Programación de los acelerómetros de dispositivos Android

Alberto Socas Mendoza

y

Antonio Jesús Ruiz Gómez

Periféricos y Dispositivos de Interfaz Humana

Índice

1. Introducción

Los dispositivos Android tienen sensores integrados de los que hablaremos en este trabajo, estos sensores pueden medir el movimiento, la aceleración, entre otros. Los sensores de Android son capaces de proporcionarnos datos y son bastante útiles para supervisar el movimiento o posicionamiento tridimensional del dispositivo.

Algunos de estos sensores se basan en hardware y otros en software, los basados en hardware son componentes físicos integrados en un dispositivo, estos miden propiedades específicas, como la aceleración, la intensidad del campo geomagnético o el cambio angular. Los sensores basados en software imitan a los basados en hardware, pero estos no son físicos, y se les denomina sensores virtuales, un ejemplo de los sensores de software son el sensor de la aceleración lineal y el sensor de gravedad.

2. Tipos de sensores en dispositivos Android

Para entrar un poco más en el tema, vamos a nombrar los diferentes tipos de sensores que puede tener un dispositivo Android. Estos admiten tres categorías de sensores:

- Sensores de movimiento

Miden las fuerzas de aceleración y las fuerzas de rotación en tres ejes (x,y,z). Incluye acelerómetros, sensores de gravedad, giroscopios y sensores del vector rotación.

- Sensores ambientales

Miden parámetros ambientales, como puede ser la temperatura y la presión del aire ambiental, la iluminación y la humedad. Pueden ser sensores como barómetros, fotómetros y termómetros.

- Sensores de posición

Miden la posición física de un dispositivo. Incluye sensores de orientación y magnetómetros.

En las siguientes imágenes (Ilustración 1 , Ilustración 2) podemos ver los distintos sensores que puede haber en un dispositivo Android:

Sensor	Tipo	Descripción	Usos habituales
TYPE_ACCELEROMETER	Hardware	Mide en m/s^2 la fuerza de aceleración que se aplica a un dispositivo en los tres ejes físicos (x, y, z), incluida la fuerza de gravedad.	Detección de movimiento (agitación, inclinación, etc.).
TYPE_AMBIENT_TEMPERATURE	Hardware	Mide la temperatura ambiente de la habitación en grados Celsius (°C). Consulta la siguiente nota.	Supervisión de la temperatura del aire.
TYPE_GRAVITY	Software o hardware	Mide en m/s^2 la fuerza de gravedad que se aplica a un dispositivo en los tres ejes físicos (x, y, z).	Detección de movimiento (agitación, inclinación, etc.).
TYPE_GYROSCOPE	Hardware	Mide en rad/s la velocidad de rotación de un dispositivo alrededor de cada uno de los tres ejes físicos (x, y, z).	Detección de rotación (agitación, giro, etc.).
TYPE_LIGHT	Hardware	Mide el nivel de luz ambiental (iluminación) en lx.	Control del brillo de la pantalla.
TYPE_LINEAR_ACCELERATION	Software o hardware	Mide en m/s^2 la fuerza de aceleración que se aplica a un dispositivo en los tres ejes físicos (x, y, z), excluyendo la fuerza de gravedad.	Supervisión de la aceleración a lo largo de un solo eje.
TYPE_MAGNETIC_FIELD	Hardware	Mide el campo geomagnético ambiental de los tres ejes físicos (x, y, z) en µT.	Creación de una brújula.
TYPE_ORIENTATION	Software	Mide los grados de rotación de un dispositivo alrededor de los tres ejes físicos (x, y, z). A partir de la API nivel 3, puedes obtener la matriz de inclinación y la matriz de rotación de un dispositivo mediante el uso del sensor de gravedad y el sensor del campo geomagnético de un dispositivo usando el sensor de gravedad junto con el método getRotationMatrix().	Determinación de la posición del dispositivo.

Ilustración 1

TYPE_PRESSURE	Hardware	Mide la presión del aire del ambiente en hPa o mbar.	Supervisa los cambios de la presión del aire.
TYPE_PROXIMITY	Hardware	Mide en cm la proximidad de un objeto con respecto a la pantalla de visualización de un dispositivo. Este sensor en general se usa para determinar si un dispositivo manual se está sosteniendo cerca del oído de una persona.	Posición del teléfono durante una llamada.
TYPE_RELATIVE_HUMIDITY	Hardware	Mide en valor de porcentaje (%) la humedad relativa del ambiente.	Supervisa el punto de condensación, la humedad absoluta y la humedad relativa.
TYPE_ROTATION_VECTOR	Software o hardware	Mide la orientación de un dispositivo mediante los tres elementos del vector de rotación del dispositivo.	Detección de movimiento y detección de rotación.
TYPE_TEMPERATURE	Hardware	Mide la temperatura del dispositivo en grados Celsius (°C). La implementación de este sensor varía según el dispositivo; en la API nivel 14 se reemplazó por el sensor TYPE_AMBIENT_TEMPERATURE.	Supervisión de temperaturas.

Ilustración 2

3. Marco de trabajo del sensor

Puedes acceder a los sensores disponibles en el dispositivo y adquirir datos sin procesar del sensor mediante el marco de trabajo del sensor de Android. Este proporciona clases e interfaces que ayudan a realizar tareas relacionadas con el sensor, veamos unos ejemplos:

- SensorManager

Esta clase sirve para crear una instancia del servicio del sensor, proporcionando varios métodos para acceder a sensores. También proporciona varias constantes del sensor que se usan para informar la exactitud del sensor y poder definir las velocidades de adquisición de datos y calibrar sensores.

- Sensor

Esta clase crea instancias de un sensor específico y proporciona diferentes métodos que permiten determinar las capacidades de un sensor.

- SensorEvent

Esta clase crea un objeto de evento de sensor, que proporciona información sobre un evento del sensor, incluye información como: los datos sin procesar del sensor, el tipo de sensor, la marca de tiempo, entre otros datos.

- SensorEventListener

Esta interfaz para crear dos métodos de devolución de llamada que reciben notificaciones cuando cambian los valores del sensor o cuando cambia la exactitud del sensor.

4. Disponibilidad de los sensores

La disponibilidad de los sensores varía de un dispositivo a otro, pero también puede cambiar de una versión a otra de Android. Vamos a resumir la disponibilidad de los sensores con la siguiente imagen:

Sensor	Android 4.0 (API nivel 14)	Android 2.3 (API nivel 9)	Android 2.2 (API nivel 8)	Android 1.5 (API nivel 3)
TYPE_ACCELEROMETER	Sí	Sí	Sí	Sí
TYPE_AMBIENT_TEMPERATURE	Sí	n/a	n/a	n/a
TYPE_GRAVITY	Sí	Sí	n/a	n/a
TYPE_GYROSCOPE	Sí	Sí	n/a[1]	n/a[1]
TYPE_LIGHT	Sí	Sí	Sí	Sí
TYPE_LINEAR_ACCELERATION	Sí	Sí	n/a	n/a
TYPE_MAGNETIC_FIELD	Sí	Sí	Sí	Sí
TYPE_ORIENTATION	Sí[2]	Sí[2]	Sí[2]	Sí
TYPE_PRESSURE	Sí	Sí	n/a[1]	n/a[1]
TYPE_PROXIMITY	Sí	Sí	Sí	Sí
TYPE_RELATIVE_HUMIDITY	Sí	n/a	n/a	n/a
TYPE_ROTATION_VECTOR	Sí	Sí	n/a	n/a
TYPE_TEMPERATURE	Sí[2]	Sí	Sí	Sí

Ilustración 3

5. El sensor Acelerómetro

Este sensor mide la aceleración aplicada al dispositivo, incluida la fuerza de gravedad. Este tipo de sensores utilizan el sensor estándar del sistema de coordinación.

Se recomienda usar el acelerómetro para supervisar el movimiento del dispositivo.

A continuación vamos a explicar como sería la configuración del sensor y mostraremos imágenes del resultado del sensor.

Utilizaremos el entorno de desarrollo AndroidStudio conectado con un dispositivo móvil para captar los resultados del sensor.

Configuración del Sensor

Para la clase de actividad principal tenemos un método onCreate en el que establecemos el diseño principal invocando setContentView.

En el siguiente paso aprovechamos la interfaz SensorEventListener , para usar la interfaz la clase de actividad main debes implementarla como se muestra en el fragmento de código siguiente.

```
1  public class Main extends Activity implements SensorEventListener {
2
3      /** Called when the activity is first created. */
4      @Override
5      public void onCreate(Bundle savedInstanceState) {
6          super.onCreate(savedInstanceState);
7          setContentView(R.layout.main);
8      }
9  }
```

Deberemos implementar dos métodos requeridos de la interfaz SensorEventListener, estos métodos son:

public void onSensor Changed(SensorEvent event);

public void onAccuracyChanged(Sensor sensor, int accuracy);

El método onSensorChanged detectará el gesto de agitar, se invocará cada vez que el sensor incorporado detecta un cambio.

En el método onCreate, inicializamos las variables que acabamos de declarar y registramos un receptor. Este sería el código.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    senSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
    senAccelerometer = senSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER
    senSensorManager.registerListener(this, senAccelerometer , SensorManager.SENSC
}
```

Para inicializar la instancia del SensorManager, invocamos a getSystemService para buscar la instancia del SensorManager del sistema, el cual a su vez usamos para acceder a los sensores del sistema. El método getSystemService se usa para obtener una referencia de un servicio del sistema al pasar el nombre del servicio. Con el administrador de sensores a nuestra disposición, obtenemos una referencia al acelerómetro del sistema invocando getDefaultSensor en el administrador del sensor y pasando el tipo de sensor que nos interesa. Luego registramos el sensor usando uno de los métodos públicos de SensorManager, registerListener. Este método acepta tres argumentos, el contexto de la actividad, un sensor y la velocidad a la que se nos envían los eventos del sensor.

Hay dos métodos que debemos anular, onPause y onResume. Estos son métodos de la clase main. Debemos anular el registro del sensor cuando la aplicación hiberne y registre el sensor nuevamente cuando la aplicación se reanude.

Detectar el gesto de sacudida

Vamos a ampliar la implementación del método onSensorChanged. Tomamos una referencia a la instancia Sensor utilizando la instancia SensorEvent. Comprobamos que obtenemos una referencia al tipo de sensor correcto, el acelerómetro.

El siguiente paso es extraer la posición del dispositivo en el espacio, los ejes x, y, z. Eje x movimiento lateral, eje y movimiento vertical, el eje z define el movimiento dentro y fuera del plano definido por los ejes x e y, otorgamos valores a esos ejes con una matriz flotante.

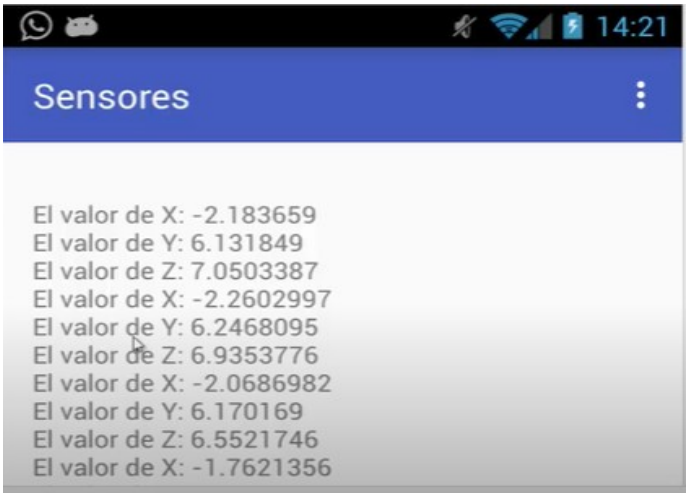
Utilizamos curTime para almacenar la hora actual del sistema y verificamos si han pasado más de 100 milisegundos desde la última vez que se invocó onSensorChanged, ya que este método se invoca varias veces por segundo y no necesitamos tantos datos.

Para acabar tenemos que detectar si le dispositivo ha sido sacudido o no. Usamos la clase math para calcular la velocidad del dispositivo. La variable SHAKE_THRESHOLD declarada estáticamente se usa para ver si se ha detectado o no un gesto de sacudida. La modificación de SHAKE_THRESHOLD aumenta o disminuye la sensibilidad.

```
public void onSensorChange(SensorEvent sensorEvent) {
    Sensor mySensor = sensorEvent.sensor;

    if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        float x = sensorEvent.values[0];
        float y = sensorEvent.values[1];
        float z = sensorEvent.values[2];

        long curTime = System.currentTimeMillis();

        if ((curTime - lastUpdate) > 100) {
            long diffTime = (curTime - lastUpdate);
            lastUpdate = curTime;

            float speed = Math.abs(x + y + z - last_x - last_y - last_z)/ diffTime * 10000;

            if (speed > SHAKE_THRESHOLD) {

            }

            last_x = x;
            last_y = y;
            last_z = z;
        }
    }

    texto.append("\n" + "El valor de X: " + last_x + "\n" + "El valor de Y: " + last_y + "\n" + "El valor
de Z:                          " + last_z)
}
```

Para mostrar los datos sólo nos quedaría añadir al método una instancia texto que utiliza TextView y llamando al método append nos muestre los valores del eje x, y, z.

Al ejecutar el programa nos aparecería esto en nuestro teléfono.



Bibliografia

code.tutsplus.com

developer.android.com