Escuela
Politécnica
Superior

Big Bad Monster

Videojuego para dispositivos Android Wear

Máster Universitario en Desarrollo de Software para Dispositivos Móviles

Trabajo Fin de Máster

Autor:
Diana de VIcente Clausell
Tutor/es:
Miguel Ángel Lozano Ortega

Junio 2015

Universitat d'Alacant
Universidad de Alicante

Índice general

Índice de Figuras

1 Introducción

1.1 Motivación y Objetivos

Desde hace un tiempo el mercado se ha visto inundado por unos nuevos dispositivos móviles llamados *wearables*. Estos dispositivos vienen para complementar al teléfono móvil apoyando y ampliando sus funciones. Pero, ¿Qué aplicaciones necesitan explícitamente el dispositivo *wearable* para poder funcionar, ampliando realmente la aplicación nativa del teléfono? Esta fue la pregunta clave que dio origen a este proyecto.

En el mercado de las aplicaciones móviles hay muy pocas que estén preparadas o ampliadas para ser utilizadas en un dispositivo *wearable*, pero además, casi no hay aplicaciones nacidas específicamente para aprovechar el potencial que nos brindan estos dispositivos.

Este proyecto nace de la ilusión por conocer esta nueva generación de *wearables* que están empezando a inundar el mercado y aprovechar su potencial combinando sus funciones con las del teléfono inteligente que hoy en día casi todos llevamos en el bolsillo.

Para ello nos basaremos en el reloj inteligente de LG, el GWatch R, y las API's de Android que Google ha desarrollado para poder trabajar con estos dispositivos.

Por tanto, los objetivos de este proyecto son:

- Sacar el máximo partido a la dualidad *wearable-smartphone* creando una aplicación donde se complementen estos dos dispositivos.
- Aprender a programar aplicaciones para *SmartWatches* basados en Android.
- Aprender los métodos que utilizan los *SmartWatches* para comunicarse con los *SmartPhones*.
- Tener una idea sobre el impacto que pueden tener los dispositivos *Wearables* en nuestra sociedad.

1.2 Explicación del Proyecto

Es un pequeño juego donde van apareciendo unos monstruos de colores. Según el color del monstruo, el usuario deberá realizar un gesto con la muñeca donde tiene puesto el reloj. Si el gesto es el apropiado para atrapar al monstruo, este desaparecerá dando lugar al siguiente. El objetivo es coger a todos los monstruos del nivel en el mínimo tiempo posible.

El juego se basa en el acelerómetro del reloj y la comunicación teléfono-reloj, complementando ambos dispositivos para crear una experiencia de usuario todavía no explotada en el mundo Android.

1.3 Motivación

La principal motivación del proyecto es aprender a programar para dispositivos *wearables*, en este caso el reloj de Android, ya que este es un mercado nuevo con muchas posibilidades aún sin explotar y mucho que aprender.

Además, dado que el ámbito del proyecto es nuevo, casi no hay aplicaciones para *SmartWatches* y las que hay, son simplemente ampliaciones de las ya disponibles, donde los desarrolladores se han limitado a hacer compatible su aplicación original con el nuevo reloj.

Por todo esto, me encantó la idea de tener la oportunidad de entrar en este nuevo mercado, aprendiendo nuevas formas de programación desarrollando una aplicación que tuviera en cuenta desde el primer momento que no iba a ser solo una aplicación para el reloj, o una aplicación para el móvil. Este juego iba a ser una aplicación híbrida que aprovechara las dos plataformas a la vez, creando una nueva experiencia de usuario mucho más rica.

2 Contextualización

En el contexto del mercado móvil actual existen muchos dispositivos para poder elegir el que mejor se ajuste a las exigencias de cada usuario. Pero si nos paramos a mirar más detenidamente, el sistema operativo móvil líder es Android, con una cuota del 76,6% del mercado mundial[1] A día de hoy, muchas marcas que apuestan por Android como su principal o único sistema operativo móvil ya han lanzado al mercado su versión del *SmartWatch* basado en él, de manera que los usuarios tenemos una amplia gama de opciones a la hora de elegir el dispositivo, pero todos se basan en lo mismo: la plataforma Android.

A esto le tenemos que sumar que actualmente su principal competidora, *Apple.Inc* todavía está empezando a lanzar su *SmartWatch* en los principales países del mundo. Esto significa que su reloj inteligente aún no ha llegado a todos los países donde está planeado que llegue, incluida España, por lo que los usuarios españoles solo tenemos dos opciones: Android, o esperar[2].

2.1 Tecnologías disponibles

En el ámbito de los teléfonos móviles todavía no existen aplicaciones que se basen en gestos hechos desde un dispositivo externo, como puede ser un mando a distancia o en este caso, un *SmartWatch*. Pero sí podemos buscar referencias en otras plataformas como la *Wii* de Nintendo.

Este dispositivo se basa en los sensores del mando para determinar la posición en la que se encuentra el jugador o el gesto que ha realizado. Inicialmente se utilizaba un acelerómetro para medir el movimiento y dos luces led que captaba la cámara dispuesta en la "barra sensora" (receptor que se ubica encima de la televisión y hacia donde hay que apuntar el mando). De esta manera se controlaba el movimiento lineal que se estaba haciendo (acelerómetro) y la distancia y tamaño aproximado de la pantalla (luces led).

---

[1] Datos obtenidos de http://androidayuda.com/2015/02/24/no-hay-quien-pueda-con-android-se-mantiene-por-encima-del-75-de-cuota-nivel-mundial/

[2] Está previsto que *Apple* lance su *SmartWatch* en España el 26 de Junio. Datos obtenidos de http://www.applesfera.com/apple-tv/ya-es-oficial-el-apple-watch-llega-el-dia-26-a-espana

Más tarde, en la siguiente versión introdujeron un giroscopio para optar a más movimientos y ganar precisión. En este caso la "barra sensora" también serviría para calibrar el giroscopio con el típico truco de "apunta a la pantalla y pulsa el botón A"[3].

Pues bien, nuestro objetivo es trasladar la experiencia de usuario de la *Wii* al entrono Android, utilizando un *SmartWatch* como mando y el móvil como receptor y gestor del juego.

2.2 Aplicaciones similares

A día de hoy existen muy pocas aplicaciones para el reloj de Android, la mayoría de ellas son adaptaciones de la versión original reducidas al tamaño de la pantalla del reloj.

Si además entramos en el terreno de los juegos, ninguno de ellos le saca partido a la combinación teléfono-reloj, por lo tanto podríamos decir que este tipo de juego es único en esta plataforma.

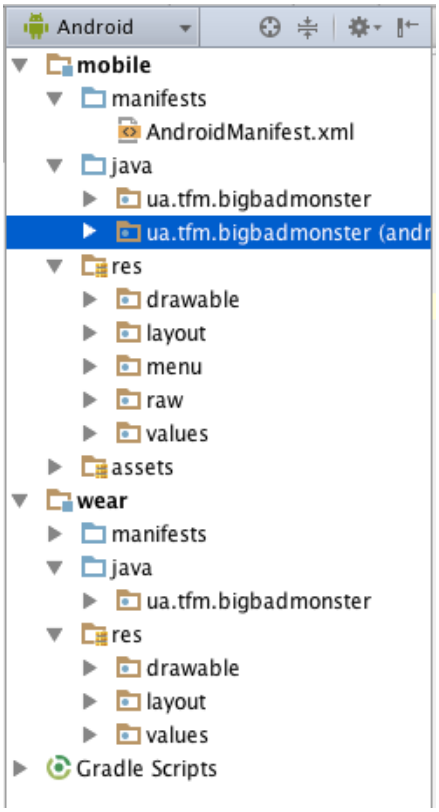
[3] Datos obtenidos de http://www.mundopepone.com/2012/12/06/como-funciona-el-mando-de-wii/

3 Análisis y Diseño

En este apartado realizaremos un estudio previo de lo que tenemos que saber para poder implementar la aplicación que queremos, realizando una labor de investigación para aprender cómo se desarrollan aplicaciones híbridas, la mejor manera de sincronizar datos entre dispositivos o cómo interpretar los movimientos reales de la muñeca para poder transformarlos en datos utilizables por la aplicación.

3.1 Estructura de una aplicación Android con *Wearables*

Programar una aplicación para dispositivos Android con *wearables* es ligeramente distinto a programar una aplicación solo para teléfonos Android. En primer lugar, el IDE que tenemos que utilizar es el nuevo *Android Studio* ya que el *Eclipse* de "toda la vida" no nos creará la estructura de aplicación necesaria para estos casos.



Esta estructura consta de dos módulos, uno para el teléfono móvil y otra para el dispositivo *wearable*. De esta manera, podemos implementar las *activities* y *layouts* que necesitemos en cada unos de los dispositivos de manera individual, teniendo en cuenta su tamaño y capacidad de la memoria y el procesador.

Partiendo de esta base, hay que elegir e implementar un buen método de intercambio de datos entre estos dos dispositivos para que la comunicación sea la adecuada para cada caso.

Figura 1: Estructura de un proyecto Android con Wearables.

3.1.1 Depuración de una aplicación híbrida

La depuración de la aplicación es ligeramente distinta a la tradicional. Para la correcta depuración de este tipo de aplicaciones debemos seguir estos pasos:

1. Conectar el teléfono al ordenador mediante un cable. *Android Studio* reconocerá que hemos introducido un dispositivo y lo pondrá en la lista de *devices*.
2. Conectar el bluetooth del móvil, y mediante la aplicación *Android Wear* (aplicación de descarga obligatoria para poder utilizar un wearable. Esta aplicación permite configurar las opciones del reloj). Conectar el móvil al reloj.
3. Activar la depuración por bluetooth desde esta misma aplicación, nos aparecerá lo siguiente:

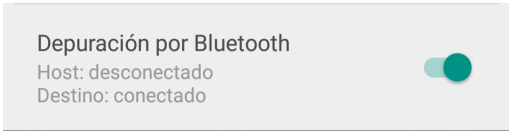


Figura 2: Captura de la aplicación Android Wear.

4. Conectar el *Host* desde terminal con los siguientes comandos:

```
adb forward tcp:4444 localabstract:/adb-hub
adb connect localhost:4444
```

Una vez seguidos estos pasos en la lista de *devices* nos aparecerán los dos dispositivos. Cuando vayamos a depurar deberemos elegir qué modulo ejecutar, el del móvil o el del teléfono. Para esto elegiremos el módulo en el desplegable de la barra de herramientas del *Android Studio* y cuando nos salga el diálogo para ejecutar, elegiremos el mismo dispositivo que pusimos en el desplegable.

Figura 3: elección de dispositivo para depurar.

3.1.2 Empaquetado de la aplicación

Este paso es igual al que seguiríamos con una aplicación tradicional, la única diferencia es que hay que asegurarse que los dos módulos son parte del mismo paquete y tienen la misma versión.

Una vez hecho esto firmamos la app y *Android Studio* se encargará de generar la APK.

3.2 Especificación de requisitos

El objetivo de cualquier juego es entretener al usuario el máximo tiempo posible, pero a veces la clave para conseguir esto es que el juego sea ligero, fácil de manejar y de rápida jugabilidad. Para ello hemos estimado unos requisitos funcionales básicos que deberá tener el juego final:

- Los menús del juego deben ser sencillos, con pocas y precisas opciones que hagan que el usuario no se pierda navegando y pueda jugar rápidamente.
- La curva de aprendizaje del juego debe ser casi nula, teniendo el usuario que aprender únicamente los gestos del reloj asociados a cada monstruo. Por otra parte, este único aprendizaje es lo que hace que este juego sea divertido.
- Este es un juego puntual, de jugada rápida, por lo que las partidas o los niveles tienen que ser cortas.
- Como el objetivo del juego es atrapar a los monstruos del nivel lo más rápido posible, la sincronización teléfono-reloj debe ser en tiempo real.

3.3 Especificaciones Técnicas

Para la correcta implementación de la idea básica del juego este debe cumplir con algunas especificaciones técnicas tales como la correcta sincronización del reloj con el teléfono, la gestión de los datos recogidos por el acelerómetro del reloj y la captura del tiempo de partida.

3.3.1 Datos del acelerómetro

La base del juego es coger a los monstruos que nos aparecen con un gesto de la muñeca donde tenemos ubicado el reloj, para ello utilizamos el acelerómetro.

Este dato no se consigue de manera trivial, lo que se ha hecho es un filtro dedicado a la lectura de los datos del acelerómetro de la siguiente manera:

- Primero capturamos el dato actual del acelerómetro para cada una de las coordenadas del espacio (x, y, z).
- Comparamos este dato con el que se ha tomado 0,75 segundos antes.
- Si la diferencia entre las variables está dentro de unos parámetros estudiados se manda al teléfono el tipo de movimiento que se ha capturado.

En esta versión de la aplicación existen cinco tipos de monstruos, por lo tanto, se han establecido cinco tipos de movimientos bien diferenciados. Para poder calibrar los movimientos, me he ayudado de la aplicación "Sensor Dashboard".

Esta aplicación captura los movimientos de los sensores que detecta en el reloj y los muestra en una gráfica en el teléfono en tiempo real. La correspondencia de colores-coordenadas es la siguiente:

- Coordenada X -> color azul
- Coordenada Y -> color morado
- Coordenada Z -> color verde

3.3.1.1 Movimiento 0

Este movimiento no está diseñado para cazar a ningún monstruo, sino para iniciar la captación de datos del reloj por parte del teléfono. De esta manera, le decimos al juego cuando puede empezar a atrapar monstruos con los gestos del reloj. Si este movimiento no estuviera, el reloj empezaría a captar datos desde el momento en el que inicializamos la aplicación en él, en el caso en el que en el teléfono todavía no estuviese activa la pantalla del juego, el reloj almacenaría los datos de los movimientos hasta poder mandarlos. De esta manera, seria posible pasarnos un nivel entero si ni siquiera haber empezado a jugar.

El funcionamiento es el siguiente: cuando lleguemos a la pantalla de juego y con el reloj mirando hacia arriba, lo ponemos unos segundos mirando hacia abajo, es decir, con la palma de la mano hacia arriba. En cuando el movimiento llegue al teléfono este emitirá el sonido de inicio del juego y se empezará a contar el tiempo.

Las graficas de este movimiento son planas, debido a que solo captamos que la Z es gravedad negativa y la X y la Y están sobre el 0.

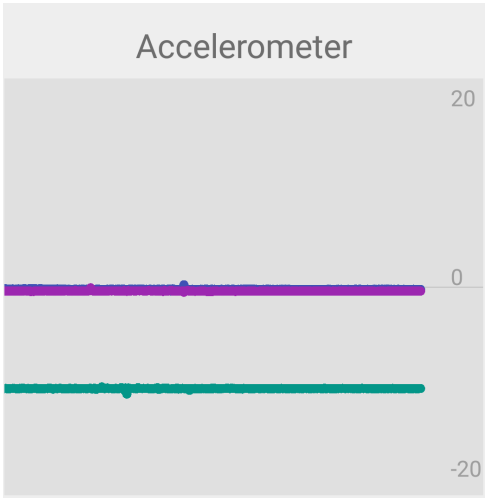
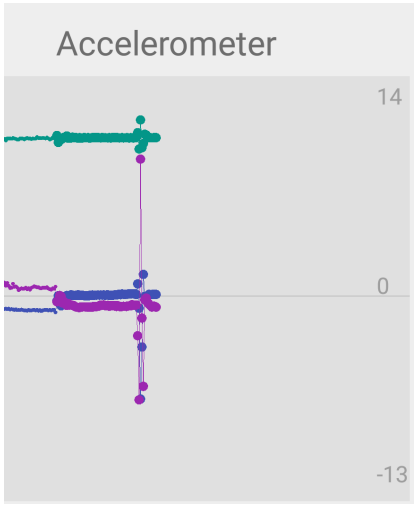


Figura 4: Gráfica del movimiento 4.

3.3.1.2 Movimiento 1

Con el reloj mirando hacia arriba: hacer un movimiento en el sentido de la coordenada X. Este movimiento equivale a las siguientes gráfica:



Como se puede apreciar, la gravedad está en el eje Z y no varía mucho. El eje X se tiene una pequeña variación, pero el que más se nota es el eje Y. Por lo tanto los parámetros clave son la gravedad en Z y la variación de la Y.

Figura 5: Gráfica del movimiento 1.

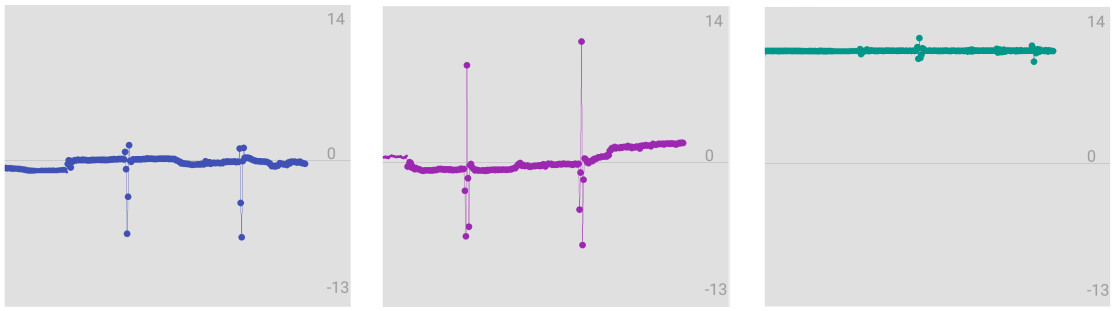
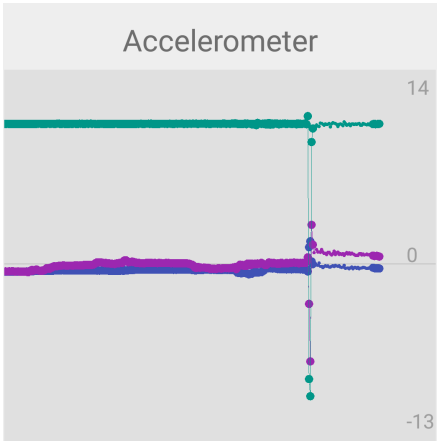


Figura 5.1: Desglose del movimiento 1. De izquierda a derecha: X, Y , Z.

3.3.1.3 Movimiento 2

Con el reloj mirando hacia arriba: hacer un movimiento en el sentido de la coordenada Y.



La gravedad sigue en el eje X, pero en este casi sí experimenta una variación importante. El eje X casi no se mueve, y el eje Y tiene una pequeña oscilación. Las coordenadas clave de este movimiento son la Z y la Y.

Figura 6: Gráficas del movimiento 2.

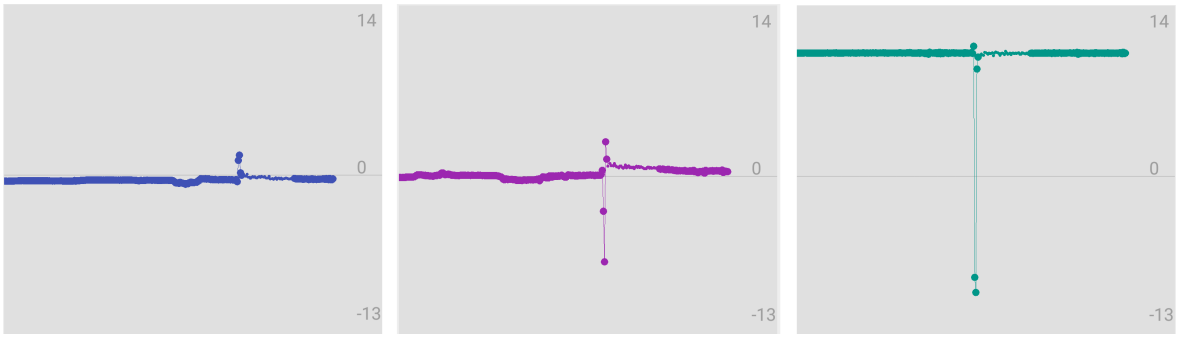
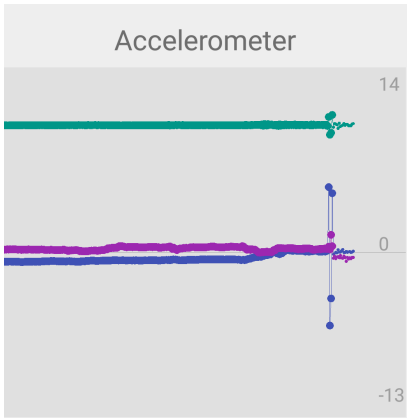


Figura 6.1: Desglose del movimiento 2. De izquierda a derecha: X, Y , Z.

3.3.1.4 Movimiento 3

Con el reloj mirando hacia arriba: hacer un movimiento en el sentido de la coordenada Z.



Con la gravedad en Z, la variación se produce en el eje X. Las oscilaciones de los ejes Y y Z son despreciables.

En este caso tomamos como referencia la gravedad en Z, el eje Y oscila entre los parámetros -3 y 3, y el movimiento se da en el eje Y.

Figura 7: Gráficas del movimiento 3.

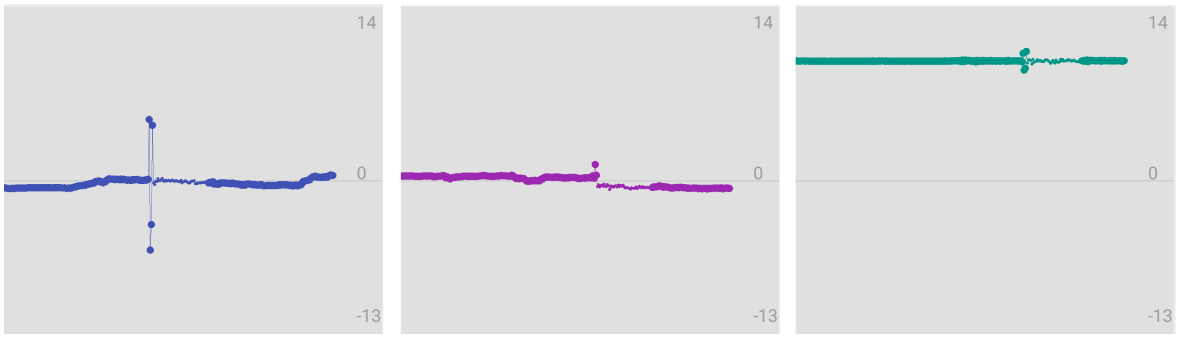
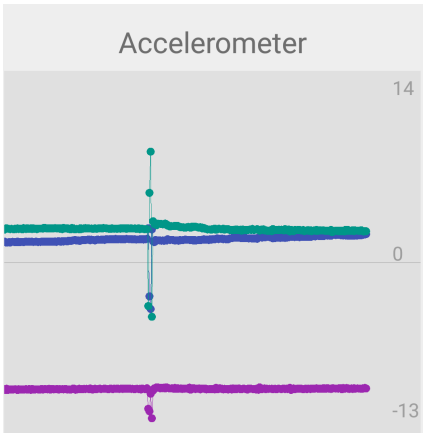


Figura 7.1: Desglose del movimiento 3. De izquierda a derecha: X, Y , Z.

3.3.1.5 Movimiento 4

Con el reloj mirando hacia la izquierda: hacer un movimiento en el sentido de la coordenada X.



Aquí la gravedad está en la coordenada Y, que además es negativa y la oscilación se da en los ejes X e Y.

Tomamos como referencia la Y como gravedad negativa, y capturamos las variaciones de la X y la Y.

Figura 8: Gráfica del movimiento 4.

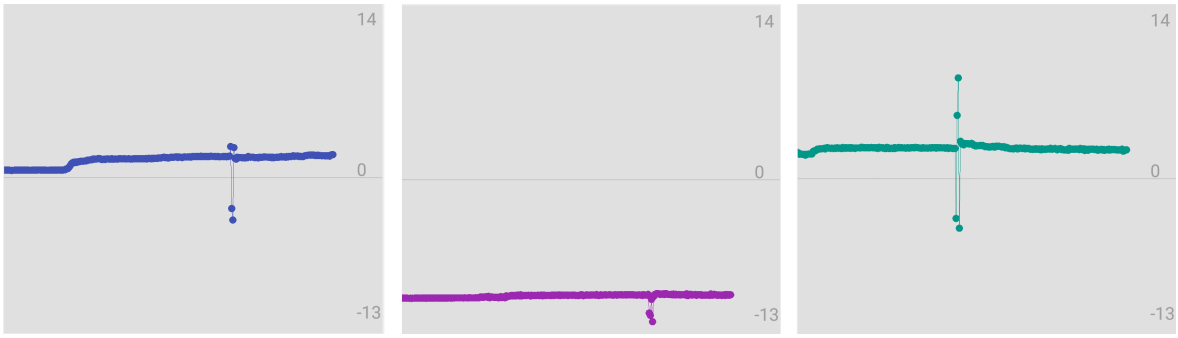
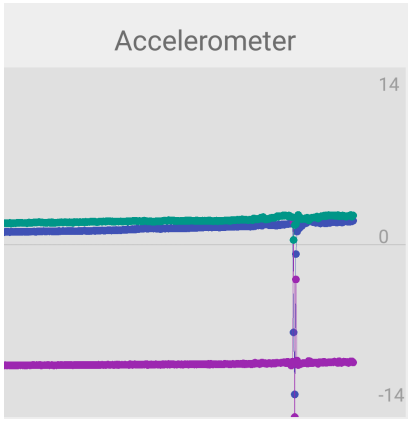


Figura 8.1: Desglose del movimiento 4. De izquierda a derecha: X, Y , Z.

3.3.1.6 Movimiento 5

Con el reloj mirando hacia la izquierda: hacer un movimiento en el sentido de la coordenada Y.



La gravedad sigue en –Y, pero en este caso es esta coordenada la que sufre la oscilación. También encontramos variación significativa en el eje X, mientras que el eje Z se mantiene casi constante.

Con la Y en gravedad negativa capturamos su movimiento, así como la oscilación del eje X. Tendremos como referencia el eje Z en un valor entre -3 y 3.

Figura 9: Gráfica del movimiento 5.

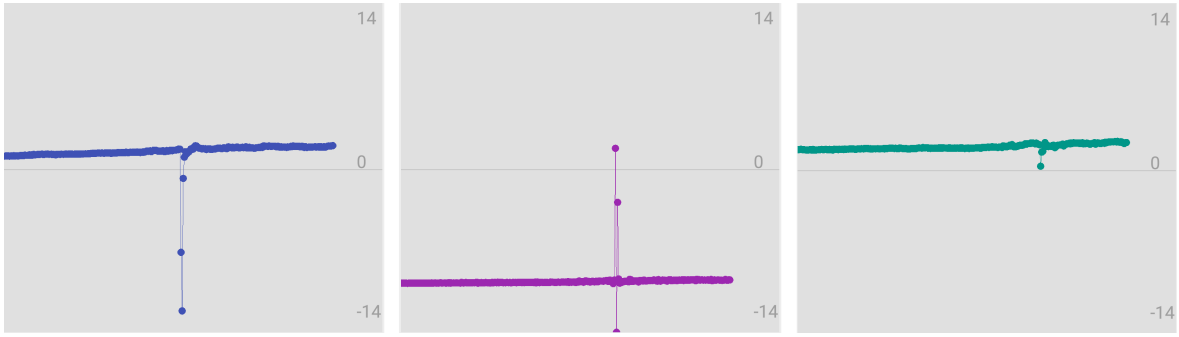


Figura 9.1: Desglose del movimiento 5. De izquierda a derecha: X, Y , Z.

3.3.2 Sincronización teléfono-reloj

Una vez hemos capturado el tipo de movimiento con el acelerómetro, el reloj debe enviar al teléfono el dato correspondiente. Android cuenta con varias formas de mandar datos de un dispositivo a otro y todas se basan un objeto llamado *DataItem*.

Un *DataItem* es un objeto utilizado para definir una interfaz de intercambio de datos entre *wearables* y teléfonos, generalmente consiste en un par clave-valor y una ruta para compartir la información que debe empezar por "/".

Una vez tenemos nuestro *DataItem*, este se puede enviar de dos maneras explicadas a continuación.

3.3.3 *DataMap*

Un *DataMap* es una clase que permite enviar información entre el teléfono y el reloj de manera inmediata utilizando pares de clave-valor. La información que podemos manejar en este caso es limitada (máximo 100KB), ya que se limita a datos pequeños como números o *byteArrays*. Esta información se puede serializar, de manera que también podemos mandar objetos creados por nosotros.

Google nos recomienda esta forma de sincronización si vamos a enviar poca información ya que es inmediata. Además, en el caso en que la conexión se pierda, los datos se guardan en un buffer y se sincronizan todos en cuanto es restablecida la conexión, de esta manera, nos aseguramos de no perder ningún dato. Por estas razones, este método de intercambio de datos es el más adecuado para nuestra aplicación.

3.3.4 *Assets*

Para mandar largas cadenas de datos binarios mediante el Bluetooth, como imágenes, usamos los *Assets*. Estos objetos manejan de forma automática el almacenamiento en caché de los datos para evitar la retransmisión y conservar el ancho de banda Bluetooth, un ejemplo común es una aplicación que necesite enviar una imagen al dispositivo *wearable*. Primero la reducirá a un tamaño adecuado a la pantalla del *wearable* y luego la enviará mediante un *Asset.*

A diferencia del *DataMap*, los *Assets* pueden ser tan grandes como queramos, pero hay que tener en cuenta que si mandamos objetos muy grandes la experiencia de usuario se verá afectada.

3.4 GameFlow

Una vez tenemos la idea principal de cómo va a ser nuestro juego, montamos la estructura que tendrá. La primera versión consta de dos niveles, el primero es un pequeño tutorial donde se especifica qué movimiento corresponde con cada monstruo y el segundo ya es un nivel normal. No obstante como hemos explicado antes añadir niveles es algo trivial, por lo que en posteriores versiones se añadirán más niveles con nuevos retos.

3.4.1 Estructura del juego en el módulo del móvil.

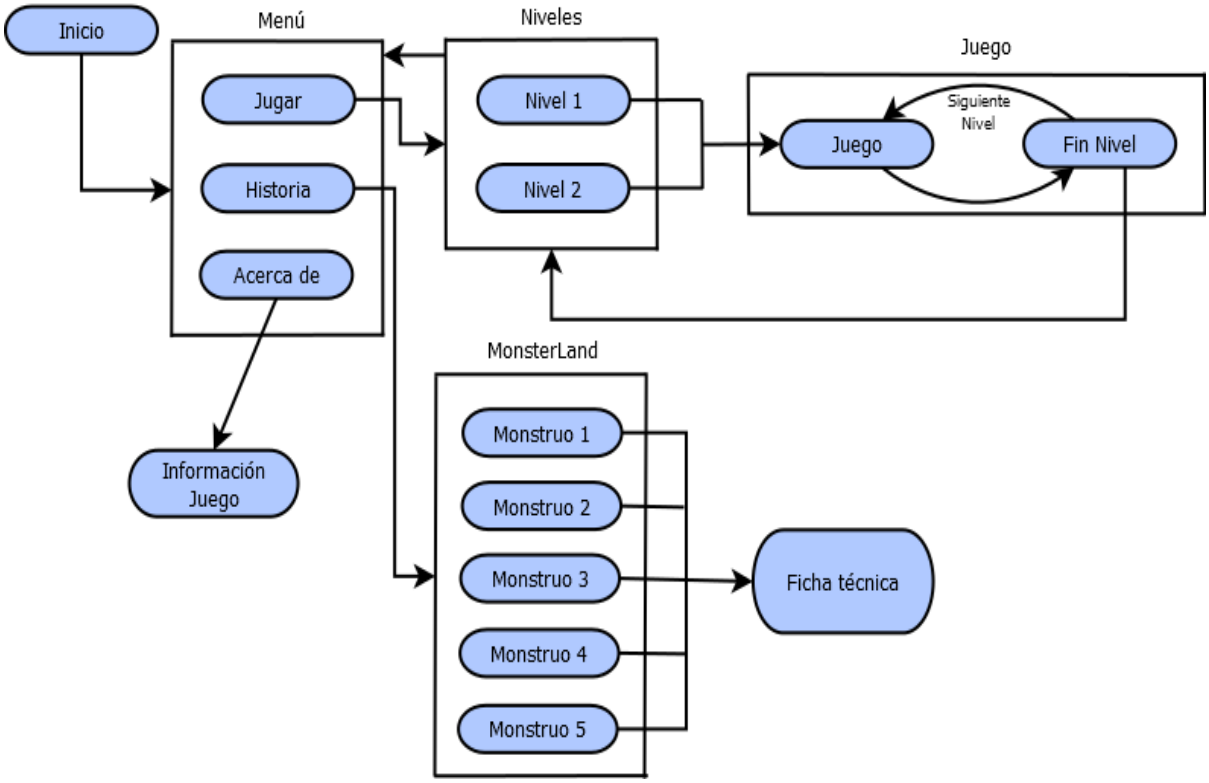


Figura 10: Estructura del juego en la parte del móvil.

3.4.2 Estructura general del juego

En este diagrama se muestra la estructura que tendrá nuestro juego, unificando el módulo del reloj y el del teléfono. Aquí se puede observar dónde se realiza la comunicación entre los dos dispositivos.

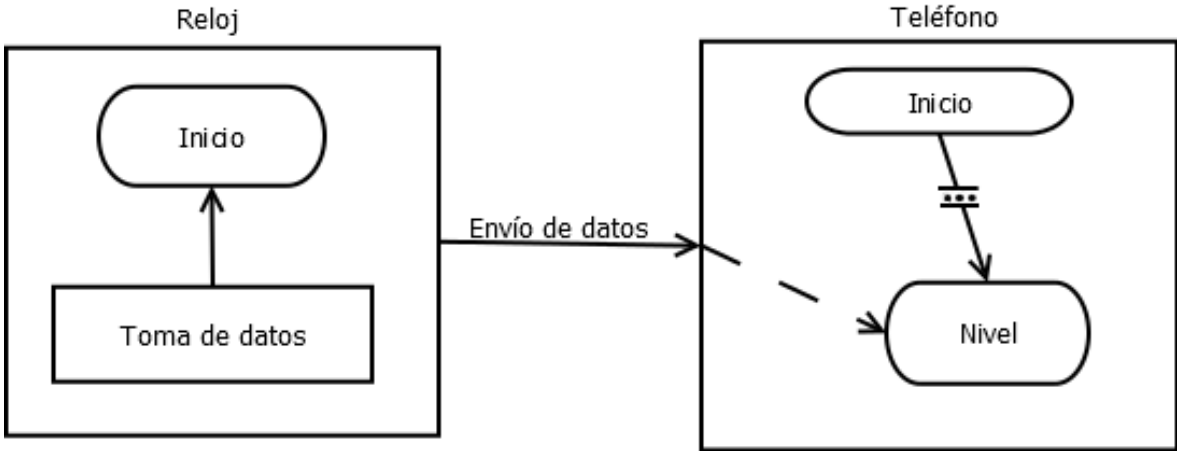


Figura 11: Estructura general del juego.

El reloj es el que se encarga de capturar los datos del acelerómetro, realiza el filtro de los datos para determinar qué movimiento se ha hecho y le manda el número del movimiento al teléfono. El teléfono solo escucha estos datos en la pantalla del juego.

3.5 Diagrama de clases

Este juego se basa en la interpretación de los gestos de la mano donde tenemos el reloj y la sincronización de estos datos con el teléfono, por lo que no necesitamos muchas clases para poder implementarlo. Por lo tanto, las clases con las que contamos son un *Singleton* que nos permite:

- Inicializar a los monstruos
- Determinar el número de monstruos de cada nivel, su orden y cuáles son mediante un *Array* de monstruos.
- Saber en qué nivel nos encontramos en cada momento.

Cabe destacar, que gracias a la manera en la que se ha implementado esto, añadir niveles al juego es sumamente fácil. Simplemente hay que definir un nuevo *Array* de monstruos que conformarán el nivel e insertarlo en un *Switch* que utilizamos para inicializar cada nivel.

La otra clase que se ha implementado es la clase "Monstruos", compuesta por:

- *id* del monstruo.
- *numRespuesta:* se utiliza para asignar un número de movimiento al monstruo. Esto quiere decir que cuando el reloj envíe un número de movimiento 1, por ejemplo, el movimiento asociado al monstruo debe ser el 1 para que la respuesta sea correcta.
- *Imagen:* utilizamos este campo para asignar un *drawable* al monstruo.
- *Texto:* este campo es para asignar un valor de cadena de texto a cada monstruo. Esto se utiliza en el apartado *Historia*, para exponer la ficha de cada monstruo.

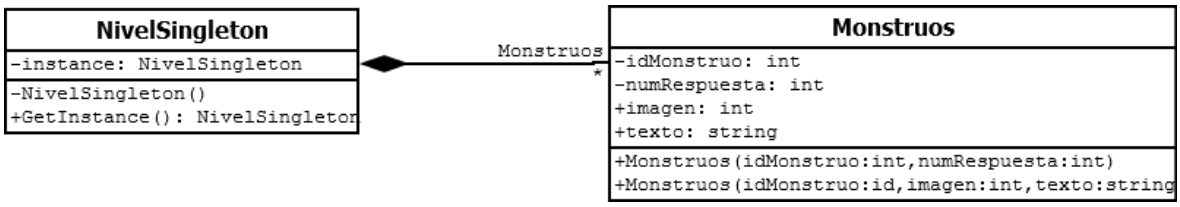


Figura 12: diagrama de clases de BigBadMonster.

4 Implementación

Como hemos explicado en el punto 3 de este documento, para implementar esta aplicación se ha tenido que investigar mucho acerca del funcionamiento interno de esta, ya que nunca antes había desarrollado para el *SmartWatch* de Android.

Como resultado de esta investigación hemos llegado a la conclusión de que lo más apropiado es utilizar la API nativa de Android ya nos provee de todo lo necesario para poder trabajar con los dos dispositivos y sincronizarlos entre si. También hemos determinado que el mejor tipo de *activity* a utilizar sea la *Activity*, presente en Android desde las primeras versiones. Hemos elegido esta en lugar de la moderna *ActionBarActivity* porque no necesitamos la barra de menú que esta implementa, a parte de que estéticamente no se integra en el tema del juego.

Hay que saber que para poder programar una aplicación que utiliza un *wearable*, lo que hace Android es dividir la solución del proyecto en dos módulos distintos: el módulo de la aplicación para el teléfono y el módulo de la aplicación para el reloj.

Para poder probar el código, al depurar se debe elegir si se prueba el módulo del reloj o del teléfono, por lo que la comunicación entre ellos hay que programarla, diciéndole dónde vas a guardar el dato, qué tipo de dato es y su clave para identificarlo.

A parte de la lógica de una aplicación híbrida, en esta en concreto, se han querido capturar movimientos reales, por lo tanto se ha necesitado de una captura de movimiento en tiempo real basándose en las coordenadas que nos da el acelerómetro. Esto implica que hay que filtrar muy bien los movimientos que se hacen con el reloj en la muñeca, ya que no todos nos sirven. Y además, hay que saber distinguir cuándo un movimiento es válido o no.

Por todo esto la implementación se ha llevado a cabo por partes, ya que cada una de estas partes conlleva una investigación previa. Cada parte se puede probar por separado, implementando todas juntas el juego final.

4.1 Calibración del acelerómetro

La primera parte de la aplicación es la captura de los datos del acelerómetro en el reloj, para ello se ha utilizado la clase *SensorManager* proporcionada por Android. Una vez capturados se hace un filtrado para diferenciar claramente seis movimientos. Esto se ha llevado a cabo en el módulo del reloj.

```
if ((oldX - 2 < x && x < oldX + 2) && (-11.8 < z && z < -8.8)) {
    //movimiento de inicio de juego = 0
    mandaDato(0);
}

if ((8.5 < oldZ && oldZ < 10.5) && (Math.abs(y - oldY) > 5) && (x < 0)){
    //movimiento x, reloj mirando arriba = 1
    mandaDato(1);


} else if ((0.0 < oldX && oldX < 5.0) && (-5.0 < oldY && oldY < 2.0) && (Math.abs(z - oldZ) > 15)) {
    //movimiento Y, desde reloj mirando hacia arriba girar la muñeca hasta reloj mirando hacia abajo = 2
    mandaDato(2);


} else if ((-3.0 < oldY && oldY < 3.0) && (8.5 < oldZ && oldZ < 10.5) && (Math.abs(x - oldX) > 5)) {
    //movimiento z, reloj mirando arriba = 3
    mandaDato(3);


} else if ((-15.8 < oldY && oldY < -8.8) && (Math.abs(x - oldX) > 10) && (Math.abs(z - oldZ) > 10)) {
    //movimiento X, reloj mirando izq = 4
    mandaDato(4);


} else if ((-3 < oldZ && oldZ < 3) && (Math.abs(y - oldY) > 10) && (Math.abs(x - oldX) > 10)) {
    //movimiento de Y, reloj mirando izq = 5
    mandaDato(5);

}
```

Figura 13: Código del filtro de movimientos del reloj.

Hay que destacar los problemas que hemos tenido a la hora de diferenciar con claridad los seis tipos de movimientos, pondremos un ejemplo:

El movimiento 2 se caracteriza por que la gravedad está inicialmente en Z y es positiva, al mismo tiempo que las coordenadas X e Y están más o menos rozando el 0. Cuando se realiza el movimiento la mayor variación de datos se obtiene de la coordenada Z que es la gravedad, la cual, es nuestro mejor referente ya que las otras dos coordenadas están más o menos en el mismo sitio.

Por otro lado, si nos fijamos en el movimiento número 1 veremos que la gravedad está en Z y que la X y la Y también están sobre el 0 de tal manera, que si no coordinamos bien el momento en el que se toman las medidas, estos dos movimientos se pueden confundir. Lo mismo pasa con los movimientos 4 y 5, en los que la gravedad está en –Y y tanto la X como la Z varían.

4.2 Sincronización con el teléfono

En la segunda parte se envían los datos de los movimientos capturados al teléfono para que este gestione la lógica del juego. Para ello se ha utilizado la clase *DataMap* de la API de Android para *wearables*.

La función implementada para mandar información al teléfono es la siguiente:

```
public void mandaDato(int dato)
{
    PutDataMapRequest putDataMapReq = PutDataMapRequest.create("/count");
    putDataMapReq.getDataMap().putInt(COUNT_KEY, dato);
    PutDataRequest putDataReq = putDataMapReq.asPutDataRequest();
    pendingResult = Wearable.DataApi.putDataItem(mGoogleApiClient, putDataReq);

    pendingResult.setResultCallback(new ResultCallback<DataApi.DataItemResult>() {
        @Override
        public void onResult(final DataApi.DataItemResult result) {
            if (result.getStatus().isSuccess()) {
                Log.d(TAG, "Data item set: " + result.getDataItem().getUri());
            } else {
                Toast.makeText(getApplicationContext(), "Fallo envio: ", Toast.LENGTH_SHORT).show();
                Log.d(TAG, "Data item set: " + result.getStatus().getStatusMessage());
            }
        }
    }, 5, TimeUnit.SECONDS);
}
```

Figura 14: Extracto de código de BigBadMonster: función utilizada para mandar datos al teléfono.

Como se puede observar en la línea número 4, se está introduciendo en un *DataMap* el conjunto de clave-valor con los datos correspondientes al movimiento efectuado en el reloj, para mandarlo al teléfono en la línea siguiente.

También se incluye una función de *callback* con el fin de tener *feedback* sobre el estado del envío, y como se puede observar se ejecuta a los 5 segundos de haber efectuado la transacción. Esto se utiliza a nivel desarrollador para la correcta depuración del código y es completamente transparente para el usuario

Una vez se ha recibido el dato en el teléfono, se compara si el tipo de movimiento corresponde con el monstruo mostrado en pantalla, si es correcto se pasará al siguiente.

Los problemas que hemos tenido en este punto no han sigo pocos, ya que para probar esto hay que tener los dos dispositivos conectados al mismo tiempo y las dos aplicaciones en primer plano. Por lo tanto, se ha especificado en el archivo

Manifest de las dos apps que mientras se esté jugando, la pantalla no se apague y por lo tanto las aplicaciones no se pausen o pasen a segundo plano si no es por orden del usuario. Así nos aseguramos de tener la conexión siempre activa.

4.3 Estructura del juego

En esta tercera parte se ha implementado toda la estructura de clases del juego y la lógica que este lleva. Esto se ha implementado en la aplicación para el teléfono. En primer lugar se ha creado una clase *Singleton* donde se gestionan qué monstruos aparecen en cada nivel y en qué nivel nos encontramos. La segunda clase utilizada especifica qué monstruo corresponde con cada imagen y movimiento.

La estructura de los niveles del juego es circular, ya que la *Activity* que se utiliza para cada nivel es la misma. Para implementar esto primero tenemos que recoger del *singleton* el número de nivel en el que estamos y el *array* de monstruos que le corresponde. Este *array* lo recorreremos en bucle mostrando todos los monstruos, cuando lo terminemos, calculamos el tiempo que hemos tardado y lo enviamos a la pantalla de fin de nivel.

Hay que remarcar que para recibir el dato enviado por el reloj utilizamos el método *onDataChanged* definido en la API *DataListener* de Android. Esto nos ocasiona un obstáculo, ya que este método se ejecuta en segundo plano y los cambio de monstruo se realizan en la interfaz de usuario, que Android ejecuta siempre en el hilo principal.

Para salvar este obstáculo se ha organizado la *activity* de la siguiente manera:

- Mostramos en primer monstruo en el método *onCreate*.
- Cuando recibimos un dato con el método *onDataChanged* llamamos a una función creada por nosotros pasándole este dato.
- En esta función especificamos que se desarrolle en el hilo principal e implementamos el bucle que recorre el *array* de monstruos, pero eliminando el primer monstruo de este, puesto que ya lo hemos mostrado.
  De esta manera conseguimos que el único dato que tienen que compartir los hilos sea el número de movimiento que ha llegado desde el reloj.

4.4 Animaciones

Para que el juego quedara más dinámico hemos introducido algunos efectos de animación. Estos los hemos aplicado a las manos y los dientes de los monstruos, para crear un efecto de movimiento en ellos. También hemos introducido unas manos animadas para mostrar el movimiento que hay que hacer con la mano del reloj para cada monstruo. Obviamente estos efectos se ven mejor en vivo, pero aquí mostramos un ejemplo de los *sprites* que hemos utilizado para cada tipo de animación.

Figura 15: Ejemplo de animación de monstruo. Se mueven las manos y los dientes.

Figura 16: Ejemplo de animación de mano. Se desplaza de izquierda a derecha.

5 Jugando a Big Bad Monster

En este apartado se va a explicar el juego para que se pueda tener una visión global de la aplicación y se puedan apreciar todos los matices antes expuestos.

Al abrir la aplicación veremos la portada del juego durante dos segundos para luego pasar al menú principal, donde podemos elegir entre jugar, conocer la historia de los monstruos o acceder al apartado *Acerca de*.

Figura 17: Capturas del juego-> Portada y menú principal

Si elegimos jugar pasaremos a la pantalla de niveles, donde elegiremos qué nivel queremos jugar. El primer nivel es un pequeño tutorial donde se presenta a cada monstruo y se explica con qué gesto atraparlo. Al entrar en el nivel te van apareciendo estos monstruos, aquí la función del jugador es realizar el gesto correspondiente a cada monstruo en el menor tiempo posible. Al finalizar el nivel, se nos mostrará el tiempo que hemos tardado y la puntuación obtenida, dándonos la posibilidad de pasar al siguiente nivel o volver a la pantalla de menú.

Figura 18: Capturas del juego-> nivel 1 y pantalla de fin de nivel.

Si elegimos la opción Historia se nos mostrará un menú para elegir la historia de qué monstruo queremos conocer. Cuando pinchemos en uno de los monstruos veremos su ficha personal, donde conoceremos du manera de actuar y se nos explicará el gesto a realizar para atraparlo mediante una animación.

Figura 19: Capturas del juego-> menú historia y ficha del monstruo verde.

En la opción *Acerca de* se ha incluido información sobre el origen del juego, autor, y compañía.

5.1 Historia

La historia del juego es sencilla, la leemos a continuación:

" Estos monstruos siempre se están metiendo en todo. Tu misión es dejarlos KO para poder atraparlos y que dejen de crear más problemas, pero ¡Ojo!, son muy listos y solo podrás ganarlos si los conoces bien y te sabes sus puntos débiles."

5.2 Tipos de monstruos

En esta versión existen cinco tipos de monstruos y ahora los vamos a conocer.

Este es *Rojus*, es el más rápido corriendo, pero siempre se despista y no mira a los lados. Un buen golpe por un flanco y ¡será tuyo!.

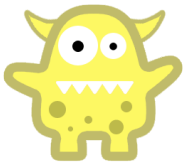
Gesto: con el reloj mirando hacia arriba realizar un movimiento en el sentido de la coordenada X.

Aquí tenemos a *Verdium,* este monstruito se dedica a corretear en círculos alrededor de la gente hasta que la desquicia, y no lo podrás tumbar empleando la fuerza. Prueba a darle la vuelta y ponerlo patas arriba. ¡No falla!

Gesto: con el reloj mirando hacia arriba realizar un movimiento en sentido antihorario.

Zulix tiene el mejor equilibrio de todos e intentará ganarte en su terreno tirándote al suelo para poder huir. Su punto débil es cuando cae de espaldas, ya que aún no ha aprendido a levantarse. ¡Sé más rápido y empújale tú primero!

Gesto: con el reloj mirando hacia arriba realizar un movimiento en el sentido de la coordenada Z.



¡Aquí viene *Yocus*! No te dejes intimidar por sus cuernos, en el fondo es un miedica. Si lo coges por un lado sin que te vea del mismo susto quedará KO.

Gesto: con el reloj mirando hacia la izquierda realizar un movimiento en el sentido de la coordenada X.

Y por último, presentamos a *Morax*. Es un poco tontete, debería ser fácil de atrapar, pero su cabeza es de hormigón puro. Imposible de atontar si no es con un buen martillazo en la mollera. ¡A por él!

Gesto: con el reloj mirando hacia la izquierda realizar un movimiento en el sentido de la coordenada Y.

5.3 Tiempo

En el juego, el papel del tiempo es muy importante, ya que nos basamos en él para puntuar al jugador. Cuanto más se tarde en atrapar a todos los monstruos de un nivel, menos puntuación se obtendrá, teniendo incluso que repetir el nivel si el resultado muy malo.

La puntuación se mide por estrellas, donde el máximo son tres estrellas, lo que significa que hemos hecho un tiempo muy bueno.

6 Trabajo Futuro

A pesar de tener una primera versión con muchos matices, siempre se puede mejorar e incluir nuevas funcionalidades. En nuestro caso, la base del juego se mantiene, pero se pueden incluir muchas opciones nuevas. Las principales mejoras que aportaría al juego serian:

- Compartir resultados: en una futura versión se incluirá la posibilidad de compartir los tiempos obtenidos con tus amigos mediante redes sociales como Facebook, Twitter, etc.
- Ranking y records: se añadirá una pantalla en el menú principal con un ranking de los tiempos obtenidos ordenados por puntuación, recalcando los récords de cada nivel.
- Multiusuario: siguiendo con la idea del ranking, se incluirá la posibilidad de guardar el perfil de varios jugadores, incluyendo sus puntuaciones en el ranking.
- Contrarreloj: se añadirá una nueva modalidad de juego llamada *Contrarreloj*, donde se tendrá que atrapar al máximo número de monstruos en el mínimo tiempo posible. Las puntuaciones también se incluirán en el ranking y se podrán compartir en las redes sociales.

7 Conclusiones

Como conclusión personal he de decir que ha sido una experiencia única, que ha permitido que pueda aprender a programar para Android de una manera nueva y diferente, ya que no solo se programa una aplicación para el móvil, sino que se tiene que tener en cuenta dos aplicaciones para plataformas distintas que tienen que funcionar de manera sincronizada, apoyándose la una en la otra.

Me ha encantado tener el privilegio de poder "trastear" con un nuevo dispositivo que a día de hoy poca gente tiene, pero que se espera que en el futuro sea uno más de la tecnología del día a día.

A parte de mi experiencia personal, en cuanto al juego desarrollado, estoy muy contenta de haber conseguido los objetivos que se plantearon al principio de este proyecto. Ya sabíamos que la idea sonaba muy bien, pero también sabíamos que el desarrollo no iba a ser fácil. Se nos planteaban obstáculos como la correcta interpretación de los movimientos del usuario y la sincronización teléfono-reloj en tiempo real.

Pero además de la parte técnica, también había que hacer un juego llamativo, con una historia que enganche y unos personajes entrañables. Creo que estos objetivos han sido cumplidos y que la aplicación final cumple las expectativas de ser un juego fresco, rápido y diferente.

8 Agradecimientos

Quiero agradecer su apoyo y ayuda a mis compañeros del máster, ya que me han ayudado con algunas ideas y me han apoyado cuando no he sabido cómo seguir adelante.

También quiero incluir en los agradecimientos al profesor F. Javier Ferrández Pastor, porque un día en su asignatura nos dejó un reloj Android y nos dio toda la clase para que pudiésemos probarlo y familiarizarnos con él. Ese día nació la idea de este juego, incluidas todas las ganas y la ilusión por aprender algo nuevo y de momento bastante exclusivo.

Y no quisiera dejar de agradecerle su apoyo a mi tutor Miguel Ángel Lozano, por prestarme su rápida ayuda siempre que la he necesitado y no ponerme ningún problema a la hora de organizarlo todo para que la universidad me pudiese prestar el reloj para mi desarrollo.

9 Bibliografía

Android Developers: http://developer.android.com/training/building-wearables.html

StackOverflow: http://stackoverflow.com

SensorDashboard:

https://play.google.com/store/apps/details?id=com.github.pocmo.sensordashboard