UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Título del Proyecto:

Laberinto

Autor:

Gasulla Mestre, Javier

Director:

Albiol Colomer, Antonio

TESINA PARA LA OBTENCIÓN DEL TÍTULO DE:

Máster en Desarrollo de Aplicaciones sobre Dispositivos Móviles

Septiembre de 2015

Contenido

Introducción 3
Descripción de la aplicación 3
Objetivos 3
Motivación 3
Arquitectura de la aplicación 4
Esquema del diseño 4
Código Java 5
Código nativo 9
Librerías generadas 10
Recursos 10
Modelo de datos 10
Vistas 11
Conclusiones 19
Anexos 20
Listado de fuentes entregadas 20

Introducción

Descripción de la aplicación

El proyecto consiste en el desarrollo de una aplicación en Android, utilizando Eclipse. La aplicación constituye un juego en el que hay que mover una pelota por un laberinto para recoger monedas y dirigirse a la meta. Al llegar a la meta pasamos al siguiente nivel, en el que aparece otro laberinto diferente. En total la aplicación consta de dos niveles.

La pelota puede moverse de tres maneras diferentes a elegir en preferencias:

- Detección de cara. Se utiliza para ello la funcionalidad proporcionada por OpenCV, así como implementación de funcionalidad en código nativo.
- Pantalla táctil.
- Sensores de aceleración.

Objetivos

El principal objetivo de la aplicación es utilizar OpenCV para procesar el movimiento de la cara del usuario (captado por la cámara del dispositivo) y en consecuencia ejecutar acciones sobre otros elementos (en este caso desplazar una pelota por un laberinto).

La implementación del juego constituye una excusa para plasmar el objetivo buscado.

Motivación

La motivación de este proyecto ha sido integrar el tratamiento de imagen con otros aspectos vistos a lo largo del curso (código nativo, threads, acelerómetros, pantalla táctil, ajuste del tamaño de imágenes, layouts para diferentes tipos de pantalla, multiidioma, etc.), en una aplicación que pueda servir de base para futuras aplicaciones de uso tanto lúdico como profesional.

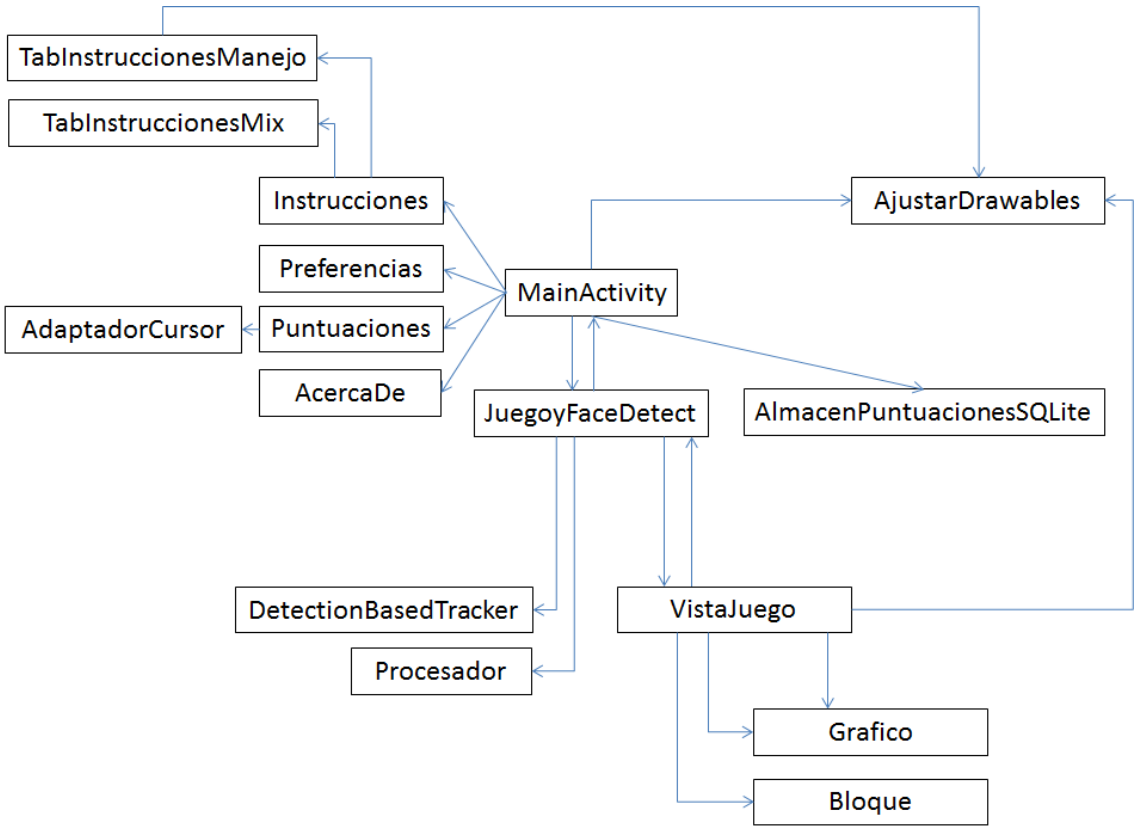
Arquitectura de la aplicación

En los apartados siguientes se va a detallar los principales elementos que forman la aplicación (clases Java, código nativo, librerías, recursos), describiendo su funcionalidad para mostrar el funcionamiento de la aplicación.

Por otra parte, en la librería externa "OpenCV Library – 2.4.10", utilizada por la aplicación, hay que modificar la clase org.opencv.android.AsyncServiceHelper para convertir el intent implícito existente en explícito. De este modo la aplicación puede usarse para terminales con nivel de API 21 o superior.

Esquema del diseño

En el esquema siguiente se muestra la relación de las diferentes clases entre sí.



Código Java

A continuación se relaciona, por orden alfabético, las diferentes clases que forman la aplicación, detallando la función de cada una de ellas.

- **AcercaDe.java**
  Lanza el Dialog con los créditos de la aplicación.

- **AdaptadorCursor.java**
  Adaptador del ListView que contiene las puntuaciones.

- **AjustesDrawables.java**
  Implementa el código necesario para ajustar el drawable indicado al tamaño deseado. Se utiliza en las clases MainActivity, VistaJuego y TabInstruccionesManejo para dimensionar los drawables en función del tamaño de pantalla del dispositivo. De este modo, independientemente del tamaño de dicha pantalla, las vistas mostradas mantendrán el mismo aspecto.

- **AlmacenPuntuaciones.java**
  Interfaz que indica los métodos para guardar una puntuación o mostrar puntuaciones, que debe implementar la clase AlmacenPuntuacionesSQLite.

- **AlmacenPuntuacionesSQLite.java**
  Se encarga de crear la base de datos que contiene las puntuaciones de los jugadores y gestionar los accesos a la misma (para guardar una nueva puntuación u obtener la relación de puntuaciones).

- **Bloque.java**
  Extiende de la clase Grafico. A esta clase pertenecen los bloques que forman las paredes del laberinto. Contiene características particulares de los bloques, como por ejemplo los diferentes tipos de bloque.

  El aspecto de los bloques es el mismo, pero se les asigna un tipo distinto en función de los lados del bloque con los que puede chocar la pelota (los lados contiguos a otro bloque no son susceptibles de que la pelota choque con ellos).

  El tipo de cada bloque, junto con la posición relativa de pelota y bloque, se utilizará para determinar si la pelota ha chocado con el bloque. Se determina también que parte de la pelota (inferior, superior, derecha o izquierda) es la que ha chocado con el bloque. La detección de colisiones implementada en la clase Grafico sirve para detectar colisiones entre objetos de forma redondeada. Al ser los bloques cuadrados y querer determinar el sentido del choque, se implementa una función particularizada en esta clase.

- **DetectionBasedTracker.java**

  Utilizada para la detección de caras en modo "nativo". Proporcionada en el ejemplo face-detection de OpenCV.

- **Grafico.java**

  Implementa los métodos necesarios para dibujar, posicionar y mover objetos de esta clase creados a partir de drawables. También detecta colisiones con otro objeto de la clase Grafico (se utiliza para detectar colisiones de pelota con moneda, o pelota con la meta).

  Se utiliza en la clase VistaJuego ya que la pelota, monedas y meta allí creados son objetos de la clase Grafico. También derivan de esta clase los objetos de tipo bloque que forman las paredes del laberinto.

- **Instrucciones.java**

  FragmentActivity que muestra los dos tabs que contienen las instrucciones para utilizar la aplicación, definidos en TabInstrucciones.java y TabInstruccionesMix.java.

- **JuegoyFaceDetect.java**

  Esta clase es llamada por MainActivity.java cuando se elige la opción de jugar una partida. Se encarga de inicializar OpenCV, cargar las librerías, realizar la detección cara, lanzar la pantalla del juego (gestionando el thread que maneja el juego, definido en el fichero VistaJuego.java), establecer la dirección y sentido del movimiento de la pelota (si se ha elegido la detección de cara como modo de manejo de la misma) y activar o desactivar los sensores de aceleración (en caso de que sea el modo elegido para manejar la pelota). También gestiona la pulsación de los botones existentes en la pantalla del juego, descritos en el apartado de vistas.

  Para realizar la detección de cara se ha utilizado el método descrito en el ejemplo face-detection de OpenCV. Para ello se utiliza un fichero LBPCascade optimizado para detección frontal de cara. Este tipo de ficheros contienen los patrones que permiten realizar la detección de cara en una imagen. Los dos tipos más utilizados son LBPCascade y HaarCascade. En nuestro caso es LBPCascade el idóneo, ya que siendo algo menos preciso, es más rápido. Esta última característica resulta fundamental, al tratarse de una aplicación que realiza el tratamiento de una imagen de video para obtener una respuesta en tiempo real (el movimiento de la pelota).

  De entre las diferentes caras detectadas, nos quedamos con la de mayor tamaño (se asume que corresponde al jugador). En función de la posición de la cara, respecto al centro de la pantalla, se establece el movimiento de la pelota. Una explicación más detallada sobre esta cuestión está disponible en el apartado de vistas.

  La detección se puede realizar de dos maneras diferentes a definir en preferencias (Java o nativa). "Java" utiliza funciones pertenecientes a OpenCV Library – 2.4.10, sin necesidad de implementar código nativo adicional. "Nativa" utiliza OpenCV Library – 2.4.10, utilizando código nativo adicional (disponible en el ejemplo de OpenCV "face-

detection"). En determinados casos en que el fondo que hay tras el usuario presenta variedad de formas y colores, puede dar lugar a detectar como falso positivo alguna de estas formas. En ese caso el modo "Java" presenta una detección intermitente de dicho falso positivo, mientras que el modo "Nativo" se mantiene inmune. En la aplicación, al utilizar la cara detectada de mayor tamaño, no notaremos diferencia entre ambos modos, ya que en caso de que parte del fondo sea susceptible de ser detectada como falso positivo, normalmente presentará un tamaño menor que la cara del usuario.

La detección utiliza también el tamaño mínimo que debe tener la cara (en relación porcentualmente al tamaño de la imagen de la cámara) para ser detectada. Este tamaño se establece en preferencias.

Esta clase inicializa OpenCV (en caso de que no esté instalada, también solicita permiso al usuario para descargar la aplicación OpenCV Manager en el terminal). Tras esto carga las dos librerías OpenCV incluidas en la aplicación:

- "detection_based_tracker". Utilizada para realizar la detección de cara utilizando funciones nativas adicionales, incluida en el ejemplo "face-detection" de OpenCV.

- "mi_libreria_opencv". Utilizada para reflejar horizontalmente la imagen mostrada en pantalla, correspondiente a la cámara frontal. De este modo, cuando el usuario mueve la cabeza a derecha o izquierda, la pelota se desplaza en ese sentido y en la imagen mostrada en pantalla la cabeza se mueve, de igual modo, en el mismo sentido. En un primer momento, implementé el hecho de reflejar la imagen utilizando código java. La detección era considerablemente más lenta, por lo que la imagen y la pelota se movían "a golpes", perdiendo todo el efecto de movimiento continuo. En cambio, al implementar esta operación en código nativo (y generar la librería correspondiente) el efecto de movimiento en tiempo real permanece intacto.

- **MainActivity.java**

Clase principal de la aplicación. Es la responsable de mostrar la pantalla inicial de la aplicación y realizar las acciones correspondientes en función del ImageButton pulsado (jugar una partida, establecer las preferencias, mostrar las diez mayores puntuaciones, instrucciones o créditos de la aplicación).

Esta clase ajusta el tamaño de los ImageButton de manera proporcional al tamaño de la pantalla. Para ello hace uso de las funciones definidas en la clase AjustesDrawables.java.

Mientras se muestra la pantalla principal, se reproduce una música de fondo, siendo esta clase la responsable de la gestión de la misma. Cuando se abandona la pantalla principal se guarda el punto de reproducción de la música de modo que, cuando regresamos, la reproducción de la música continúa en el lugar en que se interrumpió.

Cuando, tras haber jugado una partida, volvemos a la pantalla principal, la puntuación obtenida, junto con el nombre del jugador (fijado en preferencias) y la fecha y hora actual se almacenan en una base de datos local SQLite. Además, la puntuación obtenida y el nombre del jugador se muestran en un mensaje tipo Toast y se envían en una notificación que recibe el usuario (esto último constituye fundamentalmente un recurso didáctico).

Las opciones preferencias, instrucciones y puntuaciones están también disponibles a través del menú de la aplicación.

- **Preferencias.java**
  Lanza la pantalla de preferencias.

- **Procesador.java**
  Clase en la que se define el método nativo utilizado para reflejar horizontalmente la imagen a mostrar en la pantalla del dispositivo, correspondiente a lo captado por la cámara frontal del teléfono.

- **Puntuaciones.java**
  Lanza la pantalla con las puntuaciones.

- **TabInstruccionesManejo.java**
  Muestra el tab con las instrucciones para manejar la pelota. Es llamada desde la clase Instrucciones.java. Para establecer el tamaño de la imagen que muestra en pantalla, utiliza la clase AjustesDrawables, de este modo su tamaño es proporcional al tamaño de la pantalla. Es llamada desde la clase Instrucciones.java.

- **TabInstruccionesMix.java**
  Muestra el tab con instrucciones generales.

- **VistaJuego.java**
  En este fichero se definen dos clases: VistaJuego y ThreadJuego.

<u>Clase VistaJuego</u>: Define y maneja la vista que contiene el laberinto y los diferentes elementos en él existentes (pelota, monedas y meta). Esta clase gestiona los diferentes aspectos relacionados con la partida. Se ha definido la clase JuegoyFacedetect como su clase padre, ya que es la actividad que ha inflado el layout que contiene la vista VistaJuego.

Establece el tamaño de los drawables de manera proporcional al tamaño de la pantalla del dispositivo (utilizando la clase AjustesDrawables) y posiciona los diferentes elementos en pantalla.

Gestiona los sonidos a emitir cuando se coge una moneda o se alcanza la meta del nivel (la manera de gestionar la clase SoundPool se realiza de manera diferente en función del nivel de API del dispositivo, ya que cambia a partir del API 21).

Determina la dirección y sentido que debe llevar la pelota en los casos de manejo mediante pantalla táctil o acelerómetro (en el caso de detección de cara se encarga de esta cuestión la clase JuegoyFaceDetect).

En cualquiera de los tres modos de manejo de la pelota existentes se encarga de mover la pelota. Para ello se gestiona las colisiones con los bloques que forman el laberinto, recolección de monedas o pasar al siguiente nivel si se ha llegado a la meta. Se establece la velocidad de la pelota en función del modo de manejo de la misma. En el caso de detección de cara, la velocidad es la mitad que en caso de usar la pantalla táctil o acelerómetro. Esto se ha implementado así para tener mejor control sobre la pelota en el caso de detección de cara. Por otra parte, la velocidad de la pelota se mantiene constante independientemente de la resolución del dispositivo.

Al finalizar la partida se devuelve, como respuesta en el Intent, la puntuación obtenida.

Clase ThreadJuego: Esta clase implementa el thread que permite el desarrollo de la partida. La clase JuegoyFaceDetect es la encargada de poner en marcha el thread o pausarlo.

Código nativo

La carpeta jni contiene la parte de código nativo utilizada por la aplicación. Está formada por los ficheros siguientes:

- **Android.mk**
  Utilizado para construir las librerías detection_based_tracker (a partir de DetectionBasedTracker_jni.cpp) y mi_libreria_opencv (a partir de Procesador_jni.cpp)

- **Application.mk**
  Indicamos que se va a construir las librerías para las diferentes plataformas: armeabi, armeabi-v7a, x86 y mips.

- **DetectionBasedTracker_jni.cpp y DetectionBasedTracker_jni.h**
  Implementan la detección de cara mediante código nativo. Proporcionados por el ejemplo face-detection de OpenCV.

- **Procesador_jni.cpp y Procesador_jni.cpp**
  Se implementa la acción de reflejar la imagen de la cámara frontal del dispositivo, mediante código nativo. Al utilizar código nativo se mantiene el efecto de procesado en tiempo real. Si se realizara esta operación mediante código Java (como hice en un primer momento), el proceso sería considerablemente más lento, perdiendo el efecto de movimiento continuo.

Librerías generadas

En la carpeta libs se incluyen las librerías libdetection_based_tracker.so (usada para realizar la detección de cara mediante código nativo) y libmi_libreria_opencv.so (usada para reflejar horizontalmente la imagen de la cámara), para cada una de las plataformas definidas en el fichero Application.mk.

Recursos

La parte de recursos de la aplicación contiene las siguientes carpetas:

- **drawable**
  Drawables a incluir en las diferentes pantallas de la aplicación.

- **layout y layout-xlarge**
  Layouts a mostrar en las pantallas de la aplicación. Se distingue entre teléfonos y dispositivos de mayor tamaño como tablets.

- **raw**
  Contiene los ficheros de música y el fichero lbpcascade_frontalface.xml para detectar que partes de la imagen corresponden a una cara

- **values y values-es**
  Incluye ficheros de strings en castellano e inglés (idioma por defecto), arrays y estilos propios de la aplicación.

- **xml**
  Fichero con las preferencias de la aplicación.

En el apartado de Anexos se incluye la relación de archivos contenida en cada una de las carpetas anteriores.

Modelo de datos

La aplicación utiliza una base de datos, compuesta por una única tabla, donde se guarda la información con las puntuaciones obtenidas en las diferentes partidas. Esta tabla se crea, actualiza y consulta utilizando las funciones definidas en la clase AlmacenPuntuacionesSQLite.java

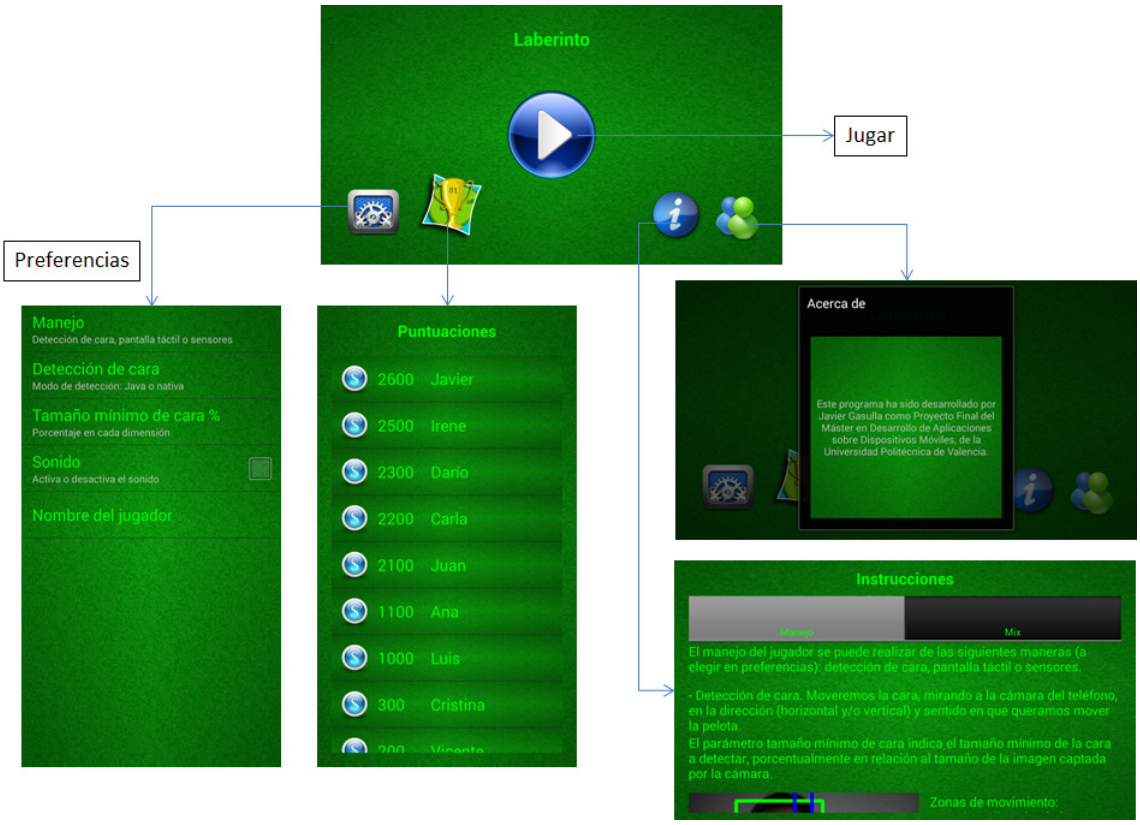
Los campos de que consta la tabla “puntuaciones” son los siguientes:

- puntos. Entero que contiene la puntuación obtenida.
- nombre. Texto con el nombre del jugador.
- Fecha. Fecha de la partida. Este campo no se ha utilizado en la aplicación.

Vistas

A continuación se muestran las diferentes pantallas de la aplicación, así como la navegación entre las mismas.

En primer lugar tenemos la **pantalla principal**:



Como se puede ver la pantalla principal nos da acceso a:

- Jugar: accedemos a la pantalla con el juego
- Preferencias: configuramos las diferentes opciones.
- Puntuaciones: obtenemos una lista con las diez mayores puntuaciones.
- Instrucciones: indica como usar la aplicación.
- Acerca de: diálogo con los créditos de la aplicación

En la **pantalla de preferencias** se configuran las diferentes opciones, explicadas a continuación

- **Manejo**.

  Forma de mover la pelota por el laberinto. Hay tres posibilidades:

  - Detección de cara. Moveremos la cara, mirando a la cámara del teléfono, en la dirección y sentido en que queramos mover la pelota. Más adelante se explica detalladamente como realizar el movimiento.

  - Pantalla táctil. Tocaremos la pantalla y arrastraremos el dedo para mover la pelota por el laberinto. Podemos situar y mover el dedo sobre la pelota o sobre cualquier otro punto de la pantalla. La pelota se moverá para acercarse a nuestro dedo.

  - Sensores de aceleración. Inclinando el dispositivo conseguiremos desplazar la pelota por el laberinto.

- **Detección de cara.**

  La aplicación puede realizar la detección de cara de dos maneras diferentes:

  - Java. Utiliza funciones pertenecientes a OpenCV Library – 2.4.10, sin necesidad de implementar código nativo adicional.

  - Nativa. Utiliza OpenCV Library – 2.4.10, utilizando código nativo adicional (disponible en el ejemplo de OpenCV "face-detection". La detección "Nativa" presenta una mayor inmunidad frente a falsos positivos que pueda provocar un fondo con variedad de formas y colores.

- **Tamaño mínimo de cara %**.

  Indica el tamaño mínimo que debe tener la cara a detectar, en relación a la imagen captada por la cámara. Se dan dos opciones 20%, y 40%. Por ejemplo, si elegimos la opción de 40%, la aplicación desechará todas las caras cuyas dimensiones sean menores que el 40% de la anchura o la altura de la imagen total captada por la cámara. Se recomienda utilizar la opción del 20%, ya que la aplicación trabaja siempre con la cara mayor de las detectadas. De este modo, si nos separamos un poco del dispositivo, éste seguirá detectándonos.

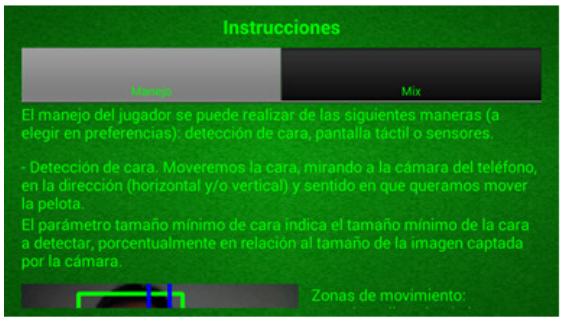
- **Sonido**.

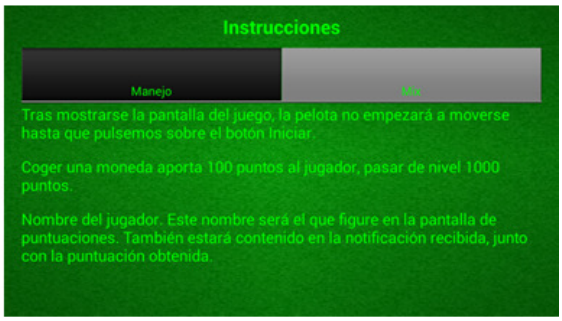
  Activa o desactiva tanto la música de la pantalla principal, como los sonidos (al recoger una moneda o llegar a la meta) de la pantalla del juego.

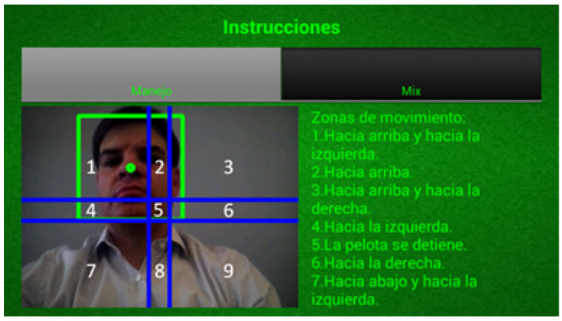
- **Nombre del jugador**.

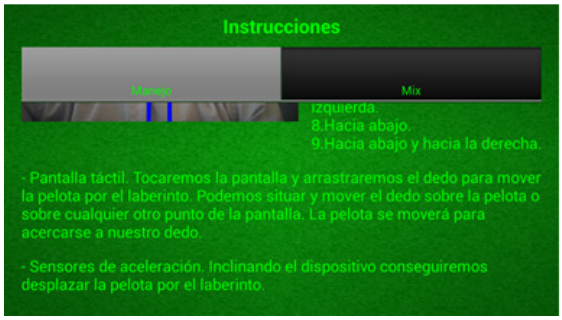
  Este nombre será el que figure en la pantalla de puntuaciones. También estará contenido en la notificación recibida, junto con la puntuación obtenida.

La **pantalla de instrucciones** indica como manejar la aplicación. La información se agrupa en dos tabs. En uno se explica como manejar la pelota y en el otro se indican aspectos generales de la aplicación.

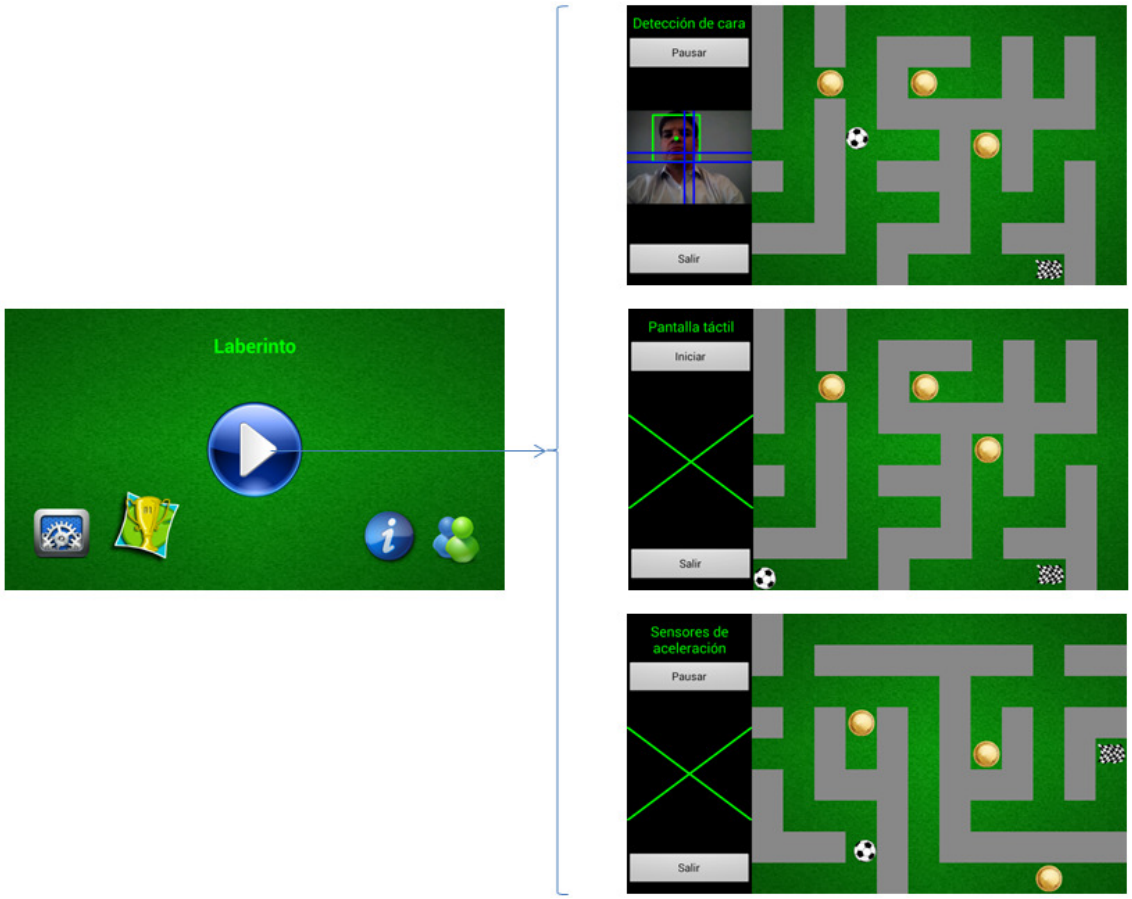








La **pantalla del juego** nos permitirá mover la pelota por el laberinto, mediante el modo elegido en preferencias: detección de cara, pantalla táctil o sensores de aceleración. El modo utilizado se indica en la parte superior izquierda de la pantalla.



Tras mostrarse la pantalla del juego, la pelota no empezará a moverse hasta que pulsemos sobre el botón “Iniciar”, además el texto del botón cambiará a “Pausar”. Si pulsamos entonces sobre el botón la pelota se detendrá. En ese caso, el texto del botón cambiará de nuevo a “Iniciar”.

En cualquier momento podemos salir de la partida pulsando el botón “Salir”. Se guardará como puntuación final de la partida los puntos que hasta ese momento hayamos conseguido. Se obtiene cien puntos por cada moneda y mil puntos por llegar a la meta de cada uno de los dos niveles de que consta el juego

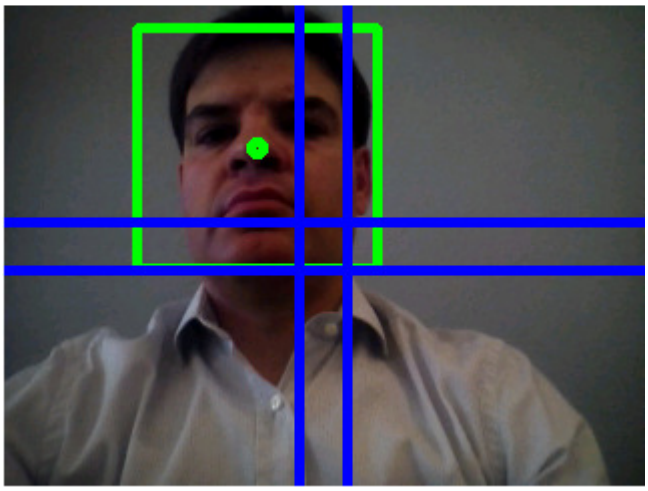
A continuación indico las particularidades de la pantalla en función del modo elegido.

- Detección de cara.

En la parte de la derecha se muestra la imagen que capta la cámara frontal. Esta imagen se ha reflejado horizontalmente, mediante implementación en código nativo. De este modo se consigue el efecto de que cuando el jugador se mueve hacia la derecha, la pelota se mueve hacia la derecha y la cara de la imagen también se mueve hacia la derecha (desde el punto de vista del usuario que está manejando el dispositivo).

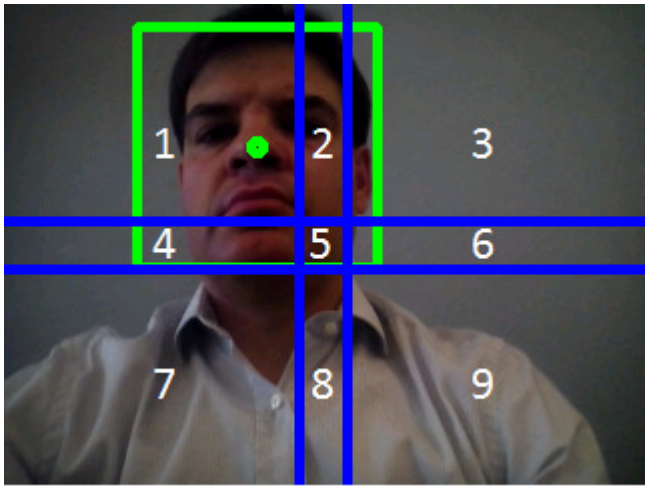
El jugador debe tratar de mirar a cámara de manera frontal, ya que la aplicación está preparada para la detección frontal de caras.

Impresionados sobre la imagen de la cámara se dibujan diferentes elementos cuya finalidad es servir de referencia al usuario, para dirigir la pelota en la dirección y sentido deseados.



El rectángulo de color verde representa la cara detectada. En el centro de dicho rectángulo se ha dibujado un punto, también de color verde. Nos tendremos que fijar en la posición de este punto para dirigir la pelota.

Las líneas de color azul delimitan diferentes zonas de la pantalla. En función de la zona en que se encuentre el punto verde se determina la dirección en que se mueve la pelota. A continuación se indica el movimiento que tendrá la pelota, para cada una de estas zonas.



1. Movimiento hacia arriba y hacia la izquierda.
2. Movimiento hacia arriba.
3. Movimiento hacia arriba y hacia la derecha.
4. Movimiento hacia la izquierda.
5. La pelota se detiene.
6. Movimiento hacia la derecha.
7. Movimiento hacia abajo y hacia la izquierda.
8. Movimiento hacia abajo.
9. Movimiento hacia abajo y hacia la derecha.

El tamaño de las diferentes zonas guarda la misma proporción, independientemente de la resolución del dispositivo.

En caso de que la aplicación deje de detectar la cara del usuario, la pelota seguirá moviéndose en la misma dirección, hasta colisionar con un bloque. Esto permite que aunque haya instantes en que no se detecte la cara del usuario, haya un movimiento continuo evitando pequeñas interrupciones en el movimiento de la pelota.

- <u>Pantalla táctil y Sensores de aceleración</u>.

En cualquiera de estos dos casos no se muestra la imagen de la cámara. En su lugar mostramos una x.

Conclusiones

En la aplicación se ha conseguido el objetivo de realizar acciones a partir de movimientos de cabeza efectuados por el usuario.

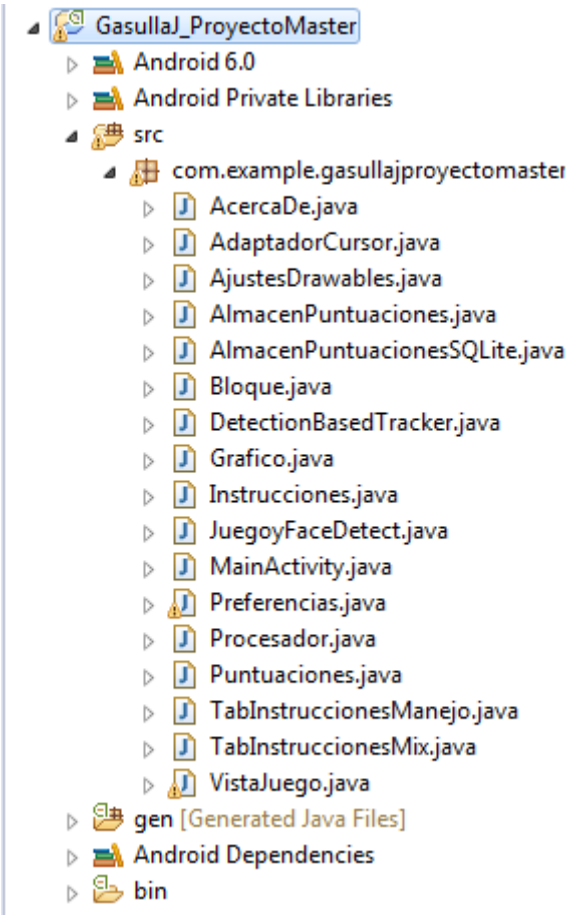
La funcionalidad implementada en el juego podría usarse para propósitos muy diferentes, como por ejemplo enviar mensajes a dispositivos externos con la finalidad de que estos ejecuten una determinada acción.

En este caso me he servido de la funcionalidad implementada en el ejemplo face-detection suministrado por OpenCV. Si quisiera detectar otro tipo de objeto tendría que implementar el clasificador realizando un proceso de entrenamiento. El entrenamiento debería ser lo suficientemente exhaustivo para minimizar el número de falsos positivos y falsos negativos.

Anexos

Listado de fuentes entregadas

A continuación se indica una relación de los archivos que componen la aplicación:



- jni
  - Android.mk
  - Application.mk
  - DetectionBasedTracker_jni.cpp
  - DetectionBasedTracker_jni.h
  - Procesador_jni.cpp
  - Procesador_jni.h
- libs
  - armeabi
    - libdetection_based_tracker.so
    - libmi_libreria_opencv.so
  - armeabi-v7a
    - libdetection_based_tracker.so
    - libmi_libreria_opencv.so
  - mips
    - libdetection_based_tracker.so
    - libmi_libreria_opencv.so
  - x86
    - libdetection_based_tracker.so
    - libmi_libreria_opencv.so
- obj
- res
  - drawable
    - acercade.png
    - fondo.jpg
    - info.png
    - meta.png
    - moneda.png
    - pelota.png
    - play.png
    - puntos.png
    - scores.png
    - settings.png
    - zonas_mvto_cara.png
  - drawable-hdpi
    - ic_launcher.png
  - drawable-mdpi
    - ic_launcher.png
  - drawable-xhdpi
    - ic_launcher.png
  - drawable-xxhdpi
    - ic_launcher.png
  - drawable-xxxhdpi
    - ic_launcher.png

- layout
  - acercade.xml
  - activity_main.xml
  - elemento_lista.xml
  - instrucciones.xml
  - juego.xml
  - puntuaciones.xml
  - tab_instrucciones_manejo.xml
  - tab_instrucciones_mix.xml
- layout-xlarge
  - acercade.xml
  - activity_main.xml
  - elemento_lista.xml
  - instrucciones.xml
  - juego.xml
  - puntuaciones.xml
  - tab_instrucciones_manejo.xml
  - tab_instrucciones_mix.xml
- menu
  - main.xml
- raw
  - lbpcascade_frontalface.xml
  - musica_fondo.mp3
  - sonido_meta.mp3
  - sonido_moneda.mp3
- values
  - arrays.xml
  - strings.xml
  - styles.xml
- values-es
  - strings.xml
- xml
  - preferencias.xml
- AndroidManifest.xml
- project.properties