas tablas tiene la siguiente forma:

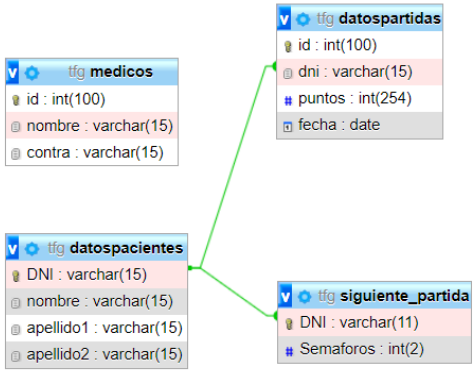

Figura 7: Relaciones Base de Datos

Como podemos observar en la Figura 5 tenemos una tabla que no tiene ninguna relación y es la tabla de los médicos. En esta tabla la clave primaria sería el id del médico y luego tendría su nombre y contraseña.

Luego las otras tres tablas se relacionan con el DNI del paciente, al utilizarse las tres para los pacientes que jueguen al videojuego.

Por un lado, tendríamos la tabla de los datos pacientes en los cuales se guardan las credenciales del paciente. Podemos observar que no tiene contraseña este y esto es porque para tener una cuenta debe haberte dado de alta el médico con tu DNI por lo que con esa autentificación es suficiente.

Por otro lado, estaría la tabla de los datos de las partidas que haya jugado un paciente, donde se guarda el DNI del paciente con los puntos que haya hecho en esa partida y con la fecha en la que se haya jugado, para poder hacer un seguimiento de la evolución del paciente.

Por último, tenemos la tabla siguiente_partida que se usa para programar cuantos semáforos quiere el médico que tenga la siguiente partida de un paciente identificándolo por el DNI.

Para finalizar este apartado hay que comentar los diferentes usuarios que van a acceder a la base de datos, sus roles y las acciones que harán sobre la base de datos. Para ello definiremos dos tipos de usuarios que querrán acceder a la base de datos, uno con el rol de médico y el otro con el rol de paciente.

El primero de estos, el médico, tendrá permisos de lectura y escritura. De lectura ya que el médico quiere visualizar por la interfaz de este la lista de pacientes registrados y a su vez los resultados de las partidas que estos hayan jugado. Además, tendrá permisos de escritura, ya que el médico al personalizar las partidas de los pacientes debe de escribir en la base de datos, de la misma forma que si quiere dar de alta a algún paciente.

El segundo de estos, el paciente, tendrá también permisos de lectura y escritura. Deberá tener permisos de lectura para poder iniciar sesión para jugar al videojuego y para que el videojuego se cargue con el número de semáforos que tenga guardado en la base de datos, y también deberá tener permisos de escritura para escribir los resultados de la partida jugada en la base de datos.

Capítulo 7 - Interfaz del médico

La interfaz del médico será aquella aplicación que usa el médico para cualquier gestión que tenga que ver con los pacientes y el videojuego. El médico usará dicha interfaz para ver la evolución de los pacientes, añadir nuevos y personalizar las partidas de estos.

7.1 Tecnologías aplicadas a la interfaz del médico

El lenguaje usado para el desarrollo de la interfaz del médico ha sido PHP y HTML. Para la escritura de todos los códigos en estos lenguajes se ha usado la herramienta Visual Studio Code.

La definición que nos ofrece del lenguaje PHP la propia página oficial de PHP es la siguiente:

PHP es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado en HTML. En lugar de usar muchos comandos para mostrar HTML (como en C o en Perl), las páginas de PHP contienen HTML con código incrustado que hace "algo". El código de PHP está encerrado entre las etiquetas especiales de comienzo y final que permiten entrar y salir del "modo PHP". [16].

PHP también destaca por la facilidad para desarrollar código en este lenguaje y el fácil y sencillo entendimiento que este tiene. En la interfaz del médico, se usa PHP para todas las consultas a la base de datos.

Por otro lado, como se ha mencionado anteriormente, para el desarrollo de la interfaz del médico también se usa HTML.

La definición que nos da, del lenguaje HTML, la página oficial de Mozilla es la siguiente:

HTML (Lenguaje de Marcas de Hipertexto, del inglés HyperText Markup Language) es el componente más básico de la Web. Define el significado y la estructura del

contenido web. Además de HTML, generalmente se utilizan otras tecnologías para describir la apariencia/presentación de una página web (CSS) o la funcionalidad/comportamiento (JavaScript). [17].

7.2 Desarrollo de la interfaz del médico

En este apartado se describirán las pantallas que forman toda la interfaz del médico. A continuación, se mostrará un diagrama de flujo de la interfaz del médico para tener una visión general del flujo principal de esta y entender de mejor forma la explicación del desarrollo de la interfaz del médico.

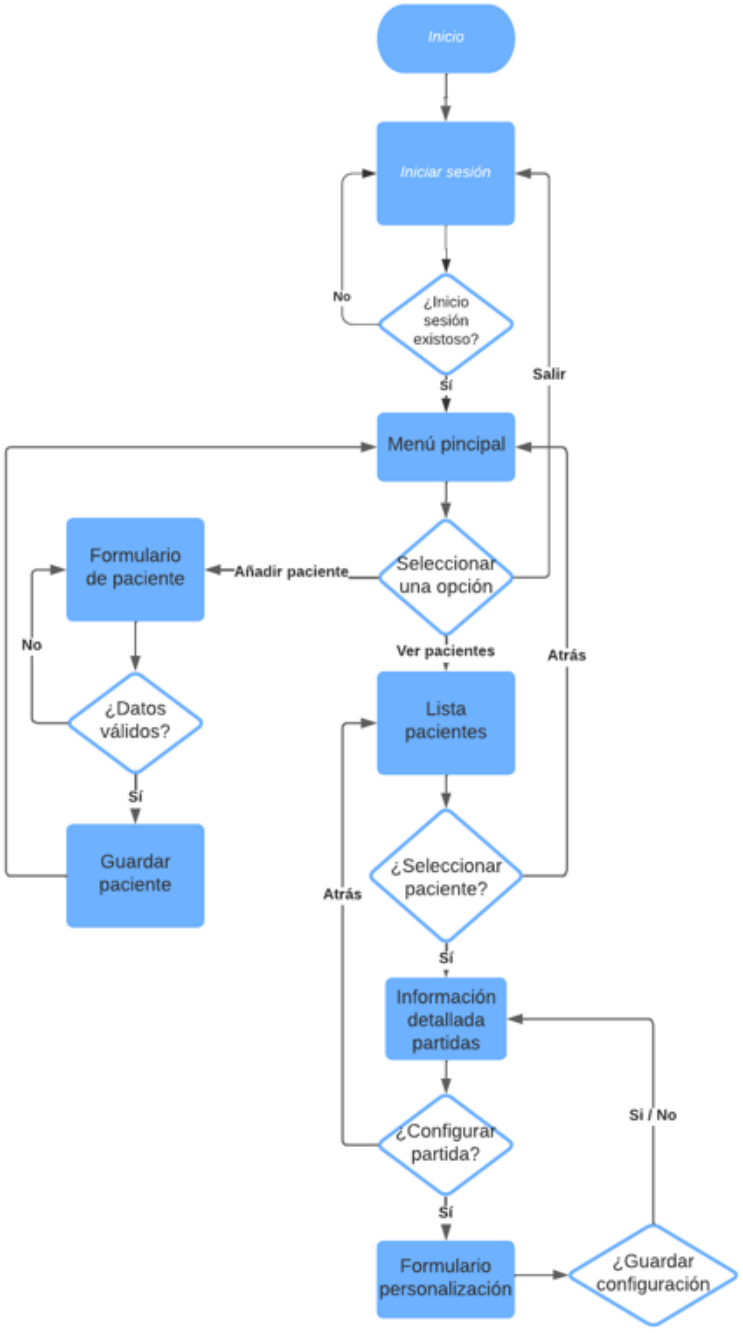

Figura 8: Diagrama de Flujo

En primer lugar, aparece un login de usuario y contraseña en el cual el médico o administrador deberá de introducir sus credenciales para entrar en dicha página web.

Una vez introducidos las credenciales y si estos son correctos se mostrará una pantalla de menú principal formada por tres botones. Estos tres botones permiten al

médico añadir a un paciente, ver a los pacientes ya registrados en el sistema y salir de la página web cerrando sesión.

Si el médico decide añadir a un nuevo paciente la pantalla que se mostraría es un formulario con los datos que se tienen que rellenar para que se pueda hacer la consulta de guardado de dicho paciente en el sistema. Luego tendría los botones de guardar en caso de querer guardar al paciente o de atrás en caso de que se haya equivocado.

Si el médico decide ver los pacientes guardados en el sistema la pantalla que se abrirá estará formada por una tabla con tantas filas como pacientes haya registrados y en las columnas saldrán los DNI, nombres y apellido de los usuarios. Además, también saldrá un buscador para que el médico pueda introducir un DNI en específico y solo saldrá dicho DNI en la tabla si es que existe, si no existe saldrá vacía.

Desde esta última pantalla el médico puede acceder a los datos de las partidas jugadas por un usuario en específico pinchando en el DNI de este. La información de las partidas saldrá en modo de tabla teniendo en columnas el DNI del paciente, los puntos realizados en la partida y la fecha en la que se jugó.

Por último, desde esta pantalla el médico tiene un botón para configurar la siguiente partida. Pinchándolo se le abrirá un formulario para personalizar la siguiente partida del usuario seleccionado, en la cual el médico deberá de poner cuantos semáforos quiere que tenga dicha partida.

7.3 Casos de uso

Según la página oficial de IBM [18],los casos de uso definen una secuencia de acciones dando lugar a un resultado observable. Estos casos de uso representan una serie de requisitos funcionales en cuanto a procesos empresariales y de sistema.

En este trabajo se pueden describir casos de uso de como un médico interactúa con la interfaz del médico para llevar a cabo todas las acciones posibles en esta. A continuación, se mostrará el diagrama que describirá todos los casos de uso de la aplicación y posteriormente se explicará en detalle cada uno de ellos.

7.3.1 Diagrama de casos de uso

En la siguiente figura se muestran los casos de uso de la interfaz del médico. En este diagrama podemos observar que el único actor principal es el médico al ser el único en tener acceso a la página web. También observamos el ecosistema de los casos de uso que es la propia interfaz del médico y dentro de esta se encuentran todos los casos de uso.

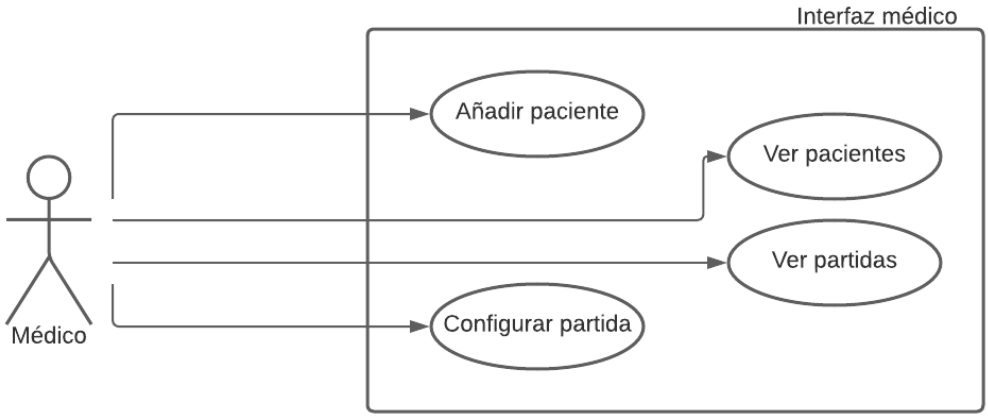

Figura 9: Diagrama Casos de uso

7.3.2 Caso de uso 1: Añadir paciente

Caso de uso 1	Añadir paciente
Objetivo	El objetivo de este caso de uso es añadir un paciente nuevo a la base de datos para que pueda jugar al videojuego.

Caso de uso 1	Añadir paciente
Precondición	El médico haya hecho el login de manera exitosa introduciendo sus credenciales. La conexión con la base de datos se haya realizado de forma correcta.
Postcondición	Se guardan los datos introducidos por el médico en la base de datos, creando un usuario para un paciente.
Actor	Médico
Flujo Normal	- El médico selecciona la opción de añadir paciente del menú principal. - El sistema saca por pantalla un formulario con los datos que tiene que rellenar el médico del paciente. - El médico introduce todos los datos solicitados y le da al botón de guardar. - El sistema hace una consulta para guardar los datos en la base de datos.

Tabla 1: Caso de uso 1: Añadir Paciente

7.3.3 Caso de uso 2: Ver pacientes

Caso de uso 2	Ver pacientes
Objetivo	El objetivo de este caso de uso es obtener una visualización de todos los pacientes que se encuentren en la base de datos
Precondición	El médico haya hecho el login de manera exitosa introduciendo sus credenciales. La conexión con la base de datos se haya realizado de forma correcta.
Postcondición	Se visualizan en forma de tabla todos los pacientes de la base de datos con todos sus datos
Actor	Médico
Flujo normal	- El médico selecciona la opción de ver pacientes del menú principal. - El sistema hace una consulta a la base de datos para sacar la información de todos los pacientes. - El sistema muestra por pantalla todos los datos de los pacientes en formato tabla.

Tabla 2: Caso de uso 2: Ver pacientes

7.3.4 Caso de uso 3: Ver partidas

Caso de uso 3	Ver partidas
Objetivo	El objetivo de este caso de uso es tener una visualización de las partidas jugadas por un paciente en concreto.
Precondición	El médico haya hecho el login de manera exitosa introduciendo sus credenciales. La conexión con la base de datos se haya realizado de forma correcta. Estar en la pantalla de ver pacientes.
Postcondición	Se visualizarán todas las partidas de un paciente determinado seleccionado por el médico.
Actor	Médico
Flujo normal	- El médico seleccionará un paciente de la lista de pacientes, sacados por pantalla en el caso de uso anterior, pinchando sobre el DNI de este. - El sistema hace una consulta a la base de datos con el DNI seleccionado para recoger la

Caso de uso 3	Ver partidas
	información de las partidas jugadas por este paciente. - El sistema saca por pantalla una tabla con la información de todas las partidas jugadas por el paciente seleccionado.

Tabla 3: Caso de uso 3: Ver partidas

7.3.5 Caso de uso 4: Configurar partida

Caso de uso 4	Configurar partida
Objetivo	El objetivo de este caso de uso es configurar las próximas partidas que vaya a jugar un paciente en específico.
Precondición	El médico haya hecho el login de manera exitosa introduciendo sus credenciales. La conexión con la base de datos se haya realizado de forma correcta. Estar en la pantalla de ver partidas.
Postcondición	Se guardarán los datos de la configuración de la siguiente partida de un paciente seleccionado por el médico.
Actor	Médico

Caso de uso 4	Configurar partida
Flujo normal	- El médico le dará al botón de configurar próxima partida que se encuentra en la pantalla de ver partidas. - El sistema sacará un formulario solicitando el número de semáforos de la siguiente partida y guardando el DNI del paciente en el que se estaba. - El médico introduce un número válido de semáforos y le da a guardar. - El sistema hace una consulta guardando el numero de semáforos introducidos por el usuario en el DNI guardado de la pantalla ver partidas.

Tabla 4: Caso de uso 4: Configurar partida

Capítulo 8 - Conclusiones y trabajo futuro

8.1 Conclusiones

Con el desarrollo de este trabajo hemos podido acercarnos a un prototipo de un videojuego para las personas que se encuentren en terapia de rehabilitación, lo cual era el objetivo principal del proyecto. Este prototipo se puede enfocar a personas que tengan que hacer uso de una coordinación entre lo que ven en el videojuego y el movimiento que se solicita por este.

Cabe destacar que se ha logrado desarrollar un proyecto que cumple todos los objetivos marcados al comienzo del desarrollo de este. Se ha conseguido obtener un videojuego orientado para personas que se encuentren en terapia de rehabilitación al jugarlo usando un sensor de movimiento. Además, se ha desarrollado una interfaz para el médico, donde este pueda observar los resultados de los pacientes que jueguen al videojuego.

Para poder jugar al videojuego creado en el trabajo, todos los archivos necesarios se encuentran en el siguiente repositorio público:

https://drive.google.com/drive/folders/1YrcPpNg7SdvE8oqGyMtYmy7nMJLtzyOB?usp=share_link

Para poder importar el proyecto sigue la guía de importación que se encuentra en Apéndice A - Guía para importar el proyecto.

8.2 Trabajo futuro

Una vez finalizado este proyecto hay que definir el trabajo que habría que hacer a partir de este punto. Para empezar el desarrollo de la base de datos se hizo sobre una base de datos cerrada sin estar en un servidor, lo que hace que solo se pueda jugar al videojuego desde un ordenador donde tenga importada dicha base de datos.

En un futuro, esta base de datos lo ideal sería que estuviera subida en un servidor de tal forma que el videojuego esté conectado a esta por internet haciendo que pueda

jugar al videojuego cualquier paciente desde su casa sin necesidad de tener dicha base de datos metida en su ordenador. Esto sería la mejor solución, ya que el proyecto también está enfocado para personas que se encuentran en sus casas y ya no en el hospital donde el ordenador sería facilitado por el propio hospital con la base de datos importada.

Otro aspecto que mejorar de cara al futuro sería la interfaz del médico, haciendo que la base de datos recoja resultados más exactos como cuantas veces se ha movido el sensor o el rango de movimiento de este. Esto se mostraría en la interfaz del médico mejorando el análisis de los resultados del paciente y mostrando en mayor detalle su evolución.

Además, también se podría implementar la opción de que el médico pueda personalizar la posición a la que el sensor contesta para cada paciente en función de la movilidad que este pueda tener en el pie. Esto ampliaría el prototipo desarrollado haciendo una mayor personalización de las partidas, logrando que estas se ajusten más a las necesidades de cada paciente.

Por último, otra mejora posible para implementar a este trabajo sería la posibilidad de jugar con dos sensores de movimiento, uno para cada pie. De esta forma, el paciente debería de frenar con uno de los pies y acelerar con el otro, pero con el mismo mecanismo y movimiento que esta implementado ahora. Con esto lo que se conseguiría sería aumentar la dificultad en el apartado de la coordinación entre lo que ve el usuario en el juego y el movimiento de debe de hacer.

Conclusions and future work

Conclusions

With the development of this work we have been able to get closer to a prototype of a video game for people undergoing rehabilitation therapy, which was the main objective of the project. This prototype can be focused on people who have to make use of a coordination between what they see in the video game and the movement requested by it.

It should be noted that we have managed to develop a project that meets all the objectives set at the beginning of its development. It has been possible to obtain a video game aimed at people undergoing rehabilitation therapy by playing it using a movement sensor. In addition, an interface has been developed for the doctor, where he/she can observe the results of the patients who play the videogame.

In order to play the video game created in the work, all the necessary files can be found in the following public repository:

https://drive.google.com/drive/folders/1YrcPpNg7SdvE8oqGyMtYmy7nMJLtzyOB?usp=share_link

In order to import the project, follow the import guide in Appendix A - Guide to import the project.

Future Work

Once this project is finished, it is necessary to define the work to be done from this point onwards. To begin with, the development of the database was done on a closed database without being on a server, which means that the videogame can only be played from a computer where the database has been imported.

In the future, this database would ideally be uploaded to a server in such a way that the videogame is connected to it via the internet, so that any patient can play the videogame from home without the need to have the database on their computer. This

would be the best solution, since the project is also focused on people who are at home and not in the hospital, where the computer would be provided by the hospital itself with the imported database.

Another aspect to improve for the future would be the interface to the doctor, making the database collect more accurate results such as how many times the sensor has moved or the range of movement of the sensor. This would be displayed on the doctor's interface, improving the analysis of the patient's results and showing in greater detail the patient's evolution.

In addition, it would also be possible to implement the option for the doctor to customise the position to which the sensor responds for each patient depending on the mobility they may have in their foot. This would extend the prototype developed by making the items more personalised, making them more tailored to the needs of each patient.

Finally, another possible improvement to implement in this work would be the possibility of playing with two movement sensors, one for each foot. In this way, the patient would have to brake with one foot and accelerate with the other, but with the same mechanism and movement that is implemented now. This would increase the difficulty in the coordination between what the user sees in the game and the movement he/she has to make.

Capítulo 9 - Bibliografía

[1] C. J. M. y. Y. K. Laurie A. Malone, «Active Videogaming Interventions in Adults with Neuromuscular Conditions: A Scoping Review,» p. 1, Junio 2022.

[2] R. T. S. J. W. D. Z. S. C. y. D. M. W. Jason K Hsu, «A ‘‘Wii’’ bit of fun: The effects of adding Nintendo Wii Bowling to a standard exercise regimen for residents of long-term care with upper extremity dysfunction,» 25 Marzo 2010.

[3] U. C. «Cátedra BOSCH-UCM en Inteligencia Artificial aplicada a Internet de las Cosas,» [En línea]. Available: https://www.ucm.es/catedrabosch/.

[4] U. Technologies, «Empieza a usar Unity,» [En línea]. Available: https://unity.com/es/learn/get-started#:~:text=Unity%20es%20m%C3%A1s%20que%20un,los%20productos%20de.

[5] Unity Technologies, «Unity Support - Unity Asset Store,» [En línea]. Available: https://support.unity.com/hc/es/articles/7645361023892-Unity-Asset-Store-qu%C3%A9-es-y-c%C3%B3mo-funciona.

[6] Unity, «Unity Asset Store,» [En línea]. Available: https://assetstore.unity.com/packages/3d/vehicles/land/simple-cars-pack-97669#description.

[7] Unity, «Unity Asset Store,» [En línea]. Available: https://assetstore.unity.com/packages/3d/environments/urban/city-package-107224.

[8] Unity Technologies, «Unity Documentation,» [En línea]. Available: https://docs.unity3d.com/es/530/ScriptReference/index.html.

[9] Unity Technologies, «Unity Documentation - Box Collider,» [En línea]. Available: https://docs.unity.cn/es/2018.4/Manual/class-BoxCollider.html.

[10] Unity Technologies, «Unity Documentation - UnityWebRequest,» [En línea]. Available: https://docs.unity3d.com/es/530/Manual/UnityWebRequest.html.

[11] Arduino S.r.l, «Tienda Oficial Arduino,» [En línea]. Available: https://store.arduino.cc/products/nicla-sense-me.

[12] Ingeniería MCI Ltda, «Arduino cl,» [En línea]. Available: https://arduino.cl/que-es-arduino/.

[13] Arduino, «Arduino Docs,» [En línea]. Available: https://docs.arduino.cc/tutorials/nicla-sense-me/web-ble-dashboard.

[14] Microsoft Support, «Support Microsoft - Access SQL,» [En línea]. Available: https://support.microsoft.com/es-es/office/access-sql-conceptos-b%C3%A1sicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671#:~:text=La%20sintaxis%20SQL%20se%20basa,Basic%20para%20Aplicaciones%20(VBA).

[15] M. Delisle, Mastering PhpMyAdmin 2. 11 for Effective MySQL Management, Packt Publishing, Limited, 2008.

[16] The Php Group, «Php,» [En línea]. Available: https://www.php.net/manual/es/intro-whatis.php.

[17] Mozilla Corporation, «HTML: Lenguaje de etiquetas de hipertexto,» [En línea]. Available: https://developer.mozilla.org/es/docs/Web/HTML.

[18] IBM , «Definición de casos de uso,» [En línea]. Available: https://www.ibm.com/docs/es/elms/elm/6.0.3?topic=requirements-defining-use-cases.

[19] Unity Technologies, «Empieza a usar Unity,» [En línea]. Available: https://unity.com/es/learn/get-started#:~:text=Unity%20es%20m%C3%A1s%20que%20un,los%20productos%20de%20su%20imaginaci%C3%B3n.

APÉNDICES

Apéndice A - Guía para importar el proyecto

Para poder jugar al videojuego de manera correcta es necesario que se descarguen todas las carpetas que se encuentran en el repositorio.

Por un lado, tendrás que descargar la base de datos que se encuentra en el repositorio y exportarla a phpmyadmin.

Luego tendrás que descargar la carpeta de la interfaz del médico y guardarla en la carpeta htdocs de la carpeta donde tengas instalado el xampp, que es la herramienta que se ha usado para hacer las conexiones con phpmyadmin. Es importante meterla en dicha carpeta, si no es posible que ciertos datos del videojuego no se guarden de manera correcta.

Para poder acceder a la interfaz del médico deberás introducir usuario y contraseña. Estas credenciales, que se encuentran en la base de datos son admin tanto para usuario como para contraseña. Una vez en la interfaz poder añadir clientes o visualizar los existentes, en este punto solo tendrás que ver un DNI creado en esta interfaz que será el que tengas que introducir en el videojuego para jugar con normalidad.

En caso de que estes intentando jugar al videojuego con otro sensor que no sea el original con el que se desarrollo el trabajo, deberás de instalar en este el Arduino que se encuentra subido al repositorio y copiar la dirección específica del sensor en el ejecutable de Python que se encuentra dentro de los scripts del videojuego de Unity.