Travaux Dirigés et Pratiques de Programmation Android n° 2

Interface et interaction avec l'utilisateur

Les objectifs de ce TD sont :

- La création d'une interface simple ;
- l'interaction avec l'utilisateur via des boutons et des zones de texte.

Nous allons réaliser pas à pas une application permettant à l'utilisateur d'entrer deux entiers, et renvoyant leur somme lorsque l'utilisateur appuie sur un bouton. Cette application sera ensuite complétée pour gérer d'autres opérations, puis en une calculatrice basique.

Mise en place

Créer une nouvelle application (nommée par exemple « TD2 ») comprenant une activité vide.

Interface

Consulter le fichier d'interface `res/layout/content_td2.xml`, dans l'onglet « Text » d'Android Studio.

L'interface est décrite par un ensemble de boîtes (« layouts ») contenues les unes dans les autres, décrites par un certain nombre d'attributs. Aux feuilles de ces boîtes, on trouve les objets comme des zones de texte, des images, ou encore des boutons.

Ici, on a une seule boîte principale de type `RelativeLayout`[1], signifiant que les autres éléments qu'elle contient (sous-boîtes ou objets) sont positionnés de manière relative les uns aux autres. Ses attributs principaux sont :

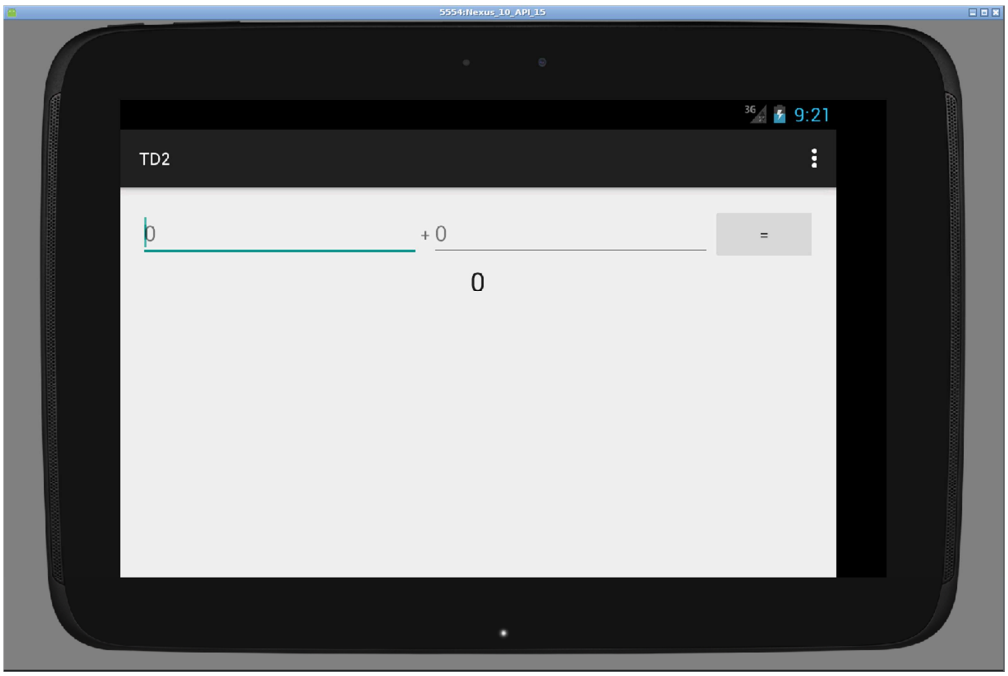
- `android:layout_width` et `android:layout_height`, pour désigner respectivement la largeur et la hauteur de cette boîte : ils ont pour valeur `match_parent`, indiquant que le `RelativeLayout` occupe tout l'espace de la boîte dont elle dépend (en l'occurrence, tout l'espace de l'écran) ;
- `android:paddingLeft`, ... etc, pour laisser une marge autour des éléments contenus dans la boîte.

Cette boîte contient un unique élément, à savoir un objet qui est une zone de texte non modifiable, de type `TextView`[2]. Il a lui-même plusieurs attributs:

- `android:layout_width` et `android:layout_height` également : cette fois, leur valeur est `wrap_content`, signifiant que la zone de texte prendra la taille requise en fonction du texte qu'elle contient ;
- `android:text`, indiquant la valeur initiale du texte ; comme vu au TD précédent, cette valeur est contenue dans la chaîne de caractères `@string/hello_world`.

Maintenant que nous avons vu le principe de fonctionnement général du fichier d'interface, nous allons remplacer cette dernière par l'interface qui nous intéresse dans le cadre de notre application :

[1] http://developer.android.com/reference/android/widget/RelativeLayout.html
[2] http://developer.android.com/reference/android/widget/TextView.html



Cette interface contient une boîte principale dont les éléments dont disposés verticalement les uns par rapport aux autres. Elle contient deux éléments, qui sont donc l'un au-dessus de l'autre :

- une sous-boîte, dont les éléments sont disposés horizontalement les uns par rapport aux autres, contenant successivement une zone de texte modifiable, une non modifiable, une modifiable, et un bouton ;
- une zone de texte non modifiable.

1. Supprimer l'objet `TextView` actuel. Modifier la boîte principale `RelativeLayout` en une boîte `LinearLayout`, en changeant les balises de début et de fin. Ajouter à cette boîte un attribut `android:orientation` avec pour valeur `vertical` afin de préciser la disposition des éléments de cette boîte.

2. Passer dans l'onglet « Design ». Par glisser-déposer, ajouter à cette boîte successivement un `LinearLayout (Horizontal)` et un `Plain TextView`, en centrant ce dernier horizontalement. Si le rendu ne fonctionne pas (notamment, sur les machines de l'IUT), ajouter directement ces éléments dans le fichier XML, comme feuilles du `LinearLayout`:

```
<LinearLayout
 android:orientation="horizontal"
 android:layout_width="match_parent"
 android:layout_height="wrap_content">
</LinearLayout>
```

```
<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/new_text"
 android:id="@+id/textView"
 android:layout_gravity="center_horizontal"/>
```

3. Retourner dans l'onglet « Text », et vérifier que les deux éléments qu'on vient d'ajouter sont bien des feuilles de notre `LinearLayout` vertical. La boîte horizontale doit avoir comme largeur `match_parent` et comme hauteur `wrap_content`, tandis que la zone de texte a la valeur

`wrap_content` pour les deux dimensions. En plus du texte initial, elle contient deux nouveaux attributs:

- `android:layout_gravity`, indiquant qu'on a centré la boîte horizontalement ;
- `android:id`: cela donne un identifiant à cette zone de texte, qui permettra d'y faire référence aussi bien dans ce code d'interface que dans le code Java. Sa valeur par défaut est `@+id/textView` : le `+` indique qu'il s'agit de la première définition de l'identifiant (on ne le mettra pas lors des références à cet identifiant), et `textView` est son nom actuel. Le modifier en un nom plus parlant, comme par exemple `result`. Changer également le texte initial en « 0 ».

4. Procéder de la même façon pour ajouter à la boîte horizontale quatre éléments: une zone de texte éditable (`Text Fields`) de type `Number (Signed)`, une zone de texte non éditable, une nouvelle zone éditable, et un bouton (`Button`). On peut également glisser-déposer vers la fenêtre « Component Tree » pour placer les éléments dans la boîte voulue. Contrôler dans le XML que les éléments ont été ajoutés au bon endroit, et les déplacer éventuellement. Si le rendu ne fonctionne pas, on pourra s'inspirer des déclarations suivantes :

```
<EditText
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:inputType="numberSigned"
 android:ems="10"
 android:id="@+id/idEditText"
 android:hint="@string/stringEditText"/>

<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/stringTextView"/>

<Button
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="@string/stringButton"
 android:id="@+id/idButton"/>
```

5. Leur donner des identifiants parlant, et choisir « + » comme chaîne de caractères par défaut de la zone de texte non modifiable, ainsi que « = » comme chaîne de caractères sur le bouton. Ajouter aux deux zones de texte éditables un attribut de type `android:hint` contenant la chaîne « + », pour indiquer la valeur par défaut.
6. Utiliser `wrap_content` pour toutes les dimensions. Qu'observe-t-on ? Pour que l'ensemble de cette boîte occupe toute la largeur de l'écran, en ajustant la largeur des zones de textes éditables comme sur l'image, on peut donner à ces dernières un poids plus important grâce à l'attribut `android:layout_weight` auquel on donne une valeur de « `1` ». Dans ce cas, la valeur de la largeur n'a plus d'importance, et on peut par conséquent la porter à « `0dp` ». Tester avec différents poids pour les différents éléments et regarder l'effet.
7. Tester l'interface sur la tablette ou l'émulateur.

Interaction avec l'utilisateur

Nous avons notre interface, mais il reste à faire en sorte que la somme des entiers que l'utilisateur aura entrés s'affiche lorsqu'il appuie sur le bouton.

Tout d'abord, nous allons, au niveau du code Java, initialiser des attributs contenant les éléments graphiques qui vous intervenir dans cette action: les deux zones de texte éditables, car il faudra lire la valeur entrée par l'utilisateur ; et la zone de texte pour l'affichage du résultat. Déclarons tout d'abord ces attributs dans la classe de l'activité principale :

```
private EditText addLHS;
private EditText addRHS;
private TextView result;
```

Ces attributs sont initialisés au moment de la création de l'activité, après avoir démarré l'interface graphique : à la fin de la méthode `onCreate`. On récupère les divers éléments graphiques en appelant la méthode `findViewById` [3] de la classe `Activity` sur les identifiants choisis dans le fichier XML, par exemple:

```
addLHS = (EditText) findViewById(R.id.addLHS);
addRHS = (EditText) findViewById(R.id.addRHS);
result = (TextView) findViewById(R.id.result);
```

1. Consulter la documentation des classes `TextView` [4] et `EditText` [5]. Comment modifier la valeur du texte d'un objet de la classe `TextView` ? Comment lire la valeur entrée par l'utilisateur d'un objet de la classe `EditText` ? Bien regarder les signatures des méthodes trouvées.
2. Écrire une méthode `public void computeAdd(View v)` lisant les entiers contenus dans `addLHS` et `addRHS`, les additionnant, et écrivant le résultat dans `result`.
   Rappel : la méthode `int Integer.parseInt(String s)` [6] permet de transformer une chaîne de caractères en entier, si possible.
   Par ailleurs, on pourra utiliser les méthodes `toString` des classes `Editable` et `Integer` pour obtenir les chaînes de caractères correspondantes.
3. Consulter la documentation de l'attribut `android:onClick` [7] de la classe `View`, dont `Button` hérite. L'utiliser pour lier la méthode `computeAdd` à l'action de l'utilisateur sur le bouton.

Tester l'interface sur la tablette ou l'émulateur. Vérifier que l'application ne ferme pas dans les cas particuliers, par exemple lorsque l'utilisateur n'entre rien.

À vous de jouer

1. Imaginer une interface permettant de retourner quatre résultats (au lieu d'un seul), un pour chacun des quatre opérations, lorsque l'utilisateur appuie sur le bouton. Implanter l'interface, puis gérer l'interaction avec l'utilisateur.
   Remarque : Pour la division, il s'agit de la division entière. On pourra choisir différentes manières de gérer une division par zéro, du moment que l'application ne ferme pas.
2. On veut maintenant permettre à l'utilisateur d'enchaîner les additions, à la manière d'une calculatrice qui n'aurait que cette opération. Imaginer une interface avec un seul champ de texte (servant à la fois d'entrée et d'affichage) mais deux boutons : toujours le bouton « = », et un bouton pour l'addition. Implanter l'interface, puis gérer l'interaction avec l'utilisateur.
3. Ajouter un bouton pour réinitialiser l'affichage et le calcul.
4. On va maintenant avoir une interface plus proche d'une calculatrice. Ajouter 10 boutons, un par chiffre. Rendre la zone de texte non éditable ; cependant, elle doit maintenant visualiser ce que

[3] http://developer.android.com/reference/android/app/Activity.html#findViewById
[4] http://developer.android.com/reference/android/widget/TextView.html
[5] http://developer.android.com/reference/android/widget/EditText.html
[6] http://developer.android.com/reference/java/lang/Integer.html#parseInt
[7] http://developer.android.com/reference/android/view/View.html#attr_android:onClick

l'utilisateur tape au fur et à mesure. Implanter l'interface, puis gérer l'interaction avec l'utilisateur. On pourra utiliser les classes TableLayout [8] et TableRow [9], ou encore la classe GridLayout [10].
5. Regarder la documentation de la classe RelativeLayout [11] et l'utiliser pour les diverses interfaces.
6. * Ajouter un bouton de soustraction, puis un bouton pour pouvoir entrer des nombres négatifs.
7. * Ajouter un bouton de multiplication. Attention à l'ordre des opérations !
8. ** Ajouter des parenthèses.

Attention : lorsqu'on a plusieurs localisations, il ne faut pas oublier de gérer les variables pour chacune d'entre elles ! Il est souvent préférable de toujours faire une passe sur les fichiers de localisation à la fin, après chaque modification d'une application.

[8] http://developer.android.com/reference/android/widget/TableLayout.html
[9] http://developer.android.com/reference/android/widget/TableRow.html
[10] http://developer.android.com/reference/android/widget/GridLayout.html
[11] http://developer.android.com/reference/android/widget/RelativeLayout.html