Utilisation d’un capteur:

Commencer par créer une nouvelle application (onglet “**File -> Close Project”** ,pour revenir au “launcher Android”).

Dans ce tutoriel nous allons voir comment implémenter un capteur, en l’occurrence un Accéléromètre, récupérer ses données et les exploiter.

A savoir que certaines dépendances seront à ajouter au cours du tutoriel, pour cela dès qu’un text apparaît en rouge faites alt + enter après avoir cliqué dessus et cliquer sur importer la classe.

Dès lors que l’on veut utiliser des fonctionnalités propres à un téléphone on va devoir le spécifier dans le manifest.xml qui se trouve dans le répertoire manifest. On va donc ouvrir le fichier et y ajouter les lignes suivantes:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="fr.macompagny.spriteandsensor">

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

Nous n'aurons pas besoin du texte disponible par défaut dans le mainActivity.xml, donc si vous le voulez supprimer le, ce sera mieux pour l’affichage final.

Avant d'ajouter quoi que ce soit dans la classe, on va devoir implémenter la classe SensorEventListener que vous pouvez voir en dessous, normalement une fois fait vous aurez une erreur, cliquez dessus , faites alt + enter et implémentez les méthodes (les deux premières).
Pour pouvoir utiliser n’importe quel capteur de notre appareil on va devoir déclarer à minima les deux premières variables en dessous de la classe, le SensorManager qui accèdera aux capteurs du mobile, et le Sensor qui correspondra au capteur que vous souhaitez utiliser:

```
public class MainActivity extends AppCompatActivity implements SensorEventListener {
    private SensorManager senSensorManager; // permet d'accéder aux sensors du support
    private Sensor senAccelerometer; //sensor du type accelerométre

    private int accX;
    private int accY;
```

Dans le onCreate(), on va donner accès à notre SensorManager aux capteurs et via celui-ci on va lier notre seconde variable avec l'Accéléromètre, soit on ajouter les lignes suivantes: *La dernière ligne met sur écoute le capteur.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);


    senSensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE); //accède aux sensors du système
    senAccelerometer = senSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER); //lie la var avec l'accelerometre
    // le laps de temps minimum entre deux activations des fonctions liées aux sensors
    senSensorManager.registerListener((SensorEventListener) this, senAccelerometer ,  samplingPeriodUs: 1000000);

}
```

On va ensuite récupérer certaines données de notre capteur via le code ci-dessous que vous allez ajouter(dans la méthode de l'interface implémentée) et les stocker dans deux variables globales que nous avons initialisé au tout début de notre programme:

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    Sensor mySensor = sensorEvent.sensor;

    if (mySensor.getType() == Sensor.TYPE_ACCELEROMETER) { //vérifie que les types correspondent
        double x = sensorEvent.values[0];
        double y = sensorEvent.values[1];
        accX = (int) x;
        accY = (int) y;
    }
}
```

Dans la deuxième fonction override nous n'aurons pas besoin d'ajouter du code:

```
@Override
public void onAccuracyChanged(Sensor sensor, int accuracy) {

}
```

Pour finir, au niveau de la base de l'implémentation d'un capteur et de la récupération de ses données, on va ajouter le code ci dessous qui permettra respectivement d'arrêter l'écoute du capteur quand l'application sera en pause, ainsi que le réécouter quand l'application se relancera:

```
protected void onResume() {
    super.onResume();
    //réutilise le sensor de la var contenant l'accéléromètre, et indique la fréquence de rafraîchissment des fonctions
    senSensorManager.registerListener( listener: this, senAccelerometer,  samplingPeriodUs: 1000000);


}

protected void onPause() {
    super.onPause();
    senSensorManager.unregisterListener(this); //arrête d'utiliser les sensors pendant que l'activité est en pause
}
```

Avant d'entamer la suite ajouter le code suivant dans votre AndroidManifest.xml disponible dans le répertoire manifest:

```
<activity android:name=".MainActivity" android:screenOrientation="portrait">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Fermez Android Studio (ou faites "Close Project" disponible dans l'onglet file).

Maintenant que vous avez vu la base, je vous mets à disposition un exemple d'utilisation du capteur dans une application, vous aurez besoin de récupérer deux fichiers disponibles ci-dessous:

https://drive.google.com/open?id=1Hl_XeViw3i3KqF4kyN5gqPaiPEYp_E7

Puis vous irez dans le dossier contenant votre application à l'adresse ci-dessous, vous remplacez le contenu par les deux fichiers:

...\DossierContenantVosProjets\NomApp\app\src\main\java\fr\macompagny\NomApp

*Évidemment ce qui est en rouge correspond au nom que vous avez mis.

Désormais vous pouvez ouvrir le projet à nouveau, vérifiez si les .java sont bien là, puis testez l'application pour vérifier qu'il n'y a pas de problème.

Résumé des changements:

Donc, parmis les deux fichiers on a ajouté une classe rectangle qui sera notre "Joueur", elle dispose d'attributs comme des coordonnées pour la deux dimensions, ainsi que des méthodes pour les manipuler, que ce soit les modifier ou bien les récupérer.

Dans la classe principale **"MainActivity"**, on va utiliser un thread en plus de celui de l'interface utilisateur pour éviter de le surcharger et donc de ralentir voir entraîner un crash de l'application (le thread se lance au passage de l'application à l'état **onResume**).
Le thread va effectuer en boucle la méthode run tant que nous ne changeons pas l'état de l'application. Cette méthode run va d'un côté appeler la méthode update qui va s'occuper de toute la partie calcul, ainsi que la méthode onDraw qui s'occupera comme son nom l'indique d'afficher ce que nous lui passons, ici notre rectangle.
Les données de l'accéléromètre sont traitées et transformées en mouvement pour le rectangle dès que l'accélération mesurée est considérée comme suffisante.