

## TD Machine - *Initiation à Android*

### Exo 1 : Hello world + bases de l'IHM

#### **Lancer l'outil Android Studio**

```
cd ~/Android/
./studio.sh
```

#### **Créer et tester un nouveau programme**

1. New Project ...
  - Application Name (Nom affiché dans Android) -> Mon Helloworld
  - Domain Name (Inverse du nom de package) -> hello.moi.cpe
  - Project location -> Mon premier programme
  - Target -> Minimum SDK : 19 (4.4)
2. Empty activity
  - Activity name : nom de la classe principale (HelloActivity)
  - Layout name : fichier XML d'interface graphique (HelloLayout)
  - **Finish**
3. Exécuter ce programme
  - Brancher un device en USB **en mode développeur**
  - Cliquer sur run (flèche verte)
  - Ce qu'il fait : Génération d'un fichier .apk + chargement/installation du fichier dans le simulateur
  - Ce que vous faites : Ôh que c'est bô ! ;)

#### **Créez votre premier visuel d'interface graphique**

1. Où est décrite l'interface graphique ?
2. Où est stocké le nom de votre application ? (Qui est affiché)
3. En utilisant l'**éditeur graphique** : modifier le programme pour que le seul texte affiché soit "*Un petit pas pour l'homme, un grand pas pour ma compréhension d'Android.*"
4. Compléter et modifier l'interface graphique pour quelle ressemble à celle présentée en Illustrations 1 et 2. Exemple de grandes étapes qui peuvent y mener :
  1. En **éditant le XML** du Layout : ajouter 3 nouvelles étiquettes (**TextView**) affichant "Premier", "Deuxième" et "Troisième".
  2. Insérez un LinearLayout **Horizontal** centré et occupant toute la largeur. Y placer les 3 textview
  3. Mettre un poids à 1 pour chacun de ces TextView, ainsi qu'un alignement centré du texte.
  4. Ajouter un bouton avec le texte adapté.

#### **Ajouter des interactions**

Faites en sorte qu'en cliquant sur le bouton, le texte du TextView central se change en "Alors ? C'est pas bô ça ? :D"

#### **Rappels/aides :**

```
findViewById(R.id....) : récupération d'un pointeur vers un objet instancié dans un XML
btn.setOnClickListener(...) : méthode d'ajout d'un écouteur à un bouton
OnClickListener : classe caractérisant un écouteur de bouton
public void onClick(View v) : action à redéfinir pour le bouton.
```

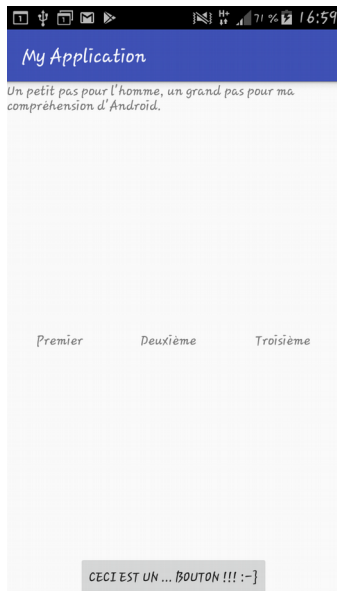


Illustration 1: Rendu de l'application

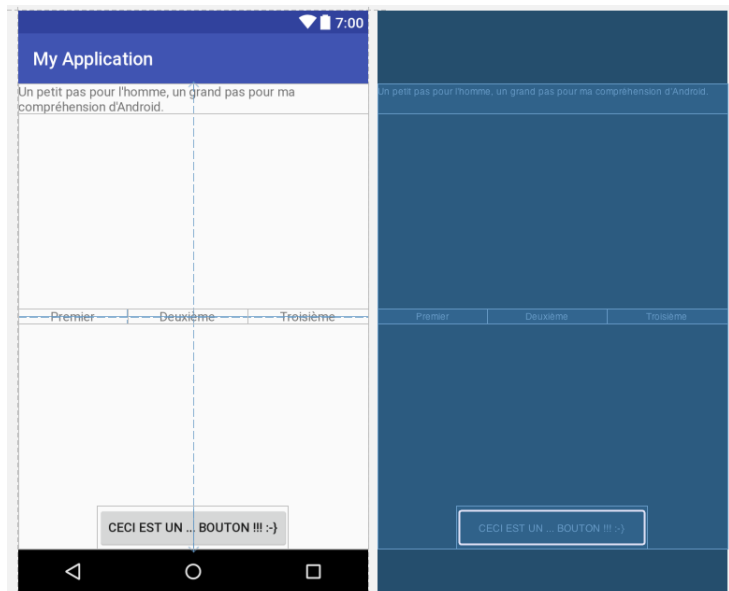


Illustration 2: Making of de l'interface attendue

## Exo 2 : Utilisation des sondes

### Version simple (1 sonde)

Faites une application affichant les données lues sur les 3 axes de l'accéléromètre.

Démarche possible :

- Réutilisez votre programme de l'exercice 1
- Implémentez l'interface `EventListener` depuis l'activité ;  
<http://developer.android.com/reference/android/hardware/SensorEventListener.html>
- Récupérez une référence vers le gestionnaire de services ;  
**Aide :** `(SensorManager) getSystemService(Context.SENSOR_SERVICE)`  
<http://developer.android.com/reference/android/hardware/SensorManager.html>
- Enregistrez l'activité en tant qu'écouteur pour le capteur voulus auprès du gestionnaire de service lors de la reprise et la dé-enregistrer à la mise en pause ;  
**Note :** pensez à choisir une fréquence de rafraîchissement pas trop rapide.  
**Aides :**
  - `sm.registerListener(...)` : Méthode d'enregistrement
  - `sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)` : Choix d'un capteur
  - `SensorManager.SENSOR_DELAY_UI` : une fréquence de rafraîchissement de l'interface graphique
- Faites en sorte que les valeurs reçues par `onSensorChanged` soient publiées sur l'interface graphique.

### Pour attendre les autres (2 sondes)

Lire les données de 2 capteurs avec votre application (Proximité et accéléromètre?) et les afficher.

**Note :** pour savoir à quel capteur est associée une valeur reçue, on peut le demander via :

`event.sensor.getType() == Sensor.TYPE_ACCELEROMETER`

## Exo 3 : Communications réseau

Nous proposons de faire une application permettant d'envoyer une chaîne de caractères par UDP. Pour tester votre code, l'ordinateur du prof héberge un serveur écoutant les messages UDP sur le port 10000. Il affiche les messages reçus et les renvoie à l'expéditeur. Si l'enseignant a oublié de le lancer et de le vidéo-projecter, demandez le lui.

### Mettre en place de l'interface graphique

- Créez un nouveau projet Android avec dans son interface avec au moins un bouton et un EditText (Illustration 3).
- Dans l'activité :
  - Déclarez 2 attributs pour faire le lien avec les objets graphiques (1) et (2)
  - Dans la méthode onCreate, faites le lien entre les attributs et les objets graphiques (findViewById)

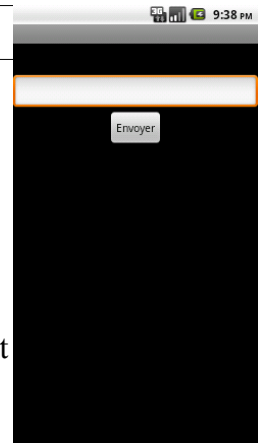


Illustration 3:  
Suggestion de  
présentation

### Émettre un message sur le réseau

- Déclarez les 4 attributs suivants pour permettre l'interaction avec le réseau :
 

```
private final String IP="192.168.1.xxx"; // Remplacer par l'IP de votre interlocuteur
private final int PORT=10000;           // Constante arbitraire du sujet
private InetAddress address;             // Structure Java décrivant une adresse résolue
private DatagramSocket UDPsocket;        // Structure Java permettant d'accéder au réseau (UDP)
```
- Au résumé, initialisez le DatagramSocket et l'adresse résolue (et libérez les ressources à la mise en pause);
- Écrivez une méthode d'envoi du message présent dans le champs texte, appelée par le bouton;

**Rappel :** Pensez à déclarer, dans le **manifeste** (AndroidManifest.xml), que votre application utilisera le réseau en ajoutant la ligne suivante **avant** l'ouverture de la balise application :

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

**Astuce :** pour vous aider au debug, vous pouvez poster des traces dans les log via :

```
Log.d("TAG (un texte quelconque)", "Message à afficher");
```

### Quelques liens utiles pour aller plus loin

- <http://developer.android.com/reference/java/net/InetAddress.html>
- <http://developer.android.com/reference/java/net/DatagramPacket.html>
- <http://developer.android.com/reference/java/net/DatagramSocket.html>
- <http://developer.android.com/reference/java/net/DatagramSocket.html#send%28java.net.DatagramPacket%29>

### Si terminé avant l'heure : Recevoir un message

Écoutez sur le réseau et afficher les messages reçus.

### Rappels des principes de la réception d'un message en UDP

- Pour recevoir un message, il est nécessaire d'avoir :
  - Le droit de le faire (la même permission que précédemment)
  - Un accès au réseau (le même DatagramSocket que précédemment)
   
<http://developer.android.com/reference/java/net/DatagramSocket.html>
  - Un espace pour stocker les informations reçues (un tableau de bytes).
  - Un processus tournant en tâche de fond en attente des messages
- Pour recevoir un message, on utilise la méthode receive(...) qui est fournie par le DatagramSocket. Chaque réception doit fournir : une zone de stockage du message attendu.
   
<http://developer.android.com/reference/java/net/DatagramSocket.html#receive%28java.net.DatagramPacket%29>

- Le type `AsyncTask` peut être utilisé pour réaliser facilement un processus d'attente.  
<http://developer.android.com/reference/android/os/AsyncTask.html>

### Petit coup de pouce pour `AsyncTask`

- Exemple de définition d'un processus d'attente de chaînes de caractères :

```
private class ReceiverTask extends AsyncTask<Void, String, Void> {
    protected Void doInBackground(Void... rien) {
        while(true){ // Boucle d'attente active de messages
            String msgRecu= ... ; // attente d'un message sur le réseau
            publishProgress(msgRecu); // Publication asynchrone du résultat
        }
    }
    protected void onProgressUpdate(String... data) {
        displayInputs(data[0]); // Mise à jour de l'affichage avec le premier message
        // cette méthode n'existe pas. À vous de l'écrire.
    }
}
```

- Exemple de lancement de ce processus :

```
(new ReceiverTask()).execute();
```

## Annexe 1 : figer la position de l'écran

Pour éviter que l'application de tourne lorsque l'on bouge l'écran, ajouter la propriété « `screenOrientation` » dans le manifest de votre projet :

```
<activity
    android:screenOrientation="portrait"
    android:label="@string/app_name"
    android:name=".MyActivity" >
```

## Annexe 2 : créer une vue et l'intégrer à l'affichage

- Créer une nouvelle classe
  - Name : `MaVue`
  - Superclasse : `android.view.View`
  - Générer les 3 constructeurs issues de la super-classes
  - [Pour le debug]** Pour rendre la vue visible, changer sa couleur de fond :  
Dans chacune des trois méthodes `public Graphique(...)` insérer la ligne :  
`setBackgroundColor(Color.GREEN);`
- Intégrer la classe dans l'interface graphique.
  - Exemple : si le package est `ot.nst.roulequiboule`, alors mettre l'objet suivant dans le `main.xml` :

```
<ot.nst.roulequiboule.MaVue
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/maBouboule" />
```

- Exécuter le programme. Il doit afficher une application vide ayant un fond vert.

## Annexe 3 : conversions

Conversions `String <-> byte[]` :

```
byte[] b = myString.getBytes();
Arrays.toString(bytes);
```

## Annexe 4 : Empêcher l'écran de se mettre en veille

Pour éviter que l'écran ne se mette en veille au bout d'un certain temps, il suffit d'ajouter la permission suivante dans le manifest de l'application :

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

ainsi que la ligne suivante dans la méthode onCreate de l'activité :

```
getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);
```

## Annexe 5 : l'envoi avec Android 3 et +

À partir d'Android 3, il est interdit de faire un envoi de message directement, mais nécessairement à travers un processus. Voici une méthode horrible mais triviale pour contourner le problème. Ensuite, à vous de faire plus propre avec des files synchrones.

```
private void UDPSend() {
    // L'émission est faite depuis un processus, à cause de restriction des
    // API 11 et +
    (new Thread() {
        @Override
        public void run() {
            byte[] data = new byte[1]; // Pour simplifier l'exemple, nous n'envoyons qu'1 octet
            data[0] = Byte.parseByte(outView.getText().toString());
            DatagramPacket packet = new DatagramPacket(data, data.length, address, port);
            try {
                UDPClient.send(packet);
            } catch (Exception e) {
            }
        }
    }).start();
}
```