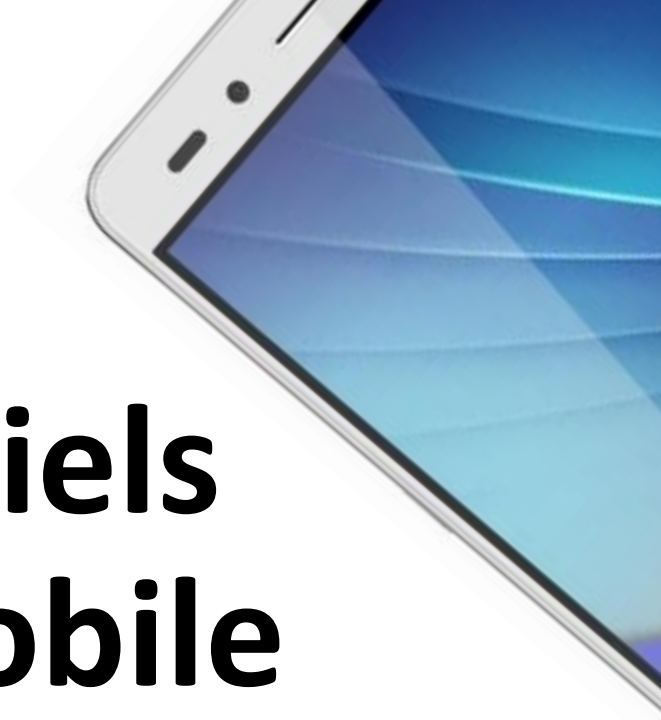

Environnements Logiciels pour l'Informatique Mobile

Android : Capteurs & Actionneurs

Présentation

- Polytech'Nice-Sophia 2012 (IAM)

- 5 ans chez Sopra-Steria
  - Développement
  - Architecture
  - Projets innovants
  - Formation

- gregory.marro@soprasteria.com

Quelles informations peut-on avoir ?

- Les capteurs
- L’état du périphérique
- Les données du périphérique
- Les données fournies par d’autres applications

Du bon usage des Listeners…

- Afin de limiter la consommation de la batterie et des ressources, il est largement conseillé d'utiliser les méthodes
  - onPause() : Suspension de l'écoute
  - onResume() : Reprendre l'écoute

- Utilisation de registerListener() et unregisterListener()

Les capteurs

- Représentent des données physiques
- Autonomes : ne requiert pas de tiers (Data, satellite…)
- 3 catégories : mouvement, environnement, position
- Attention, certains capteurs ne sont pas disponibles sur tous les périphériques et la précision n’est pas la même !
  - Penser systématiquement à en vérifier la présence

L’utilisation des capteurs

- A déclarer dans le Manifest.xml : attribut *required* : permet que le capteur soit un prérequis

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

- Utilisation du SensorManager pour accéder aux capteurs

```
SensorManager sensorManager = (SensorManager)getSystemService(Context.SENSOR_SERVICE);
```

- Utilisation de la classe Sensor pour chaque capteur

```
Sensor accelerometre = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELETOMETER);
if(accelerometre != null)
```

L’utilisation des capteurs

- Création d’un listener sur le capteur :
  - Pour le changement de valeur
  - Pour le changement de précision

```
final SensorEventListener mSensorEventListener = new SensorEventListener() {
  public void onAccuracyChanged(Sensor sensor, int accuracy) {
    // Que faire en cas de changement de précision ?
  }

  public void onSensorChanged(SensorEvent sensorEvent) {
    // Que faire en cas d'évènements sur le capteur ?
  }
};
```

- Enregistrement du Listener sur notre capteur :

```
  mSensorManager.registerListener(mSensorEventListener, mAccelerometer,
SensorManager.SENSOR_DELAY_NORMAL);
```

Les capteurs de déplacement

Type	Unité	Valeurs	Description
Accéléromètre	Mètre/seconde²	[x][y][z]	Inclus la gravité
Gyroscope	Rad/seconde	[x][y][z]	
Gravité	Mètre/seconde²	[x][y][z]	
Gyroscope non calibré	Rad/seconde	[rate.x][rate.y][rate.z][drift.x][drift.y][drift.z]	
Accélération linéaire	Mètre/seconde²	[x][y][z]	Exclus la gravité
Compteur de pas	pas	[x]	

Les capteurs de déplacement

Type	Unité	Valeurs	Description
Champ magnétique	µTesla	[x][y][z]	Mesure le champ magnétique
Orientation	Degrés	[x][y][z]	Mesure l'angle depuis le Nord
Proximité	Mètre	[x]	Mesure la distance entre le capteur et l'objet le plus proche
Rotation	-	[x][y][z]	Rotation du périphérique selon les différents axes

Les capteurs d’environnement

Type	Unité	Valeurs	Description
Humidité	%	[x]	Mesure de l’humidité ambiante
Lumière	Lux	[x]	Mesure la luminosité
Pression	KPascal	[x]	Mesure la pression
Température	Celsius	[x]	Mesure la température du périphérique
Température ambiante	Celcius	[x]	Mesure de la température ambiante

TD : Lister les capteurs

- Créer une application Android qui liste les capteurs du SensorManager

- Au clic sur un des éléments, un Toast affiche les valeurs du capteur

- Quand une valeur d'un capteur change, l'élément de la liste correspondant change de couleur quelques instants

L’état du périphérique

- Permet de connaître l’état du périphérique
- Relations avec les tiers (GSM, GPS…)
- Données sur le périphérique

La géolocalisation

- Mise en commun de deux méthodes :
  - Par le réseau (distance des antennes, point d'accès WIFI à proximité)
  - Par le GPS

- L'utilisation des 2 possibilités simultanément permet :
  - Un gain de vitesse
  - Un gain de batterie
  - Une solution de « backup »

La géolocalisation

- Ajout des permissions dans le Manifest.xml
  - Attention, FINE implique COARSE

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

- Utilisation du LocationManager

```
LocationManager locationManager =
(LocationManager)getSystemService(Context.LOCATION_SERVICE);
```

- Demander la mise à jour de la position, avec son Listener

```
locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 60000, 150, new
LocationListener() {

  @Override
  public void onStatusChanged(String provider, int status, Bundle extras) {

  }

  @Override
  public void onProviderEnabled(String provider) {
```

L’accès au réseau

- Mise à jour des permissions dans le Manifest.xml
- Utilisation du ConnectivityManager
- Permet de tester la connexion Wifi & GSM
- Rester à l’écoute de l’état de la connexion avec le BroadcastReceiver
- Attention, les accès réseaux doivent être réalisés sur un Thread séparé

TP :

- A partir de l'application précédente, ajouter 1 indicateur permettant de visualiser l'état de la connexion en bas de l'écran:
  - Vert : WIFI + GSM
  - Orange : 1 des 2
  - Rouge : Aucun

- Ajouter ensuite 1 champ permettant de visualiser les coordonnées de la géolocalisation en bas de l'écran

- Ajouter un bouton permettant de forcer la mise à jour de la localisation

Les actions sur le périphérique

- Prises d’images
- Utiliser le réseau GSM/Data
- Faire vibrer le périphérique
- Utiliser l’USB
- …

Les actions sur le périphérique

- Utilisation des Intents
- Fonctionnalités présentent grâce au getSystemService()
- Doc : https://developer.android.com/reference/android/content/Context.html

Les Intents

- Permet la communication entre composants :
  - De manière explicite
  - De manière implicite

- Contient les informations sur les actions et les données à fournir au destinataire

Les Intents explicites

- Gestion de plusieurs activités au sein d'un même projet
  - Ajouter la nouvelle activité dans le Manifest

- Permet d'imposer le destinataire de l'Intent

```
Intent intent = new Intent(Activite_de_depart.this, Activite_de_destination.class);
```

- Avec ou sans retour
  - Sans :

```
Intent secondeActivite = new Intent(FirstActivity.this, secondeActivity.class);

startActivity(secondeActivite);
```

Les Intents explicites

- Avec :

```
Intent secondeActivite = new Intent(FirstActivity.this, secondeActivity.class);

startActivityForResult(secondeActivite, REQUEST_ID);
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

  if (requestCode == REQUEST_ID) {

    if (resultCode == RESULT_OK) {

      // Traitement du résultat
    }
  }
}
```

Les Intents implicites

- Permet de déclencher une action sans connaitre le destinataire
- Création d’une URI
- Utilisation d’une Action

Les Intents implicites

- L'URI est composée :

```
<schéma> : <information> { ? <requête> }
```

  - D'un shéma : http:// sms: tel: geo: …
  - D'une information (coordonnées GPS, numéro de téléphone …)
  - D'une requête (optionnelle)

```
Uri exampleUri = Uri.parse("sms:0606060606,0606060607?body=Hello,World");
```

Les Intents implicites

- L'Action est une constante de la classe Intent

```
example.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    Uri telephone = Uri.parse("tel:0606060606");
    Intent secondeActivite = new Intent(Intent.ACTION_DIAL, telephone);
    startActivity(secondeActivite);
  }
});
```

TP :

- Créer une application qui permet de :
  - Faire vibrer le téléphone
  - Envoyer un SMS
  - Ouvrir l’activity des capteurs réalisée précédemment
  - Ouvrir un navigateur pour une URL saisie
  - Prendre une photo