ESIGELEC irseem

Android et Innovation

Adnane Cabani

D1 272

cabani@esigelec.fr

Les capteurs

Introduction

- Trois types de capteurs :
    - Les détecteurs de mouvement
    - Les capteurs environnementaux
    - Les capteurs de position

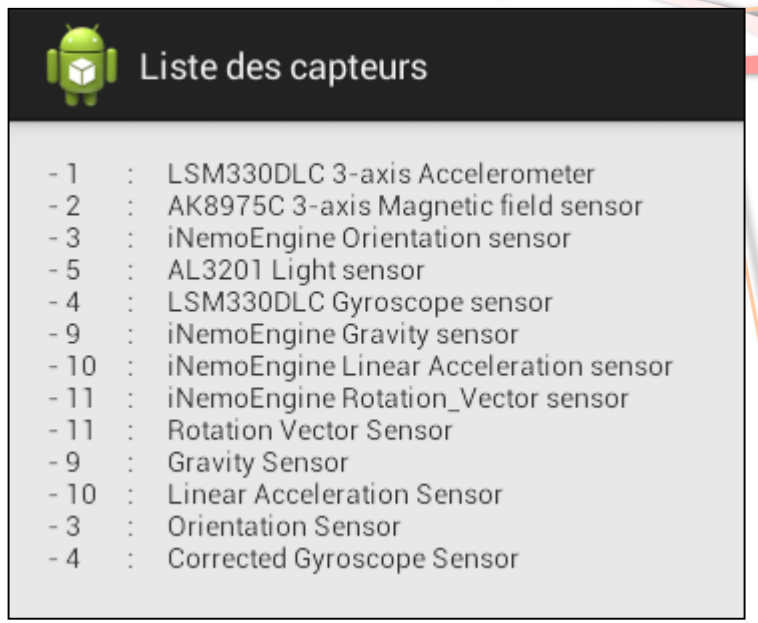
Quelques types de capteurs supportés

Capteur	Type	Description	Utilisation
TYPE_ACCELEROMETER	M	Mesure la force d'accélération en m/s^2	Détecter les mouvements
TYPE_GYROSCOPE	M	Mesure le taux de rotation en rad/s sur les trois axes x, y et z	Détection l'orientation de l'appareil
TYPE_LIGHT	M	Mesure le niveau de lumière ambiante en lux (lx)	Adapter la luminosité de l'écran
TYPE_ORIENTATION	L	Mesure le degré de rotation que l'appareil effectue sur les trois axes	Déterminer la position de l'appareil

Type M : matérielle
Type L : logicielle (se base sur plusieurs capteurs pour déduire et calculer des données)
http://developer.android.com/guide/topics/sensors/sensors_overview.html

Identifier les capteurs

```
public class MainActivity extends Activity {

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

        // Créer une instance de SensorManager
        SensorManager mySensorManager =
                (SensorManager) getSystemService(Context.SENSOR_SERVICE);

        // Récupérer la liste des capteurs
        List<Sensor> listeCapteurs = mySensorManager
                .getSensorList(Sensor.TYPE_ALL);

        // Affichage de la liste des capteurs
        TextView tv = (TextView) findViewById(R.id.textView1);
        for (Sensor sensor : listeCapteurs) {
            tv.append(" - " + sensor.getType()
                    + "\t : \t " + sensor.getName() + "\n");
        }
    }
}
```



Configurer l'application

```
<uses-feature
  android:name="string"
  android:required=["true" | "false"]
  android:glEsVersion="integer" />
```

- android:name : Indique un seul matériel ou fonctionnalité du logiciel utilisé par l'application
- android:required : (valeur par défaut : true)
  - true : présence indispensable
  - false : présence souhaitable
- android:glEsVersion : version openGL requise par l'application (valeur par défaut : OpenGL ES 1.0)

Exemple :

http:/

```
<uses-feature android:name="android.hardware.sensor.gyroscope"
    android:required="true" />
```

ıt.html

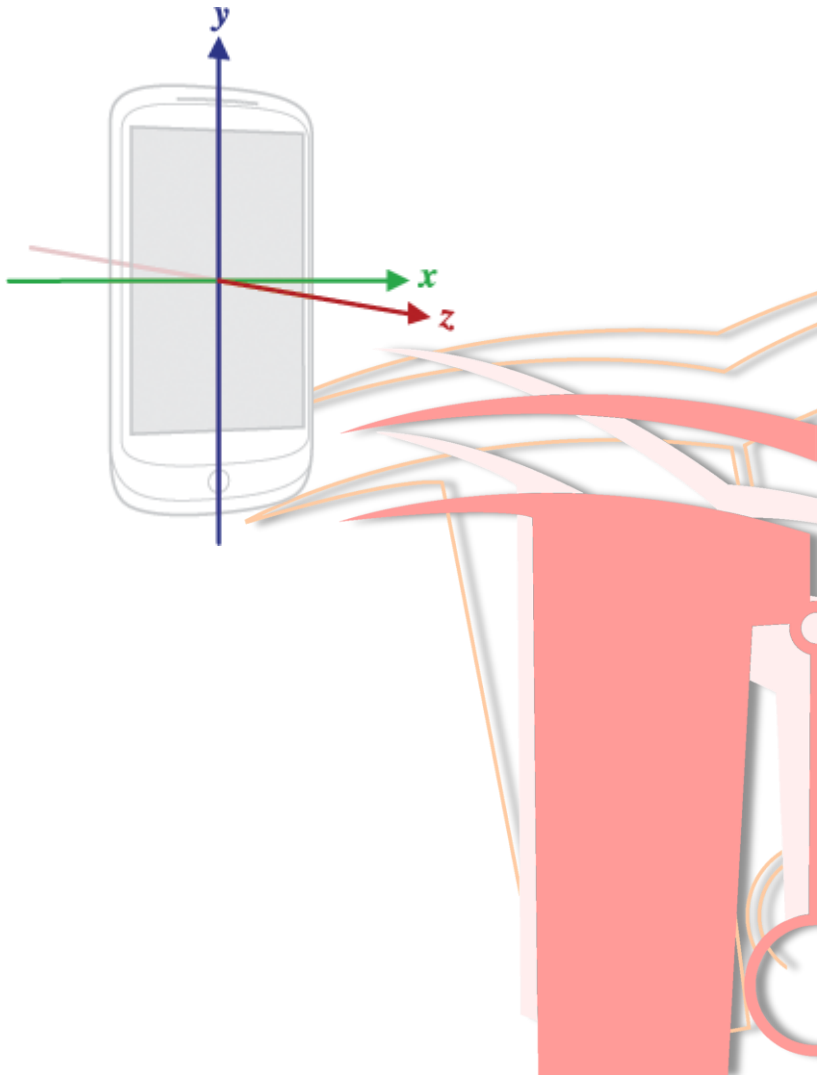
Vérifier l'existence d'un capteur

```
public class MainActivity extends Activity {

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

        // Créer une instance de SensorManager
        SensorManager mySensorManager = (SensorManager)
                getSystemService(Context.SENSOR_SERVICE);

        // Obtenir le capteur gyroscope par défaut,
        // retourne NULL s'il n'existe pas
        Sensor gyroscope = mySensorManager
                .getDefaultSensor(Sensor.TYPE_GYROSCOPE);

        if (gyroscope == null)
            Toast.makeText(this, "Pas de gyroscope !", Toast.LENGTH_LONG)
                    .show();
        else
            Toast.makeText(this, "Il existe au moins un gyroscope.",
                    Toast.LENGTH_LONG).show();
    }
}
```

Système de coordonnées

- Vérifier toujours l'orientation par défaut de l'appareil.

- Orientation :
  - Portrait (en général les Smartphones)
  - Paysage (beaucoup de tablette)



Bonnes pratiques

1. Annuler l'enregistrement de l'écouteur du capteur :
   - à la fin d'utilisation
   - activité en pause

   Si non,
   - L'acquisition des données continue

   ➔ Consommation de la batterie

```
private SensorManager mySensorManager;
  ...
@Override
protected void onPause() {
  super.onPause();
  mySensorManager.unregisterListener(this);
}
```

Bonnes pratiques

2. Ne pas bloquer la méthode **onSensorChanged()**
   Peut être appelée fréquemment (selon la fréquence d'utilisation du capteur).
3. Éviter d'utiliser des méthodes ou types de capteurs obsolètes
4. Vérifier la présence du capteur avant son utilisation
5. Choisir la bonne fréquence d'utilisation du capteur

Utiliser les capteurs

- Implémenter l'interface **SensorEventListener**
- S'abonner aux événements du capteur

  public boolean registerListener (SensorEventListener listener, Sensor sensor, int rate)

  - listener : l'objet SensorEventListener
  - sensor : le capteur à écouter
  - rate : fréquence d'écoute en microsecondes

Les capteurs de mouvement

- Tous les capteurs de mouvement retournent des tableaux multidimensionnels

Capteur	donnée	Description	Unité de mesure
TYPE_ACCELEROMETER	SensorEvent.values[0]	Acceleration force along the x axis (including gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (including gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (including gravity).	
TYPE_GRAVITY	SensorEvent.values[0]	Force of gravity along the x axis.	m/s^2
	SensorEvent.values[1]	Force of gravity along the y axis.	
	SensorEvent.values[2]	Force of gravity along the z axis.	

http://developer.android.com/guide/topics/sensors/sensors_motion.html

Exemple - Accéléromètre

```
public class MainActivity extends Activity implements SensorEventListener {

    private SensorManager mySensorManager;
    private Sensor accelerometre;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Récupérer le gestionnaire de capteurs
        mySensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
        // Récupérer l'accéléromètre
        accelerometre = mySensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    }

    ...

}
```

Exemple - Accéléromètre

```
@Override
protected void onPause() {
    super.onPause();
    mySensorManager.unregisterListener(this);

}

@Override
protected void onResume() {
    super.onResume();
    mySensorManager.registerListener(this, accelerometre, SensorManager.SENSOR_DELAY_UI);
}

@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {
    Toast.makeText(MainActivity.this, "onAccuracyChanged()", Toast.LENGTH_SHORT).show();
}
```

Exemple - Accéléromètre

```
@Override
public void onSensorChanged(SensorEvent event) {
    float x, y, z;
    if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
        x = event.values[0];
        y = event.values[1];
        z = event.values[2];

        TextView tvX = (TextView) findViewById(R.id.tvX);
        TextView tvY = (TextView) findViewById(R.id.tvY);
        TextView tvZ = (TextView) findViewById(R.id.tvZ);

        tvX.setText(Float.toString(x));
        tvY.setText(Float.toString(y));
        tvZ.setText(Float.toString(z));
    }
}
```

Il est utile d'utiliser des filtres passe-bas et passe-haut pour éliminer les forces gravitationnelles et réduire le bruit.