Tutoriel - Capteurs de mouvement

LAMOOT Romain (FI)
21/11/2018

Résumé

Les capteurs font partis de notre quotidien, que ce soit pour récupérer notre position à un instant T ou pour la rotation automatique de l'écran de nos smartphones, nous utilisons des capteurs sans parfois même s'en apercevoir.

Dans ce TP, nous verrons comment utiliser des capteurs (dans notre cas : l'accéléromètre et le gyroscope) et comment récupérer leurs données. Nous verrons aussi comment créer une petite animation graphique grâce aux données transmises par l'accéléromètre.

Pré-requis

- *Programmation en langage Java / Android*
- *Savoir utiliser Android Studio*

Code source

Code source **initial** : https://github.com/rOmAiin062/TP-Capteurs_mouvement/tree/Version_init

Code source **final** : https://github.com/rOmAiin062/TP-Capteurs_mouvement/tree/Version_finale

Explications du TP

Tout d'abord, on retrouve 3 grandes catégories de capteurs :

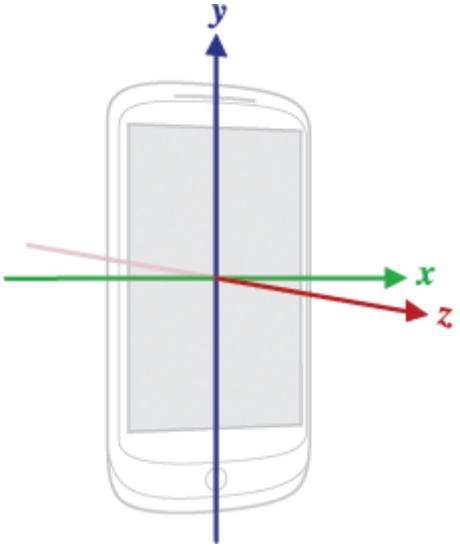
- Les capteurs de mouvements : Accéléromètre, capteur de gravité, gyroscope...
    - Ces capteurs mesurent les forces d'accélération et de rotation selon trois axes.
- Les capteurs de positions : Magnétomètre, capteur d'orientation...
    - Ces capteurs mesurent la position physique d'un appareil.
- Les capteurs d'environnements : Température, pression de l'air, éclairage, humidité…
    - Ces capteurs mesurent divers paramètres environnementaux.

Dans ce TP, nous allons travailler sur deux types de capteurs : l'accéléromètre et le gyroscope. L'accéléromètre permet de mesurer l'accélération linéaire tandis que le gyroscope permet de calculer la vitesse angulaire du smartphone.

L'accéléromètre mesure la force d'accélération en m/s² appliquée à l'appareil sur les trois axes physiques (x, y et z), y compris la force de gravité. Il est utilisé pour détecter les mouvements du smartphone (secousse, inclinaison, etc)

Le gyroscope mesure la vitesse de rotation d'un appareil en radian/s autour de chacun des trois axes physiques (x, y et z). Il est utilisé pour détecter les mouvements de rotation.

Système de coordonnées (par rapport au smartphone) utilisé par l'API du capteur.
L'axe des X est l'axe horizontal de l'écran, l'axe des Y est le vertical et l'axe des Z est orthogonal à l'écran.



(Sources : https://developer.android.com/reference/android/hardware/SensorEvent)

<u>Les étapes pour utiliser les capteurs du smartphone :</u>

- Étape 1 : Créer une activité qui implémente l'interface "SensorEventListener"
- Étape 2 : Instancier un gestionnaire de capteurs (SensorManager)
- Étape 3 : On récupère le capteur que l'on veut utiliser via le SensorManager
- Étape 4 : S'abonner/se désabonner aux événements du capteur
- Étape 5 : Implémenter les méthodes de l'interface "SensorEventListener" (onAccuracyChanged, onSensorChanged)

Voir l'exemple ci-dessous.

Etape 1: Déclarations des capteurs et abonnements

Dans un premier temps, nous allons instancier un gestionnaire de capteurs grâce auquel nous allons pouvoir instancier des capteurs sur lesquels nous allons travailler ensuite. Nous pourrons alors récupérer les informations des capteurs instanciés.

Nous allons donc déclarer **3 attributs** : le premier pour gestionnaire de capteurs et les deux autres pour les capteurs. Nous allons travailler ici avec l'accéléromètre et le gyroscope.

```
// Gestionnaire de capteurs
private SensorManager mSensorManager;
// Capteur 'Accélérometre'
private Sensor sensorAccelerometer;
// Capteur 'Gyroscope'
private Sensor sensorGyroscope;
```

Il faut maintenant instancier ces attributs dans la méthode "protected void onCreate(Bundle savedInstanceState)" :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   // Instanciation du gestionnaire de capteurs
   mSensorManager = (SensorManager)
   getSystemService(Context.SENSOR_SERVICE);
   // Instanciation des deux capteurs utilisés (Accéléromètre +
   Gyroscope)
   sensorAccelerometer =
   mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
   sensorGyroscope =
   mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
   // Récupération des dimensions de l'écran
   getScreenDimension();
}
```

N.B : Ne tenez pas compte de la méthode getScreenDimension(); pour le moment.

Maintenant que nous avons instancié et récupéré nos capteurs. Nous allons nous "abonner" à ceux-ci. C'est-à-dire que nous allons écouter et recevoir les données à chaque changement d'état du capteur / chaque variation du capteur.

Dans la méthode **protected void onResume()**, on ajoute des 'Listeners' sur les capteurs instanciés plus haut.

```
// On s'abonne aux capteurs - si un événement arrive, on le récupère et on le traite
mSensorManager.registerListener(this, sensorAccelerometer,
mSensorManager.SENSOR_DELAY_UI);
mSensorManager.registerListener(this, sensorGyroscope,
mSensorManager.SENSOR_DELAY_UI);
```

Ensuite, il faut faire la même chose mais dans le sens inverse : on se désabonne des événements du capteur lorsqu'on quitte l'application.

Dans la méthode **protected void onPause()**, on se désabonne des 'Listeners' sur les capteurs instanciés plus haut.

```
// On se "désabonne" des capteurs - si un événement arrive, on ne s'en occupe plus
mSensorManager.unregisterListener(this, sensorAccelerometer);
mSensorManager.unregisterListener(this, sensorGyroscope);
```

Pensez-vous à vérifier que les permissions pour accéder aux capteurs (accéléromètre / gyroscope) soient déclarées dans le fichier ***AndroidManifest.xml***.

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

Etape 2: Récupération des données transmises par les capteurs

Maintenant que nous avons instancier nos capteurs et que nous nous sommes abonnés à leurs événements, nous allons récupérer les données envoyées par les capteurs.

Vous l'avez peut être remarqué, notre MainActivity implémente l'interface **SensorEventListener**.

Cette interface est utilisée pour recevoir des notifications de SensorManager lorsqu'il y a de nouvelles données émises par capteur. La réception des données se fait plus particulièrement grâce à la méthode **public void onSensorChanged(SensorEvent event)** qui est appelée automatiquement à chaque nouvel événement déclenché par le capteur associé au gestionnaire de capteurs.

Dans notre méthode "**onSensorChanged(SensorEvent event)"**, nous allons traité les deux cas :

```
// Cas 1 : On traite un événement venant du capteur 'Accéléromètre'
    if (event.sensor.getType() == sensorAccelerometer.TYPE_ACCELEROMETER) {
        // Récupération des valeurs du capteur 'Accéléromètre'
        float axe_x = event.values[0]; // axe x
        float axe_y = event.values[1]; // axe y
        float axe_z = event.values[2]; // axe z
}
// Cas 2 : On traite un événement venant du capteur 'Gyroscope'
      else if(event.sensor.getType() == sensorGyroscope.TYPE_GYROSCOPE){
          // Récupération des valeurs du capteur 'Gyroscope'
          float vitesse_x = event.values[0]; // vitesse angulaire autour de x
          float vitesse_y = event.values[1]; // vitesse angulaire autour de y
          float vitesse_z = event.values[2]; // vitesse angulaire autour de z
} else { return; }
```

N.B : Il est aussi possible de récupérer la précision du capteur (event.accuracy) et le moment où l'événement a été lancé (event.timestamp)

Etape 3: Affichage des données récupérées

Après avoir récupérées les données transmises par les capteurs, nous allons les afficher dans nos textView (cf activity_main.xml).

Dans la méthode *onSensorChanged(SensorEvent event)*, dans le cas d'un événement émis par l'accéléromètre : (Cas 1 - TYPE_ACCELEROMETER)
On récupère les TextView du layout et on ajoute les valeurs récupérées :

```
// Affichage des valeurs récupérées ci-dessus
        TextView tx = (TextView) findViewById(R.id.accelerometre_x);
        TextView ty = (TextView) findViewById(R.id.accelerometre_y);
        TextView tz = (TextView) findViewById(R.id.accelerometre_z);
        tx.setText(round(axe_x) + "m/s^2");
        ty.setText(round(axe_y) + "m/s^2");
        tz.setText(round(axe_z) + "m/s^2");
```

Toujours dans la méthode *onSensorChanged(SensorEvent event)*, on fait la même chose pour le gyroscope : (Cas 2 - TYPE_GYROSCOPE)

```
// Affichage des valeurs récupérées ci-dessus
         TextView tx = (TextView) findViewById(R.id.gyro_x);
         TextView ty = (TextView) findViewById(R.id.gyro_y);
         TextView tz = (TextView) findViewById(R.id.gyro_z);
         tx.setText(round(vitesse_x) + "rad/s");
         ty.setText(round(vitesse_y) + "rad/s");
         tz.setText(round(vitesse_z) + "rad/s");
```

Maintenant, si vous lancer votre application, vous allez voir changer les valeurs écrites en fonction des mouvements que vous faites avec votre smartphone.

Vous savez maintenant comment utiliser les capteurs (accéléromètre et gyroscope), récupérer et afficher leurs valeurs.

Pour plus d'informations, consulter la rubrique "*Informations complémentaires*".

Etape 4: Animation d'une image en fonction des données de l'accéléromètre

Pour aller plus loin, je vous propose maintenant d'animer une petite image en fonction des valeurs reçues par l'accéléromètre. Dans ce cas, seuls deux axes nous intéressent : l'axe x (horizontal) et l'axe y (vertical).

Il faut tout d'abord ajouter deux nouveaux attributs à notre MainActivity. Ces attributs contiendront la dernière position de l'image (x,y).

```
// Derniere position de l'image
private float last_x, last_y;
```

Il faut ensuite récupérer l'imageView et mettre à jour ses coordonnées en fonction des données reçues depuis l'accéléromètre.

Dans la méthode *onSensorChanged(SensorEvent event)* , dans le cas de l'accéléromètre (Cas 1 - TYPE_ACCELEROMETER), il faut ajouter à la suite :

```
// Animation de l'image en fonction des valeurs récupérées
            ImageView img = (ImageView) findViewById(R.id.imageView);
            last_x -= axe_x;
            last_y += axe_y;
             // On met à jour l'image sur l'écran
            if((0 <= last_x) && (last_x <= screen_witdh - 40)){
                img.setX(last_x);
            }
            if((0 <= last_y) && (last_y <= screen_height - 40)){
                img.setY(last_y);
            }
```

N.B : C'est ici qu'intervient la méthode getScreenDimension(); qui va récupérer les dimensions de l'écran et va mettre les valeurs dans les attributs 'screen_width' et 'screen height' qui sont utilisés ci-dessus.

Vous pouvez à nouveau exécuter l'application et tester l'animation de l'image.

Informations complémentaires

- https://developer.android.com/guide/topics/sensors/ : Documentation officielle concernant l'utilisation des capteurs.

- https://mathias-seguy.developpez.com/tutoriels/android/utiliser-capteurs : Tutoriel en français avec utilisations d'autres capteurs.

- https://github.com/googlesamples/android-AccelerometerPlay : Petit exemple d'animation possible avec les capteurs.

- https://www.youtube.com/watch?v=zVoxUMitLV8 : Pour mieux comprendre les données reçues depuis le gyroscope.

- https://fr.wikipedia.org/wiki/Acc%C3%A9l%C3%A9rom%C3%A8tre : Pour plus d'informations sur l'accéléromètre

- https://fr.wikipedia.org/wiki/Gyroscope : Pour plus d'informations sur le gyroscope