ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Ingeniería Informática

Virtual Golf

Realizado por
Álvaro Rivero Bablé
Macarena Noriega del Río

Dirigido por
José Ramón Portillo

Departamento
Matemática Aplicada I

Sevilla, (03/09/2013)

Resumen

Esta documentación corresponde a un proyecto final de carrera consistente en la creación de un videojuego de simulación de golf con Android en el que se estudian y se usan las técnicas de localización, realidad aumentada, uso del sensor acelerómetro y uso del sensor de orientación.

La finalidad de este proyecto es entretener al usuario y el estudio de las diferentes técnicas antes nombradas.

A lo largo de esta memoria se detallará el procedimiento seguido para la creación partiendo de cero de la aplicación:

Primero, empezaremos comentando los objetivos que nos marcamos al realizar la aplicación.

Seguidamente, pasaremos al análisis de requisitos, diseño y entraremos en profundidad en la implementación donde haremos mención a la librería *Look!* utilizada en el proyecto y cómo hemos conseguido integrar en una aplicación aspectos como utilización de una librería (realidad aumentada y localización), acelerómetro y orientación.

Más adelante hablaremos sobre los antecedentes, la aportación que hemos realizado y analizaremos temporalmente el proyecto. También comentaremos el dispositivo móvil que hemos utilizado para el desarrollo de la aplicación y explicaremos brevemente los sensores que necesitamos del teléfono para poder entender mejor lo resumido anteriormente.

Para seguir, aportaremos todas las pruebas realizadas que constatan que la aplicación realiza aquello que nos propusimos en los inicios.

Incluiremos un manual de uso de la aplicación para la correcta utilización por los usuarios.

Y para finalizar, somos conscientes de que este proyecto puede ser la base de un videojuego mucho más completo del que actualmente es y daremos las posibles mejoras y avances que podrían incluirse en futuras actualizaciones.

Agradecimientos

Resulta muy complicado para los autores de este proyecto enumerar a todos aquellos que han contribuido al buen fin del mismo. Por este motivo queremos disculparnos si alguien que ha resultado ser un apoyo a lo largo del proceso no aparece explícitamente en estas líneas.

Sobre todo agradecer a nuestras familias la paciencia y la comprensión sin la cual no hubiera sido posible superar aquellas situaciones adversas que nos han surgido. Y por supuesto a todos esos amigos, compañeros de trabajo, etcétera. Muchas gracias a todos.

Índice general

Índice de figuras

Índice de código

CAPÍTULO 1

Definición de objetivos

Para comenzar este capítulo, debemos mencionar que con el desarrollo del proyecto objeto de esta documentación se pretende elaborar una aplicación Android, basada en un videojuego de simulación deportiva de golf.

La aplicación nos dará la opción de jugar al golf desde cualquier sitio, sólo con el uso de un móvil Android con el videojuego instalado.

Para elaborar esta aplicación hemos usado las siguientes técnicas : localización , realidad aumentada, acelerómetro y orientación. A continuación mostramos un breve resumen de cada una de ellas.

- Con la localización determinaremos la ubicación de la bola y la ubicación del hoyo, con esto calcularemos la distancia a la que se encuentran. La localización se define como el emplazamiento de un objeto espacial en un sistema de coordenadas determinado, este proceso es utilizado frecuentemente en los Sistemas de Información Geográfica (SIG). En nuestro proyecto la localización tendrá un papel importante ya que tenemos que controlar en todo momento la situación de la bola. Usaremos la librería *Look!*, que explicaremos más adelante.

- La realidad aumentada se usa para definir una visión directa o indirecta de un entorno físico del mundo real. En 1962, Morton Heilig - director de fotografía - creó un simulador de moto llamado Sensorama con imágenes, sonido, vibración y olfato, lo más avanzado que podemos encontrar en el mercado relacionado con la realidad aumentada son las gafas que empezó a diseñar Google en el 2012, *Project Glass*, en Virtual Golf lo usaremos para poder comprobar en todo momento la situación de la bola y la del banderín virtual.

- Para hacer el lanzamiento de la bola de golf usaremos el acelerómetro del móvil que nos permitirá simular de alguna forma que tenemos

el palo de golf en la mano y con el movimiento del móvil podemos golpear la bola. También tendremos en cuenta la orientación del teléfono con el sensor de orientación para controlar en todo momento hacia dónde queremos dirigir el disparo.

Los objetivos que hemos buscado al realizar esta aplicación han sido:

- Demostrar que se puede utilizar el acelerómetro de los móviles para lograr una sensación parecida al mando de la Wii, ese ha sido el principal objetivo de esta aplicación.
- Cerciorarnos de que la aplicación pueda seguir creciendo debido al amplio potencial que tiene y la multitud de características que se le pueden incluir, independizando y modularizando lo máximo posible la inclusión de nuevas funcionalidades.
- Conseguir el acoplamiento de diferentes sensores y utilización de librerías en una misma aplicación Android.
- Asegurar el fácil, intuitivo y fluido manejo de la aplicación teniendo en cuenta la cantidad de herramientas del móvil que utiliza.

CAPÍTULO 2

Análisis de requisitos, diseño e implementación

Empezamos desarrollando algunas aplicaciones de prueba para irnos familiarizando con el problema a tratar, investigando el funcionamiento de diversas funcionalidades de Android que necesitamos para poder realizar nuestra aplicación. Vamos a pasar a explicar la realización de nuestra aplicación dividiéndola en tres apartados.

Primero pasaremos a desglosar el análisis de los requisitos necesarios para llevar a cabo la realización del proyecto. Indicando las tecnologías necesarias que hemos utilizado durante la etapa de desarrollo hasta los conocimientos que hemos tenido que ampliar para combinar estas tecnologías y conseguir un buen funcionamiento de la aplicación.

Seguidamente describiremos el diseño de nuestra aplicación, en él se analizará la estructura elegida para nuestro sistema, justificándola y explicando la relación de cada clase con cada módulo. Tras una fase teórica y otra teórico-práctica, pasaremos a la práctica, el desarrollo de la aplicación.

Será entonces cuando comentaremos con más detalle el código de la aplicación, pudiendo afianzar los conocimientos anteriormente expuestos.

2.1– Análisis de requisitos

El objetivo de VirtualGolf es el de conseguir acercarse a un juego virtual de golf. Usaremos los sensores del móvil para mover la pelota y el campo de juego lo visualizaremos desde la cámara, pintando tanto la pelota como el hoyo a través de la realidad aumentada.

Ahora vamos a pasar a explicar todas las tecnologías o componentes que utilizamos en VirtualGolf para poder llevar a cabo lo anteriormente expuesto:

- **Realidad aumentada:** En el mundo de los dispositivos móviles se

conoce por Realidad Aumentada a la tecnología que permite la superposición, en tiempo real, de imágenes generadas por ordenador sobre imágenes del mundo real. Estos datos superpuestos pueden ser tanto información relativa a elementos reales que están siendo visualizados, como datos independientes asociados a referentes físicos.

En un entorno de este tipo, el usuario puede además, interactuar con los objetos virtuales, para alterarlos u obtener información de ellos. Para ello, necesitamos el uso de la cámara.

- **Cámara:** Es un sensor que nos proporciona imágenes de nuestro entorno. Como en el caso del GPS, podemos encontrar fácilmente cámaras en los dispositivos móviles desde hace varios años, más aún en los de nueva generación.

  Este tipo de sensores son útiles tanto para tomar fotografías como para grabar vídeos, que podemos compartir con nuestros amigos o almacenarlos para realizar procesamientos posteriones. A partir del análisis digital de una imagen, podemos extraer una serie de parámetros que nos den información relevante sobre nuestro entorno, lo que puede ser útil para una aplicación determinada (detección de objetos, cálculo de magnitudes visuales, búsqueda de patrones, etc.).

  Por otro lado, también resultan útiles para la representación de ciertos objetos de manera mucho más intuitiva para el usuario. Por ejemplo, recursos como tiendas, restaurantes o cualquier otro objeto cuyas coordenadas geográficas sean conocidas, pueden ser presentados en la pantalla del dispositivo como un dibujo o imagen sobre una vista previa de la cámara, mostrando claramente en qué posicion se encuentran en relación a la localización del usuario.

- **Acelerómetro:** Se presentan como unos sensores capaces de medir la aceleración instantánea de la gravedad terrestre. Las medidas son realizadas a lo largo de los tres ejes cartesianos por lo que tenemos tres componentes, devueltas en un vector, que corresponden con la fuerza aplicada por el dispositivo sobre los ejes (x, y, z). Su fundamento es el siguiente: como sabemos que el módulo de la aceleración de la gravedad es, aproximadamente, 9.8 m/s2, el módulo percibido por el terminal en reposo debe ser el mismo. Por tanto, valores suficientemente mayores o menores indican que el portador del teléfono comienza a caminar o se detiene. Como sabemos la dirección y el sentido del mayor cambio, podemos estimar la dirección hacia la que se mueve.

  Por otro lado, como sabemos la magnitud de la gravedad en esas tres componentes, podemos calcular la inclinación del dispositivo en reposo. Por ejemplo, situando el terminal en posición horizontal, con

la parte positiva del eje Z apuntando hacia el techo, tenemos que sólo ese eje se ve afectado por la gravedad. Si comenzamos a inclinarlo alrededor de cualquiera de los otros ejes, veremos que la magnitud del eje Z decae, incrementándose en los otros ejes. Haciendo una medida de proporcionalidad entre estos valores, podemos saber el grado de inclinación del dispositivo.

- **Orientación:** No es realmente un sensor, sino una abstracción de los valores capturados por los acelerómetros y el sensor de campo magnético, con el objetivo de construir una brújula digital, que puede determinar la dirección hacia la que está orientado el usuario. Sin embargo, no sólo devuelve la orientación, también la inclinación del dispositivo.

  Entre las magnitudes retornadas por este sensor, tenemos:

  Azimut: Mide en grados la rotación alrededor del eje Z. Ese ángulo puede ser traducido en puntos cardinales.

  Pitch: Mide en grados la rotación alrededor del eje X. Su rango de medida es [-180º, 180º].

  Roll: Mide en grados la rotación alrededor del eje Y. Su rango de medida es [-90º, 90º].

2.2– Diseño

Tras el Análisis de Requisitos, vamos a realizar el diseño de nuestro proyecto, distribuyendo las funcionalidades previstas de la forma más óptima posible para un fácil mantenimiento y proporcionando un entendimiento fácil e intuitivo.

Hemos organizado el proyecto en 4 paquetes:

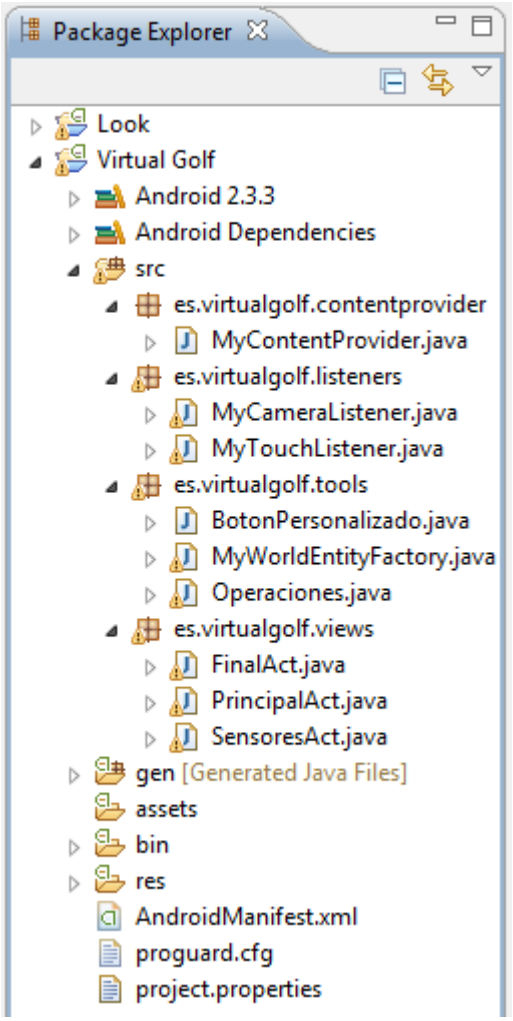

Figura 2.1: Estructura del proyecto

El paquete *es.virtualgolf.views* es donde tenemos las actividades de la aplicación. Cuenta con una clase principal llamada *PrincipalAct*, es la clase (como su propio nombre indica) principal y será donde accedemos al arrancar la aplicación. En esta actividad podremos ver la situación de la bola y el hoyo. Tiene un botón *Golpear* para empezar a jugar y también nos muestra el número de golpes que llevamos. La siguiente clase se denomina *SensoresAct* donde aparece el botón para poner en marcha los sensores y también se muestra el valor de cada parámetro del acelerómetro y de la orientación para poder mejorar nuestros tiros. Por último en este paquete, está también la clase *FinalAct* que nos indicará el final de la partida, o bien por haber introducido la bola en el hoyo o bien porque nos hemos pasado del número máximo de disparos.

El paquete *es.virtualgolf.contentprovider* sirve para cuando queramos

que los elementos que han sido añadidos durante la ejecución, sean recordados en subsecuentes ejecuciones, utilizando una base de datos. La clase *LookSQLContentProvider* representa una base de datos genérica, que extendemos con nuestra clase *MyContentProvider*.

El paquete *es.virtualgolf.listeners* lo utilizamos para añadir interacciones con las entidades. Cada una de las entidades permiten que le sean añadidos una serie de *listeners* que respondan a los distintos eventos que puedan surgir durante la ejecución de la aplicación. Para añadir procesado de eventos táctiles usamos la clase *MyTouchListener*. Cuando una entidad reciba un evento táctil, pasará ese evento a todos los *listeners* de su lista de *TouchListener*. La interfaz define métodos para los tres tipos de eventos táctiles principales (*onTouchDown*, *onTouchUp*, *onTouchMove*). Cada uno de los métodos recibe como parámetro la entidad que recibió el evento y las coordenadas de pantalla donde sucedió. Para añadir eventos de cámara usamos la clase *MyCameraListener*. Los eventos de cámara son cuando el usuario enfoca (o deja de enfocar) directamente una entidad con el centro de su dispositivo. *CameraListener* cuenta con dos eventos: uno para cuando el usuario enfoca la entidad: *onCameraEntered*, y otro para cuando deja de enfocarla: *onCameraExited*.

El paquete *es.virtualgolf.tools* es donde vamos a albergar todas las clases auxiliares. La clase *BotonPersonalizado* sirve para la funcionalidad del botón de recogida de datos (que se apague y se encienda visualmente, que sólo esté activo cuando sea pulsado durante un tiempo...). La clase *MyWorldEntityFactory* es donde vamos a crear las entidades en nuestra aplicación. En nuestro proyecto, hemos necesitado crear solamente las entidades *bola* y *banderín*. Por último en la clase *Operaciones* vamos a incluir todos los métodos auxiliares necesarios para el correcto funcionamiento de la aplicación que quitamos del resto de clases para una mayor modularidad.

2.3– Implementación

2.3.1. Librería Look!

Look! es un framework de Realidad Aumentada para Android creado para resolver los problemas comunes encontrados en el desarrollo de aplicaciones de este tipo. Es de código abierto (Licencia GPL v3) y con capacidad de extensión. Integra las siguientes características para que puedas crear tu aplicación de una manera simple y rápida. Permite:

- Dibujado de gráficos en dos y tres dimensiones.
- Posibilidad de integrar los gráficos con la cámara.
- Interacción con los Objetos Virtuales.

- Construcción de Entidades representables en Realidad Aumentada.

Para poder realizar aplicaciones atractivas y vistosas, el framework define herramientas para el dibujado de elementos tanto en 2D como 3D, que si se desea se pueden superponer a la cámara.

Con Look! se pueden incluir de manera muy simple objetos comunes en el dibujado, como textos y formas básicas, además ofrece herramientas para definir colores y texturas. También provee de funcionalidades geométricas, para facilitar labores comunes en el desarrollo de gráficos: puntos, vectores, matrices, planos y rayos, y todas las operaciones relacionadas.

También, aunque no lo hayamos necesitado, soporta:

Localización en Interiores de Edificios. Integración con Servicios Remotos. Servicio de Persistencia de Datos.

2.3.2. PrincipalAct

Comienzo del juego:

Es la pantalla principal del juego. Comenzamos recuperando la información entre actividades, si es nula es que estamos comenzando un nuevo juego. Mostramos el número de golpes (que en este caso es 0), creamos el nuevo escenario, la entidad bola, la entidad banderín (inicializando sus propiedades) y las añadimos a la base de datos. Por último, actualizamos la base de datos.

```
1  @Override
2      public void onCreate(Bundle savedInstanceState) {
3          super.onCreate(savedInstanceState);
4
5          //Recuperamos la informacion pasada en el intent
6          Bundle bundle = this.getIntent().getExtras();
7          if(bundle == null){
8
9              tv_golpes.setText("Numero de golpes: " + golpes);
10
11             LookData.getInstance().setWorldEntityFactory(new
                   MyWorldEntityFactory());
12
13             // Creamos la bola
14             bola = new EntityData("bola");
15             bola.setLocation(5, 0, 2);
16             bola.setPropertyValue(MyWorldEntityFactory.NAME, "bola
                   ");
17             bola.setPropertyValue(MyWorldEntityFactory.COLOR, "
                   blanco");
18             LookData.getInstance().getDataHandler().addEntity(bola)
                   ;
```

```
19
20              // Creamos el banderin
21              banderin = new EntityData("banderin");
22              banderin.setLocation(5, 3, 4);
23              banderin.setPropertyValue(MyWorldEntityFactory.NAME, "
                    banderin");
24              banderin.setPropertyValue(MyWorldEntityFactory.COLOR, "
                    verde");
25              LookData.getInstance().getDataHandler().addEntity(
                    banderin);
26
27              // Actualizamos BD
28              LookData.getInstance().updateData();
29
30          }
31  ...}
```

Código 2.1: Comienzo del juego

Seguimiento del juego:

En el caso de que la información entre actividades no sea nula, es que el juego ya ha comenzado. De nuevo mostramos el número de golpes que llevemos, recuperamos de la base de datos las entidades, actualizamos la posición de la bola tras el último golpe y las volvemos a insertar y actualizamos la base de datos.

Tras eso, comprobamos que el número de golpes no sea superior a 10 o que la bola haya entrado en el banderín. Si se cumplen alguna de estas dos condiciones, pasamos a la actividad del final del juego. Según sea una u otra, enviamos un valor para diferenciarlas y mostrarlo en pantalla.

```
1   // Seguimiento del juego
2           }else{
3
4               golpes = bundle.getInt("golpes");
5               tv_golpes.setText("Numero de golpes: " + golpes);
6
7               // Recuperamos posiciones
8               float[] posicion_bola = (float[]) bundle.get("
                    posicion_bola");
9               float x1 = posicion_bola[0];
10              float y1 = posicion_bola[1];
11              float z1 = posicion_bola[2];
12
13              // Recuperamos la bola y el banderin
14              bola = LookData.getInstance().getDataHandler().getLista
                    ().get(0);
15              banderin = LookData.getInstance().getDataHandler().
                    getLista().get(1);
```

```
16
17              // Actualizamos la posicion de la bola
18              LookData.getInstance().getDataHandler().updatePosition(
                    bola, x1, y1, z1);
19
20              //Introducimos en la BD
21              LookData.getInstance().getDataHandler().addEntity(bola)
                    ;
22              LookData.getInstance().getDataHandler().addEntity(
                    banderin);
23
24              // Actualizamos BD
25              LookData.getInstance().updateData();
26
27              //Si hemos dado mas de 10 golpes o hemos metido la bola
                     -> FIN
28              if(golpes > 9 || Operaciones.estaDentro(bola, banderin)
                    ){
29                  Integer val = 0;
30                  if(golpes > 9){
31                      val = 0;
32                  }
33                  if(Operaciones.estaDentro(bola, banderin)){
34                      val = 1;
35                  }
36                  Bundle b = new Bundle();
37                  b.putInt("fin", val);
38
39                  //Creamos el Intent
40                  Intent intent = new Intent(PrincipalAct.this,
                        FinalAct.class);
41
42                  //Anadimos la informacion al intent
43                  intent.putExtras(b);
44
45                  //Iniciamos la nueva actividad
46                  startActivity(intent);
47              }
48
49          }
```

Código 2.2: Seguimiento del juego

Botón golpear:

Para empezar la partida pulsamos el botón, este nos enviará a la actividad para golpear. Debemos pasarle la posición de la bola y el número de golpes que llevamos (se aumenta en uno debido al próximo golpeo).

```
btn_golpear.setOnClickListener(new OnClickListener() {

            public void onClick(View v) {
                //Creamos el Intent
                Intent intent = new Intent(PrincipalAct.this,
                    SensoresAct.class);

                //Creamos la informacion a pasar entre actividades
                float[] posicion_bola = new float[3];
                posicion_bola[0] = bola.getLocation().x;
                posicion_bola[1] = bola.getLocation().y;
                posicion_bola[2] = bola.getLocation().z;

                //Aumentamos numero de golpes
                golpes = golpes + 1;

                Bundle b = new Bundle();
                b.putFloatArray("posicion_bola", posicion_bola);
                b.putInt("golpes", golpes);

                //Anadimos la informacion al intent
                intent.putExtras(b);

                //Iniciamos la nueva actividad
                startActivity(intent);

            }
        });
```

Código 2.3: Botón golpear

MyWorldEntityFactory

Es una clase auxiliar para la creación de forma geométrica de las entidades. En ella nos creamos el objeto de tipo *Ring* para el banderín virtual y la bola la creamos desde un archivo realizado con el programa *Blender*. Por último les ponemos el color a las entidades a través de RGB.

```

//Esfera
    ObjMesh3D sphere = DrawablesDataBase.getInstance().
        getDrawable3D(R.raw.sphere);

@Override
    public WorldEntity createWorldEntity(EntityData data) {
        WorldEntity we = new WorldEntity( data );
        Entity3D drawable3d = null;

        if (data.getType().equals("banderin")) {
```

```
11
12              drawable3d = new Entity3D(new Ring(new Point3(3.0f, 3.0
                    f, 3.0f), 1.0f, 3.0f, 4, new Color4(0.0f, 1.0f, 0.0f
                    )));
13          } else if (data.getType().equals("bola")) {
14              drawable3d = new Entity3D(sphere);
15              drawable3d.setMaterial(new Color4(1.0f, 1.0f, 1.0f));
16          }
17
18          String color = data.getPropertyValue(COLOR);
19
20          if (color != null)
21              if (color.equals("blanco")) {
22                  drawable3d.setMaterial(new Color4(1.0f, 1.0f, 1.0f))
                        ;
23              } else if (color.equals("verde")) {
24                  drawable3d.setMaterial(new Color4(0.0f, 1.0f, 0.0f))
                        ;
25              }
26
27          we.setDrawable3D(drawable3d);
28
29          we.addTouchListener(new MyTouchListener( ));
30          we.addCameraListener(new MyCameraListener( ));
31
32
33          return we;
34      }
```

Código 2.4: Bola y banderín

2.3.3. SensoresAct

Es la pantalla donde utilizamos el acelerómetro y la orientación, en ella hemos puesto un botón con la siguiente funcionalidad: captura datos (acelerómetro y orientación) solamente cuando está siendo pulsado.

Creación del botón para la captura de datos

En el método *onCreate* nos creamos el botón personalizado y capturamos las posiciones del banderín y la bola que se pasan desde la actividad *PrincipalAct*. En el caso que sea pulsado, se lanza el método *setOnLongClickListener()* donde cambiamos el color del botón para indicar que está siendo pulsado e iniciamos el *Handler* para la captura de datos.

```
1   @Override
2      public void onCreate(Bundle savedInstanceState) {
3          super.onCreate(savedInstanceState);
4          setContentView(R.layout.sensores_act);
```

```
5           this.setRequestedOrientation(ActivityInfo.
                SCREEN_ORIENTATION_PORTRAIT);
6
7           mMyButton = (BotonPersonalizado) findViewById(R.id.
                my_button);
8
9           mMyButton.setOnLongClickListener(new OnLongClickListener()
                {
10              public boolean onLongClick(View v) {
11                  boton = true;
12                  mHandler.post(mRunnable);
13                  mMyButton.setBackgroundDrawable(getResources().
                        getDrawable(R.drawable.button_on));
14                  return true;
15              }
16          });
17
18          mMyButton.setMain(this);
19
20          Bundle bundle = this.getIntent().getExtras();
21
22          if(bundle != null){
23              posicion_bola = bundle.getFloatArray("posicion_bola");
24              posicion_banderin = bundle.getFloatArray("
                    posicion_banderin");
25              golpes = bundle.getInt("golpes");
26          }
27      }
```

Código 2.5: Botón captura de datos

Fin de la captura de datos

Una vez que soltamos el botón, se lanza el método *cancelLongPress* en el que dejamos el color del botón en su situacion inicial y prepara la información para pasársela a la actividad *PrincipalAct*. En este método llamamos al método *calculaMovimiento* de la clase auxiliar *Operaciones* que devolverá los valores calculados según los parámetros generados por los sensores acelerómetro y de orientación y esos valores calculados se los sumamos a la posición donde se encontraba la bola anteriormente, así conseguimos la nueva posición de la bola.

```
1       public void cancelLongPress() {
2           boton = false;
3           mMyButton.setBackgroundDrawable(getResources().getDrawable(
                R.drawable.button_off));
4
5           //Creamos el Intent
```

```
6           Intent intent = new Intent(SensoresAct.this, PrincipalAct.
                class);
7
8           //Creamos la informacion a pasar entre actividades
9           Bundle b = new Bundle();
10
11          //Capturamos movimientos y se lo anadimos
12
13          ArrayList<Float> movimiento = Operaciones.calculaMovimiento
                (x1, y1, z1, x2, y2, z2);
14          posicion_bola[0] = posicion_bola[0] + movimiento.get(0);
15          posicion_bola[1] = posicion_bola[1] + movimiento.get(1);
16          posicion_bola[2] = posicion_bola[2] + movimiento.get(2);
17          b.putFloatArray("posicion_bola", posicion_bola);
18          b.putFloatArray("posicion_banderin", posicion_banderin);
19          b.putInt("golpes", golpes);
20
21          //Anadimos la informacion al intent
22          intent.putExtras(b);
23
24          //Iniciamos la nueva actividad
25          startActivity(intent);
26      }
```

Código 2.6: Fin captura de datos

onSensorChanged()

Este es el método más importante de toda la clase, vamos a capturar un nuevo dato siempre que existan cambios en los sensores (como vemos, sirve tanto para el sensor acelerómetro como el de orientación). Consiste básicamente en ir guardando las diferencias (entre el valor nuevo y el antiguo) en una lista para capturar los diferentes "vectores de movimiento", a esa lista le pasaremos una función de la clase auxiliar *Operaciones*. Por último, mostramos por pantalla el resultado para poder mejorar los disparos en un futuro.

```
1       public void onSensorChanged(SensorEvent event) {
2           if (boton) {
3               synchronized (this) {
4                   switch (event.sensor.getType()) {
5                   case Sensor.TYPE_ACCELEROMETER:
6
7                       ac_curX = event.values[0];
8                       ac_curY = event.values[1];
9                       ac_curZ = event.values[2];
10
11                      //Inicializamos los valores
```

```
12                  if (ac_prevX == 0 && ac_prevY == 0 && ac_prevZ
                        == 0) {
13                      ac_prevX = ac_curX;
14                      ac_prevY = ac_curY;
15                      ac_prevZ = ac_curZ;
16                  }
17                      //Hallamos diferencias con el valor anterior
18                      float ac_difX = ac_curX - ac_prevX;
19                      float ac_difY = ac_curY - ac_prevY;
20                      float ac_difZ = ac_curZ - ac_prevZ;
21
22                      //Insertamos en las listas
23                      ac_listaX.add(ac_difX);
24                      ac_listaY.add(ac_difY);
25                      ac_listaZ.add(ac_difZ);
26
27                      //Actualizamos ultimo valor
28                      ac_prevX = ac_curX;
29                      ac_prevY = ac_curY;
30                      ac_prevZ = ac_curZ;
31
32                  break;
33              case Sensor.TYPE_ORIENTATION:
34
35                  // event.values[0] = Circular
36                  or_curX = event.values[0];
37                  // event.values[1] = Ecuador
38                  or_curY = event.values[1];
39                  // event.values[2] = Meridiano
40                  or_curZ = event.values[2];
41
42                  //Inicializamos los valores
43                  if (or_prevX == 0 && or_prevY == 0 && or_prevZ
                        == 0) {
44                      or_prevX = or_curX;
45                      or_prevY = or_curY;
46                      or_prevZ = or_curZ;
47                  }
48
49                      //Hallamos diferencias con el valor anterior
50                      float or_difX = or_curX - or_prevX;
51                      float or_difY = or_curY - or_prevY;
52                      float or_difZ = or_curZ - or_prevZ;
53
54                      //Insertamos en las listas
55                      or_listaX.add(or_curX);
56                      or_listaY.add(or_curY);
57                      or_listaZ.add(or_curZ);
58
59                      //Actualizamos ultimo valor
```

```
60                      or_prevX = or_curX;
61                      or_prevY = or_curY;
62                      or_prevZ = or_curZ;
63                      break;
64                  }
65              }
66          }
67
68          x1 = Operaciones.obtieneAceleracion(ac_listaX);
69          y1 = Operaciones.obtieneAceleracion(ac_listaY);
70          z1 = Operaciones.obtieneAceleracion(ac_listaZ);
71          x2 = Operaciones.obtieneOrientacion(or_listaX);
72          y2 = Operaciones.obtieneOrientacion(or_listaY);
73          z2 = Operaciones.obtieneOrientacion(or_listaZ);
74
75          //Mostramos los datos
76          ((TextView) findViewById(R.id.txtAccX)).setText("
                Acelerometro X: "+ x1 + " m/s2");
77          ((TextView) findViewById(R.id.txtAccY)).setText("
                Acelerometro Y: "+ y1 + " m/s2");
78          ((TextView) findViewById(R.id.txtAccZ)).setText("
                Acelerometro Z: "+ z1 + " m/s2");
79          ((TextView) findViewById(R.id.txtOrX)).setText("Orientacion
                 Circular: "+ or_curX + "o");
80          ((TextView) findViewById(R.id.txtOrY)).setText("Orientacion
                 Ecuador: "+ or_curY + "o");
81          ((TextView) findViewById(R.id.txtOrZ)).setText("Orientacion
                 Meridiano: "+ or_curZ + "o");
82
83          }
```

Código 2.7: Capturar nuevo dato

2.3.4. Operaciones

Esta clase es un conjunto de funciones auxiliares a la clase *PrincipalAct*. Vamos a ir explicando una a una.

estaDentro()

Esta función sirve para comprobar si la bola y el hoyo están en la misma posición, comparando sus tres coordenadas.

```
1       public static boolean estaDentro(EntityData e1, EntityData e2)
            {
2           boolean p = false;
3
4           if ((e1.getLocation().x == e2.getLocation().x)
5                   && (e1.getLocation().y == e2.getLocation().y)
```

```
6                 && (e1.getLocation().z == e2.getLocation().z)) {
7             p = true;
8         }
9         return p;
10
11    }
```

Código 2.8: Comprobar bola y hoyo

obtieneAceleracion()

En esta sumamos todas las diferencias de lo captado por el acelerómetro que se ha insertado previamente en una lista. El primer valor no es interesante al ser 0 debido al inicio de la captura de datos. Lo pasamos a valor absoluto porque nos interesa el módulo de la aceleración.

```
1     public static Float obtieneAceleracion(ArrayList<Float> lista){
2         float acu = 0;
3         //El primer valor no es interesante debido a que va a sumar
              en el acumulador un 0.
4         for(int i = 1; i < lista.size(); i++){
5             acu = lista.get(i) + acu;
6         }
7         return acu;
8     }
```

Código 2.9: Acelerómetro

obtieneOrientacion()

En esta con la implementación actual, devolvemos el último valor que se obtiene de todo lo capturado por la orientación ya que tras las pruebas realizadas funciona bastante bien. No obstante, en futuras iteraciones se podría conseguir mayor sensibilidad partiendo de la implementación que presentamos. Serviría más que nada para añadir efectos a la bola, por ejemplo.

```
1     public static Float obtieneOrientacion(ArrayList<Float> lista){
2         float dif = 0;
3         float f1 = 0;
4         float f2 = 0;
5         if (lista.size() != 0) {
6             f1 = lista.get(0);
7             f2 = lista.get(lista.size() - 1);
8             dif = f2 - f1;
9         }
10        return f2;
11    }
```

Código 2.10: Obtiene orientación

calculaMovimiento()

Es la función central del juego, recibe seis parámetros. Tres son los pertenecientes a la aceleración (ac_x, ac_y, ac_z) y los otros tres a la orientación (or_x, or_y, or_z). El objetivo con estos seis parámetros es devolver la posición que ha de añadirse a la actual. Es decir, pasar de aceleración y orientación a posición.

Para la orientación vamos a utilizar un valor que varíe entre {-1, 1}. Tendrá el valor "1"para indicar que se va a mover respecto a esa coordenada lo máximo. Por el contrario, el valor "-1", indica que retrocederá lo máximo sobre esa coordenada.

La orientación de la coordenada x y la z la vamos a obtener a través de la orientación circular (or_x). Tras hacer pruebas concluimos en que:

Para la x, cuando esté en los 90º va a haber un movimiento máximo hacia esa posición. Cuando marque 270º va a haber un movimiento máximo negativo hacia esa posición. Y por ello, dividimos en 4 los 360º de la circunferencia. Utilizamos la constante que va a valer "1/90", esto nos va a servir para que, según el grado de la orientación circular, el valor devuelto esté comprendido entre los citados anteriormente: {-1, 1}. Es decir:

- Desde los 0º hasta los 90º: Para 0º vamos a devolver: 0. Para 90º vamos a devolver: 1. Al aumentar los grados, vamos tendiendo a 1.
- Desde los 90º hasta los 180º: Para 90º vamos a devolver: 1. Para 180º vamos a devolver: 0. Al aumentar los grados, vamos tendiendo a 0.
- Desde los 180º hasta los 270º: Para 180º vamos a devolver: 0. Para 270º vamos a devolver: -1. Al aumentar los grados, vamos tendiendo a -1.
- Desde los 270º hasta los 360º: Para 270º vamos a devolver: -1. Para 360º vamos a devolver: 0. Al aumentar los grados, vamos tendiendo a 0.

```
1          //Calculo preliminar x respecto de la orientacion circular
2          if (or_x >= 0 && or_x < 90) {
3
4              x = 0f + (cte * (or_x - 0));
5
6          } else if (or_x >= 90 && or_x < 180) {
7
```

```
8             x = 1f - (cte * (or_x - 90));
9
10        } else if (or_x >= 180 && or_x < 270) {
11
12            x = 0f - (cte * (or_x - 180));
13
14        } else {
15
16            x = -1f + (cte * (or_x - 270));
17
18        }
```

Código 2.11: Calcula movimiento

Para la z, cuando esté en los 0º va a haber un movimiento máximo hacia esa posición. Cuando marque 180º va a haber un movimiento máximo negativo hacia esa posición. Es decir:

- Desde los 0º hasta los 90º: Para 0º vamos a devolver: 1. Para 90º vamos a devolver: 0. Al aumentar los grados, vamos tendiendo a 0.
- Desde los 90º hasta los 180º: Para 90º vamos a devolver: 0. Para 180º vamos a devolver: -1. Al aumentar los grados, vamos tendiendo a -1.
- Desde los 180º hasta los 270º: Para 180º vamos a devolver: -1. Para 270º vamos a devolver: 0. Al aumentar los grados, vamos tendiendo a 0.
- Desde los 270º hasta los 360º: Para 270º vamos a devolver: 0. Para 360º vamos a devolver: 1. Al aumentar los grados, vamos tendiendo a 1.

```
1         //Calculo preliminar z respecto de la orientacion circular
2         if (or_x >= 0 && or_x < 90) {
3
4             z = 1f - (cte * (or_x - 0));
5
6         } else if (or_x >= 90 && or_x < 180) {
7
8             z = 0f - (cte * (or_x - 90));
9
10        } else if (or_x >= 180 && or_x < 270) {
11
12            z = -1f + (cte * (or_x - 180));
13
14        } else {
15
16            z = 0f + (cte * (or_x - 270));
```

```
17
18          }
```

Código 2.12: Cálculo orientación circular

Para la y, utilizamos la orientación ecuador. De nuevo el valor que devuelve va a estar comprendido entre {-1, 1}. Para valores extremos, cuando valga -90º va a valer "-1", cuando valga 90º valdrá "1". En el caso de que no esté comprendido entre esos grados, no lo tomamos como tiro válido.

```
1  //Calculo preliminar y respecto de la orientacion ecuador
2          if (or_y >= -90 && or_y < 90) {
3
4              y = cte * or_y;
5
6          }else{
7              y = 0.0f;
8          }
```

Código 2.13: Orientación ecuador

Una vez tengamos la orientación para las tres coordenadas, vamos a añadirle la aceleración. Ahora vamos a usar una nueva variable auxiliar que denominamos "fm", sus valores van a estar comprendidos entre {0, 1}. Cuando la orientación circular sea de 0º, valdrá "1", cuando tienda a 90º o -90º valdrá "0". Esto es, cuando la orientación circular es 0º, el tiro va a desplazarse totalmente en los parámetros x y z, pero en el parámetro y (altura) no va a verse modificado. Sin embargo, cuando sea de 90º o -90º, va a tomar toda la altura y en cambio para los parámetros x y z no va a aumentar.

```
1  // Calculamos movimiento de x y z segun y. abs_y va del 0 al 90.
2          // Si abs_y tiende a 0, factor de multiplicacion de x y z
               tiende a 1.
3          // Si abs_y tiende a 90, factor de multiplicacion de x y z
               tiende a 0.
4          Float abs_y = Math.abs(or_y);
5          if (abs_y < 90) {
6
7              Float fm = cte * (90 - abs_y);
8              x = fm * x * ac_y;
9              y = y * ac_z;
10             z = fm * z * ac_y;
11         } else {
12
13             x = 0.0f;
14             y = 0.0f;
15             z = 0.0f;
16         }
```

```
17
18          movimiento.add(x);
19          movimiento.add(y);
20          movimiento.add(z);
21
22          return movimiento;
```

Código 2.14: Orientación coordenadas

2.3.5. FinalAct

Es la pantalla final, en ella se informa del fin de la partida. Bien porque se haya introducido la bola en el hoyo o por haber dado más de 10 golpes.

```
1  public void onCreate(Bundle savedInstanceState) {
2          super.onCreate(savedInstanceState);
3          setContentView(R.layout.final_act);
4          tv_final = (TextView) findViewById(R.id.tv_final);
5
6           Bundle bundle = this.getIntent().getExtras();
7           if (bundle != null){
8               Integer val = bundle.getInt("fin");
9               Integer golpes = bundle.getInt("golpes");
10              if(val == 0){
11                  tv_final.setText("Fin de la partida. Has dado ms de
                        10 golpes.");
12              }else{
13                  tv_final.setText("Enhorabuena! Has metido la bola
                        en el hoyo. Has necesitado dar " + golpes + "
                        golpes.");
14              }
15          }
16      }
```

Código 2.15: Fin del juego

2.3.6. Problemas encontrados

Durante el desarrollo del juego, nos hemos encontrado tres problemas básicamente.

El primero que como ya comentamos en las pruebas, hemos resuelto, era que al dar más de 10 golpes el juego fallaba debido a un error en el reciclado de los bitmaps (gráficos). Lo solventamos poniendo en 10 el número máximo de golpes permitidos.

El segundo problema fue el que al abandonar la actividad y volver, desaparecen los elementos. Eso si, en el momento que vuelves a golpear, aparecen de nuevo los elementos. Por más horas que le hemos dedicado,

no hemos conseguido arreglar este error y es probable que pueda ser por la librería utilizada.

El último es sobre la navegabilidad de la aplicación, si le dábamos hacia atrás una vez comenzado y llevando ya varios golpes dados, el juego se vuelve inestable. Este problema lo hemos resuelto parcialmente evitando que se pueda navegar hacia atrás y sólo te permita seguir adelante.

CAPÍTULO 3

Análisis de antecedentes y aportación realizada

En primer lugar en este apartado debemos mencionar que en la actualidad no existe ningún antecedente de videojuego para Android en el que se junte el uso del acelerómetro, la cámara (realidad aumentada) y la orientación. Como consecuencia este proyecto supone una innovación en su ámbito dentro de dicha Universidad y resulta imposible la comparación de este proyecto con cualquier otro de carácter similar.

En cambio sí que es posible hallar un extenso número de aplicaciones en el que se usan las tres por separado.

Para la realización del proyecto comenzamos buscando información acerca de aplicaciones de realidad aumentada en Android. Tras analizar todas las alternativas, intentamos desplegar una aplicación sencilla con cada librería, nos decantamos por utilizar la libreria *Look!* debido al excelente tutorial para incluirla en una aplicación Android que encontramos en su documentación, como también por las diferentes aplicaciones que han lanzado sus creadores. Gracias a esto, pudimos, poco a poco, empezar a aprender la extensa funcionalidad que aporta la librería.

Una vez controlada la realidad aumentada, pasamos a investigar acerca del acelerómetro. No tuvimos demasiados problemas ya que en la red se puede encontrar bastante código Android para capturar los parámetros del acelerómetro. También utilizamos varias aplicaciones tales como *My Android Sensors* o *CPU-Z* para entender realmente cómo se comportaba cada parámetro del acelerómetro según los movimientos que realizábamos con el móvil, con estas aplicaciones nos cercioramos de que la aplicación de acelerómetro que habíamos desarrollado, en efecto, se comportaba igual que aplicaciones ya asentadas en el *Play Store* de Google. A partir de este punto es donde empieza nuestra aportación tras lo que había anteriormente.

Comenzamos insertando un botón que capturase, mientras estuviese

siendo pulsado, los valores del acelerómetro cada un cierto intervalo de tiempo. Tras muchas pruebas, le dimos un tratamiento a esos datos, como si se estuviese golpeando a una bola de golf, intentando un primer acercamiento a la funcionalidad del mando de la *Wii*.

Una vez controlada la parte de realidad aumentada y la de acelerómetro, las fusionamos en una misma aplicación y comenzamos a ver el comportamiento de la bola sobre el terreno de juego virtual. No quedamos del todo satisfechos y tras investigar concluimos en que tendríamos que contar también no sólo con los datos del acelerometro sino también del sensor de orientación. Tras realizar los mismos pasos que con el sensor de acelerómetro (aplicación independiente, pruebas en otras aplicaciones y tratamiento de datos para asemejarlo al problema que estábamos tratando) lo incluimos en la aplicación obteniendo unos resultados mucho más satisfactorios.

CAPÍTULO 4

Análisis temporal y costes de desarrollo

4.1– Análisis temporal

En este apartado hemos analizado el tiempo que hemos dedicado a la realización del proyecto. Especificaremos cuanto tiempo hemos estado investigando acerca de todas las cuestiones que abordan la aplicación desarrollada en este proyecto, el tiempo dedicado en realizar la memoria, los distintos manuales y las pruebas correspondientes para comprobar que nuestra aplicación funciona correctamente.

La realización del proyecto la hemos dividido en tres partes: investigación, desarrollo y finalización.

- La primera parte del proyecto la dedicamos exclusivamente a la investigación, esto fue lo más extenso ya que tuvimos que estudiar muchos aspectos relacionados con la realidad aumentada, la orientación y el acelerómetro. En esta parte también hemos investigado acerca de diferentes librerías con las que poder desarrollar la aplicación. En la investigación podríamos dividir el tiempo invertido en cada uno de las diferentes prestaciones requeridas por el videojuego.

- En el desarrollo comenzamos a montar la aplicación y probar las diferentes librerías estudiadas anteriormente. Comenzamos desarrollando el proyecto con una aplicación sencilla para ir aprendiendo a manejarnos con la librería de realidad aumentada. Por otro lado también realizamos otra aplicación para el uso del acelerómetro del móvil. Más adelante, unimos realidad aumentada y el acelerómetro, al cabo del tiempo nos dimos cuenta que sería mejor usar la orientación como mejora de ésta, finalmente lo incluímos y requirió un tiempo extra que no esperábamos.

- Con la finalización del proyecto pulimos la aplicación, completamos

la memoria, e hicimos las pruebas pertinentes para buscar los errores con sus respectivas soluciones.

En el siguiente cuadro podemos ver el analisis temporal de nuestro proyecto.

	Tiempo(Horas)
Investigación(gps)	150
Investigación(orientación)	180
Investigación(realidad aumentada)	120
Investigación(acelerómetro)	150
Desarrollo	450
Documentación	150
Total	1200

4.2– Costes de desarrollo

En este proyecto no ha habido costes de desarrollo ya que disponíamos de un dispositivo Android para probar la aplicación y ordenadores para su desarrollo. Si hubiéramos carecido de éstos el coste del proyecto podría haber ascendido a los 2000 euros como mínimo. El dispositivo Android que vamos a utilizar es el HTC One X. Con el emulador de Android no podíamos hacer pruebas debido a que necesitábamos el uso de cámara y de diferentes sensores que sí están presentes en el móvil.

Las especificaciones completas del terminal son las siguientes:

- Medidas: 134.4 x 69.9 x 8.9 mm.
- Peso: 130 g.
- Batería: Standard, Li-Po 1800 mAh.
- Pantalla: 4.7 pulgadas con 720 x 1280 píxeles.
- Procesador: Nvidia Tegra 3 quad-core 1.5GHz.
- Cámara trasera: 8 MP, 3264x2448 píxeles, autofocus, flash LED, captura de vídeo y fotos simultáneos, vídeo 1080p a 30fps estéreo.
- Cámara frontal: 1.3MP a 720p.
- Sistema operativo: Android, versión 4.1.1 Ice Cream Sandwich.
- Almacenamiento interno: 32 GB.
- RAM: 1 GB.
- Red: GSM 850 / 900 / 1800 / 1900 - HSDPA 850 / 900 / 1900 / 2100.

Figura 4.1: HTC One X

Es un terminal de gama alta y, aunque ya hay en el mercado móviles superiores, tiene potencia de sobra para realizar este proyecto. El coste del software utilizado es nulo al tratarse de herramientas gratuitas.

4.3– Software utilizado

En esta sección vemos las herramientas utilizadas durante el desarrollo de VirtualGolf. Para cada herramienta se ofrece una pequeña descripción y se adjuntan las razones por las cuales ha sido elegida para el desarrollo frente a sus competidoras.

- **Eclipse**

  Es un Entorno Integrado de Desarrollo, del inglés Integrated Development Environment(IDE), para todo tipo de aplicaciones libres. Inicialmente desarrollado por IBM, y actualmente gestionado por la Fundación Eclipse. Herramienta para el programador desarrollada principalmente para el desarrollo de aplicaciones Java. Es además el entorno más utilizado por los desarrolladores de aplicaciones Android, por ésto ha sido elegido para la realización de este proyecto.

- **Blender**

  Blender es un programa informático multiplataforma, dedicado especialmente al modelado, animación y creación de gráficos tridimen-

sionales. El programa fue inicialmente distribuido de forma gratuita pero sin el código fuente, con un manual disponible para la venta, aunque posteriormente pasó a ser software libre. Actualmente es compatible con todas las versiones de Windows, Mac OS X, GNU/-Linux, Solaris, FreeBSD e IRIX.

CAPÍTULO 5

Comparación con otras alternativas

Hemos encontrado varias alterantivas similares a VirtualGolf pero ninguna de ellas es comparable con nuestra aplicación. Con las siguientes alternativas vemos como podemos encontrar las diferentes técnicas que usamos en nuestro proyecto y como se les da uso en estos diferentes videojuegos de móvil y de la videoconsola *Wii*.

- Como alternativas podemos encontrar juegos como *Let's golf!* que no usa ni realidad aumentada ni el acelerómetro pero la finalidad del juego es la misma, meter la bola en un hoyo ubicado en un campo de golf. Al ser el objetivo del juego el mismo, podemos ver similitudes en el desarrollo del juego, una de ellas sería la asignación de puntos o el tiro de la bola hacia el hoyo. En relación con los gráficos de este juego podríamos compararlo con nuestra aplicación.

Figura 5.1: Let's Golf

- Una clara alternativa a la aplicación que podría compararse con Vir-

tual Golf pero el soporte es totalmente diferente es el juego *Tiger Woods* de la *Nintendo Wii*, en la que se usa un mando como palo de golf con el que virtualmente se golpea la bola para intentar introducirla en el hoyo, un hoyo que a diferencia de Virtual Golf encontramos en un campo de golf simulado en la pantalla. En nuestra aplicación de forma parecida usamos el mando como palo pero al ser una aplicación para un dispositivo móvil, éste hace la función del mando usando el acelerómetro y la orientación.

Figura 5.2: Tiger Woods

- Otra alternativa a VirtualGolf que encontramos en el mercado es *Parallel Kingdom AOT* en el que el objetivo del juego es buscar aventuras para sumar experiencias. Lo bueno es que si alguien más se encuentra en la zona de influencia del jugador, va a poder ver su casa y sus construcciones, e incluso atacarlas o declararle la guerra. El sistema en el que está desarrollado es el mismo que nuestra aplicación y usa la geolocalización, además de la realidad aumentada. Aunque el juego en sí sea bastante diferente podemos compararlos en base a sus características.

Figura 5.3: Parallel Kingdom AOT

Definiremos como el más parecido a nuestro proyecto el juego *Tiger Woods*, ya que nuestro objetivo en un principio era que el móvil pudiera ser usado como el mando de la videoconsola *Wii*. En relación a esta videoconsola podemos verla como alternativa general a lo que VirtualGolf nos permite realizar con un dispositivo móvil al que se lo instalemos.

En definitiva, no podemos comparar en igualdad de condiciones y de forma imparcial algún otro videojuego con Virtual Golf ya que sus características no las iguala ninguna aplicación que podamos encontrar ahora mismo en el mercado. Esta es una de las principales razones por las que nos encantó la idea inicial de VirtualGolf porque nos ha dado opción a investigar sobre las técnicas que usamos de forma conjunta viéndolas por separado en otros juegos.

CAPÍTULO 6

Pruebas

Una vez completado el desarrollo de una aplicación es necesario llevar a cabo una fase de pruebas que, aunque no garanticen totalmente el correcto funcionamiento de la aplicación, podemos decir que al menos en las condiciones que han sido desarrolladas ha funcionado como esperábamos.

El periodo de pruebas ha sido interesante a la par que resolutorio, ya que encontramos varios problemas que conseguimos arreglar gracias a estas pruebas. Uno de los problemas encontrados fue que al realizar un número superior de 12 golpes la aplicación lanzaba un error debido a que Android no gestiona bien el reciclado de Bitmaps (en nuestro caso la bola y el hoyo) en la memoria, por lo que como propuesta y solución final hemos puesto un límite máximo de golpes de 10 para a la vez de subsanar el error, darle mayor dificultad al juego.

Las pruebas que hemos realizado podemos dividirlas en dos secciones que explicamos a continuación: pruebas unitarias, en las que probábamos cada parte de la aplicación por separado y de integración en las que jugábamos al juego por completo. Hemos representado en las pruebas el hoyo como un cubo para una mayor facilidad visual.

6.1– Pruebas unitarias

Como pruebas unitarias tenemos las siguientes:

- **Pruebas de localización:** En las pruebas con la localización queríamos probar cómo funcionaba la libreria utilizada respecto a la inserción de tres coordenadas. Primero posicionamos la pelota en {5, 0, 4} y el banderín en {5, 3, 4}.

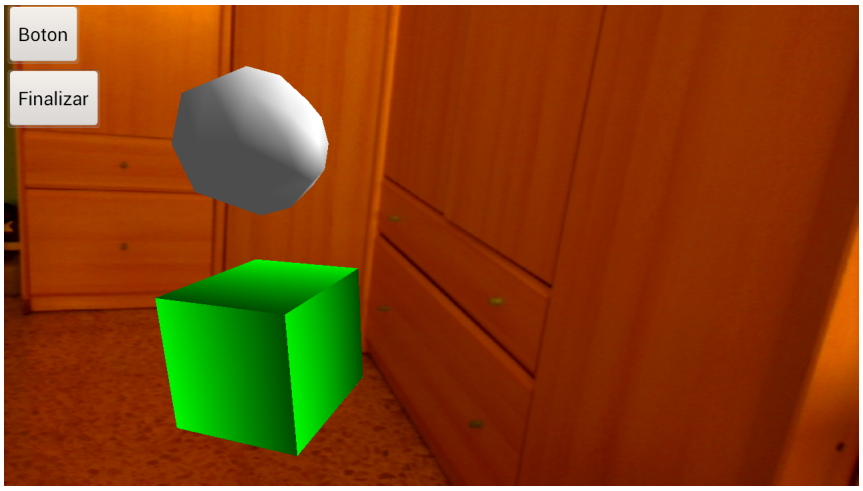

Figura 6.1: Posición inicial

Vamos a ir modificando los valores de x, y, z de la pelota y vamos a tomar como referencia el banderín.

Pelota en {9, 0, 4}:

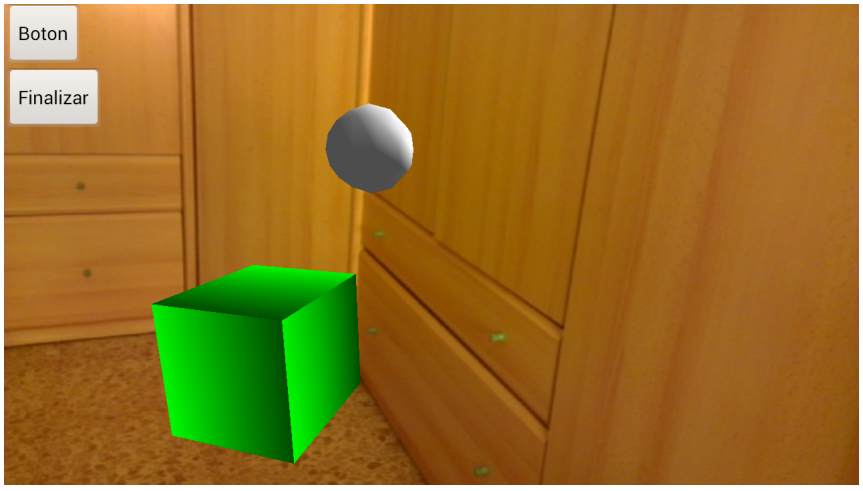

Figura 6.2: Desplazamiento de la X

La pelota aparece desplazada hacia el fondo en el eje de las x.

Pelota en {3, 0, 4}:

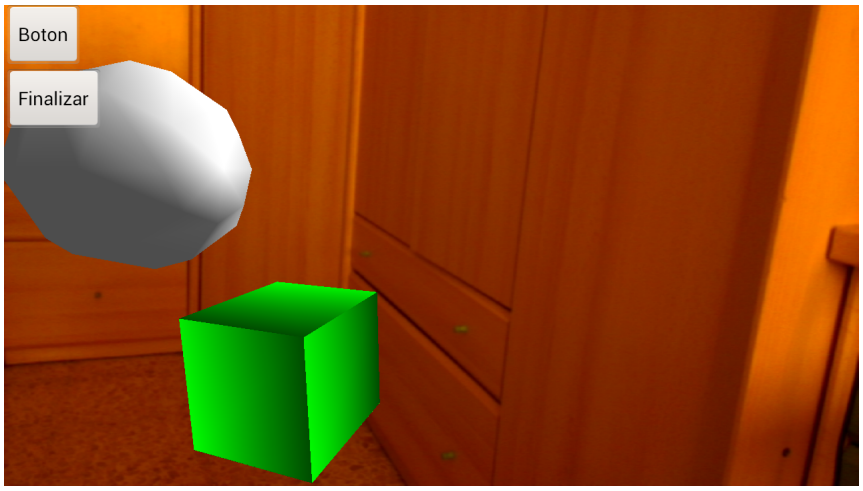

Figura 6.3: Desplazamiento de la X

Como esperábamos ahora está desplazada hacia nosotros respecto al eje de las x.

Pelota en {5, 7, 4}:

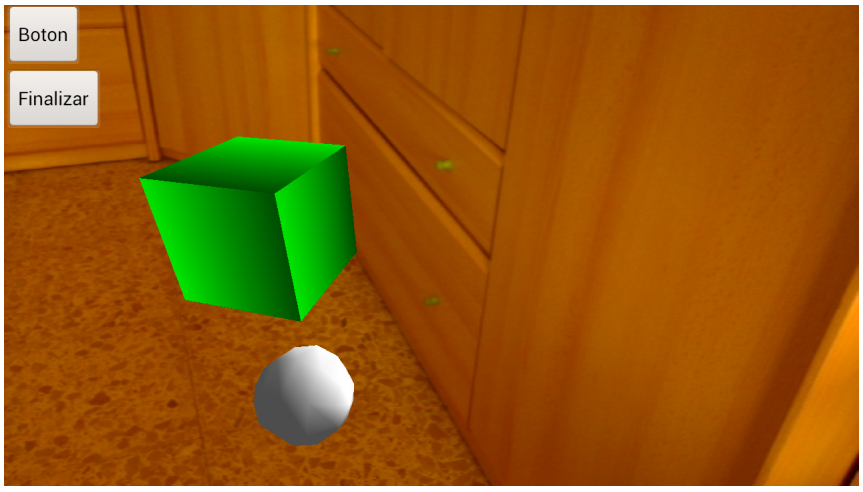

Figura 6.4: Desplazamiento de la Y

Respecto al eje de las y ya podemos comprobar comparando la bola y la pelota desde la posición inicial, como al aumentar la coordenada en el eje de la y, disminuimos la altura.

Pelota en {5, -1, 4}:

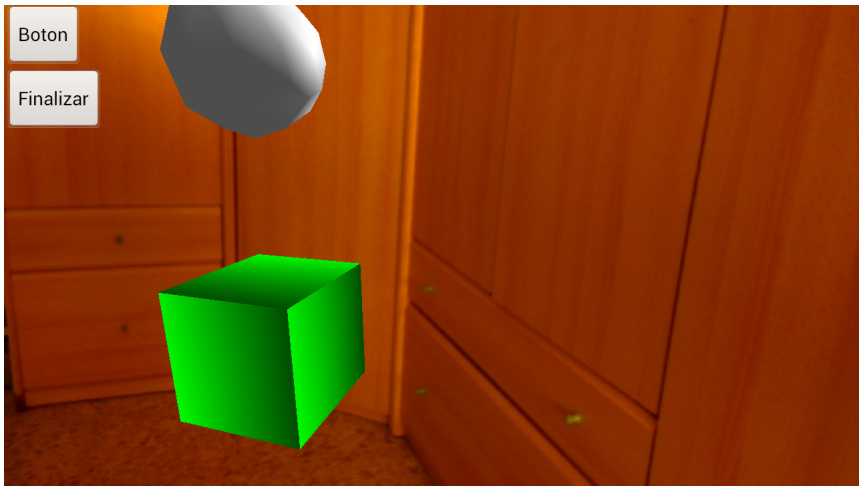

Figura 6.5: Desplazamiento de la Y

Por el contrario, al disminuir la y, aumentamos la altura.

Pelota en {5, 0, 7}:

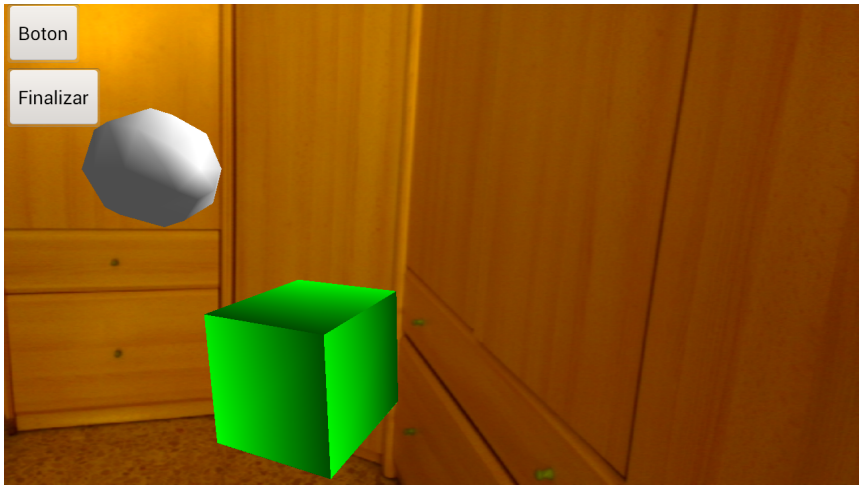

Figura 6.6: Desplazamiento de la Z

Respecto a la coordenada z, al aumentarla, vemos como se desplaza a la izquieda respecto al cubo en el eje de las z.

Pelota en {5, 0, 2}:

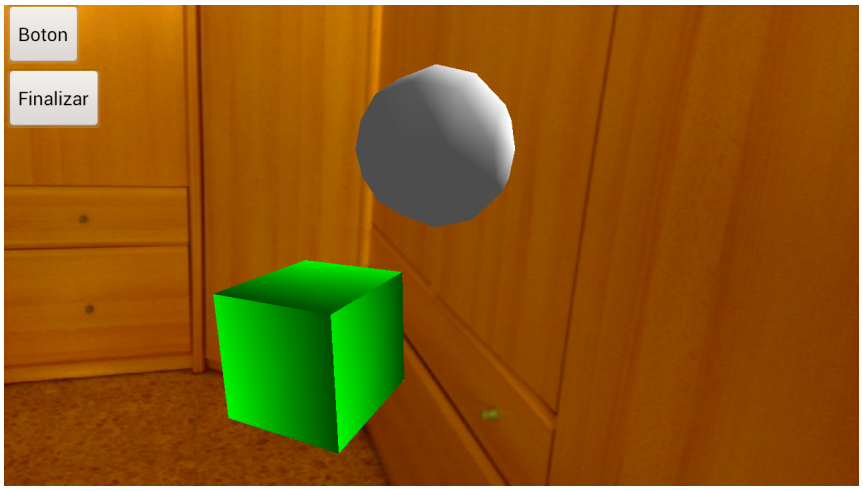

Figura 6.7: Desplazamiento de la Z

Sin embargo al disminuir, queda desplazada a la derecha respecto al banderín.

- **Pruebas de orientación:** Las pruebas de orientación consistieron en comprobar los valores devueltos por nuestra aplicación de orientación y compararlos con los de otras aplicaciones, resultando satisfactorio.

- **Pruebas de realidad aumentada:** De realidad aumentada realizamos pruebas respecto a todos los diferentes gráficos que la librería permite representar (cubos, esferas, anillos, etcétera). Así pudimos conseguir tanto el aspecto final de la pelota como del banderín.

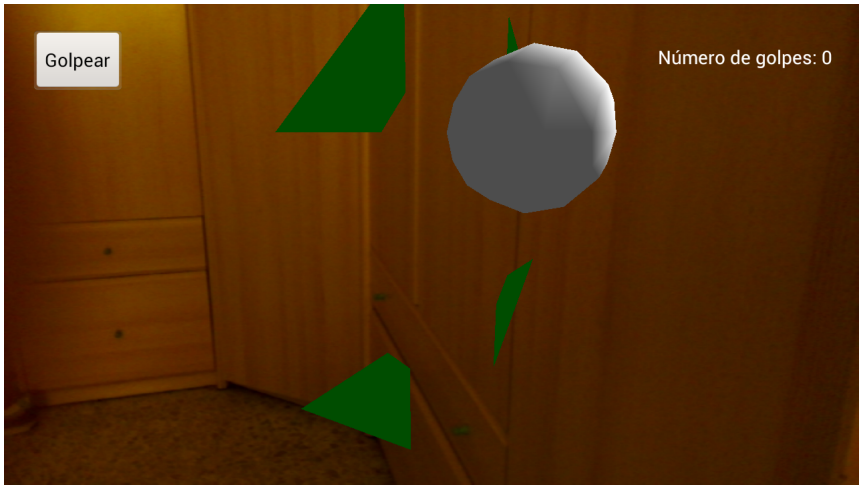

Figura 6.8: Bola y banderín

- **Pruebas de acelerómetro:** Igual que con las pruebas de orientación, las de acelerómetro las comparamos y el resultado fue el esperado respecto a otras aplicaciones.

  También a la vez que probamos el acelerómetro y la orientación, probamos el botón que sólo captura datos cuando se mantiene pulsado, funcionando como se esperaba en la implementación.

6.2– Pruebas de integración

En estas pruebas comprobamos que la aplicación funciona como esperamos, en primer lugar en cada una de las pruebas realizadas iniciamos la aplicación, en este punto de inicialización nunca hemos tenido problemas. Al comenzar a jugar observamos que al llegar a 12 toques falla la aplicación, pero lo resolvimos como bien explicamos al comienzo de esta sección. Ahora vamos a comenzar uniendo las pruebas de localización y orientación, multiplicando por un factor constante para la comprobación de que la bola se va moviendo hacia el lugar que deseamos orientando el móvil.

- **Pruebas de localización y orientación:** Primero posicionamos la pelota en {5, 0, 4} y el banderín en {5, 3, 4}.

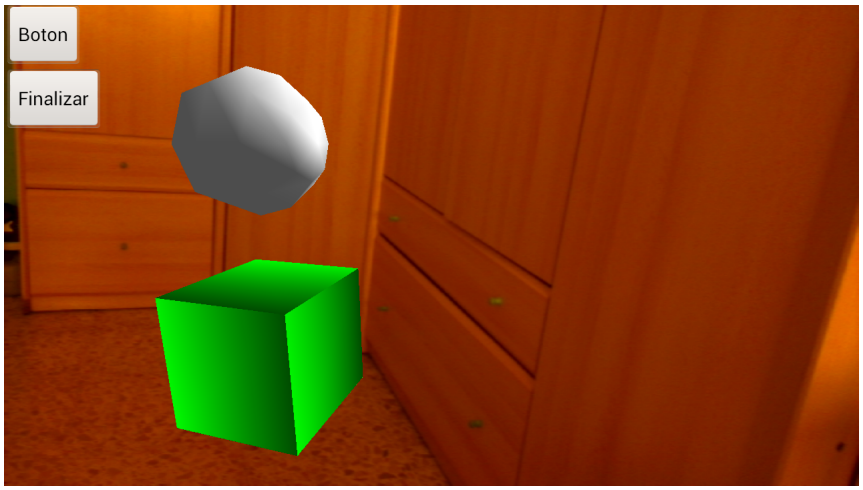

Figura 6.9: Posición inicial

Intentamos mover la bola hacia el fondo a la derecha. Resultando satisfactorio al segundo intento.

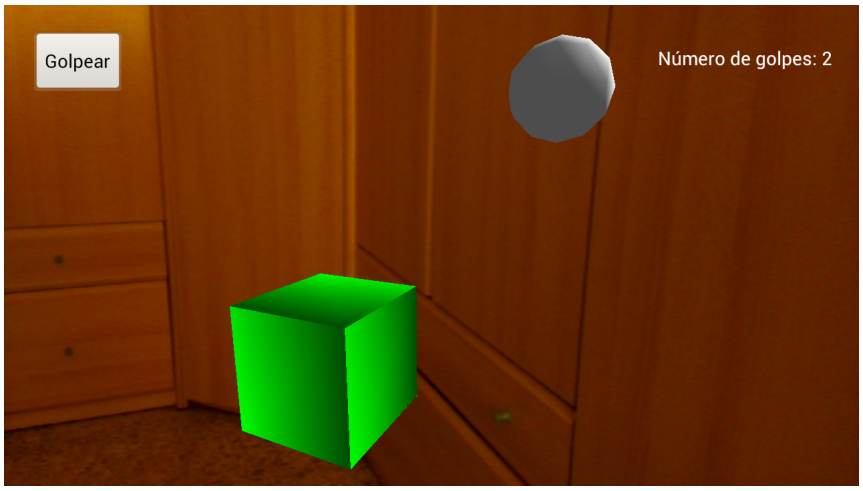

Figura 6.10: Posición final 1

Movemos hacia abajo del hoyo, tras de nuevo dos intentos, conseguimos el objetivo.

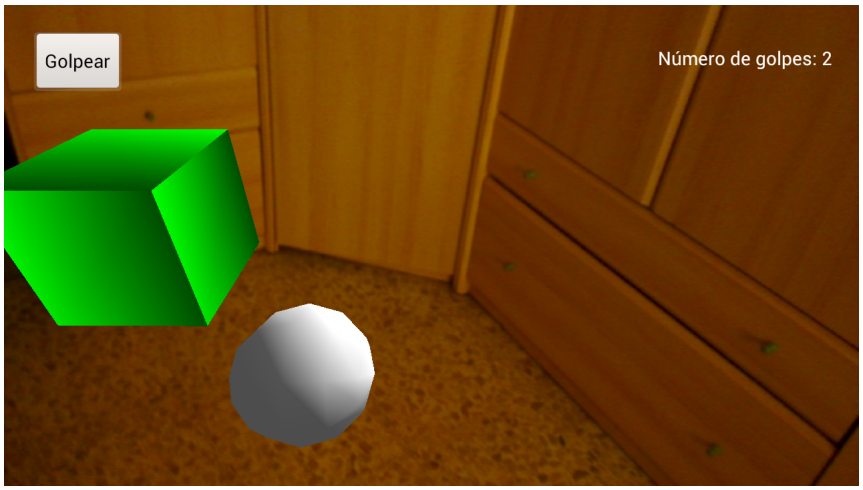

Figura 6.11: Posición final 2

- **Pruebas de localización, orientación y acelerómetro:**

  Añadimos el acelerómetro para comprobar que el comportamiento sea el esperado. Tuvimos que modificar el planteamiento inicial al no obtener los resultados esperados. Tras la modificación en el código, conseguimos un resultado satisfactorio. En este ejemplo, intentamos llevar la bola lo más alta posible en varios golpes.

Figura 6.12: Posición final 3

- **Pruebas de finalización del juego:** El juego termina para bien o para mal. Si introduces la bola en el hoyo antes de 10 golpes, llegaremos a la pantalla de finalización del juego. Se felicitará al usuario y mostrará el número de golpes que hemos necesitado. Tras muchos intentos, finalmente conseguimos introducirla en el hoyo.

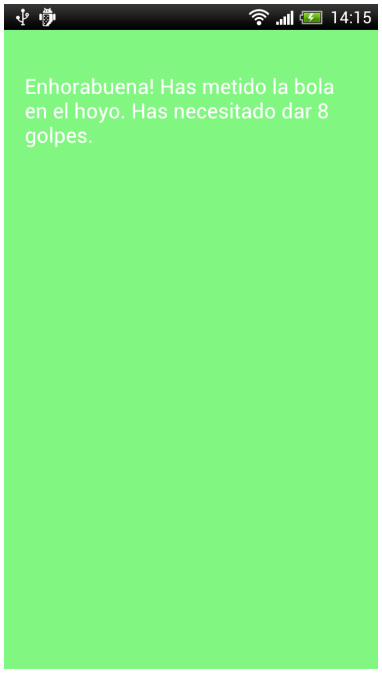

Figura 6.13: Finalización tras introducción en el hoyo

Sin embargo, ha sido muy fácil dar más de 10 golpes y comprobamos que satisfactoriamente al décimo golpe nos lanzaba a la pantalla de fin del juego con el siguiente mensaje.

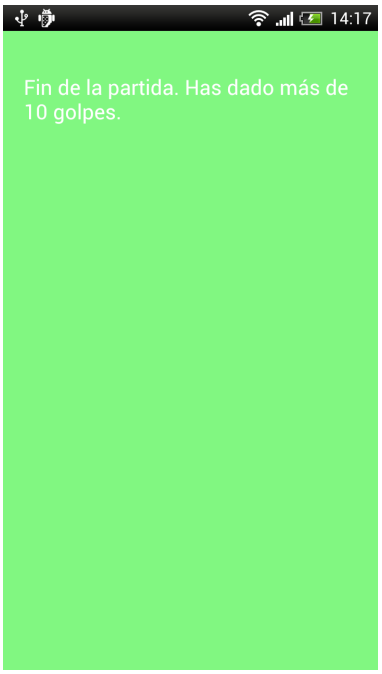

Figura 6.14: Finalización tras dar más de 10 golpes

CAPÍTULO 7

Manual

Finalizada la implementación de la aplicación y comprobando su correcto funcionamiento, es necesario redactar un manual de uso, de forma que, aun teniendo un uso intuitivo, cualquier usuario sea capaz de familiarizarse fácilmente con la interfaz y su correcto funcionamiento.

7.1– Requisitos

Para poder jugar a la aplicación tan sólo será necesario el uso de un móvil con sistema operativo Android, que posea una cámara para poder utilizar la Realidad Aumentada y tener activados los sensores acelerómetro y orientación.

7.2– Funcionamiento

Una vez iniciada la aplicación, vamos a visualizar la bola, el banderín, un botón para golpear la bola y el número de golpes que llevamos.

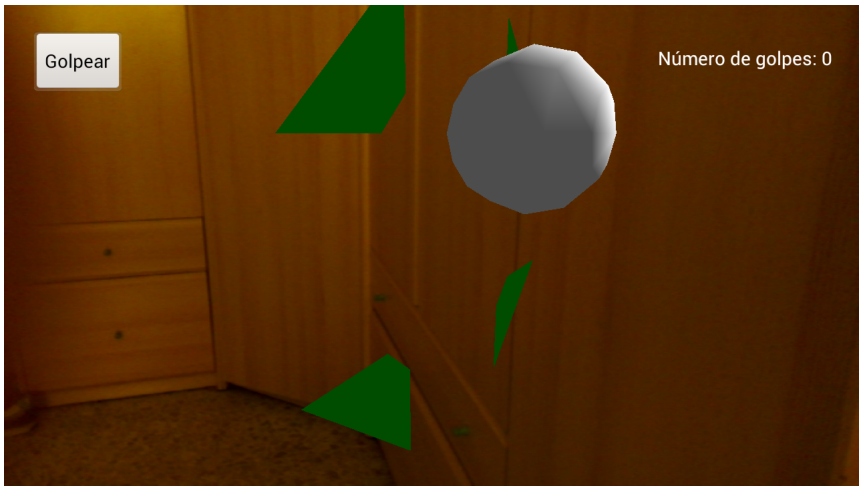

Figura 7.1: Pantalla inicial

Al pulsar en golpear, se muestra la siguiente pantalla. El botón rojo, cuando esté pulsado se pondrá de color verde e indicará que los sensores están tomando datos. Abajo veremos en todo momento lo que capturan los dos sensores. Cuando dejamos de pulsar el botón, nos vuelve a la pantalla principal y si hemos realizado un tiro satisfactorio, la pelota se habrá movido.

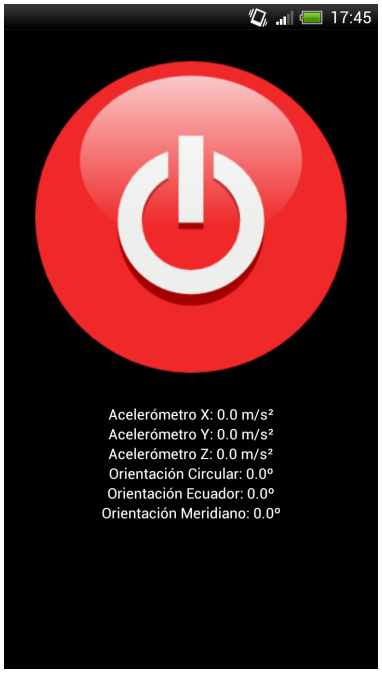

Figura 7.2: Pantalla golpear sin tener pulsado el botón

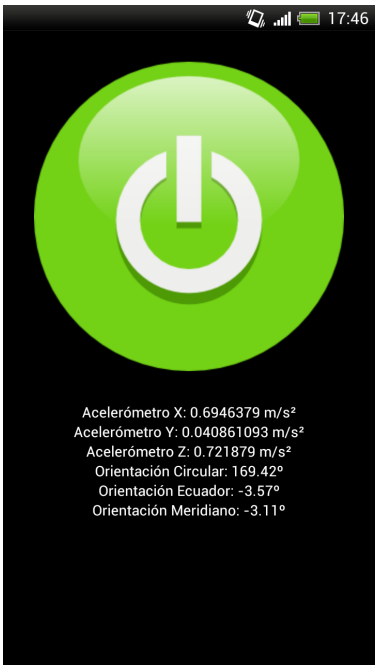

Figura 7.3: Pantalla golpear pulsando el botón

En el caso de insertar la bola en el banderín, esta es la pantalla de finalización que se nos muestra.

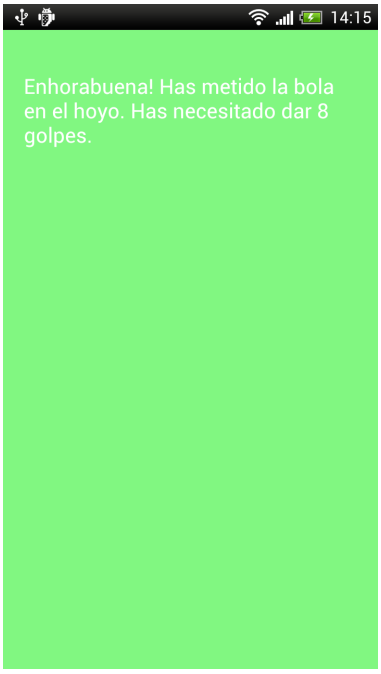

Figura 7.4: Pantalla final habiendo metido la bola

En cambio, si nos pasamos más de 10 golpes sin haber podido introducir la bola en el banderín, veremos esta pantalla de finalización.

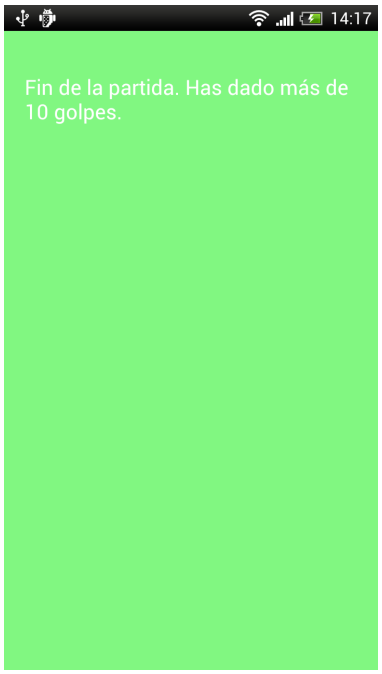

Figura 7.5: Pantalla final habiendo superado el máximo de golpes

CAPÍTULO 8

Conclusiones y desarrollos futuros

8.1– Conclusiones

Las conclusiones que hemos sacado a partir de este proyecto son numerosas y se puede comprobar a lo largo de esta memoria. Nuestro objetivo principal con el proyecto de fin de carrera era poder además de crear una aplicación que simulara al mando de la videoconsola Wii, investigar sobre ampliaciones en la aplicación que permitieran una mayor usabilidad. Con mayor usabilidad nos referimos a que el videojuego ademas de ser más facil de usar, sea más atractivo al usuario, ya que como proyecto uno de los objetivos principales es que su acogida en el mercado sea satisfactoria, por lo que un buen desarrollo futuro podría generar altos beneficios.

Todo ello que nos propusimos en un principio ha sido posible, demostrándose además un interés bastante notable en el público al conocer la aplicación y su funcionamiento. Que nuestro alrededor nos alentara en este proyecto ha sido algo a destacar ya que siempre hemos tenido a personas queriendo ver la evolución y su desarrollo.

La realización de este proyecto nos ha servido para coger una base de cara a futuros desarrollos, que seguro serán más rápidos y ágiles. También hemos podido comprobar que es posible desarrollar una aplicación desde cero y utilizando exclusivamente herramientas libres.

Desarrollando este proyecto hemos disfrutado, investigado y aprendido mucho sobre un tema que nos apasiona. Además hemos encontrado un sector en el que no nos importaría que se desarrollase nuestro futuro laboral ya que se puede trabajar en diferentes campos y sobre diferentes temas.

Al comienzo de este proyecto decidimos hacerlo todo conjuntamente ya que como hemos aprendido en la escuela con numerosas asignaturas siempre es mejor formar parte de un equipo. Al realizar todo entre los dos ha sido más fácil ver errores de programación y solucionar cuestiones que

nos surgían. Sabemos que podríamos habernos dividido el trabajo y que podríamos haber sido más rápidos pero no tan eficientes, estamos muy contentos de haber escogido esta forma de trabajo ya que el resultado ha sido óptimo.

8.2– Desarrollos futuros

Como bien hemos explicado a lo largo de esta memoria este proyecto ha tenido mucha investigación y en relación a los desarrollos futuros más aún. Lo que nos hubiera gustado hacer si hubiéramos tenido medios y tiempo habría sido una red social de VirtualGolf en la que los usuarios pudiesen jugar al golf en su casa solamente con el uso de su dispositivo móvil.

Con la idea de la red social podríamos ampliar muchas cosas del videojuego, las siguientes son las que creemos más interesante:

- **Mejora del entorno:** Un punto fundamental en un videojuego es su apariencia y en el prototipo que hemos realizado para este proyecto para su puesta en el mercado deberíamos mejorar su estética, como por ejemplo, que se viese el palo de golf en cada tiro y también quedaría bastante bien si la parte inferior que vemos como superficie del banderín tuviese aspecto de césped.

- **Elección del palo de golf:** Sería interesante que el usuario pudiese elegir su palo de golf e incluso personalizarlo. Con esta ampliación el jugador se sentiría dentro del juego y esto es un punto importante en la realización de un videojuego. La finalidad de la elección del palo sería poder adecuarlo a nuestras necesidades.

- **Viento:** Sería un factor aleatorio del juego que le daría mayor dificultad y realismo al juego.

- **Concepto de amistad:** Como red social los usuarios tendrían amigos a los que retarían a jugar cuando y donde quisiesen. Es importante que se involucre a gente del entorno del jugador en el videojuego ya que de esta forma tendrán un uso continuado del juego, retándose y consiguiendo superar el nivel de sus amigos.

- **Niveles:** Cada usuario comenzaría en un nivel base y a partir de ahí con los retos o partidas de entrenamiento se conseguirían puntos y se iría subiendo de nivel incentivando al usuario a jugar. El usuario podrá ver en que posición del ranking se encuentra en relación con sus amigos.

- **Minijuegos:** Con los minijuegos se conseguirían puntos, y según el nivel en el que se encuentre el jugador habría diferentes minijuegos,

estos estarían relacionados con el golf y de fácil uso. Serían sencillos de realizar y usarían algunas de las técnicas investigadas en el proyecto.

Sabemos que todo esto sería posible pero como ya hemos dicho sería necesario tiempo y medios. Nos hubiera encantado poder continuar hasta llegar a terminar los puntos anteriormente señalados y no descartamos que en un futuro lo intentemos.

Bibliografía

[1] Alberto Pérez Barranco. Geolocalización perfiles nbff. https://forja.rediris.es/projects/nbff. 02-07-2013.

[2] Bytehangar. My android sensors. http://www.bytehangar.com/android/my_android_sensors.html. 02-07-2013.

[3] Alberto Beiztegui Casado. Videojuego android acelerómetro. https://forja.rediris.es/projects/android-acel. 12-02-2013.

[4] Adrián Catalán. Trabajando con acelerómetros en android. http://www.slideshare.net/ykro/trabajando-con-acelermetros-en-android. 27-04-2013.

[5] CPUz. http://www.cpuid.com/softwares/cpu-z.html. 27-04-2013.

[6] Maestros del Web. Curso android sensores, trabajar con acelerómetro. http://www.maestrosdelweb.com/editorial/curso-android-sensores-trabajar-con-acelerometro. 16-05-2013.

[7] Eclipse. www.eclipse.org. 16-05-2013.

[8] Google. Página oficial de android. http://www.android.com. 27-04-2013.

[9] HTC. http://www.htc.com/es. 16-05-2013.

[10] José Luis Navas Jiménez. Localizador con realidad aumentada. https://forja.rediris.es/projects/josnavjim. 12-02-2013.

[11] Sergio Bellón Alcarazo. Jorge Creixell Rojo. Ángel Serrano Laguna. Página web oficial look!. http://www.lookar.net/tutorial/. 12-02-2013.

[12] Nintendo. http://www.nintendo.es/wii/wii-94559.html. 02-07-2013.

[13] Salvador Gómez Oliver. Curso programación android. http://www.sgoliver.net/blog/?page_id=3011. 12-02-2013.

[14] Francisco Javier Martin Otero. Geotask. https://forja.rediris.es/projects/geotask. 16-05-2013.

[15] Juan León Padilla. Androio: tool to announce arrival. https://forja.rediris.es/projects/androidarrival. 12-02-2013.

[16] Ernesto Lage Tejero. Jose María Valle Rodríguez. Videojuego para android. https://forja.rediris.es/projects/androgame. 27-04-2013.

[17] David Sastre. Sensores en android: Acelerómetro. http://sekthdroid.wordpress.com/2013/03/12/sensores-en-android-acelerometro. 27-04-2013.

[18] Google Play Store. https://play.google.com/store?hl=es_419. 02-07-2013.

[19] Vma. Tutorial android: Acelerómetro, accelerometer (sensor). http://www.tutorialeshtml5.com/2012/10/tutorial-android-acelerometro.html. 02-07-2013.

[20] Wikipedia. Ángulos de navegación, wikipedia. http://es.wikipedia.org/wiki/ 02-07-2013.

[21] Fernando Antonio Caro Vega y Francisco Javier Delgado Villegas. Tutor: José Ramón Portillo Fernández. Clase latex para proyecto fin de carrera. 16-05-2013.