# Sensores y eventos - Ejercicios

## Índice

## 1. Pantalla táctil

Vamos a implementar una nueva aplicación `eventos`, en la que mostraremos una caja en la pantalla (un rectángulo de 20x20) y la moveremos utilizando la pantalla táctil. Se pide:

*a)* Empezar haciendo que se mueva la caja al punto en el que el usuario pone el dedo y comprobar que funciona correctamente (sólo hace falta reconocer el evento `DOWN`).

*b)* Implementar ahora también el evento de movimiento (`MOVE`), para hacer que la caja se desplace conforme movemos el dedo.

*c)* Sólo querremos que la caja se mueva si cuando pusimos el dedo en la pantalla lo hicimos sobre la caja. En el evento `DOWN` ya no moveremos la caja, sino que simplemente comprobaremos si hemos pulsado encima de ella. Si es así, nos anotaremos el *id* del gesto actual para mover la caja cuando se produzca un evento `MOVE` dentro de este gesto. Un evento `MOVE` cuyo gesto no se inició pulsando sobre la caja no deberá moverla.

**Ayuda**

Esto último se puede conseguir de dos formas diferentes. La primera de ellas es devolviendo `true` o `false` cuando se produzca el evento `DOWN`, según si queremos seguir recibiendo eventos para ese gesto o no. Si se pulsa fuera de la caja podemos devolver `false` para así no recibir ningún evento de movimiento correspondiente a ese gesto. La segunda consiste en anotarnos el identificador del gesto que se inicie dentro de la caja, y sólo permitir arrastrar la caja a los eventos de movimiento de dicho gesto. Esta segunda forma nos permitirá controlar mejor los eventos en dispositivos multitouch.

## 2. Gestos

Continuaremos trabajando con el proyecto anterior, en este caso para reconocer gestos. Se pide:

*a)* Reconocer el gesto de *double tap* aplicado sobre la pantalla. Cada vez que se produzca este gesto, haremos que la caja se posicione en el centro de la pantalla.

**Atencion**

Para reconocer el gesto *double tap*, deberemos devolver `true` en el evento `onDown` del detector de gestos. Por defecto devuelve `false`, y esto produce que todos los gestos sean descartados desde su comienzo.

**Importante**

Según cuando llamemos al `onTouchEvent` del detector de gestos conseguiremos un comportamiento u otro. Hay que llevar cuidado en cuándo se devuelve `true` o `false` para un determinado gesto. Por ejemplo, si procesamos primero los eventos `DOWN` y `MOVE`, y después

llamamos al detector de gestos, si al reconocer los eventos de pulsar sobre la caja o arrastrarla estamos devolviendo directamente `true`, y por lo tanto finalizando la ejecución del método, el detector de gestos nunca llegará a ejecutarse. El resultado será que cuando estemos encima de la caja el *double tap* no será reconocido. Si no devolvemos `true` para que pueda llegar a ejecutarse el detector, deberemos llevar cuidado de que después de ejecutarse el método devuelva `true`, ya que de no ser así no nos llegarían más eventos de dicho gesto.

*b)* Reconocer el gesto *fling* ejercido sobre la pantalla. Cuando esto ocurra mostraremos un vector (línea) saliendo de la posición en la que terminó el gesto indicando la velocidad y dirección con la que se lanzó.

*c)* De forma optativa, se puede hacer que al realizar el gesto *fling* sobre la caja ésta se lance con cierta inercia. Para hacer esto necesitaremos un hilo o temporizador que vaya actualizando la posición de la caja según su velocidad, e irá dismunuyendo su velocidad debido al rozamiento.

## 3. Manipulación de gráficos 3D

Vamos a recuperar la aplicación `graficos` de la sesión anterior, en la que veíamos una figura rotando, y vamos a hacer que en lugar de rotar automáticamente la podamos rotar nosotros conforme arrastremos el dedo por la pantalla. Comenzaremos haciendo que rote alrededor del eje Y al desplazar el dedo hacia la izquierda o hacia la derecha. Una vez esto funcione, añadiremos la rotación en el eje X al arrastrar hacia arriba o hacia abajo.

**Ayuda**

Si utilizamos `GLSurfaceView` ya no podremos sobrescribir el método `onTouchEvent`, ya que no hemos creado una subclase, sino que simplemente hemos definido un `Renderer`. Para poder recibir los eventos podemos definir un objeto `OnTouchListener` y registrarlo en el `GLSurfaceView`. Puede ser buena idea hacer que la misma clase del `Renderer` actúe también como *listener*, y así el mismo evento de *touch* podrá modificar directamente el ángulo de rotación actual.

## 4. Geolocalización

Implementar una nueva aplicación `geolocalizacion` que nos localice geográficamente utilizando GPS, y nos muestre tanto nuestras coordenadas como nuestra dirección en forma de texto.

Para poder probar esto en el emulador deberemos indicarle manualmente al emulador las coordenadas en las que queremos que se localize. Esto lo podemos hacer de dos formas: mediante línea de comando o mediante la aplicación DDMS. Vamos a verlas a continuación.

Para comunicar las coordenadas al emulador mediante línea de comando deberemos

conectarnos a él mediante `telnet`. Por ejemplo, si nuestro emulador está funcionando en el puerto `5554`, haremos un `telnet` a `localhost` y a dicho puerto:

```
telnet localhost 5554
```

Una vez dentro de la línea de comando del emulador, invocaremos el comando `geo` para suministrarle las coordenadas. Por ejemplo, las siguientes coordenadas corresponden a la Universidad de Alicante:

```
geo fix -0.51515 38.3852333
```

Si no queremos tener que ir a línea de comando, podemos utilizar la aplicación DDMS a la que se puede acceder de forma indepdiente o desde dentro de Eclipse. Dado que estamos ejecutando el emulador desde Eclipse, deberemos lanzar DDMS también dentro de este entorno. Para ello deberemos mostrar la vista *Emulator Control*. En ella veremos unos cuadros de texto y un botón con los que enviar las coordenadas al emulador, siempre que esté en funcionamiento.

**Advertencia**

Debido a un *bug* del SDK de Android, el DDMS no envía correctamente las coordenadas al emulador si nuestro *locale* no está configurado con idioma inglés. Para solucionar esto de forma sencilla, podemos editar el fichero `eclipse.ini` y añadir dentro de él la opción `-Duser.language.en`. Si no hacemos esto, el emulador recibirá siempre las coordenadas `0, 0`.

Para utilizar el *geocoder*, deberemos utilizar un emulador que incorpore las APIs de Google (para así poder acceder a la API de mapas). Además, dado que necesitará conectarse a Internet para obtener las direcciones, deberemos solicitar el permiso `INTERNET`.

**Advertencia**

En el emulador de la plataforma Android 2.2 no funciona correctamente el *geocoder*. Funcionará correctamente si utilizamos, por ejemplo, un emulador con Google APIs de nivel 3 (versión 1.5 de la plataforma).

## 5. Reconocimiento del habla

Implementar una nueva aplicación `habla` con un campo de texto y un botón. Al pulsar sobre el botón se lanzará el módulo de reconocimiento del habla, y una vez finalizado mostraremos lo que se haya reconocido en el campo de texto. Sólo podremos hacer este ejercicio de contamos con un micrófono.

**Nota**

Sólo podremos hacer este ejercicio si contamos con un dispositivo real, ya que el emulador no

soporta el reconocimiento del habla.