Universidad de La Laguna

Escuela Superior de Ingeniería y Tecnología

Grado en Ingeniería Electrónica Industrial y Automática

Librería de integración sensorial para creación de interfaces y registros de señales biométricas

Alumna:
Nazaret E. Miranda López

Tutores:
D. Pedro A. Toledo Delgado
Dra. Carina S. González González

Marzo de 2016

D. **Pedro Antonio Toledo Delgado**, con N.I.F. 45725874-B, profesor Ayudante adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna y Dª. **Carina Soledad González González**, con N.I.F. 54064251-Z profesora Titular de Universidad adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna

C E R T I F I C A N

Que la presente memoria titulada:

«Librería de integración sensorial para creación de interfaces y registros de señales biométricas »

ha sido realizada bajo su dirección por Dª. Nazaret E. Miranda López, con N.I.F. 54111657-V.

Y para que así conste, en cumplimiento de la legislación vigente y surta a los efectos oportunos, firman la presente en San Cristóbal de La Laguna a 4 de marzo de 2016.

Abstract

The decrease in the prices and size of biometric sensors has led to their popularization and market consolidation. These devices have been successfully used for the control of health condition and sports performance. It is worth considering, then, the extent to which this sensors can contribute to improve the field of exergames (that is, video games involving physical exertion).

This paper is aimed at the research on the potential of a heart rate sensor to be used as an exergame controller, monitoring the heart rate of the user and thus estimating the level of exertion. The main purpose of this is to consequently adjust the intensity of the physical activities carried out by the user. The estimation of the exertion, together with other factors such as energy expenditure, is based on methods researched in the literature, requesting the user to input the necessary data at the beginning of the game.

For this purpose a basic exergame prototype has been developed using the Unity3D platform. In this videogame, the user interacts mainly through a Kinect v1 controller, while the heart rate sensor incorporates ANT/+ technology.

Resumen

El abaratamiento de costes y la reducción de tamaño de los sensores biométricos ha llevado durante los últimos años a su popularización y consolidación en el mercado. Utilizados por usuarios para controlar su salud y su rendimiento deportivo, cabe preguntarse qué pueden aportar estos sensores al campo de los exergames (videojuegos basados en la actividad física).

El objetivo de este proyecto es estudiar el potencial de un pulsómetro como controlador de un exergame, utilizándolo para registrar el ritmo cardíaco del usuario y así estimar el esfuerzo físico realizado, regulando la intensidad de la actividad posterior en consecuencia. La estimación del esfuerzo físico, junto con la de otros factores como el consumo calórico, se realiza de acuerdo a métodos presentes en la literatura, solicitando al usuario los parámetros necesarios al comienzo del juego.

Para ello se desarrolló un prototipo básico de exergame mediante la plataforma Unity3D, donde la interacción principal se realiza mediante un sensor Kinect v1, mientras que el sensor incorpora tecnología de ultra-bajo consumo ANT/+.

Agradecimientos

Quisiera agradecer a todas las personas que directa o indirectamente, e incluso sin saberlo, me han ayudado a superar este reto.

A mis padres, por ayudarme a tener los pies en la tierra, y a mi hermana por enseñarme grandes valores y ser un modelo a seguir.

A Deriman por estar siempre dispuesto a darme ideas, a Izquierdo por presentarse voluntario para probar el prototipo y a Sergio, por estar siempre ahí para escucharme y darme palabras de ánimo.

También a todas las personas de inmensa calidad humana con las que he tenido la suerte de compartir este largo viaje, y que han hecho que resultara una experiencia gratificante.

Por último, quisiera mostrar el más profundo de los agradecimientos hacia mis tutores, por su paciencia y por ayudarme más allá de lo razonable a lo largo de todas las etapas del proyecto.

Índice general

Lista de Figuras

Capítulo 1

Introducción

1.1 Contexto

1.1.1 Sensores biométricos

En los últimos años se ha producido la popularización del uso de sensores biométricos en el sector deportivo. Estos dispositivos, caracterizados por su reducido tamaño, permiten a los usuarios registrar variables fisiológicas así como otro tipo de datos relacionados con la práctica deportiva.

1.1.1.1 Evolución

Los primeros sensores biométricos aplicados al deporte fueron lanzados al mercado durante la década de los ochenta por la marca Polar, y nacieron con el propósito de monitorear la frecuencia cardíaca en una pista de esquí durante los entrenamientos. El segundo modelo en lanzamiento, bautizado Sport Tester PE3000, ya permitía a los usuarios analizar *a posteriori* los datos recolectados mediante un ordenador personal. Dos años más tarde, IBM desarrolló el primer software que permitía el análisis de la frecuencia cardíaca y, ya en 1991, Polar lanzó su propio software para el análisis de entrenamiento.

A partir de los 2000, el sector de los sensores biométricos comienza a expandirse, desarrollándose nuevos tipos de sensores que permiten medir factores como la velocidad y distancia de carrera. A este respecto cabe destacar la entrada al mercado de la gama Forerunner de la empresa Garmin, que permitirían registrar datos de distancia, velocidad, tiempo y pulsaciones, entre otros, incorporando un sistema GPS.

Actualmente, los sensores biométricos forman parte de un mercado en expansión

que se estima alcanzará un valor de 95.000 millones de dólares para el año 2020, y en el comienzan a involucrarse marcas que no están tradicionalmente ligadas al deporte como pueden ser Apple o Samsung. Hoy en día se encuentran a la venta smartphones con sensores biométricos así como hardware que permite la conectividad con dispositivos de algunas de las tecnologías existentes (ANT o bluetooth smart), y distintas aplicaciones que nos permiten visualizar la información recogida o hacer uso de la misma con distintos propósitos.

También es cada vez más común la incorporación de estos sensores a la vestimenta deportiva (conocidos como *wearable sensors*), así como el desarrollo de sensores biomecánicos, que permiten analizar el movimiento de los usuarios.

1.1.1.2 Funcionamiento

La existencia de los sensores biométricos a los que nos referimos a este proyecto está supeditada a la existencia de un recíproco que permita al usuario visualizar la información que el primero envía, estableciéndose un proceso comunicativo en el que un emisor, a través de un canal de comunicación inalámbrica, transmite los datos a un receptor.

Aunque existen varias, las dos tecnologías de transmisión inalámbrica más populares en la actualidad son ANT+ y Bluetooth Smart, caracterizados por un bajo consumo energético así como por transmitiendo en las bandas ISM, de uso abierto para aplicaciones industriales, científicas y médicas.

Los protocolos de comunicación de estas tecnologías permiten la transmisión de datos de uno varios sensores biométricos a un smartwatch, smartphone o cualquier otro dispositivo capaz de almacenar y/o mostrar dicha información, de lo cual se deduce que los sensores a los que aludimos en este documento pertenecen al nodo emisor de los sistemas de monitoreo y, por ende, los dispositivos a los que va destinada la información constituirán el nodo receptor, como se explica en el capítulo 2.

1.1.1.3 Usos

Aunque como ya se ha mencionado la historia de estos sensores comenzó con la necesidad de medir el ritmo cardíaco, en la actualidad existe una gran variedad de dispositivos en lo que a la medida de parámetros biométricos se refiere. Así, se encuentran sensores capaces de medir la potencia o velocidad de pedaleo, el número de brazadas o de pasos ejecutados durante los entrenamientos de natación y *running*, respectivamente, o incluso presión sanguínea y niveles de glucosa en sangre, entre otros.

La diversidad de variables mensurables junto con unos costes relativamente bajos (ej. podómetro Garmin 010-11092-00 alrededor de 55 €) han convertido a los sensores biométricos en elementos clave para dos principales campos: por un lado, permiten a deportistas tanto profesionales como aficionados llevar un control de las variables que afectan al rendimiento durante la práctica deportiva, optimizando así los entrenamientos, y ayudan a llevar un seguimiento de los progresos; por otro lado, permiten a personas con problemas de salud llevar un registro de sus constantes para así tener constancia de la evolución de la enfermedad, predecir situaciones de peligro y mantener informado al personal sanitario de la situación del paciente.

1.1.2 Obesidad

Multiplicándose más del doble desde los años ochenta, la obesidad ha llegado a convertirse en uno de los mayores problemas de salud a nivel mundial, siendo un factor de riesgo en el desarrollo de múltiples enfermedades como la hipertensión, la diabetes, osteoartritis, así como enfermedades cardiovasculares o incluso algunos tipos de cáncer [1].

Aunque la obesidad es una enfermedad multifactorial en la que pueden intervenir la predisposición genética, los desequilibrios hormonales u otros factores externos, normalmente viene asociada principalmente a un desequilibrio energético motivado dos hechos: por un lado, un consumo excesivo de calorías y, por otro, un estilo de vida sedentario [2].

En España el exceso de peso (sobrepeso u obesidad) se estima afectaba al 43 % de la población infantil entre los 7 y 8 años en el año 2013, según el estudio ALADINO (Estudio de Vigilancia del Crecimiento, Alimentación, Actividad Física, Desarrollo Infantil y Obesidad en España, [3]). Este 43 % es el resultado de un 24,6 % del total de la muestra en la que prevalece el sobrepeso, junto con un 18,4 % de prevalencia de obesidad. Los resultados de este mismo estudio asocian un mayor número de horas frente al televisor con la fracción en situación de obesidad, así como disponer de uno de estos aparatos o DVD en su habitación.

La lucha contra la obesidad supone, por tanto, una revisión de los desbalances nutricionales en la población, así como la incorporación de actividad física al estilo de vida. Si además se tiene en cuenta el hecho de que son los niños sedentarios y/o con exceso de peso aquellos con peor salud mental y emocional [4] y, por otro lado, la asociación que realizan muchos estudios entre la practica de ejercicio físico con el bienestar emocional, así como la mejora de los procesos cognitivos [5], cabe considerar la actividad física como un arma valiosa contra la obesidad.

1.1.3 Exergames

Como alternativa a los modos de ejercicio tradicionales, en los últimos años se ha incrementado la presencia en el mercado de los denominados *fitness games* o, más comúnmente, *exergames*.

Los *exergames* son un tipo de videojuegos orientados a fomentar los niveles de actividad física del usuario. Este objetivo se consigue, por lo general, implementando en la interfaz tecnologías de interacción directa que requieren el movimiento de grandes extensiones del cuerpo, haciendo depender el éxito del jugador de su grado de implicación física.

1.2 Objetivos

Este proyecto viene motivado, en definitiva, por dos aspectos: por un lado, la idea de extraer la información proporcionada por sensores biométricos sin necesidad de utilizar software propietario; por otro, investigar las posibilidades de aplicación de estos sensores en el área de los *exergames*. Los objetivos definidos son los siguientes:

1. Revisar del estado del arte en materia de sensores biométricos de bajo consumo así como analizar brevemente el contexto de mercado.
2. Estudiar el funcionamiento de la tecnología subyacente al sensor deportivo (pulsómetro) utilizado en este proyecto.
3. Investigar las posibilidades de la integración del mismo en la interfaz de un videojuego, estudiando los posibles beneficios que de ello pueden derivar.
4. Construir un prototipo de exergame en el que, basándonos en la investigación previa, la información extraída se utilizará activamente para tratar de controlar la dificultad del juego, regulando así la intensidad de la actividad física realizada por el usuario durante la partida.
5. Realizar una validación del prototipo y valorar las limitaciones y ventajas encontradas o previsibles respecto a los exergames tradicionales.

1.3 Resumen de las tecnologías de soporte

A continuación se enumeran las tecnologías utilizadas a lo largo del proyecto. Una descripción detallada y la justificación de su elección puede encontrarse en los correspondientes

capítulos:

- Software:
  - IDE Microsoft Visual C# 2008 Express Edition
  - IDE Microsoft Visual C++ 2008 Express Edition.
  - Herramientas ANTware II 4.100 y SimulANT+ 1.6.
  - Motor gráfico Unity3D.
  - Kinect for Windows SDK v1.8.
- Hardware:
  - ANTUSB2 Sticks de la marca Geonaute (dos unidades).
  - Pulsómetro Geonaute, modelo SHRM1G, tecnología ANT+.
  - Sensor Kinect v1.

Capítulo 2

Registro y comunicación de señales biométricas y la tecnología ANT/ANT+

2.1 ¿Por qué ANT?

ANT es una tecnología propietaria de red de sensores inalámbrica que integra una pila de protocolos del mismo nombre, y que fue desarrollada en el año 2003 por la compañía Dynastream Innovations Inc.

La pila de protocolos fue creada con el fin de transmitir los datos tomados por un sensor deportivo a un reloj actuando como receptor, minimizando el consumo de energía. De hecho, esto dio lugar a lo que ahora se denomina protocolo inalámbrico de ultra bajo consumo (*ultra-low power wireless protocol*), que es una de las principales ventajas y motivo por el cual se utiliza esta tecnología como base del proyecto. Este bajo consumo se consigue mediante ciclos de trabajo cortos y modos *deep-sleep*.

Asimismo, el protocolo se caracteriza por su gran robustez, la detección de errores (CRC-16) y su capacidad para permitir que coexistan varios dispositivos operando a la misma frecuencia gracias al uso de transmisión TDMA (*Time Division Multiple Access*, acceso múltiple por división de tiempo). Además, pueden existir cientos de dispositivos transmitiendo sin que se produzcan colisiones.

2.1.1 ANT+

ANT+ es una capa de interoperabilidad añadida al protocolo ANT. Los dispositivos ANT que implementan ANT+ son desarrollados de acuerdo a un perfil de dispositivo (*device profile*) que define qué representa la información enviada, y que es el mismo para todos los ANT+ Adopters. Esto significa que dos dispositivos de igual perfil, aunque fueran producidos por distintos fabricantes, podrían interpretar la información el uno del otro [6].

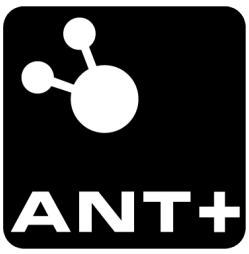


Figura 1: Logo ANT+

2.2 Simulación de la comunicación entre sensores

Establecer con éxito una conexión entre dos nodos ANT requiere configurar correctamente el canal de comunicación. Simular la comunicación entre sensores tiene dos objetivos: el primero es comprender cómo se configuran los canales; el segundo es aislar, partiendo de la correcta configuración, posibles problemas no previstos en la comunicación.

Los nodos presentes en una red ANT establecen una relación maestro/esclavo. Existe la posibilidad de que un mismo nodo pueda tomar cualquiera de los dos roles, de manera que en una red ANT se puede disponer de emisores, receptores y transceptores (*transceivers*). El rol de maestro se suele asignar a un sensor, como por ejemplo un podómetro, mientras que el rol de esclavo es propio de los receptores, como los relojes deportivos.

La conexión de los nodos ANT se realiza a través canales ANT (*ANT Channel*), pudiendo un solo canal comunicar múltiples nodos. Para establecer la comunicación, *los parámetros de configuración del canal deben coincidir en cada uno de los nodos.* Los parámetros a configurar son:

1. **Tipo de canal** (*Channel Type*) – 8 bit. Viene dado por el tipo de comunicación que se desea establecer entre los nodos. Pueden ser:
   - Canal bidireccional maestro/esclavo (*Bidirectional Slave/Master Channel*). El

tipo de nodo (maestro/esclavo) indica hacia dónde se dirigirá principalmente la información.

- Canal bidireccional maestro/esclavo compartido (*Shared Bidirectional Slave/Master Channel*). Este tipo de canal se establece cuando un nodo debe recibir datos desde múltiples nodos.
- Canal de sólo recepción/transmisión (*Transmit/Receive Only Channel*). La comunicación es unidireccional y su uso queda relegado a aplicaciones muy concretas.

2. **Radiofrecuencia de transmisión** (*RF Frequency*) – 8 bit. Puede adquirir valores válidos entre 0 y 124. Este valor representa un incremento en el valor 2400 MHz, de manera que la frecuencia podrá variar entre los 2400 MHz hasta los 2524 MHz (2400 MHz + 124 MHz).

3. **ID del canal** (*Channel ID*) – 4 byte. Consta de tres campos: tipo de transmisión, tipo de dispositivo y número de dispositivo. La comunicación entre un maestro y un esclavo solo tendrá lugar si el maestro posee los parámetros de ID de canal que el esclavo ha especificado.

4. **Periodo de canal** (*Channel Period*) – 16 bit. El periodo de canal es el ritmo al que el maestro envía los paquetes de datos. ANT permite enviar mensajes a frecuencias entre los 0.5 Hz y hasta un máximo de 200 Hz (según las características computacionales de los dispositivos) aunque el valor por defecto es de 4 Hz, optimizando el consumo energético.

5. **Identificación de red** (*Network*). Las redes a través de las cuales se comunican los nodos vienen identificadas por un un campo de 8 byte de valor único que recibe el nombre de *Network Key*, de tal manera que dos nodos solo podrán comunicarse si están conectados a la misma red, i.e. ambos tienen el mismo valor en el campo *Network Key*. Un dispositivo puede almacenar varias *Network Key* -ocho máx.-, y cada una de ellas será asociada a un valor entre 0 y 7, almacenado en el campo de 8 bit denominado *Network Number*. De esta manera, que configuraremos un canal para conectarse a un determinado *Network Number*.

2.2.1 Procedimiento

El principal objetivo es lograr la emisión y recepción de mensajes utilizando dos ANTUSB2 Stick, asignando a uno de ellos el rol de maestro y, al otro, el de esclavo.

ANT pone a disposición de los desarrolladores el *ANT Windows Library Package*, un paquete el que podemos encontrar una serie de librerías que facilitan el desarrollo de

aplicaciones. En este paquete encontramos también el código de las demo, aplicaciones construidas sobre la librería ANT y que ilustran cómo se establece la comunicación.

Además, disponemos de una aplicación precompilada que consta de una interfaz gráfica denominada SimulAnt+, mediante la cual podemos asignar un *device profile* a un stick, y permitir así simular un sensor que emite datos a través de un canal.

Para proceder, un primer stick se configura como maestro directamente asignádole un rol mediante la aplicación SimulANT+. En la figura 2 podemos ver el resultado de asignar a un stick el rol de pulsómetro (*heart rate sensor*).

Un segundo stick se ha de configurar entonces como esclavo (función de «monitor»), haciendo uso de la demo. Para ello, debemos asignar correctamente en el código de ésta aplicación todos los parámetros del canal. Los valores que corresponden a cada perfil de dispositivo, a excepción de la *network key* y el *network number*, pueden encontrarse en los documentos puestos a disposición de los desarrolladores en la web de ANT. Para el caso de dispositivos de monitorización del ritmo cardíaco debe consultarse el documento «ANT+ Device Profile – Heart Rate Monitor». Los valores de estos parámetros se pueden encontrar en la tabla 2.1.

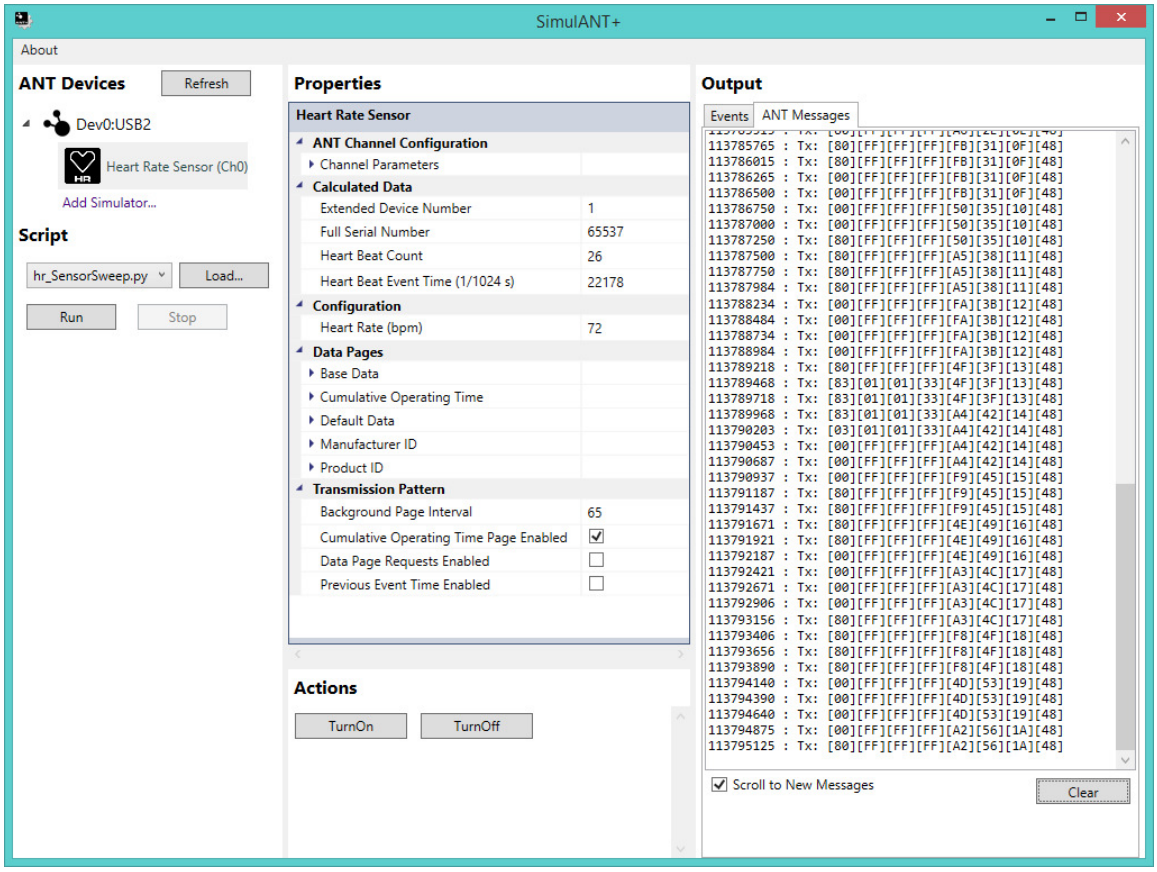


Figura 2: ANTUSB2 Stick emitiendo como pulsómetro en SimulANT+ 1.6

Cuadro 2.1: Valores de configuración de canal para monitor de ritmo cardíaco

Parámetro	Valor	Observación
Tipo de canal	0x00	Esclavo bidireccional (receptor).
Radiofrecuencia	57	Corresponde a 2557 MHz.
ID de canal		
Tipo de transmisión	0	Se asigna cero para pairing.
Tipo de dispositivo	120 (0x78)	
Número de dispositivo	0	Se asigna cero para búsqueda. Podemos utilizar el valor real si es conocido (1–65535)
Periodo de canal	4 Hz	Expresado como 8070 counts.
Network Key	ANT+ Managed Network Key	Para obtener valor numérico correspondiente a esta network key se debe estar registrado como desarrollador en la web de ANT.

Como ya se ha comentado, para que la comunicación sea correcta también deben serlo los parámetros del canal, y además estos deben coincidir entre maestro y esclavo. La modificación de los parámetros en SimulAnt+ resulta trivial a través de su interfaz gráfica. Sin embargo, al utilizar la aplicación compilada debemos tener en cuenta cómo se soporta la modificación de los parámetros: en el caso del periodo de canal, aunque se define una variable al comienzo del código, éste no se setea en el código original, lo que causa errores de transmisión. Para solucionar este problema se debe modificar el código[1] incluyendo la configuración del *channel period* en la secuencia de configuración de canal. En la figura 3a se puede observar el código original frente al fragmento de código correcto, en la figura 3b.

2.3 Transmisión de datos a aplicaciones externas

El manejo de la información recogida por el ANTUSB2 Stick se realiza mediante aplicación DEMO_NET (versión de la demo desarrollada en C#). Para facilitar la transmisión de la información al videojuego se pueden contemplar, en principio, dos opciones: bien construir esta aplicación de recogida de datos dentro del propio juego -incluyendo las llamadas a las DLL de comunicaciones ANT- , obteniendo una única aplicación o bien mantenerla como una aplicación independiente y transmitir la información mediante un socket.

La primera alternativa, aunque simplifica la estructura del proyecto, tiene dos principales inconvenientes: en primer lugar, las librerías deben ser compiladas con las herramientas de Unity, lo cual es susceptible de causar errores de compilación internos. En segundo lugar, se estarían ligando fuertemente las DLL, de manera que si existiera una actualización de las mismas ésta debería ser actualizada específicamente en el proyecto.

[1]Resuelto en: `http://www.thisisant.com/forum/viewthread/4412/`

```
case MESG_CHANNEL_RADIO_FREQ_ID:
{
   if(stMessage.aucData[2] != RESPONSE_NO_ERROR)
   {
      printf("Error configuring Radio Frequency: Code 0%d\n", stMessage.aucData[2]);
      break;
   }
   printf("Radio Frequency set\n");
   printf("Opening channel...\n");
   bBroadcasting = TRUE;
   bStatus = pclMessageObject->OpenChannel(USER_ANTCHANNEL, MESSAGE_TIMEOUT);
   break;
}
```

(a) Código original.

```
    case MESG_CHANNEL_RADIO_FREQ_ID:
{
    if (stMessage.aucData[2] != RESPONSE_NO_ERROR)
    {
        printf("Error configuring Radio Frequency: Code 0%d\n", stMessage.aucData[2]);
        break;
    }
    printf("Radio Frequency set\n");
    printf("Setting Channel Period...\n");
    bStatus = pclMessageObject->SetChannelPeriod(USER_ANTCHANNEL, 8070, MESSAGE_TIMEOUT);
    break;
}

case MESG_CHANNEL_MESG_PERIOD_ID:
{
    if (stMessage.aucData[2] != RESPONSE_NO_ERROR)
    {
        printf("Error configuring Channel Period: Code 0%d\n", stMessage.aucData[2]);
        break;
    }
    printf("Channel Period set\n");
    printf("Opening channel...\n");
    bBroadcasting = TRUE;
    bStatus = pclMessageObject->OpenChannel(USER_ANTCHANNEL, MESSAGE_TIMEOUT);
    break;
}
```

(b) Código con configuración de periodo de canal.

Figura 3: Corrección de error de transmisión.

2.3.1 Solución propuesta: socket TCP

Para evitar esta clase de errores se decidió mantener la aplicación de recogida de datos como una aplicación independiente del juego pero capaz de comunicarse con él (un esquema general de la solución se puede encontrar en la figura 4, y más detalles sobre la conexión en la sección 5.3.2.1). Esta comunicación se realiza mediante el uso de sockets, utilizando un servicio TCP.

Así, la aplicación servidor pone los datos recogidos por la aplicación a disposición de los clientes que requieran el servicio, en este caso el juego y, además, tenemos la ventaja de tener una comunicación bidireccional full-duplex.

Para que esto sea posible, el juego lanza la aplicación de recogida de datos al ejecutarse, de manera que no es necesario cargar ambas aplicaciones manualmente. Una vez lanzada y creado el servidor, el cliente se conectará a el gracias a la especificación de parámetros y comenzará la escucha.

Aunque en este caso el único cliente será el juego que creemos utilizando Unity3D, el servidor también podría proporcionar simultáneamente estos datos a otros procesos, con las ventajas y posibilidades que ello supone.

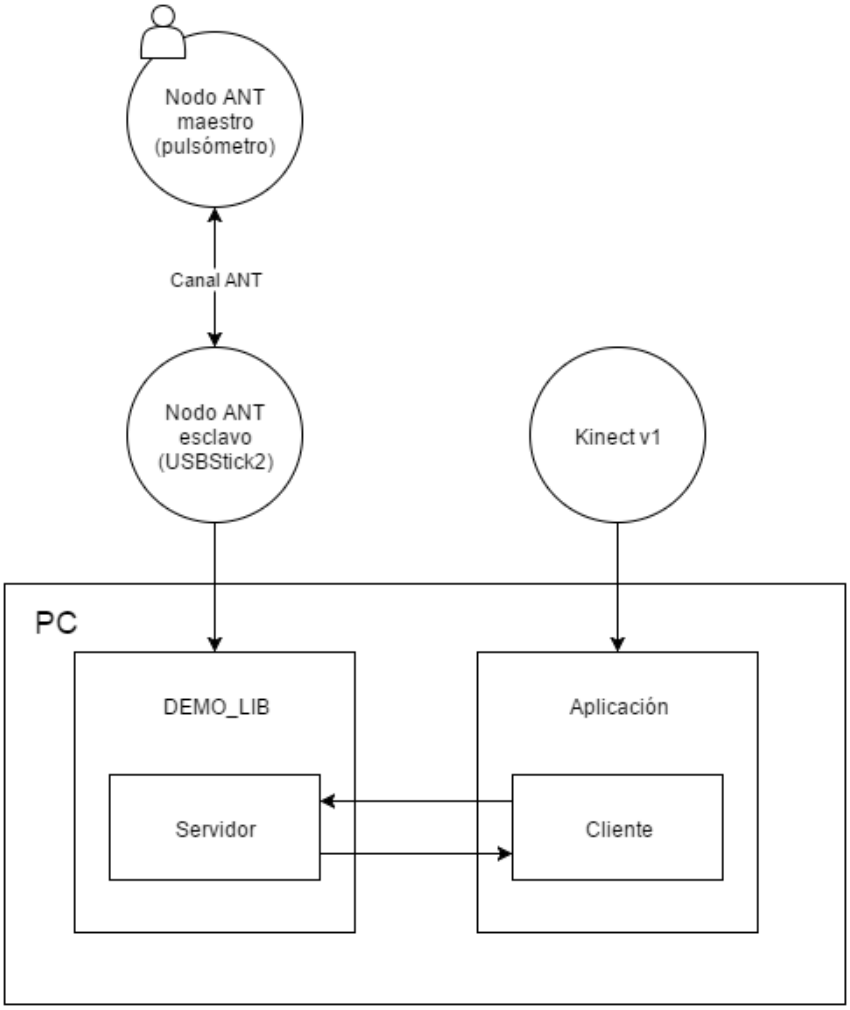


Figura 4: Esquema simplificado de la solución adoptada.

2.3.1.1 *Framing: length prefixing y delimeter*

Para transmitir los datos entre las aplicaciones, se realizará una serialización de los mismos para poder enviarlos a través de streams. Debido a ello, y dado que los tamaños de cada paquete enviado pueden ser potencialmente diferentes, no es posible garantizar la transmisión o recepción de paquetes de información mediante escrituras o lecturas de tamaño fijo en o de los streams de datos. Para proceder a la correcta interpretación de

la información recibida se utiliza el método conocido como *length prefixing*, para lo cual procedemos como sigue:

1. Extremo emisor:
   (a) Generar el dato que queremos enviar y determinar su longitud en bytes.
   (b) Añadir al dato una cabecera de longitud conocida que indica la longitud del mismo. En este caso, la longitud de la cabecera se fija a 4 bytes, por ser el tamaño del tipo `int32`. Asimismo, añadimos un *delimeter* también de longitud fija que indicará el final del mensaje. El propósito de este último es permitir la resincronización de la comunicación en caso de pérdida de datos.
   (c) Enviamos el mensaje a través del socket.
2. Extremo receptor:
   (a) Leemos del stream los 4 bytes correspondientes a la cabecera y convertimos a `int32`, obteniendo la longitud del cuerpo del mensaje.
   (b) Leemos tantos bytes como nos indique la cabecera.
   (c) Leemos los bytes correspondientes al *delimeter* e interpretamos. Si efectivamente corresponden al final del mensaje, interpretamos el contenido del cuerpo; en caso contrario descartaremos el mensaje y continuaremos leyendo del *stream* hasta encontrar el siguiente *delimeter*.

Capítulo 3

Tecnologías de desarrollo del *exergame*

3.1 Elección del entorno de desarrollo

La elección del entorno de desarrollo viene condicionada por el alcance del proyecto. Dado que el objetivo principal era la creación de un videojuego controlado por el ritmo cardíaco, y no necesariamente un exergame, se contempló la utilización del entorno SFML.

SFML es una API multiplataforma y multilenguaje implementada en C++ y estructurada en cinco grandes módulos, que está destinada a "[...] facilitar el desarrollo de videojuegos y aplicaciones multimedia" [7]. Estos cinco módulos son:

1. System. Es el módulo básico y tiene varios cometidos, desde el control de tiempo, hilos y mutex hasta utilidades para el manejo de vectores y cadenas entre otros.

2. Window. Permite definir así como manejar los eventos de la ventana de la aplicación, además de gestionar la información de entrada proveniente de sensores, ratón, teclado, joystick, etc.

3. Graphics. Permite la manipulación de gráficos 2D dentro de la ventana, como sprites, textos, el dibujo de formas básicas o colores.

4. Audio. Dedicado a la reproducción y grabación de sonidos.

5. Network. Facilita la comunicación basada en sockets TCP y UDP, y proporciona clases para el manejo de protocolos de alto nivel (HTTP, FTP).

Esta API orientada a objetos reúne las características básicas necesarias para el desarrollo de un videojuego de estas características, al incluir el módulo "network", crítico para transmitir la información del pulsómetro al juego.

Por otro lado, los medios de interacción estarían limitados a los mecanismos clásicos además del ya mencionado ritmo cardíaco. Aunque, como veremos más adelante, se han llevado a cabo proyectos con éxito (ver 4.1.2.1) combinando estas dos clases de interacción, los mecanismos de interacción tradicionales sí que son un factor condicionante para la creación de un videojuego en el que se requiera actividad física por parte del usuario, ya que limitan la libertad de movimientos.

Dado que es precisamente la actividad física la que permite obtener beneficios para la salud del usuario, añadir ésta al objetivo principal, resultando en la creación de un exergame, supone un aporte positivo. Sin embargo, ello requiere sustituir los métodos de interacción tradicionales. En este caso se dispuso utilizar el propio cuerpo del usuario como controlador a través de Kinect.

SFML requiere la creación de una librería de integración con Kinect por lo que se decidió utilizar un motor de videojuegos propiamente dicho en el que ya existan proyectos llevados a cabo con éxito que utilizan Kinect como parte de la interfaz. El cumplimiento de estos requisitos junto con la existencia de una comunidad relativamente grande y activa hicieron de Unity el entorno elegido.

3.2 Unity3D

Unity es un motor gráfico de videojuegos desarrollado por Unity Technologies. Aunque exite una versión Pro, en este proyecto se ha utilizado la última versión gratuita estable hasta la fecha en la que se realizaron las pruebas: Unity 5.1.0 Personal Edition.

Una de las muchas ventajas este entorno es permitir la publicación de un juego para múltiples plataformas a partir de un único proyecto, sencillamente seleccionando la plataforma de destino mediante la herramienta *Build Settings*. Así, es posible publicar los proyectos de Unity para Windows, Windows Store Apps, Mac, Linux/Steam OS, iOS, Android, Windows Phone 8, BlackBerry 10, Tizen, PS3, PS4, PS Vita, Xbox One, Xbox 360, Wii U, Web Player, WebGL e incluso Oculus Rift, Gear VR y Microsoft Hololens [8].

En esta sección se describen brevemente los aspectos más relevantes de Unity en lo que atañe a esta memoria.

3.2.1 Interfaz del editor de Unity

La interfaz de Unity, como se muestra en la figura 5 se puede dividir en:

1. Vista de escena. Esta pestaña nos permite movernos por la escena y modificar algunas de las propiedades de los objetos que la conforman.
2. Vista de juego. Muestra la escena tal cual la verá el jugador. Si pulsamos el play que se encuentra en la barra de herramientas podremos iniciar el modo juego sin necesidad de usar *build*, lo que permite depuración en cualquier etapa de desarrollo, junto con la pestaña de consola (en la imagen, junto a la pestaña de escena).
3. Pestaña de jerarquía. Se muestra una lista de los objetos de la escena, anidados en caso de que estén compuestos por hijos.
4. Pestaña de proyecto. Aquí podemos ver las carpetas del proyecto con los recursos, como scripts, prefabs o materiales.
5. Inspector de objeto. Aquí podemos ver los *Components* del objeto que hayamos seleccionado en la pestaña de jerarquía.

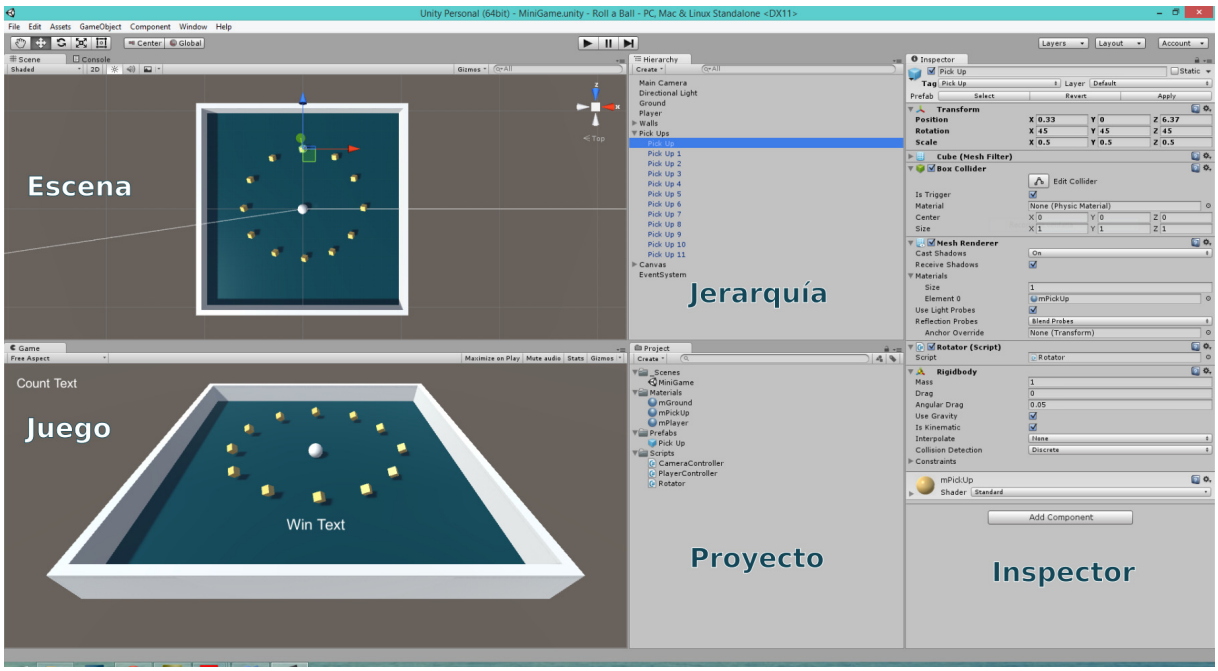


Figura 5: Editor de Unity3D

3.2.2 *Assets* de Unity

Los *assets* son las herramientas esenciales que posibilitan la producción de videojuegos en Unity.

3.2.2.1 *Scenes*

Cada escena de Unity puede ser vista como un escenario de interacción creado a base de GameObjects, y lo usual es que un juego esté conformado por distintas escenas, produciéndose transiciones de una a otra.

3.2.2.2 *GameObjects*: *Components* y *Prefabs*

Los GameObjects, u objetos, son las entidades elementales que componen las escenas y pueden representar desde personajes hasta botones, attrezzo o elementos de iluminación, entre otros.

Aunque pueden estar vacíos (ver 3.2.2.3), el papel de los objetos viene definido por los componentes (Components) que lo conforman y sus propiedades. Scripts, materiales, o elementos que determinan las formas de interacción con otros objetos, como los *colliders*, son solo algunos ejemplos de los muchos tipos de componentes.

UI UI es un conjunto de herramientas (objetos) de Unity que permiten crear interfaces de usuario. Una UI consta de un *Canvas* o lienzo en el cual se insertan los elementos de la UI que pueden ser visuales, como texto, o interactivos de control como los botones o los campos de inserción de datos (InputFields).

Prefabs Los *prefabs* son «moldes» que nos permiten crear tantos objetos iguales como necesitamos. Esto se consigue arrastrando el prefab sobre el panel de jerarquía (*Hierarchy*) o instanciando a través de los scripts.

3.2.2.3 *Scripts*

En Unity, los scripts nos pemiten, entre otras cosas, definir el comportamiento y las características de los objetos (*GameObjects*).

Aunque pueden estar escritos bien en C#, bien en UnityScript (este último basado en Javascript), en este proyecto únicamente se utilizará el primero y se programará a través de la IDE MonoDevelop, ya que es la que utiliza Unity por defecto.

Cada script C# creado con Unity implementa directamente una clase hija de otra clase integrada, MonoBehaviour, y el código no será ejecutado hasta que no asociemos una instancia del script a un GameObject (no existe un constructor como tal) [9]. Esto

implica que, aunque el código que queramos ejecutar no vaya a controlar un objeto en sí, sino otro aspecto del juego (por ejemplo, comunicación mediante sockets), debemos adjuntarlo obligatoriamente a un objeto, aunque este esté vacío.

3.3 Kinect for Windows SDK y el *asset* Kinect with MS-SDK

La versión 1.8 de este SDK ofrece un conjunto de librerías que nos permiten el desarrollo de aplicaciones en Windows que hacen uso de la Kinect v1.

Para facilitar la integración de estas librerías con el proyecto de Unity, se tomó como punto de partida el asset «Kinect with MS-SDK»[1], un proyecto de Unity que incluye ejemplos de funcionalidades del SDK mencionado anteriormente. Los más relevantes para este proyecto son el control de avatar así como la detección de gestos.

[1]Versión 1.12, publicada el 3 de febrero de 2015 por RF Solutions. Disponible en la Asset Store de Unity: `https://www.assetstore.unity3d.com/en/#!/content/7747`

Capítulo 4

Fundamentos de diseño del *exergame*

Dado que el objetivo principal del juego es mantener al usuario en un rango de intensidad de esfuerzo físico moderado utilizando su propio cuerpo como resistencia cabe, en primer lugar, concretar el método de interacción con el mundo virtual y, en segundo lugar, cómo se utilizará la información proporcionada por los sensores biométricos para mantener al usuario a dicha intensidad. Por último, podemos plantearnos la posibilidad de proporcionar al usuario *biofeedback* adicional que resulte útil para que éste perciba los resultados de su esfuerzo.

4.1 Introducción

4.1.1 Exergaming

El concepto de *exergaming* representa la idea de aunar videojuegos con actividad física mediante el uso uno o más de sensores, que participarán activamente en el control de algún aspecto del mismo.

Ejemplos primitivos de este tipo de juegos se pueden remontar hasta el año 1982 con el Puffer Project de la empresa Atari, proyecto que no llegó a materializarse debido al crash de la industria, y en el que el pedaleo de una bicicleta controlaría la velocidad del personaje. Sin embargo, es a partir los años 2005 y 2006, con el lanzamiento del *Eye toy: Kinetic* (Sony Computer Entertainment) y la Nintendo Wii ®, cuando los *exergames* alcanzan una mayor notoriedad.

Numerosos estudios han demostrado la eficacia de estos juegos para fomentar la actividad física y la quema de calorías en niños. Este hecho, junto con la preocupación sobre el sedentarismo y su estrecha relación con los altos índices de obesidad infantil

actuales, actúa como catalizador para el desarrollo de nuevos *exergames* así como de otras formas de interacción persona-máquina y de variables de control de videojuegos, incluyendo el pulso cardíaco.

4.1.2 Efectividad y sensación de reto. El ritmo cardíaco como controlador

Para que un juego de este tipo sea efectivo desde el punto de vista fisiológico, debe exigir un nivel de esfuerzo físico al jugador tal que se produzca un gasto energético significativo. A su vez, los sobreesfuerzos, sobre todo durante tiempos prolongados, deben evitarse activamente desde el desarrollo del juego para impedir no sólo una sobreexigencia que afecte a la motivación del jugador, sino las consecuencias fisiológicas de un esfuerzo cercano o superior al umbral anaeróbico.

De acuerdo con [10], el uso interfaces persona-máquina biocontroladas no convencionales puede reportar beneficios en el rendimiento o la ergonomía, entre otros.

Por otro lado, el ritmo cardíaco ha demostrado ser válido a la hora de estimar la carga fisiológica durante el ejercicio físico y de hecho, de acuerdo a Hoffman *et al.* [11], diferentes estudios demuestran la existencia de una relación lineal entre el pulso y las intensidad del ejercicio submáximo (*i.e.*, por debajo del 85 % de la frecuencia cardíaca máxima).

Todo lo expuesto anteriormente conduce a la idea de que el uso de un pulsómetro puede ayudarnos a mejorar la efectividad fisiológica del juego.

4.1.2.1 Ejemplos

La idea de desarrollar un juego controlado en casi su totalidad por el ritmo cardíaco ha sido llevada a cabo con éxito en distintas investigaciones, e incluso existen en el mercado aplicaciones basadas en este principio. Los mencionados a continuación son algunos ejemplos de ellos:

- **Skip a Beat** [12]. Skip a Beat es una aplicación para smartphones que dispongan de pulsómetro cuyo objetivo es entrenar al usuario para que aprenda a controlar su ritmo cardíaco. Para ello, el juego está dividido en distintos niveles según el rango de pulsaciones. Las pulsaciones controlan el tamaño y el movimiento del personaje (la rana Skip), y manteniéndonos dentro del rango seleccionado logramos una mayor puntuación final.

- **Pulse Masters Biathlon** [10]. Este prototipo de juego, desarrollado por investigadores de la Universidad Tecnológica de Helsinki, consiste en recorrer una pista de esquí en el menor tiempo posible, alternándolo con pruebas de disparo. El ritmo cardíaco se utiliza, por un lado, para controlar la velocidad a la que se recorre la pista, de manera que a mayor ritmo tanto mayor será la velocidad; por otro lado un mayor pulso afectará negativamente a la precisión en los disparos. La principal ventaja de este tipo de interacción es que se puede jugar practicando muy diversos tipos de actividades, como una bicicleta estática o carrera en el sitio. Sin embargo, el juego no está específicamente orientado a mantener las pulsaciones de los jugadores dentro de un rango.

- **Flitz!** [13] Flitz! (desarrollado por P. Pollinger y F. Osterwalder) es un exergame en el que el usuario debe evitar los items de penalización y pulsar los botones correspondientes del controlador (una plataforma compuesta de dos columnas con botones a diferentes alturas) para conseguir los items de recompensa. El juego será más o menos rápido según el pulso del jugador, y es el concepto más cercano al que se trata de implementar en este trabajo.

4.2 Diseño

4.2.1 Interfaz de usuario: métodos de interacción

El uso conjunto de la Kinect y sus librerías permite el *tracking* de las articulaciones del usuario en el espacio, lo que hace posible el control del avatar en el juego mediante dos técnicas fundamentales. La primera de ellas consiste en traducir las posiciones de las articulaciones rastreadas a las posiciones de las articulaciones del avatar en el espacio virtual. La principal ventaja de este método estriba en una reproducción más detallada de las diversas posturas adoptadas por los usuarios, dentro de las limitaciones del propio sensor. Sin embargo ciertas combinaciones de posiciones articulares y secuencias de movimientos (como los saltos) no son reproducidas de manera suficientemente fidedigna, llegando a producirse el solapamiento de extremidades del avatar, además de no ser posible la aplicación efectiva de gravedad al mismo. Todo ello disminuye significativamente la jugabilidad.

El segundo método, el más extendido, es un control indirecto del avatar mediante la detección de gestos. En este caso utilizamos el tracking y comparamos la posición relativa de las articulaciones que nos interesan. Si las articulaciones pasan por dichas posiciones dentro de un intervalo de tiempo determinado, entonces damos el gesto por completado.

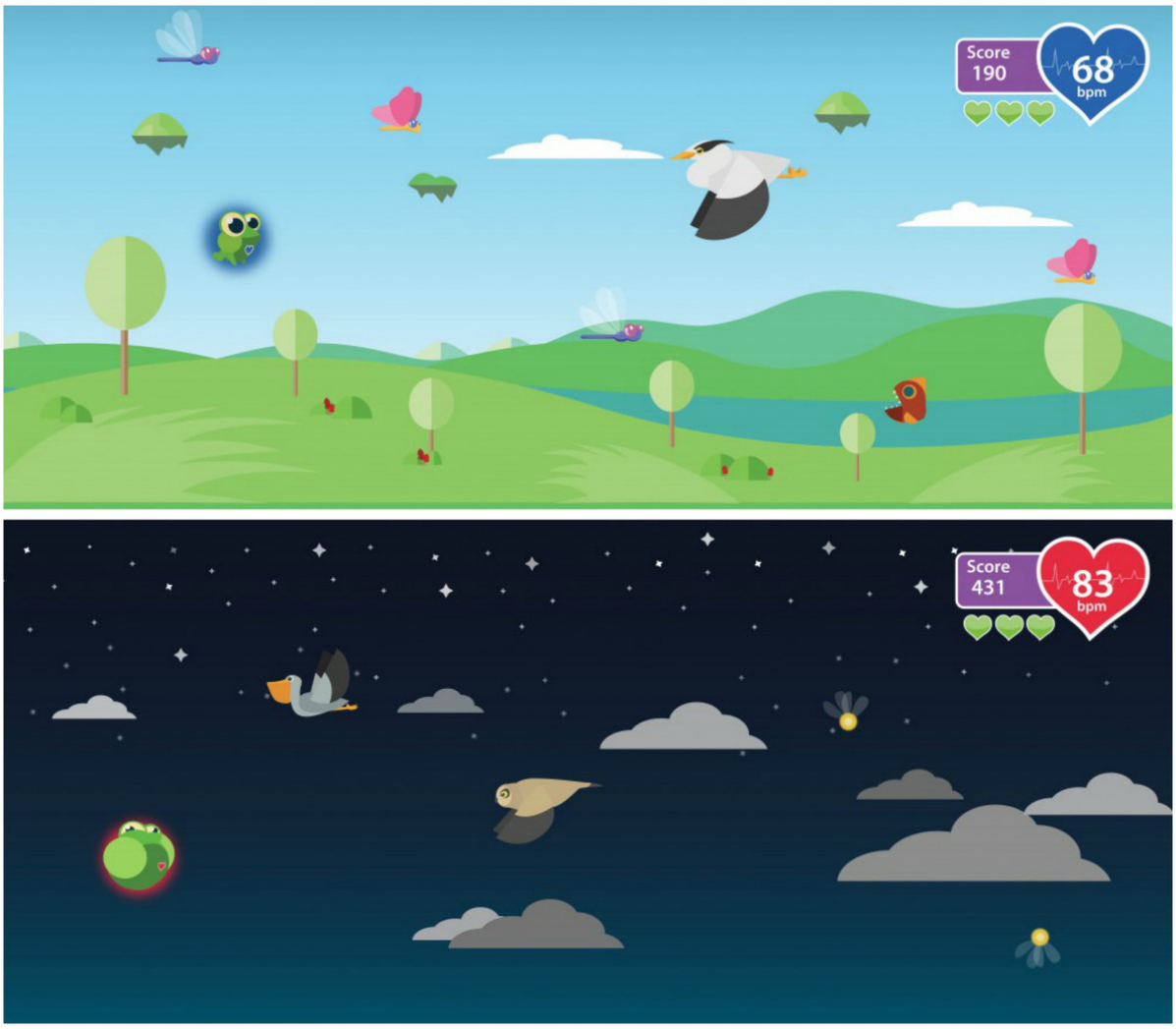


Figura 6: Captura de Skip a Beat. Recuperado de [14]

4.2.1.1 Método adoptado: detección de gestos

A pesar de que el segundo método supone tiempos de reacción superiores, permite un mejor control de los movimientos del avatar, y es por ello por lo que es el método adoptado en este caso. Así, la compleción de un gesto se utiliza para disparar la animación correspondiente.

4.2.2 Frecuencia cardíaca máxima y rangos de intensidad

Como se mencionó en 4.1.2, la frecuencia cardíaca es un buen indicador del esfuerzo físico al que está sometido el cuerpo. La frecuencia cardíaca a partir de la cual se pueden producir daños severos a la salud es conocida como frecuencia cardíaca máxima (FC_{max}), y nos sirve de base para calcular los rangos de intensidad a los que estamos trabajando.

Aunque para un cálculo preciso de la FC_{max} deberíamos someter a cada individuo

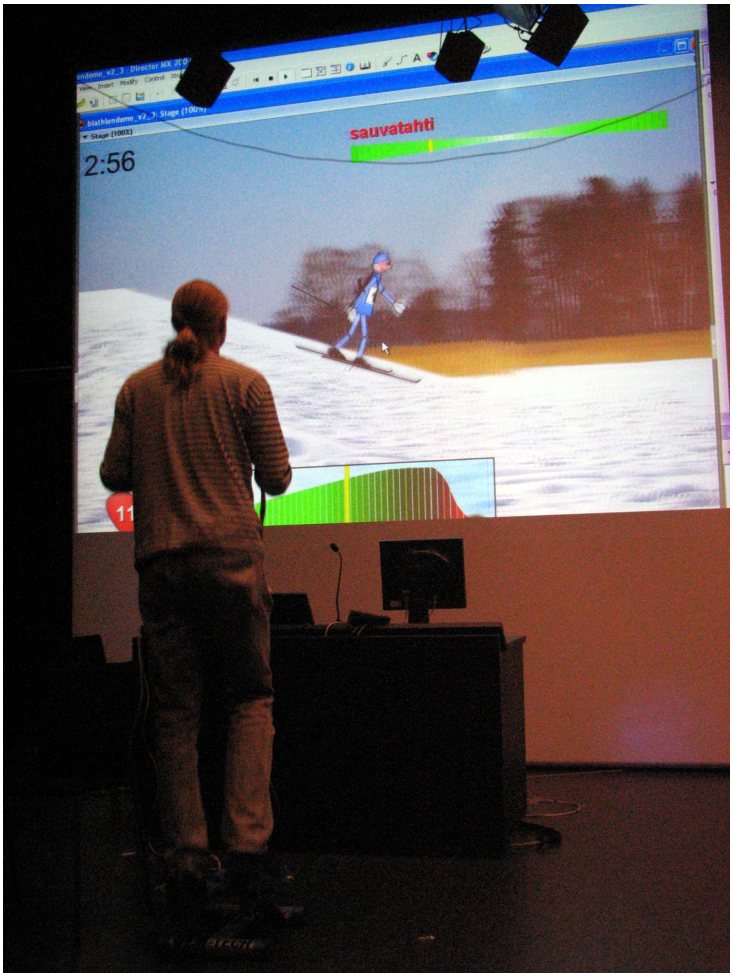


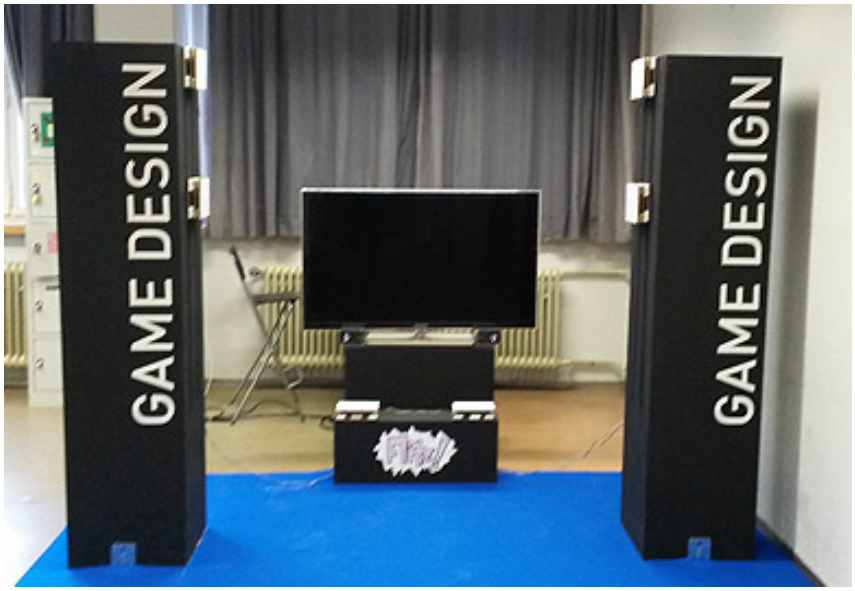
Figura 7: Test de Usuario de Pulse Masters Biathlon [10].

a un test de esfuerzo, existen una serie de fórmulas que nos permiten estimarla con cierta precisión. Frente a la tradicional ecuación $FC_{max} = 220 - edad$, la ecuación $FC_{max} = 208 - (0{,}7 * edad)$ ha demostrado ser más precisa entre niños y adolescentes de 10 a 16 años [15] y jóvenes adultos [16]. Es por ello que es esta última la utilizada en este proyecto.

Según el porcentaje de la frecuencia cardíaca máxima, podemos establecer los rangos de intensidad que se muestran en la tabla 4.1.

4.2.3 Estimación del consumo energético

Al igual que sucede con la frecuencia cardíaca máxima, un cálculo preciso del consumo energético durante la actividad física sólo puede conseguirse realizando test de calorimetría a cada individuo. Sin embargo, podemos hacer una aproximación mediante calorimetría indirecta utilizando el pulso cardíaco, ya que existe una relación lineal entre el pulso y el gasto energético en el rango de 90–150ppm, aproximadamente [18].



(a) Plataforma de control.

(b) Captura del juego.

Figura 8: Flitz! Recuperado de [13]

Cuadro 4.1: Rangos de intensidad según porcentaje de FC_{max} [17]

Zona	Rango	Características
1	50 %–60 % FC_{max}	Baja intensidad. Indicada para calentamiento y recuperación tras el ejercicio.
2	60 %–70 % FC_{max}	Intensidad moderada. Zona indicada para el aumento del metabolismo del tejido adiposo.
3	70 %–80 % FC_{max}	Zona de entrenamiento vascular y de la resistencia aeróbica.
4	80 %–90 % FC_{max}	Límite aeróbico.
5	90 %–100 % FC_{max}	Aquí comienza la zona de entrenamiento anaeróbico (déficit de oxígeno).

El estudio citado, llevado a cabo por Keytel *et al.*, estimó una ecuación de predicción de gasto energético (*energy expenditure*, EE) a partir del pulso, la edad, el sexo y la masa corporal, y cuya expresión es:

$$EE = sexo * (-55{,}0969 + 0{,}6309 * FC + 0{,}1988 * masa + 0{,}2017 * edad) + (1 - sexo) * (-20{,}402 + 0{,}4472 * FC - 0{,}1263 * peso + 0{,}074 * edad) kJ/min$$

donde $sexo = 1$ en el caso de los varones, y $sexo = 0$ para las mujeres.

Capítulo 5

Diseño, implementación y validación del juego

5.1 Descripción funcional

Para que el juego se pueda ejecutar correctamente es un requisito imprescindible, en primer lugar, definir un perfil de usuario en el que se especifiquen el sexo, el peso y la edad del jugador. En segundo lugar, se debe lograr establecer una conexión con un *heart rate sensor* (pulsómetro) a través de una interfaz USB – ANT+.

El principal objetivo del juego consiste en mantener al usuario realizando un esfuerzo físico (actividad aeróbica) dentro de un rango de pulsaciones saludables. Para que esto suceda el usuario debe realizar determinados ejercicios que implican el movimiento de grandes grupos musculares, principalmente del tren inferior, definidos a continuación:

- Salto. La articulación central de la cadera se eleva por encima de 12 cm desde su posición anterior en un intervalo de 1.5 segundos.
- Sentadilla. La articulación central de la cadera desciende por debajo de 15 cm desde su posición anterior en un intervalo de 1.5 segundos.
- Saltos laterales. Se parte de una posición con los pies juntos (distancia entre las articulaciones de los pies menor a 30 cm) y se acaba en esta misma posición pero habiéndose desplazado 25 cm a la derecha o izquierda, según corresponda.
- Puñetazo. Las manos izquierda o derecha avanzan en un intervalo de 1.5 segundos.

La interfaz del juego muestra una pequeña área de la escena en la que está centrado el avatar, y no se producen más desplazamientos de este exceptuando los pasos laterales

que se deben ejecutar para esquivar ciertos objetos. Superpuesto se encuentra un head-up display (HUD) a través del cual se proporciona al jugador un feedback sobre ciertos aspectos del juego, léase:

- HR (ritmo cardíaco). Situado en la esquina superior derecha, indica las pulsaciones del usuario sobre un corazón, que irá cambiando de color de azul a rojo en función del rango de pulsaciones en el que se encuentre.
- *Calories* (calorías): indica el consumo energético acumulado desde el inicio del juego.
- *Score* (puntuación): fracción que indica el número de objetos sorteados o destruidos satisfactoriamente respecto al total de objetos que han aparecido.
- *Time Left* (tiempo restante): Tiempo para finalizar el juego.

Para incitar al usuario a realizar los ejercicios, se presentan ante el avatar una serie de ítems que deben ser esquivados o destruidos. Cuando el usuario ejecuta una de las acciones, el avatar la emula y, en caso de que se trate de una de las acciones adecuadas al objeto, éste será esquivado o destruido satisfactoriamente. A lo largo del juego se irán contando todos los objetos que han aparecido así como el número de ellos que ha sido sorteado/eliminado gracias a las acciones ejecutadas por el usuario; esta relación (presentada en forma de fracción y de porcentaje) es lo que aquí tomamos como puntuación. Los objetos y su acción o acciones correspondientes son:

- Cilindros («troncos» o «barriles»): esquivar mediante saltos.
- «Bloques»: destruir mediante golpe frontal (puñetazo).
- «Shuriken»: esquivar mediante sentadillas.
- Esferas: destruir mediante saltos laterales. Se presentan en series y siempre aparecen frente al usuario.

A medida que las pulsaciones del usuario van acercándose al límite superior, el usuario deberá esquivar menos objetos. Al contrario ocurre cuando las pulsaciones del usuario comienzan a ser demasiado bajas.

Esta situación se mantendrá hasta finalizar el juego, es decir, una vez que transcurra el período de tiempo programado (cinco minutos). Tras esto, el usuario tendrá la opción de reintentar o abandonar el juego.

5.2 Justificación

La mecánica del juego sirve al propósito de mantener al jugador activo, y todos sus aspectos se construyen directa o indirectamente entorno a esta premisa, abogando por la sencillez sin olvidar que el objetivo final es demostrar la utilidad de la incorporación de sensores biométricos.

El menú de inicio, además de preparar al usuario para el comienzo del juego, tiene la función de garantizar que la intensidad del ejercicio es adecuada y que las calorías son estimadas correctamente, pues como se ha hecho patente en el capítulo 4: por una parte, conocer la edad nos permite calcular la frecuencia cardíaca máxima y con ella los intervalos de intensidad; por la otra, además de la edad, necesitamos conocer el sexo y el peso del usuario para realizar una estimación del gasto energético. Es por ello también que existe cierto rango de valores admisibles para cada campo.

Durante el juego, al avatar emula los movimientos del jugador a modo de espejo, por considerarse éste el método más natural y ya que el avatar se encuentra de frente al jugador para permitir una mejor visualización de los objetos.

Además, los elementos del HUD están dispuestos de tal manera que el usuario pueda acceder fácilmente a la información, pero perturbando lo menos posible la escena.

Calcular los intervalos de intensidad sirve de base para el control del resto del juego. Como se puede deducir de la sección anterior, aumentar o disminuir la frecuencia con la que aparecen los objetos no tiene otro fin más que intensificar o reducir el nivel de esfuerzo que realiza el usuario, respectivamente, para así mantener su ritmo cardíaco dentro del rango ideal.

Así, el hecho de que la puntuación del juego se represente como una proporción es una consecuencia directa de que, por un lado, el juego se desarrolle durante un periodo de tiempo previamente asignado y, por otro, que el tiempo entre la aparición de dos obstáculos distintos dependa del pulso del jugador, ya que ello significa que durante el tiempo programado el número de obstáculos totales puede variar.

A su vez, fijando un periodo de tiempo de juego, sin contemplar la finalización del mismo como penalización, se responde al indicado objetivo de mantener al jugador en movimiento manteniendo la simplicidad.

5.3 Implementación

5.3.1 Menú de inicio

En el menú de inicio disponemos de una UI que permite al usuario introducir los datos para crear su perfil. De acuerdo a lo estipulado en las secciones anteriores, constará de los siguientes elementos:

- `InputField` para la edad. Solo admite enteros.
- `InputField` para el peso. Solo admite enteros.
- `ToggleGroup` para el sexo. Excluyentes.
- Botón de Start.

Añadido a un objeto con el mismo nombre, encontramos el script MenuActions.cs, que determina el comportamiento de este menú de inicio. Al evento de fin de edición de los campos anteriormente mencionados se ha añadido un listener (función `UnityEvent.AddListener`) que permite comprobar que los valores de peso y edad introducidos se encuentran dentro de rangos aceptables. En caso negativo los campos erróneos se destacan en rojo, y lo mismo ocurre si no se selecciona un valor para el sexo. Además, se bloqueará el avance hacia la siguiente escena, que en un caso válido se realiza mediante el botón de Start. Es también mediante este botón que se lanza la demo de decodifica los datos del sensor.

5.3.2 Escena de juego

5.3.2.1 Conexión con el servidor y estado de comunicación con el sensor: el canvas de conexión

Tal y como se expuso en 2.3.1, necesitamos implementar un socket cliente desde nuestro juego que reciba los datos emitidos por el servidor, y que resulta de la decodificación de las señales transmitidas por el sensor . Esta implementación se realiza por medio de dos scripts: «TCPConnection» y «socketScript».

En `TCPConnection.cs` disponemos de las funciones necesarias para crear un socket TCP de tipo stream e iniciar una nueva instancia de las clases StreamWriter y StreamReader mediante los cuales nos comunicamos con el servidor. También es en este script donde

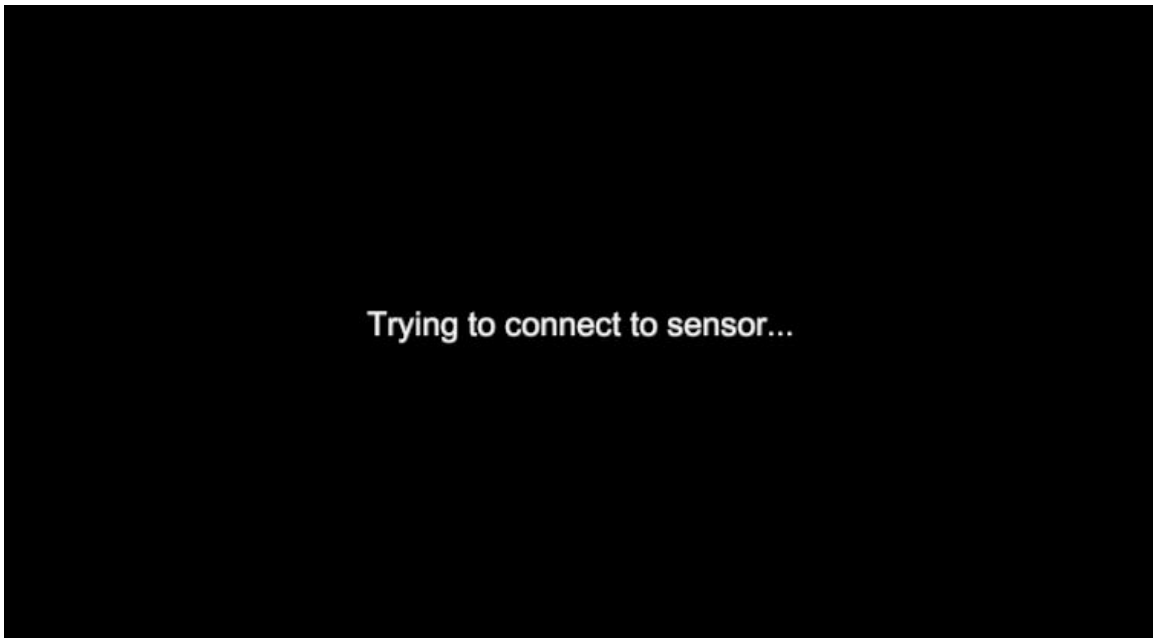
Figura 9: Menú de inicio

definimos los métodos mediante los cuales haremos uso de estas instancias, así como el cierre del socket y la reconexión en caso de interrupción inesperada.

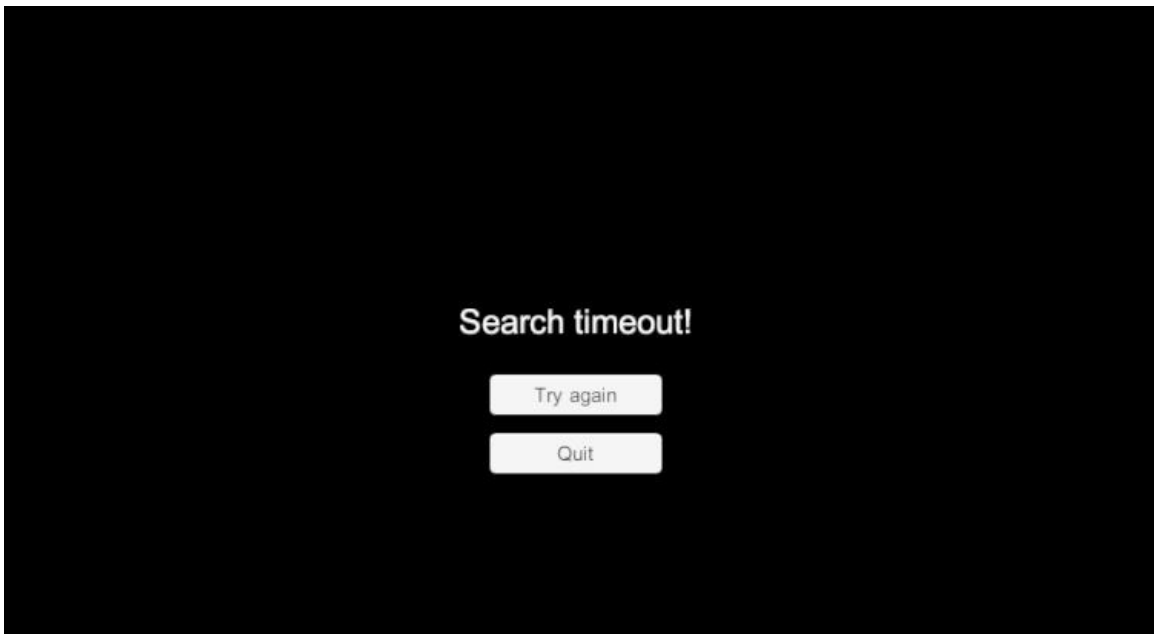
Desde `socketScript.cs` creamos un socket TCP instanciando la clase `TCPConnection`. Una vez hecho esto intentamos conectarnos automáticamente con el servidor. El método SocketResponse es el que convierte el stream de bytes recibido en tipo entero y asociándolo a una variable. Esta variable se utiliza más adelante para mostrar la información en el Head-up display.

Una vez que se ha lanzado la demo, ésta inicia la búsqueda de un sensor cuyos parámetros coincidan con los que hemos programado (tabla 2.1). Mientras se lleva a cabo esta búsqueda, se mantiene activado un canvas que indica al usuario que estamos tratando de conectar con el sensor. Aquí se pueden dar dos situaciones: la primera, que la demo se conecte satisfactoriamente al sensor; la segunda, que no sea capaz de encontrar un dispositivo válido dentro del tiempo de búsqueda. En ambos casos, la demo comunica al juego la situación enviando un mensaje a través del servidor.

Si se da el primer caso, se desactivará el canvas de conexión, se activará el head-up display y dará comienzo el juego. En caso contrario, se comunicará al usuario que no se ha podido establecer la conexión, y se le dará la opción de abandonar el juego o reintentar la conexión.



(a) Información intento de conexión



(b) Estado del canvas tras fallo en conexión

Figura 10: Canvas de conexión.

5.3.2.2 Head-up display (HUD)

El head-up display es un Canvas construido sobre la escena mediante el cual proporcionamos al jugador diferentes tipos de información. En este caso la información mostrada se controla a través del script `HUDController.cs`, que es un componente del objeto de igual nombre.

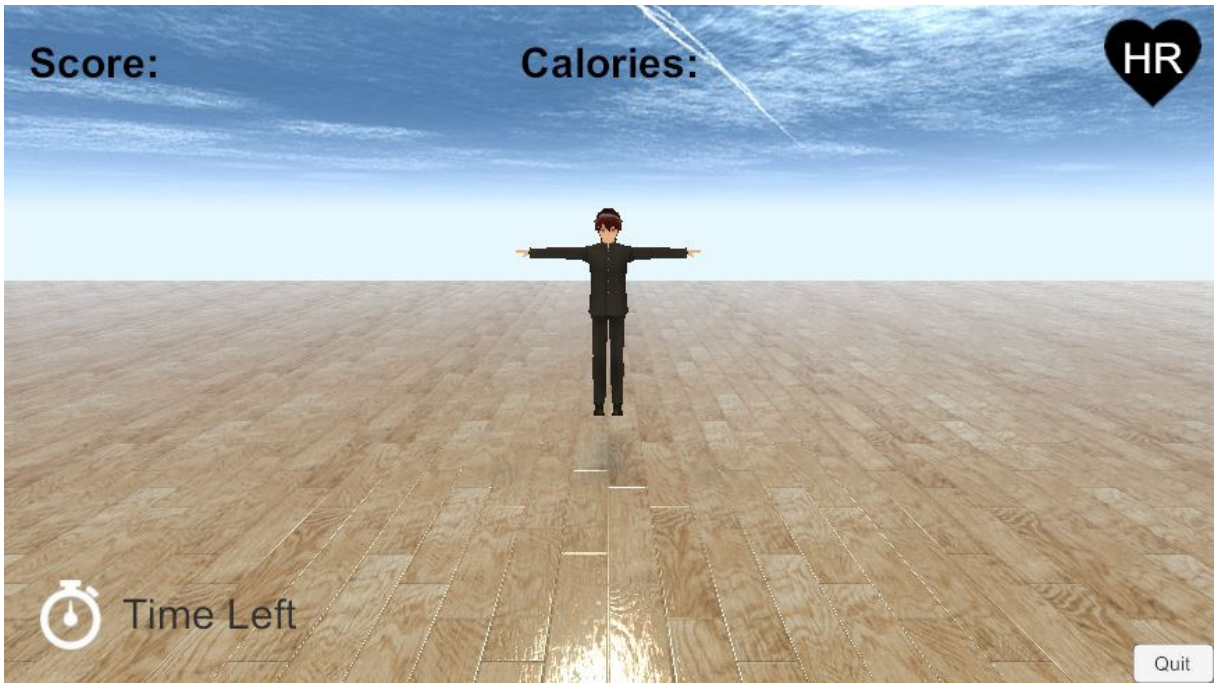


Figura 11: Head-up display

Frecuencia cardíaca

En la esquina superior derecha se muestra el ritmo cardíaco actual del jugador. La figura toma colores desde el azul hasta el rojo según el rango de intensidad en el que se encuentre, como se ha mencionado anteriormente, y de acuerdo con la tabla 4.1.

Para calcular estos rangos se ha calculado en primer lugar la FC_{max} a partir de los datos insertados en el menú de inicio, como ya se ha explicado.

Consumo energético

Para el cálculo del consumo energético utilizamos el método denominado `EnergyExpenditure`, que calcula el gasto calórico mediante la ecuación vista al final de la sección 4.2.3, pasando a unidades de kJ/ns, y un `Stopwatch`, de la siguiente manera:

1. Detenemos el stopwatch.
2. Obtenemos los ticks de reloj transcurridos.
3. Reiniciamos el stopwatch.
4. Calculamos el gasto calórico por unidad de tiempo (kJ/ns).
5. Multiplicamos el tiempo transcurrido por el gasto calórico por unidad de tiempo convirtiendo a kCal, y lo sumamos al total de calorías consumidas.

6. Pasamos a string y actualizamos el valor en el HUD.

5.3.2.3 Control de los movimientos del avatar

Detección de gestos

Dentro del paquete *Kinect with MS-SDK* encontramos el script KinectManager.cs, que nos permite especificar los gestos que se esperan de los jugadores (en nuestro caso un único jugador) así como el tiempo mínimo que debe transcurrir entre cada gesto.

Por otro lado, el proceso de detección de cada uno de los gestos está definido en el script `KinectGestures.cs`. Los gestos reconocibles por defecto que se utilizan en el juego, que ya se han descrito en la sección 5.1, son los siguientes:

- Push (puñetazo).
- Jump (salto).
- Squat (sentadilla).

Además de los mencionados anteriormente, la librería permite la implementación de nuevos gestos. Para ello se trabaja con una máquina de estados, en la que cada estado se reconoce mediante la información de las posiciones de unas articulaciones respecto a otras. Cuando una posición es alcanzada, se informa del evento a la librería para que pase al estado siguiente del reconocimiento.

Haciendo uso de esta alternativa, se han implementado dos nuevos gestos: SideJumpRight (salto lateral hacia la derecha) y SideJumpLeft (salto lateral hacia la izquierda), que detectan saltos o pasos laterales rápidos. Para añadir estos gestos, primero los añadimos el nombre del gesto al enum `KinectGestures.cs`, y pasamos a implementar el gesto dentro de la función `CheckForGesture()`. En la figura 12 podemos ver el fragmento de código resultante. En estos fragmentos de código está basada la descripción de los saltos laterales de la sección 5.1.

A este componente debemos añadirle un *Gesture Listener* previamente implementado. En este caso partimos de un script ya creado (`SimpleGestureListener.cs`) que modificamos para habilitar los gestos deseados.

Animación del avatar

```
1242        case Gestures.SideJumpLeft:
1243            switch(gestureData.state)
1244            {
1245            case 0:  // gesture detection - phase 1
1246                if(jointsTracked[leftKneeIndex] && jointsTracked[rightKneeIndex] &&
1247                   (Mathf.Abs(jointsPos[leftKneeIndex].x - jointsPos[rightKneeIndex].x) < 0.3f ))
1248                {
1249                    SetGestureJoint(ref gestureData, timestamp, leftKneeIndex, jointsPos[leftKneeIndex]);
1250                    gestureData.progress = 0.5f;
1251                }
1252                break;
1253
1254            case 1:
1255                if((timestamp - gestureData.timestamp) < 1.5f)
1256                {
1257                    bool isInPose = jointsTracked[leftKneeIndex] && jointsTracked[rightKneeIndex] &&
1258                        (gestureData.jointPos.x - jointsPos[leftKneeIndex].x  > 0.25f);
1259                    if(isInPose)
1260                    {
1261                        Vector3 jointPos = jointsPos[gestureData.joint];
1262                        CheckPoseComplete(ref gestureData, timestamp, jointPos, isInPose, 0f);
1263                    }
1264                }
1265                else
1266                {
1267                    // cancel the gesture
1268                    SetGestureCancelled(ref gestureData);
1269                }
1270                break;
1271            }
```

(a) Gesto SideJumpLeft.

```
1210        case Gestures.SideJumpRight:
1211            switch(gestureData.state)
1212            {
1213            case 0:  // gesture detection - phase 1
1214                if(jointsTracked[leftKneeIndex] && jointsTracked[rightKneeIndex] &&
1215                   (Mathf.Abs(jointsPos[leftKneeIndex].x - jointsPos[rightKneeIndex].x) < 0.3f ))
1216                {
1217                    SetGestureJoint(ref gestureData, timestamp, leftKneeIndex, jointsPos[leftKneeIndex]);
1218                    gestureData.progress = 0.5f;
1219                }
1220                break;
1221
1222            case 1:
1223                if((timestamp - gestureData.timestamp) < 1.5f)
1224                {
1225                    bool isInPose = jointsTracked[leftKneeIndex] && jointsTracked[rightKneeIndex] &&
1226                        (jointsPos[leftKneeIndex].x - gestureData.jointPos.x > 0.25f);
1227                    if(isInPose)
1228                    {
1229                        Vector3 jointPos = jointsPos[gestureData.joint];
1230                        CheckPoseComplete(ref gestureData, timestamp, jointPos, isInPose, 0f);
1231                    }
1232                }
1233                else
1234                {
1235                    // cancel the gesture
1236                    SetGestureCancelled(ref gestureData);
1237                }
1238                break;
1239            }
1240            break;
1241
```

(b) Gesto SideJumpRight

Figura 12: Implementación de dos nuevos gestos.

Para traducir el movimiento del usuario al avatar, se ha referenciado al componente `Animator` de este último un asset del tipo `Animator Controller`, que maneja las transiciones y las condiciones de disparo de las animaciones haciendo uso de una máquina de estados de mecanim. El resultado final lo podemos ver en la figura 13. Los *triggers* que permiten dichas transiciones se setean cuando el jugador ha completado el gesto correspondiente y, como se observa en la figura, se reproducirá la animación por defecto (idle_20) mientras el jugador está en reposo a efectos del juego.

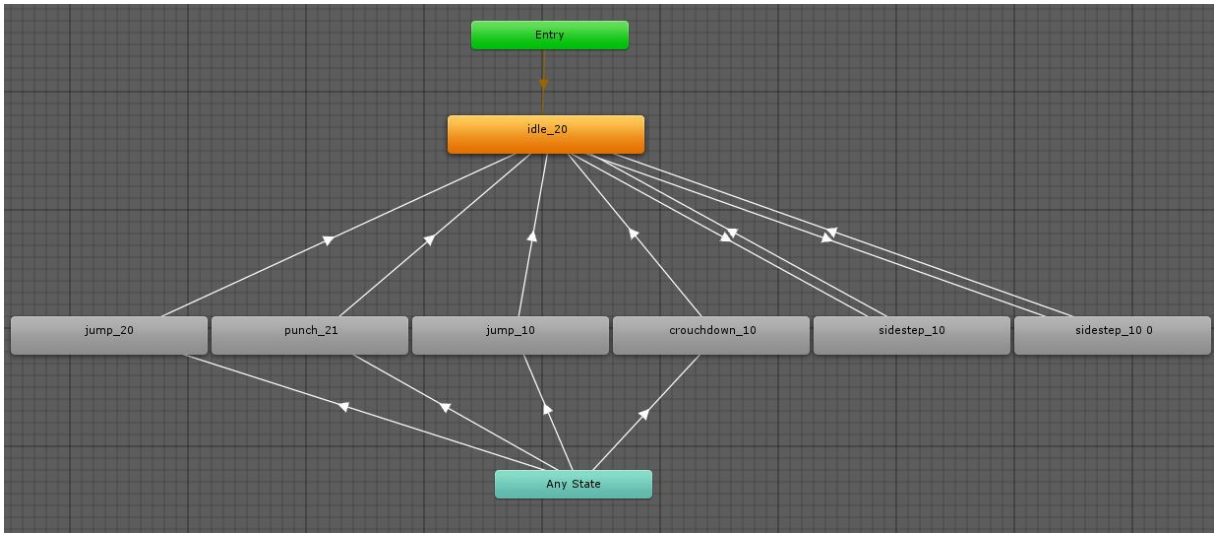


Figura 13: Máquina de estados de mecanim (animación del avatar).

5.3.2.4 Controlador del ritmo cardíaco

El control del ritmo cardíaco se realiza manipulando el tiempo de espera entre la instanciación de cada objeto enemigo (spawnTime), de manera que este tiempo irá disminuyendo siempre que el pulso del usuario esté por debajo del deseado (que, recordemos, es el 70 % de su frecuencia cardíaca máxima) e irá aumentando si nos encontramos sobre ésta.

En particular, este tiempo aumentará más o menos rápido acorde a cuánto diste la frecuencia cardíaca actual de la deseada, para lo cual se ha creado una serie de rangos como los que aparecen en la tabla 4.1. En esencia, cuanto más se aleja el valor real del deseado, mayor será el incremento o el decremento del tiempo entre instancias, según la diferencia sea por exceso o por defecto, respectivamente. Por ejemplo, al acercarnos a la zona de baja indensidad, decrementaremos hasta un máximo de 4 segundos el intervalo.

5.3.2.5 Finalización del juego

Al finalizar el tiempo se dejan de generar objetos y se activa un canvas de fin de juego, en el que se muestra al jugador la proporción de objetos esquivados satisfactoriamente además de la estimación de calorías consumidas.

Mediante este canvas se ofrece también al jugador la posibilidad de comenzar un nuevo juego (botón `Retry`) o de abandonar la aplicación (botón `Quit`). Seleccionando la primera opción cerraremos la demo y volveremos al menú de inicio; seleccionando la segunda, además de cerrar la demo saldremos de la aplicación. En la figura 14 se muestra este canvas.

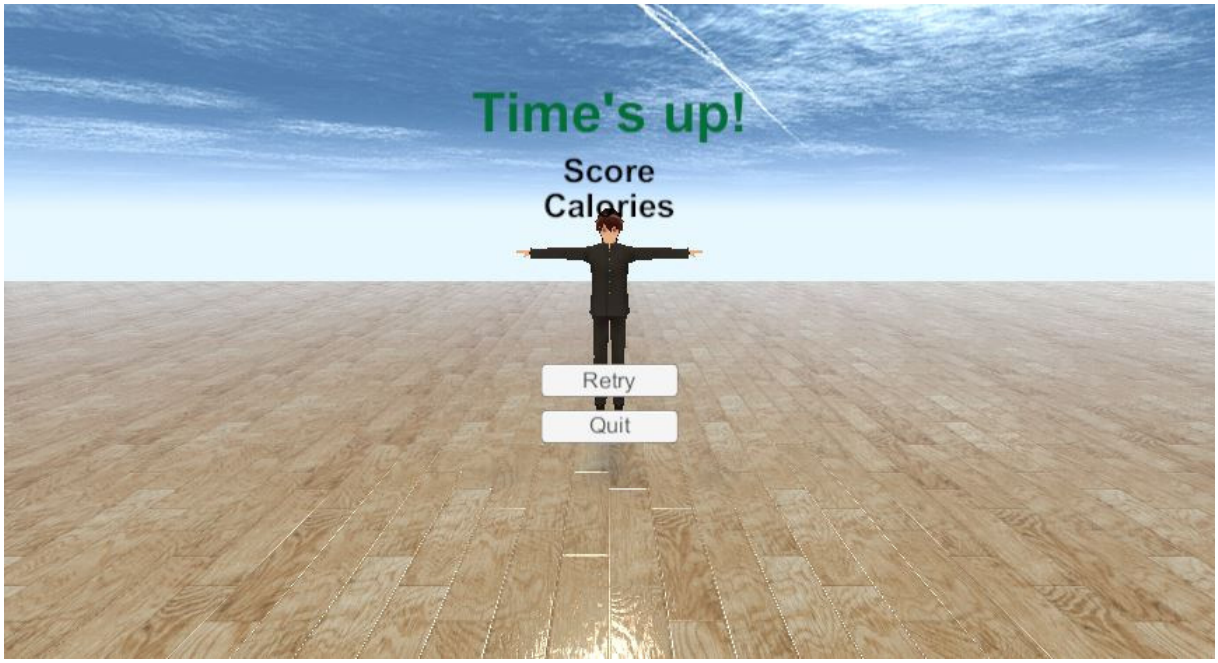


Figura 14: Canvas de fin de juego sobre la escena.

5.4 Resultados

Con el fin de comprobar la efectividad del juego, se registró el ritmo cardíaco de tres sujetos. Los datos analizados se corresponden a una única partida de cinco minutos de duración llevada a cabo en un entorno y condiciones naturales para los sujetos, con el único requisito de no haber realizado ningún tipo de ejercicio físico en las horas previas. Aunque evidentemente es una muestra demasiado pequeña para obtener resultados concluyentes, puede servir de punto de partida para detectar posibles fallas graves en el control del pulso. A continuación se muestran dichos resultados.

5.4.1 Sujeto 1

5.4.1.1 Perfil

- Sexo: Mujer
- Edad: 23
- Peso (kg): 56

5.4.1.2 Resultado

En la gráfica de la figura 15 podemos observar la evolución del ritmo cardíaco registrado (en pulsaciones por minuto, ppm) frente al tiempo transcurrido. Para un sujeto de este perfil, la frecuencia que se desea mantener es de unas 134 ppm, de acuerdo a la ecuación vista en 4.2.2 y sabiendo que deseamos mantenernos idealmente al 70 % de la FC_{max}. Como se observa en la gráfica, el sujeto parte de 93 ppm y alcanza las 120 ppm aproximadamente 14 segundos después del comienzo del juego.

Desde entonces y hasta que finaliza el tiempo, el ritmo se mantiene entre un mínimo de 120 y un máximo de 141 ppm, con un promedio de 129 ppm desde el comienzo del juego. No se excede el límite superior de la zona aeróbica ni se baja de la zona de intensidad moderada.

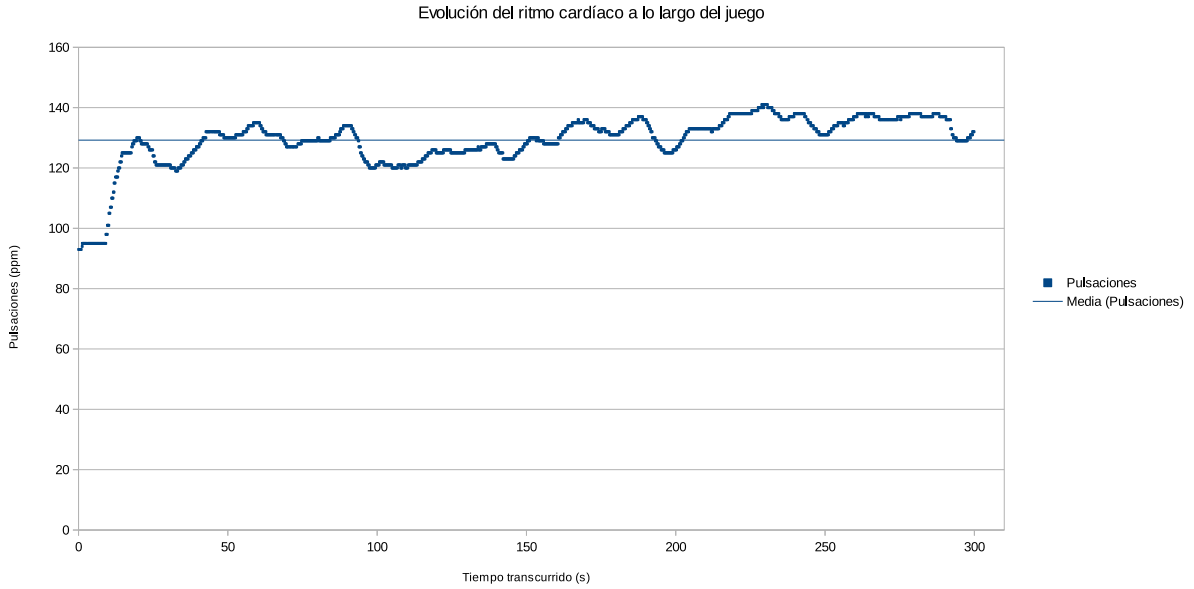


Figura 15: Pulso frente al tiempo de juego transcurrido, sujeto 1.

5.4.2 Sujeto 2

5.4.2.1 Perfil

- Sexo: Hombre
- Edad: 22
- Peso (kg): 115 kg

5.4.2.2 Resultado

La evolución del ritmo cardíaco del sujeto número dos la podemos observar en la gráfica de la figura 16. En este caso el inicio de la zona aeróbica se encuentra en las 135 ppm. El ritmo cardíaco inicial son 99 ppm, y se observa un incremento lento aunque relativamente constante en el mismo, alcanzándose por primera vez las 120 ppm a los 88 segundos.

Las pulsaciones se mantienen entre las 98 ppm del comienzo un un máximo de 138. El promedio es de 121 ppm, algo inferior al deseado, aunque en este caso tampoco se excede el límite superior de la zona aeróbica ni se baja de la zona de intensidad moderada.

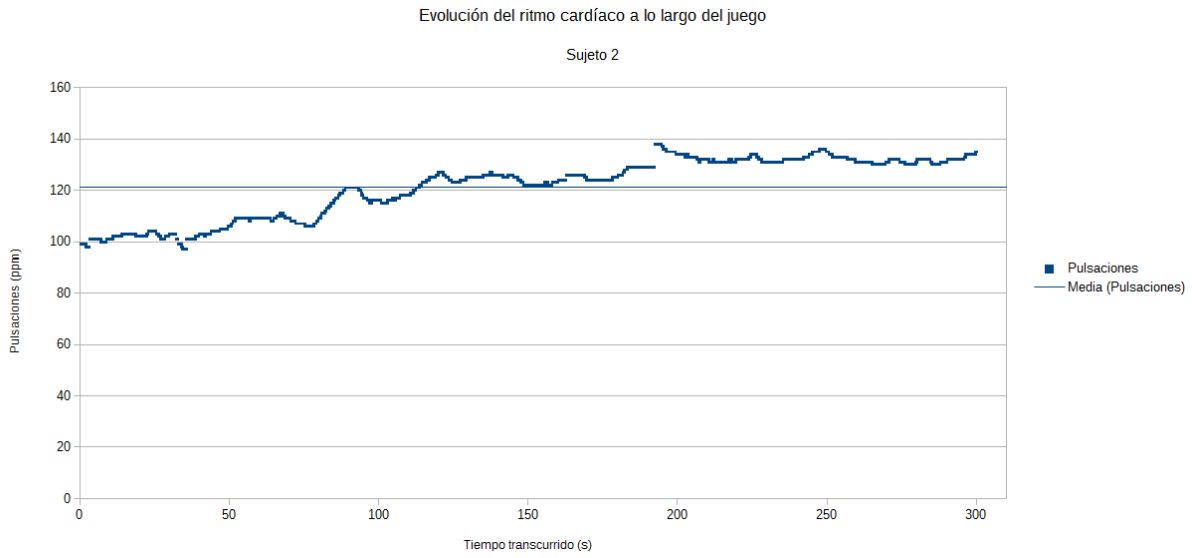


Figura 16: Pulso frente al tiempo de juego transcurrido, sujeto 2.

5.4.3 Sujeto 3

5.4.3.1 Perfil

- Sexo: Hombre
- Edad: 22
- Peso (kg): 85 kg

5.4.3.2 Resultado

Los datos obtenidos para este tercer sujeto se presentan en gráfica de la figura 17. Como en el caso anterior, el inicio de la zona aeróbica se encuentra en las 135 ppm. Partiendo de

93 ppm, alcanzamos las 120 ppm a los 63 segundos. El promedio también son 120 ppm, manteniendo la constante de los casos anteriores de no exceder ni rebajar los límites de la zona aeróbica ni de intensidad moderada, respectivamente, con pulsaciones entre los 92 y 132.

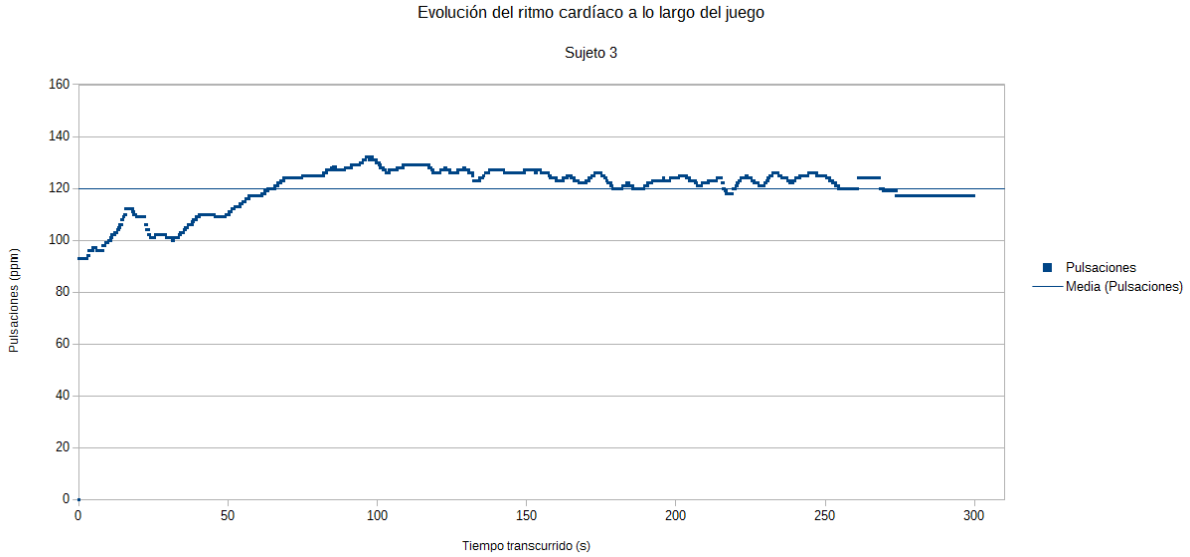


Figura 17: Pulso frente al tiempo de juego transcurrido, sujeto 3.

Capítulo 6

Conclusiones y trabajos futuros

Integrar la información recopilada por un sensor biométrico y que además ésta sea utilizada de forma no pasiva es un hábito poco practicado en los videojuegos. Sin embargo, resulta razonable pensar que, si esta tecnología es utilizada con éxito en el control del rendimiento deportivo, también pueda resultarlo en el control del rendimiento durante sesiones de juego con exergames. Comprobar la veracidad de esta premisa fue uno de los objetivos programados al comienzo de este proyecto, y es la intención de este capítulo verificar el cumplimiento de los mismos.

En primer lugar se llevó a cabo una revisión del estado del arte en materia de sensores biométricos de bajo consumo y un breve análisis del contexto de mercado, cuyos resultados fueron expuestos en el capítulo de introducción. Además, se realizó un estudio del funcionamiento de la tecnología ANT+, del que se deriva gran parte de información que podemos encontrar en el capítulo 2. Esto supone lograr dos de los cinco objetivos principales.

El hecho de que exista un rango de pulsaciones en el que la relación con el esfuerzo físico es lineal es la principal razón por la que tiene sentido utilizar un pulsómetro en el videojuego. Estudiar la relación entre el ritmo cardíaco y la actividad física y antecedentes de otros videojuegos, entre otros, cumplir satisfactoriamente con el objetivo de investigar las posibilidades de integración del sensor en un videojuego. Esta información hace posible la construcción del prototipo tal y como se describe en el capítulo cuarto.

Aunque se requiere un mayor número de muestras para determinar si es posible mantener a la mayoría de los usuarios dentro del rango de pulsaciones deseado, el análisis llevado a cabo parece apuntar en esta dirección. Esto, junto con las valoraciones en las limitaciones analizadas a continuación, satisface el objetivo de validar el prototipo y valorar sus ventajas a inconvenientes.

6.1 Líneas futuras

A pesar de cumplirse los objetivos propuestos al inicio, como ya se ha comprobado, y de que el prototipo sirva al propósito para el que fue creado, no deja de tratarse de un modelo básico muy limitado en aspectos de jugabilidad. El desarrollo de este proyecto permite predecir algunas de las limitaciones que pueden aparecer en éste y otros respectos relacionados con el desarrollo y mejora de videojuegos de estas características en general y de éste en particular.

Para este prototipo, los dos primeros aspectos susceptibles de mejora están relacionados con la detección y la ejecución a nivel virtual de sus animaciones: inestabilidad en la detección de gestos y excesivo tiempo de respuesta a los mismos.

Por un lado, un pequeño porcentaje de gestos no es detectado o no es interpretado correctamente. Esto supone un problema particularmente en los gestos que implican desplazamientos laterales, dado que su falsa detección o la ausencia de la misma puede llevar al usuario a una posición fuera del rango de rastreo del sensor de movimiento. La causa de este problema es desconocida, pero es probable que esté relacionada con la propia definición de los gestos en el código.

Por otro lado, se puede percibir cierto retraso o «lag»en la respuesta al gesto, es decir, el tiempo que transcurre entre que el usuario ejecuta la acción y la animación correspondiente comienza a ejecutarse. Esto podría mejorarse con ciertas labores de optimización o incluso disminuyendo el tiempo de detección de gesto implementando algún tipo de predicción de los mismos, aunque siempre existirá cierta latencia [19].

Para diseñar un juego como tal partiendo del prototipo, de manera que éste resulte atractivo a los usuarios, debemos modificar las dinámicas y mecánicas del juego aportando mayor nivel de dinamismo, teniendo en cuenta el aspecto más limitante para conseguir esto: el hecho de que estamos condicionados a mantener cierto ritmo cardíaco.

Dejando de lado el prototipo y situándonos en un punto de vista más genérico, abarcando el diseño de exergames basados en el principio de control de ritmo cardíaco, se pueden prever otras limitaciones en las etapas de diseño de videojuegos de mayor complejidad. Si por ejemplo quisiéramos implementar un modo multijugador y partiéramos de la técnica utilizada en este proyecto, es decir, controlar la aparición de objetos de acuerdo al nivel de esfuerzo estimado, un modo de juego competitivo puede ser percibido como, y en efecto resultar, injusto. En este hipotético caso parece más adecuado desarrollar el juego bajo principios colaborativos.

Por otra parte, las características de ultra-bajo consumo del protocolo ANT, la

posibilidad de coexistencia de un gran número de dispositivos en funcionamiento en un espacio reducido, así como la integración de esta tecnología a los smartphones supone una base sólida para el desarrollo de exergames de realidad aumentada, lo cual abre otro abanico de posibilidades en el que se pueden incluir gran variedad de sensores.

Capítulo 6

Conclusions

Incorporating the data transmitted by a biometric sensor in addition to it being used in a non-passive way is an unusual practice in the design of video games. Nevertheless, it is reasonable to think that, if this technology is successfully used for the measure of sports performance, it could also be useful for the performance control during exergaming sessions. Proving this assumption correct was one of the five main objectives set out at the begining of this paper. This chapter aims to demonstrate the fullfilment of said objectives.

Firstly, we reviewed the state of the art in matters of ultra-low power biometric sensors and analyzed in the current market context. This information can be found in the introductory chapter. Later on in chapter 2, we delved into the concepts of the ANT technology built into the heart rate sensor selected for this project. This implies the achievment of two of the five goals.

According to the literature, there exists a heart beat interval for which the physical exertion is linear. This is the main reason why using a heart rate sensor in an exergame makes sense. The study of literature related to this topic as well as preceding projects on heart rate monitored exergames, among others, attains the aim to research on the integration of the sensor in a video game. This information makes the imlpementation of the prototype -as described in chapter 4- possible, therefore fulfilling the goal of creating a game prototype.

Although much intensive and extensive research is required to draw solid conclusions on whether the heart rate of the majority of users can be kept within the desired interval, the tests carried out seems to point to the right direction. This, together with the following analysis, achieves the goal of validating the prototype and evaluate its advantages and limitations.

6.1 Future guidelines

Despite the accomplishment, as proven, of the main goals, and the model achieving he purpose it was created for, it is still a basic prototype very limited regarding playability. The development of this project allows the prediction of some of the restraints regarding this and other topics concerning the improvement and development of this kind of exergames in general and this prototype in particular.

As far as this prototype is concerned, the first to aspects susceptible to improvement are related to the detection and the execution of the animations: unsteady gesture detection and excesive response time.

On one hand, a small percentage of the gestures is not detected or interpreted correctly. This poses a problem particularly for those gestures that imply side shuffles, as the incorrect detection or lack of it may cause the user to move outside the tracking area of the movement sensor. The cause to this problem remains unknown, but it is likely to be related to how the gestures are defined codewise.

On the other hand, certain lag in the execution of gestures -that is, between the user executes the gesture and the animation triggers – can be perceived. This problem may be alleviated through optimization or reduction of the gesture detection time by implementing some kind of gesture prediction, although certain latency will always remain [19].

In order to design a proper game based on the prototype so that it results attractive to users, the game mechanics and dynamics must be modified, so as to increase the levels of dynamism. This requires overcoming one of the most limiting aspects – the condition of maintaining a certain heart rate.

Setting aside the prototype and from a general point of view, covering the design of heart rate controlled exergames, some limitations on the design of complex exergames can be foreseen. For instance, if we wanted to implement a multiplayer mode based on the methods used in this project, that is, controlling the instantiation of items according to the exertion levels, competitive dynamics can be perceived, and actually turn out to be, unfair. In that case the implementation of collaborative dynamics appears to be more adequate.

Finally, there are three factors that set a solid base for the development of augmented reality exergames: the ultra-low power characteristics of the ANT protocol, the possibility for a wide number of devices to coexist in a reduced area, and the implementation of this technology in smartphones. This kind of exergames create allow for the use

of many different sensors profiles, setting a wide range of new possibilities.

Bibliografía

[1] Obesidad y exceso de peso, 2006.

[2] Obesidad y sobrepeso (nota descriptiva nº311), 2015. Disponible en Internet: `http://www.who.int/mediacentre/factsheets/fs311/es/`.

[3] Seguridad Alimentaria y Nutrición Agencia Española de Consumo. Estudio aladino 2013: Estudio de vigilancia del crecimiento, alimentación, actividad física, desarrollo infantil y obesidad en españa 2013. Technical report, Ministerio de Sanidad, Servicios Sociales e Igualdad, Madrid, 2014.

[4] Arturo Rodríguez-Hernández, Ernesto De la Cruz-Sánchez, Sebastián Feu, and Raúl Martínez-Santos. Sedentarismo, obesidad y salud mental en la población española de 4 a 15 años de edad. *Revista Española de Salud Pública*, 85(4):373–382, 2011.

[5] Daniel Riquelme, Carlos Sepúlveda, Manuel Muñoz, and Mauricio Valenzuela. Ejercicio físico y su influencia en los procesos cognitivos. *Motricidad y Persona*, (13):70–74, 2013. Universidad Central de Chile.

[6] Thisisant.com. *ANT/ANT+ Defined*. Disponible en Internet: `http://www.thisisant.com/developer/ant-plus/ant-antplus-defined/`.

[7] Sfml-dev.org. *Simple And Fast Multimedia Library*. Disponible en Internet: `http://www.sfml-dev.org/faq.php#grl-whatis`.

[8] Unity3d.com. *Unity - Multiplatform*. Disponible en Internet: `http://unity3d.com/unity/multiplatform`.

[9] Unity3d.com. *Unity Documentation - Manual: Creating and using scripts*. Disponible en Internet: `http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html`.

[10] Ville Nenonen, Aleksi Lindblad, Ville Häkkinen, Toni Laitinen, Mikko Jouhtio, and Perttu Hämäläinen. Using heart rate to control an interactive game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pages 853–856, New York, NY, USA, 2007. ACM.

[11] Katrin Hoffmann, Josef Wiemeyer, Sandro Hardy, and Stefan Göbel. Personalized adaptive control of training load in exergames from a sport-scientific perspective. In Stefan Göbel and Josef Wiemeyer, editors, *Games for Training, Education, Health and Sports*, volume 8395 of *Lecture Notes in Computer Science*, pages 129–140. Springer International Publishing, 2014.

[12] Happitech. Skip a beat game. Disponible en Internet: `http://skipabeatgame.com/`.

[13] Fabienne Osterwalder and Patricia Pollinger. Flitz! adaptative exergame. *Echo Grafik*. Disponible en Internet: `http://www.echo-grafik.ch/flitz/index.html`.

[14] Luke Smith. *Arcticesmpire.ca*.

[15] Fabiana Andrade Machado and Benedito Sergio Denadai. Validity of maximum heart rate prediction equations for children and adolescents. *Arquivos Brasileiros de Cardiologia*, 97:136 – 140, 08 2011.

[16] Stephen James Roy. Validation of maximal heart rate regression equations. Master's thesis, University of Pittsburg, 2009.

[17] flc.losrios.edu. *Understanding heart rates and cardio training zones*. Disponible en Internet: `http://wserver.flc.losrios.edu/~willson/fitns304/handouts/heartRates.html`.

[18] L. R. Keytel, J. H. Goedecke, T. D. Noakes, H. Hiiloskorpi, R. Laukkanen, L. van der Merwe, and E. V. Lambert. Prediction of energy expenditure from heart rate monitoring during submaximal exercise. *Journal of Sports Sciences*, 23(3):289 – 297, 2005.

[19] Richard Leadbetter. What went wrong with kinect? *Eurogamer.net*. Disponible en Internet: `http://www.eurogamer.net/articles/digitalfoundry-what-went-wrong-with-kinect`.

[20] Jeff Sinclair, Phillip Hingston, and Martin Masek. Considerations for the design of exergames. In *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, pages 289–295. ACM, New York, 2007.

[21] Sung Yeun (Su) Kim, Nathan Prestopnik, and Frank A. Biocca. Body in the interactive game: How interface embodiment affects physical activity and health behavior change. *Computers in Human Behavior*, 36(0):376–384, 2014.

[22] ANT Alliance. *ANT Basics (1)*, 2010. Material grabado durante el ANT+ Alliance Symposium 2010. Disponible en Internet: `https://youtu.be/uw-_UGjER1I?list=PLA4AA10B39DB19291`.

[23] ANT Wireless. *ANT Message Protocol and Usage*. Dynastream Innovations Inc., 2014.

[24] Thisisant.com. Network keys and the ant+ managed network. *ANT Tech Bulletin*, 12 de julio de 2013. Disponible en Internet: `http://www.thisisant.com/developer/resources/tech-bulletin/network-keys-and-the-ant-managed-network`.

[25] Wikipedia.org. *ANT (network)*. Disponible en Internet: `http://en.wikipedia.org/wiki/ANT_(network)#Technical_information`.