Ejercicios de pantalla táctil

Índice

1 Pantalla táctil..... 2
2 Gestos..... 2
3 Gestos personalizados (*).....3
4 Acelerómetro (*)..... 4

1. Pantalla táctil

Vamos a trabajar con la aplicación `Touch`, en la que mostraremos una caja en la pantalla (un rectángulo de 50x50) 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. Se deberá sobrescribir el método que trata los eventos de la pantalla táctil en la vista `VistaTouch`, y dentro de él sólo será necesario reconocer el evento `DOWN`. Para mover la caja deberemos modificar los valores de los campos `x`, `y`, haciendo que se posicione en las coordenadas en las que ha tocado el usuario.

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.

Ayuda

Esto último se puede conseguir de forma sencilla 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 función `collidesRectangle` ya definida en la vista nos permite comprobar si las coordenadas en las que se ha tocado quedan dentro de la caja.

2. Gestos

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

a) Modificar el ejercicio anterior para utilizar un detector de gestos para desplazar la caja. Utilizaremos el evento `onDown` para determinar si el gesto ha comenzado sobre la caja, y `onScroll` para desplazarla. Comenta el código del ejercicio anterior, para pasar a utilizar únicamente el reconocimiento de gestos.

b) Reconocer el evento *tap* realizado sobre la caja. Cuando esto ocurra se deberá cambiar el color de la caja (cambiar el valor de la propiedad *booleana* `colorAzul` cada vez que se produzca este evento, para así alternar entre color rojo y azul).

c) Reconocer el gesto *fling* ejercido sobre la caja. 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. Para ello deberemos asignar a las propiedades `vx`, `vy` el vector de velocidad del lanzamiento. Haciendo esto se dibujará el vector de velocidad

sobre la caja.

3. Gestos personalizados (*)

En el proyecto `LatasBox2D` tenemos implementado un videojuego que hace uso de un reconocedor de gestos propio, similar a `OnGestureListener`, con los siguientes gestos:

- `onDown`: El dedo se pone en pantalla.
- `onSingleTap`: Se da un toque corto en un punto de la pantalla.
- `onFling`: Realiza un lanzamiento.
- `onScrollMove`: Se mantiene el dedo y se desplaza.
- `onScrollUp`: Se levanta el dedo tras un *scroll*.

Estos dos últimos gestos no estaban en `OnGestureListener`. Nos permitirán saber cuándo se termina de hacer un *scroll*.

En el juego deberemos lanzar una bola para derribar una pila de latas. Vamos a implementar distintas formas de manejo. El esqueleto del oyente de los eventos definidos anteriormente se puede encontrar al final de la clase `GameScene`. Utiliza los eventos del reconocedor de gestos que consideres oportunos en cada uno de los siguientes casos:

a) Al pulsar rápidamente sobre un punto de la bola deberá lanzarse (como si fuese un billar). El código necesario para lanzar la bola dadas las coordenadas *(x,y)* del impacto es el siguiente:

```
if(simulation.hit(x, height - y)) {
    simulation.resetCurrentDamage();
    state = GameSceneState.GAME_SCENE_STATE_SIMULATION;
    shots++;
}
```

b) Hacer que la bola se lance con una determinada velocidad cuando la impulsemos con el dedo. Se lanzará en la dirección en la que hayamos movido el dedo, con el vector de velocidad (vx,vy) que le hayamos dado al lanzamiento. Utilizaremos un código como el siguiente en este caso:

```
simulation.launch(vel_x, -vel_y);
simulation.resetCurrentDamage();
state = GameSceneState.GAME_SCENE_STATE_SIMULATION;
shots++;
```

c) Hacer que funcione en modo "tirachinas", al estilo de *Angry Birds*. Para ello deberemos en primer lugar detectar que al poner el dedo en la pantalla se pone sobre la bola:

```
if(simulation.testBallPosition(x, height - y)) {
    isGrabbed = true;
}
```

Mientras movamos el dedo por la pantalla con la bola sujeta, moveremos la bola a la posición de nuestro dedo:

```
if(isGrabbed) {
    simulation.setBallPosition(x, height - y);
}
```

Cuando soltemos el dedo, la bola saldrá impulsada en la dirección opuesta a la dirección en la que la hayamos arrastrado:

```
if(isGrabbed) {
    int mul = 25;
    simulation.launch(mul*(x_ini - x_fin), mul*(y_fin - y_ini));
    simulation.resetCurrentDamage();
    state = GameSceneState.GAME_SCENE_STATE_SIMULATION;
    shots++;
}
```

4. Acelerómetro (*)

Hacer que la aplicación `Acelerometro` muestre en una tabla los valores de aceleración para las coordenadas X, Y, Z. Deberemos crear un oyente de eventos del acelerómetro, programar las lecturas con una periodicidad normal, y cada vez que obtengamos una lectura la mostraremos en los campos `tvAcelerometroX`, `tvAcelerometroY`, y `tvAcelerometroZ`.

Nota

Si sólo contamos con el emulador podemos conseguir que haya un cambio en la acelaración si cambiamos la orientación del dispositivo entre vertical y horizontal (pulsando *fn + ctrl + F11*)