Android : premiers contacts

Exercice 1 – Hello World

En suivant le tutorial de `http://developer.android.com/training/basics/firstapp/creating-project.html`, créer votre première application qui affiche simplement un message de bienvenue "Hello World" et exécutez-la dans un émulateur (ou sur votre dispositif mobile).

- Entrez le nom que vous désirez dans la case `ApplicationName`
- Sélectionnez Android 2.2 pour le Build SDK et le Minimum required SDK
- Le reste est à configurer à votre convenance.

1. A quoi sert le fichier `AndroidManifest`.xml ?

2. Quelle est la classe java principale de votre application et de quelle classe hérite-t-elle? Trouvez sur le site de `http://developer.android.com/` la documentation de cette classe mère. Quelle est la méthode principale à implémenter ?

3. Dans quel fichier est décrite l'apparence graphique de votre activité principale et quel est sont type?

4. A quoi sert le fichier `string.xml` contenu dans le dossier `values` ?

Exercice 2 – My First App

1. En suivant le tutorial de `http://developer.android.com/training/basics/firstapp/building-ui.html`, créer l'application illustrée figure 1 :

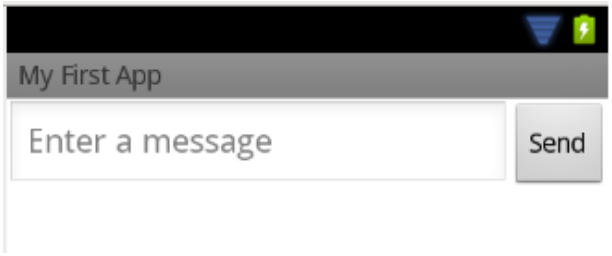

Figure 1: My First App

2. On souhaite modifier l'application pour que le bouton `Send` réagisse au clic en affichant le message dans une fenêtre `popup` appelée Toast `http://developer.android.com/guide/topics/ui/notifiers/toasts.html`.

   Pour cela, il faut :

   a. Ajouter l'attribut `onClick` au bouton dans le fichier `activity_main.xml` et lui associer le nom de la méthode qui sera appelée, par exemple `displayMessage` qui n'est pas encore définie : `android:onClick="displayMessage"`
   b. Définir la méthode `displayMessage` suivante dans la classe `MainActivity` et presser `Ctrl+Shift+O` pour importer les classes manquantes :

```
/**
        * Called when the user clicks the send button
*/

        public void displayMessage(View view){
                Context context = getApplicationContext();
                EditText editText = (EditText) findViewById(R.id.edit_message);
                String message = editText.getText().toString();
                CharSequence text = "Votre message est :\n" + message;
                int duration = Toast.LENGTH_SHORT;
                Toast toast = Toast.makeText(context, text, duration);
                toast.show();
}
```

3. Que fait la méthode `findViewById` et quel paramètre attend-elle ? A quoi correspond la classe java `R.java` contenue dans le dossier `gen` ?

Exercice 3 – Personnalisation

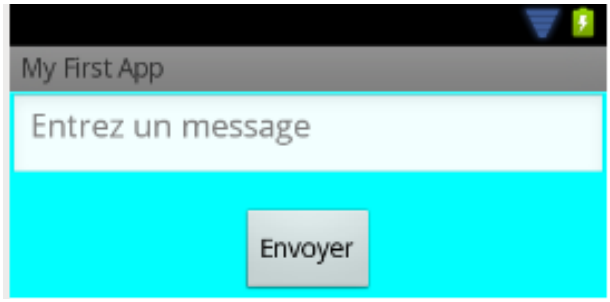

Figure 2: Personnalisation de `MyFirstApp`

Les constantes (chaînes de caractères, couleur, dimension, ...) sont définies dans le fichier `string.xml` contenu dans le dossier `values` et ensuite associées aux éléments graphiques dans le fichier `activity_main.xml`.

Modifier les éléments suivants de votre première application pour obtenir le résultat de la figure 2:

- Le message par défaut de `EditText` devient "Entrez un message"
- Les éléments sont placés verticalement dans le `layout`.

- Le message est positionné en haut de la vue (attribut `android:gravity`).

- Pour associer une couleur avec un élément graphique, il faut d'abord définir la couleur bleu (élément `color`) dans `string.xml` et ensuite utiliser l'attribut `android:background`. Modifier le fond du `layout` (attribut `android:background`) pour qu'il soit bleu (#`00FFFF`).

- Centrer le bouton (attribut `android:layout_gravity`).

Exercice 4 – Mise en pause et reprise d'application

Consulter
`http://developer.android.com/training/basics/activity-lifecycle/pausing.html`
et répondez aux questions suivantes :

1. Quels sont les états possibles d'une application android ?

2. Quelles sont les méthodes principales d'une activité android ?

3. Reprendre l'application `MyFirstApp` et implémenter toutes les méthodes précédentes en affichant simplement un message indiquant quelle méthode a été appelée à l'aide d'un `Toast`.

    a. Démarrer votre application. Quelles sont les méthodes successivement appelées ?

    b. Pour cacher votre application, cliquer sur l'icone de l'accueil. Quelles méthodes sont appelées ?

    c. Est-il possible de détruire manuellement l'application ?

    d. Modifier un peu votre application, par exemple en changeant les messages `Toast` des méthodes `onXXX`(). Que se passe-t-il si on démarre une nouvelle instance de votre application via Eclipse ?

Exercice 5 – Envoi de SMS (à rendre, après la deuxième séance)

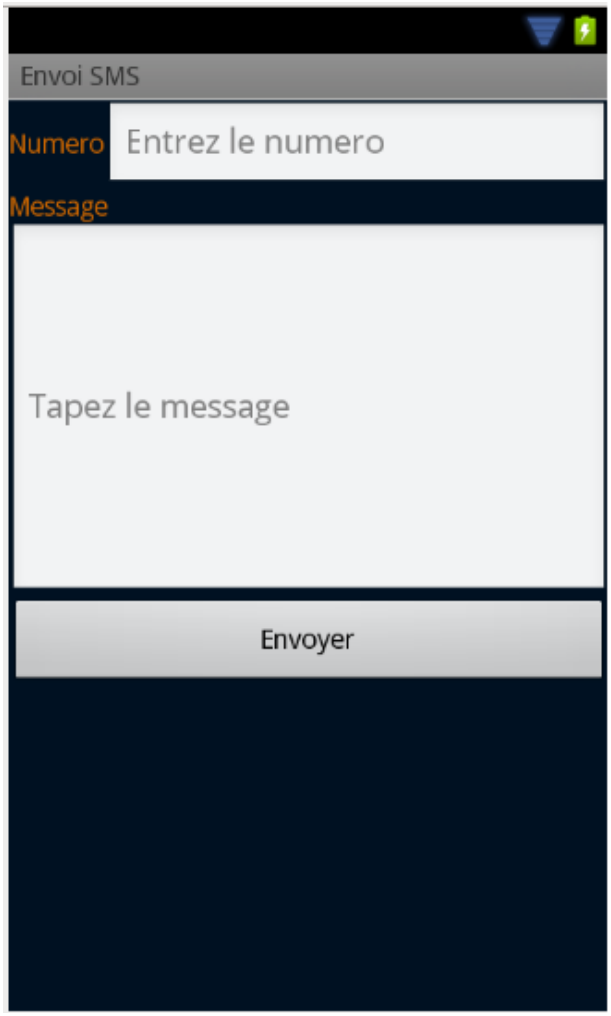

Figure 3: Application android d'envoi de SMS

Nous allons développer une petite application d'envoi de SMS qui a l'apparence de la figure 3.

1. Créer un nouveau Projet intitulé TP1_nom1_nom2, où nom1 et nom2 sont les noms du binôme, puis modifier le fichier activity_main.xml et string.xml pour créer une activité ayant l'apparence de la figure 3.

2. Pour envoyer un SMS contenant le message message à un numéro numero, il suffit d'utiliser la classe SMSManager de la façon suivante :
   SmsManager.getDefault().sendTextMessage(numero, null, message, null, null);
   Ajouter une méthode sendSMS qui réagit à un clic sur le bouton Envoyer en envoyant le message par SMS au destinataire. La méthode vérifiera que le numéro contient au moins 4 chiffres et que le message n'est pas vide.

3. Pour tester votre application avec un émulateur, il faut définir un deuxième émulateur et démarrer l'application sur le 1er émulateur. Pour envoyer un message d'un émulateur à un autre, il vous suffit de composer comme numéro les 4 chiffres qui se trouvent sur la fenêtre de l' émulateur comme dans la figure 4.

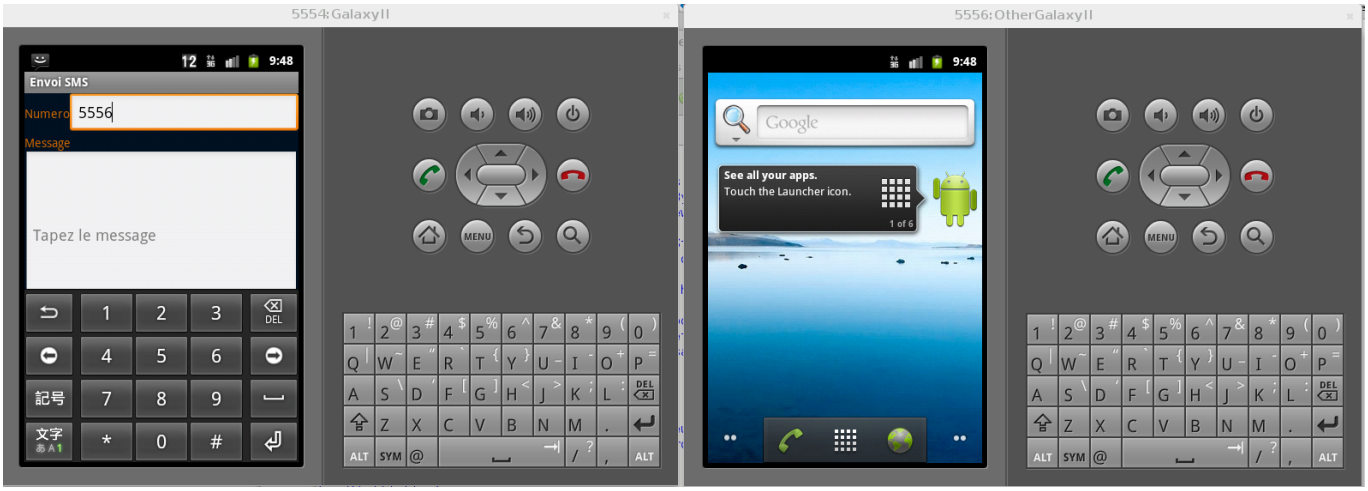

Figure 4: Application android d'envoi de SMS

4. Modifier votre application pour que l'utilisateur puisse entrer plusieurs numéros séparés par un point virgule (;). Le message devra être envoyé à tous les destinataires. Tester votre application en utilisant 3 émulateurs et un utilisateur qui envoi un message aux 2 autres.
   ATTENTION : votre clavier doit être en français, sinon vous risquez des problèmes d'encodage de certains caractères.

5. Les applications android sont « packagées » dans des archives .apk. Pour exporter un projet, clic droit sur le nom du projet et clic sur Android Tools -> Export Unsigned Application Package. Pour réellement publier un projet sur Play Store par exemple, il faudrait d'abord générer une clé (keystore), ce que nous ne ferons pas en TP. Pour soumettre votre TP, dans le workspace d'eclipse, retrouver votre projet et compresser le dossier en .tar.gz. et envoyez-le par mail à votre chargé de TP, en précisant l'objet suivant : **[ENSEIRB - AppMob] TP1 nom1 nom2**

   Attention : pour ce TP et pour les suivants, toujours supprimer le .apk avant de compresser le fichier. Sinon, le volume de votre archive sera trop grand.