Environnements Logiciels pour l'Informatique Mobile

Android : capteurs, actionneurs et vues

Présentation

- Polytech’Nice-Sophia 2012 (IAM)

- 6 ans chez Sopra-Steria
  - Architecture
  - Projets innovants
  - Formation

- gregory.marro@soprasteria.com

Capteurs

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

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

Actionneurs

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);
  }
});
```

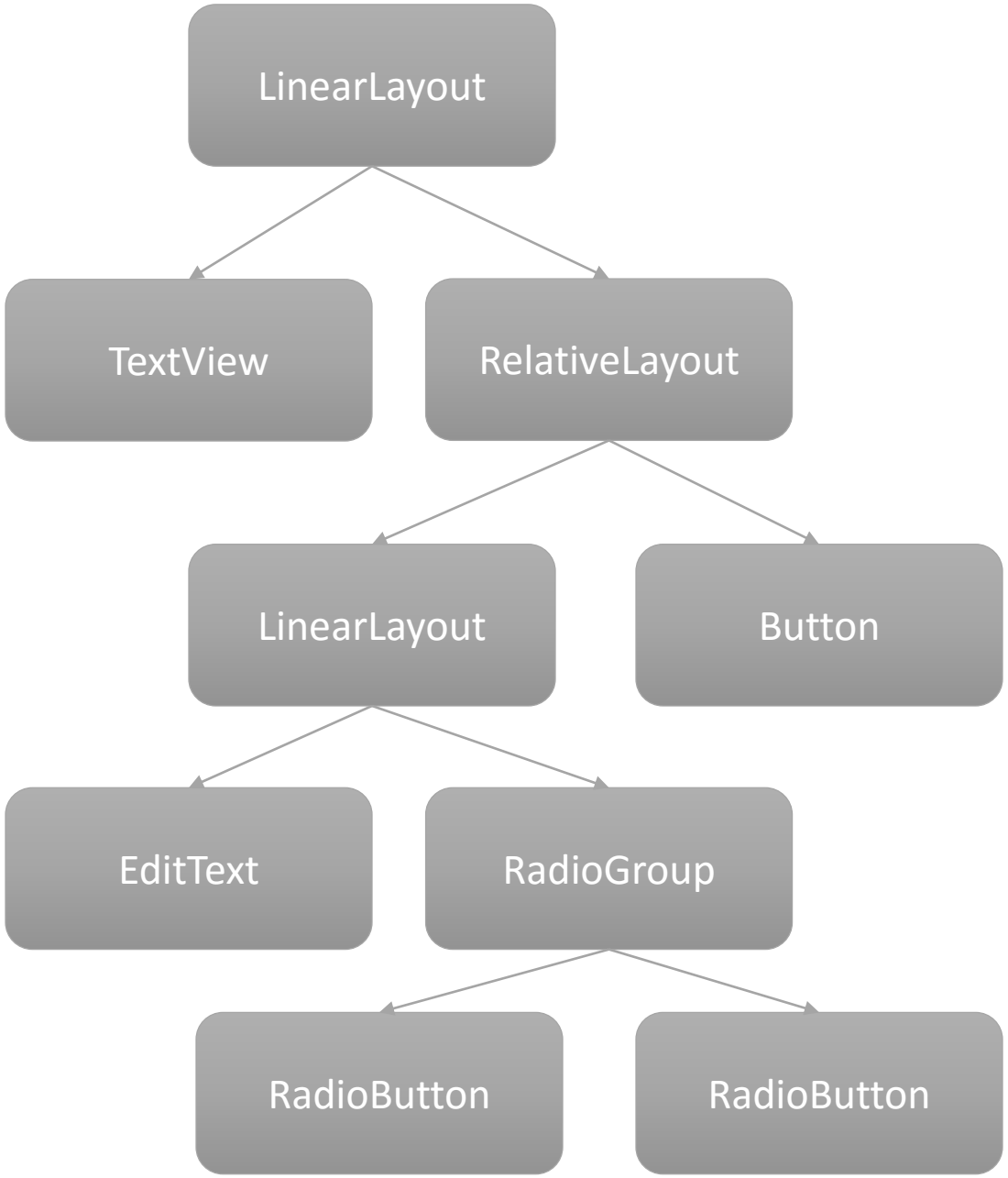
Les vues

Les listes

- Fait appel à des Adapter qui vont construire, pour chaque élément de la liste, une vue :
  - ArrayAdapter : pour les listes d'éléments simples
  - SimpleAdapter : pour les objets plus complexes
  - CursorAdapter : pour les requêtes en base de données
- Chacune de ces vues sera ensuite liée par un AdapterView qui permet la création de la liste, la gestion du scroll etc. On peut citer :
  - ListView (liste des contacts)
  - GridView (gallerie d'images)
  - Spinner

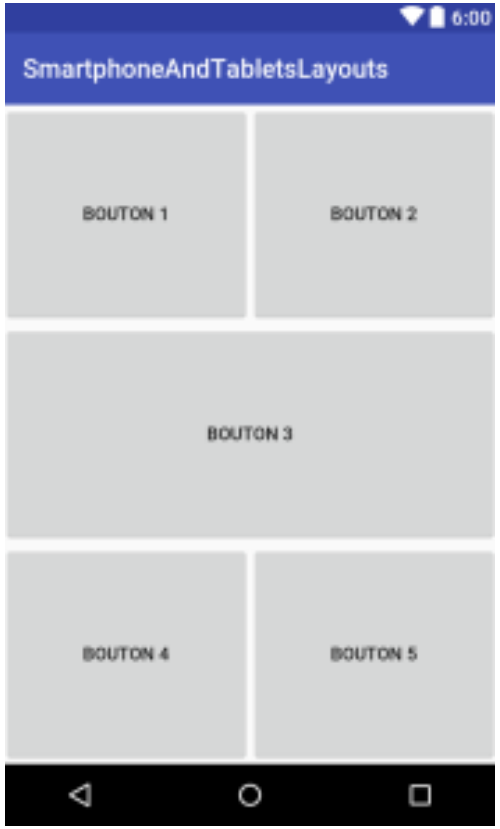
L’utilisation des Layouts

- Permet de disposer les Layouts/Widgets dans la vue
- Plusieurs manières de réaliser une même vue
- Toujours penser aux différentes tailles d’écran



LinearLayout

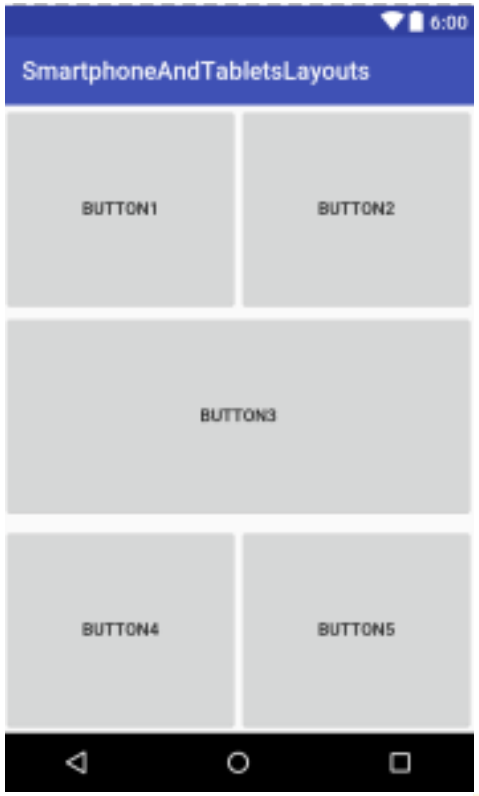
- Horizontal ou vertical
- Pas de chevauchement possible
- Penser au ScrollView
- Attention, si un élément fait toute la hauteur, il cache les autres




```
<LinearLayout xmlns:android="http://schemas.android.co
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight=".33">

        <Button
            android:id="@+id/un"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".5"
            android:text="Bouton 1" />

        <Button
            android:id="@+id/deux"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".5"
            android:text="Bouton 2" />
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
```

RelativeLayout

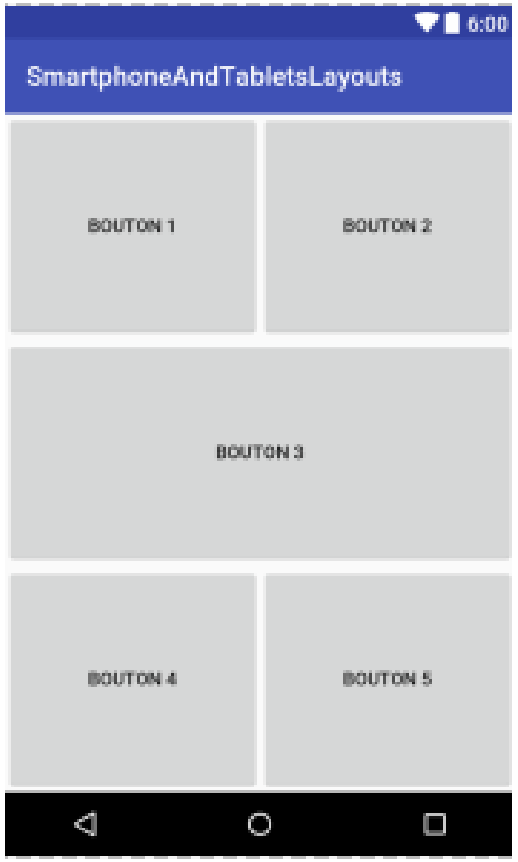
- Relations avec ses parents et pairs
- Nécessite l'utilisation des ids :
  - **android:id**= "**@+id**/idElem"
- Très puissant et performant
- Peut être sensible aux modifications, plus compliquer à maitriser




```
<android.support.percent.PercentRelativeLayout xmlns:and
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent" android:layout_h

    <Button
        android:text="Button1"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="50%"
        android:layout_alignParentTop="true"
        android:id="@+id/button1" />

    <Button
        android:text="Button2"
        android:layout_alignParentTop="true"
        android:layout_toRightOf="@id/button1"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="50%"
        android:id="@+id/button2" />

    <Button
        android:text="Button3"
        app:layout_heightPercent="33%"
        app:layout_widthPercent="100%"
        android:layout_below="@id/button2"
        android:id="@+id/button3" />

    <Button
        android:text="Button4"
        app:layout_heightPercent="33%"
```

TableLayout

- Layout sous un format de tableau
- Proche du « table » HTML
- Fusion de cellules avec layout_span



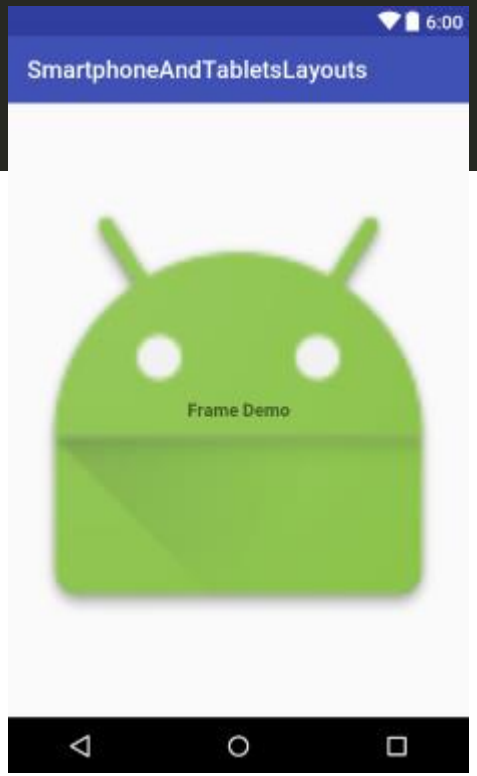

```
<TableLayout xmlns:android="http://schemas.android.com/apk/re
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight=".33">

        <Button
            android:id="@+id/un"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".5"
            android:text="Bouton 1" />

        <Button
            android:id="@+id/deux"
            android:layout_width="0dp"
            android:layout_height="match_parent"
            android:layout_weight=".5"
            android:text="Bouton 2" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="0dp"
```

FrameLayout

- Permet l'affichage d'une vue (exemple : visualisation des images)

- ou d'une superposition de vues (exemple : Google Maps)

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:src="@mipmap/ic_launcher"
        android:scaleType="fitCenter"
        android:layout_height="match_parent"
        android:layout_width="match_parent"/>

    <TextView
        android:text="Frame Demo"
        android:textSize="30px"
        android:textStyle="bold"
        android:layout_height="match_parent"
        android:layout_width="match_parent"
        android:gravity="center"/>
</FrameLayout>
```



ScrollView

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="wrap_content">
  <LinearLayout>
    <!-- contenu du layout -->
  </LinearLayout>
</ScrollView>
```

- Malgré son nom, c’est un Layout
- Permet de rendre un widget/layout scrollable
- Attention aux conflits si l’élément enfant gère déjà le scroll
- Il est fortement conseillé de n’avoir qu’un enfant pour l’utilisation de ce layout

TD

11h40 : évaluation avec M. Tigli

- 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
- 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
- Faire vibrer le téléphone au changement d'état de la connexion
- Partage de la valeur d'un capteur par SMS