Android - TP1 : les premiers pas

- Récupérez sur Chamillo le projet sur lequel vous allez travailler : TP1.zip, placez le dans un dossier SIL1.
- Lancez l'IDE android Studio et ouvrez le projet
  - Open project (le TP1 téléchargé)
- Vous trouverez sur le web toute la documentation nécessaire pour ce module
  - Documentation Java : http://docs.oracle.com/javase/7/docs/api
  - Guide User Interface Android : http://developer.android.com/guide/topics/ui/index.html
  - Android référence : http://developer.android.com/reference/android/package-summary.html

L'émulateur Android

Pour développer nos applications Android nous vous conseillons d'utiliser un device Android pour soulager votre ordinateur portable lors du développement. Mais ce n'est pas une obligation l'émulateur marche très bien (il est seulement gourmand en ressource !). Plusieurs émulateurs sont disponibles : Nexus 4, 5, 6, 7, generic devices, etc. Pour ce module, nous vous conseillons d'utiliser l'émulateur « Nexus One API 15 X86 IceCreamSandwich » (moins gourmand en ressource).

Première chose à réaliser, lancer l'émulateur en suivant les étapes suivantes :

1. Lancer le manager d'émulateurs : Tools > Android > AVD Manager (Android Virtual Devices Manager)
   a. *Create virtual Device*
   b. Choisir Nexus One puis *Next*
   c. Choisir IceCreamSandwich API 15 version X86 puis *Download*
   d. Attendre l'installation puis *Finish*
   e. Choisir IceCreamSandwich API 15 version X86 puis *Next*
   f. Show Advanced Setting puis modifier la memory ram à 700 puis *Finish*
2. Lancer l'émulateur « Nexus One API 15 ARM » en appuyant sur la flèche.

Le lancement de l'émulateur est long. Vous pouvez lire et commencer la suite.

ATTENTION : l'émulateur ne sera à lancer qu'une seule et unique fois pendant la séance de travail.

Prise en main d'Android Studio

Pour chaque exercice, des fichiers incomplets vous sont fournis à vous de les compléter.

Exercice 1 : Première activité

Pour réaliser cet exercice, vous aurez à modifier Exercice1Activity.java et activity_exercice1.xml (respectivement la classe représentant une activité et sa vue associée).

Scénario : un utilisateur entre son prénom (par exemple Jérôme) dans un objet de type EditText. Lors du clic sur le bouton valider, le texte « Hello world ! » sera remplacé par « Hello Jérôme ! » (voir figure 1).

Figure 1

Etape 1 : **créer** l'interface graphique de la figure 1 dans le fichier activity_exercice1.xml en respectant l'arborescence suivante :

LinearLayout

- TextView
- EditText
- Button
- TextView

Pour ce faire, vous modifierez le fichier activity_exercice1.xml soit à l'aide de l'outil de design d'Android Studio, soit directement sur le fichier XML. Par exemple, ci-dessous un exemple d'objet graphique de type TextView :

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/exercice1_text"
    android:gravity="center"/>
```

La valeur *match_parent* permet de prendre toute la place laisser par le parent, *wrap_content* prend juste la place nécessaire. La valeur *@string/exercice1_text* représente une chaîne présente dans le fichier res/values/strings.xml. anroid:*gravity*[1] est une propriété de l'objet de Type TextView (représente la méthode *setGravity()*) qui permet de spécifier l'alignement du texte dans la vue.

Nous vous conseillons de tester les différentes propriétés et valeurs des objets graphiques et de vérifier les résultats sur l'affichage « preview » proposée par Android Studio. Référez-vous aussi à la documentation pour connaître les propriétés des classes[2][3].

Etape 2 : **réaliser** la partie événementielle de cette activité dans le fichier Exercice1Activity.java.

Pour rappel : un utilisateur entre son prénom (par exemple Jérôme) dans un objet de type EditText. Lors du clic sur le bouton valider, le texte « Hello world ! » sera remplacé par « Hello Jérôme ! »

Première chose à faire, **créer une écoute (abonnement)** sur l'objet qui va provoquer un événement click, ici le bouton Valider. Par exemple ci-après, on ajoute au bouton d'identifiant (id) *exercice1_valider* une méthode *exerciceValider* qui « écoutera » les événements que ce bouton produira en onClick.

[1] http://developer.android.com/reference/android/widget/TextView.html#setGravity(int)
[2] http://developer.android.com/reference/android/package-summary.html
[3] http://developer.android.com/guide/topics/ui/index.html

```
<Button
    android:id="@+id/exercice1_valider"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/exercice1_bouton"
    android:onClick="exercice1Valider"/>
```

Deuxième chose à réaliser, **avoir accès aux composants graphiques** de l'interface qui seront à modifier dans le fichier Java :

- dans le XML donner un id a votre objet :

  `android:id="@+id/exercice1_hello"`

- dans le Java, récupérer l'id avec `R.id.exercice1_hello` et l'objet en utilisant les instructions suivante :

  `TextView hello = (TextView) findViewById(R.id.exercice1_hello);`

Troisième chose, compléter la méthode d'écoute pour **modifier cet objet graphique** :

`hello.setText("Hello " + prenom.getText() + " !");`

Pour résumer la méthode d'écoute exercice1Valider ressemblera à :

```
public void exercice1Valider(View view) {

    // On récupère les objets de l'arbre graphique (à l'aide de leur id)
    TextView hello = (TextView) findViewById(R.id.exercice1_hello);
    TextView prenom = (TextView) findViewById(R.id.exercice1_prenom);

    // On affiche le message si un prenom a été défini
    if (!TextUtils.isEmpty(prenom.getText())) {
        hello.setText("Hello " + prenom.getText() + " !");
    }
}
```

Tester votre application en lançant la commande de l'IDE Run > Run 'app'. Cette commande chargera votre application (.apk) sur votre émulateur.

Premières réalisations

Exercice 2 : Question-réponse

Pour réaliser cet exercice, vous aurez à modifier Exercice2Activity.java et activity_exercice2.xml (respectivement la classe représentant une activité et sa vue associée).

Scénario : Cette activité proposera une question et 3 choix possibles (voir figure 2). Une fois la réponse choisie, l'utilisateur validera pour vérifier s'il a la bonne réponse.

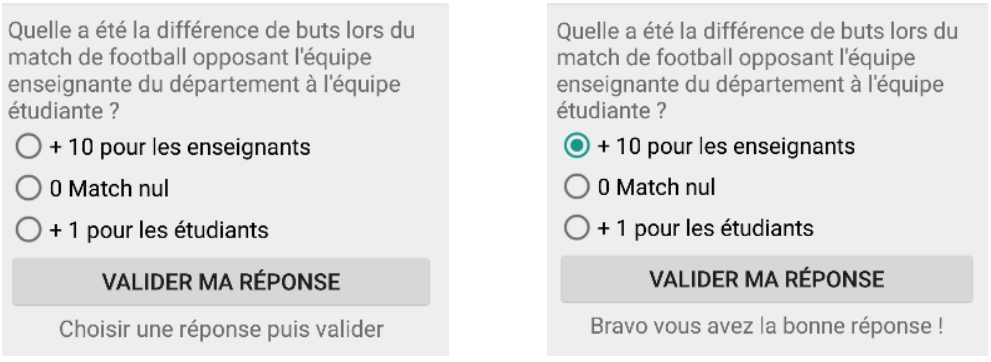

Figure 2

Etape 1 : **créer** l'interface graphique de la figure 2 dans le fichier activity_exercice2.xml. L'outil de design proposé par Android Studio vous permettra de connaître toute la palette d'objets graphiques à votre disposition.

AIDE : placez les objets de Type RadioButton dans un container de Type RadioGroup.

Etape 2 : **réaliser** la partie événementielle de cette activité dans le fichier Exercice1Activity.java.

AIDE : vous pouvez comparer des composants graphiques entre eux. Par exemple, pour vérifier que l'utilisateur à la bonne réponse vous pouvez comparer les identifiants de l'objet de type RadioButton sélectionné et de l'objet de type RadioButton pourtant la bonne réponse :

```
if (radioGroup.getCheckedRadioButtonId() == R.id.exercice2_bonne_reponse)
{…}
```

Exercice 3 : Pierre-Feuille-Ciseaux

Pour réaliser cet exercice, vous aurez à modifier Exercice3Activity.java et activity_exercice3.xml (respectivement la classe représentant une activité et sa vue associée).

En vous inspirant de l'exercice 2, réaliser le jeu pierre-feuille-ciseaux (voir figure 3). Pour plus d'information : http://fr.wikipedia.org/wiki/Pierre-feuille-ciseaux

Figure 3

Pour vous aider, les classes Jeu et Resultat (le modèle !!!) vous sont données dans le projet (une archive TestJeu.zip vous est également fournie pour que vous puissiez tester ces classes sur le terminal – lire TestJeu.java). Les images à utiliser sont présentes dans le dossier res/drawable.

Etape 1 : **créer** l'interface graphique de la figure 3.

AIDE : Pensez à utiliser des Layouts[4] pour agencer vos objets graphiques : LinearLayout, RelativeLayout, etc. Utiliser les propriétés *orientation*, *gravity*, etc.

[4] Guide Layout : http://developer.android.com/guide/topics/ui/declaring-layout.html

Etape 2 : **réaliser** la partie événementielle de cette activité.

AIDE : Un objet de Type ImageView est clickable. Donc on peut associer une méthode d'écoute à un objet de Type ImageView.

Etape 3 : **réaliser** une vue spécifique pour le mode landscape/paysage (voir figure 4).

Pour ce faire, créer un dossier représentant le landscape dans le dossier res :

- New > Android Resources Directory ;
- Ressource type, choisir layout ;
- Available qualifiers, choisir orientation ;
- Screen orientation, landscape.

Un dossier layout-land sera créer dans le système de fichier. Vous n'aurez plus qu'à créer une nouvelle vue XML pour cette activité :

- New > Android Resource File ;
- Ressource type, choisir layout ;
- Available qualifiers, choisir orientation ;
- Screen orientation, landscape ;
- Donner le même nom de fichier activity_exercice3.xml que la version « normale ».

Ce fichier sera automatique chargé par l'activité quand le device sera en landscape. Modifier donc ce fichier pour correspondre à la figure 4.

Figure 4

Pour tourner l'émulateur : sur Windows, left-ctrl+F11, sur Linux ctrl+F11on Linux.

INPORTANT : notez que les résultats sont effacés lors du changement de position du device. En effet, l'activity est recrée lors du changement. Nous verrons dans les prochaines séances comment enregistrer des informations et comment détecter les modifications survenues sur le device.

Exercice 4 : Autres réalisations possibles

Morpion, Sudoku, Mots fléchés.

Attention : Pensez à réaliser le modèle (tel que les classes Jeu et Resultat dans le jeu précédent) pour séparer la partie présentation/interaction de la partie modèle/donnée.