Travaux Dirigés et Pratiques de Programmation Android n° 3

Vie des activités

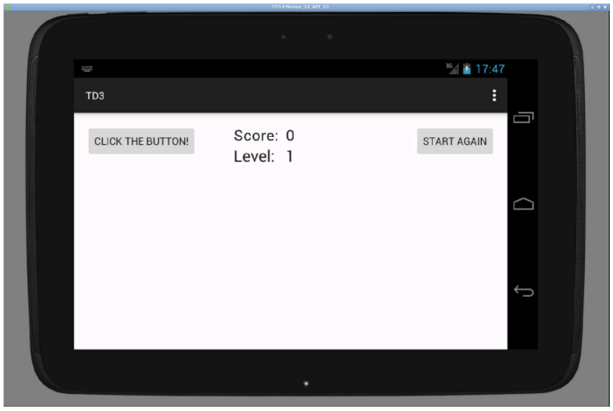
Les objectifs de ce TD sont :

- Le lancement et la fermeture des activités ;
- Leur cycle de vie, et son parcours lors de l'interaction de l'utilisateur ;
- La transmission de données entre activités.

Nous allons considérer une application implantant un « jeu » très simple : le joueur marque 1 point à chaque fois qu'il clique sur un bouton, et gagne un niveau à chaque fois qu'il a marqué 5 points.

Mise en place

Réaliser une nouvelle application (nommée par exemple « TD3 ») pouvant ressembler par exemple à cela :



Initialement, le score est de 0 et le niveau de 1. Lorsque l'utilisateur clique sur le bouton de gauche, le score augmente de 1, et tous les 5 clics, le niveau augmente de 1 également. Lorsque l'utilisateur clique sur le bouton de droite, le score et le niveau sont réinitialisés.

Lancement d'une deuxième activité

On souhaite afficher une alerte lorsque le joueur gagne un niveau. Pour cela, on va lancer une nouvelle activité.

Dans Android Studio, onglet « Project », faire un clic droit sur « app », choisir « New → Activity → Blank Activity », et lui donner un nom (par exemple, « LevelActivity »). Contrôler qu'elle apparaît bien dans le manifeste (fichier `AndroidManifest.xml`).

Au niveau de l'interface de cette nouvelle activité, utiliser uniquement des zones de texte pour dire que l'utilisateur a gagné un nouveau niveau, et afficher son niveau actuel.

Au niveau du code en Java :

- Dans l'activité principale (`TD3Activity`), lorsque le niveau est mis à jour, lancer cette nouvelle activité, en lui passant en paramètre le nouveau niveau du joueur :
  1. Construire une intention correspondant à l'activité « LevelActivity » :
     ```
     Intent intent = new Intent(this, LevelActivity.class);
     ```
  2. Lui ajouter comme extra la valeur du niveau actuel :
     ```
     intent.putExtra(EXTRA_LEVEL, level);
     ```

Il faut alors déclarer en tête de la classe `EXTRA_LEVEL` comme un attribut statique et non modifiable (`final`) contenant une chaîne de caractères explicitant l'intérêt de cet extra :
`public final static String EXTRA_LEVEL = "EXTRA_LEVEL";`

3. Enfin, lancer la nouvelle activité : `startActivity(intent);`

– Dans la nouvelle activité (`LevelActivity`), il faut récupérer la valeur du niveau est l'afficher. Pour cela, à la fin de la méthode `onCreate` :
  1. Récupérer l'intention qui a lancé l'activité : `Intent intent = getIntent();`
  2. Récupérer la valeur du niveau contenue dans les extras de l'intention :
     `int level = intent.getIntExtra(TD3Activity.EXTRA_LEVEL, 1);`
  3. Récupérer la zone de texte contenant le niveau et lui affecter cette valeur, comme vu au TD2.

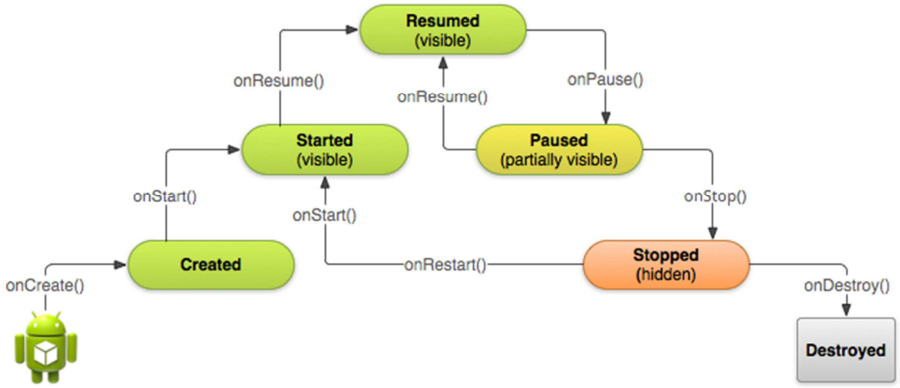
Tester l'application.

Moment de la création d'une activité

Ajouter un attribut **statique** à la classe `LevelActivity`, initialisé à 0, représentant le nombre de fois que cette activité est créée pour une même instance de la classe `LevelActivity`. L'incrémenter et l'afficher (sur une nouvelle zone de texte) à chaque passage dans la méthode `onCreate`. Tester. Qu'observe-t-on lorsqu'on passe plusieurs niveaux ? Qu'observe-t-on lorsqu'on fait tourner la tablette alors qu'on est dans l'activité `LevelActivity` ? (Pour faire tourner l'émulateur, utiliser Ctrl+F11/Ctrl+F12 (sous Mac, Fn+Ctrl+F11/Fn+Ctrl+F12).) En déduire le nombre d'instances de la classe `LevelActivity`, ainsi que les cas où l'on fait appel à la méthode `onCreate`.

Observation du cycle de vie d'une activité

Les classes héritant de la classe Activity ont toutes un ensemble de méthodes, dont la méthode onCreate, pour gérer leur cycle de vie :



© android.com

Surcharger toutes ces méthodes dans les deux classes `TD3Activity` et `LevelActivity` de manière à:

– appeler la méthode correspondante de la super-classe ;
– afficher sur la sortie standard que l'on a appelé cette méthode de cette classe.

Attention : Pour les méthodes de démarrage (`onCreate`, `onStart`, `onResume`, `onRestart`), il est préférable d'appeler la méthode correspondante de la super-classe en premier, alors que pour les méthodes d'arrêt (`onPause`, `onStop`, `onDestroy`), il est préférable de l'appeler en dernier.

La méthode `void onCreate(Bundle savedInstanceState)` a cette interface, mais les autres ne

prennent pas d'argument (par exemple, `void onStart()`).

Observer l'application :

- en utilisation normale ;
- en faisant tourner la tablette (ou l'émulateur) ; et
- en lançant une autre application, puis en revenant à notre application.

Arrêt d'une activité

Ajouter un bouton « Back » à l'activité `LevelActivity` pour l'arrêter, en faisant appel à la méthode finish()[1] de la classe Activity [2]. Tester.

Résultat renvoyé par une activité

Nous avons vu comment passer des données à une activité que l'on crée, grâce aux intentions ; mais l'activité, lorsqu'elle se termine, peut également renvoyer des données à l'activité qui l'a lancée. Nous allons observer cela en faisant en sorte que l'activité `LevelActivity` renvoie le nombre de fois où elle a été lancée lorsqu'elle termine, et que l'activité `TD3Activity` récupère cette valeur et l'affiche.

Au moment du lancement de `LevelActivity`, remplacer l'appel à `startActivity` par :
`startActivityForResult(intent, NUMBER_OF_LAUNCHES_REQUEST);`

- la méthode est différente, pour explicitement préciser qu'elle attend un résultat de l'activité lancée ;
- elle prend un argument supplémentaire indiquant à l'application lancée quel résultat on attend d'elle.

Comme pour `EXTRA_LEVEL`, définir cet attribut statique et final de la classe `TD3Activity` comme l'entier de votre choix.

Dans la classe `LevelActivity`, surcharger la méthode `public void finish()` :

1. Créer une nouvelle intention, non liée à une classe cette fois-ci :
   `Intent data = new Intent();`
2. Lui associer à un extra de votre choix la valeur du nombre de fois où elle a été lancée, de manière similaire à l'intention créée pour lancer `LevelActivity`.
3. Renvoyer cette intention comme résultat (dans ce cas, l'activité s'est exécutée correctement) :
   `setResult(RESULT_OK, data);`
4. Faire un appel à la méthode `finish()` de la super-classe.

Enfin, dans la classe `TD3Activity`, surcharger la méthode `onActivityResult` [3]de manière à afficher la valeur souhaitée (uniquement dans les cas voulus) :

```
@Override
protected void onActivityResult(int requestCode,int resultCode,Intent data) {
    if (requestCode == NUMBER_OF_LAUNCHES_REQUEST && resultCode == RESULT_OK) {
      int launches = data.getIntExtra(EXTRA_LAUNCHES, 0);
      lnumberV.setText(launches + "");
    }
}
```

où `lnumberV` est une nouvelle zone de texte à définir affichant le nombre de fois que `LevelActivity` a été

[1] http://developer.android.com/reference/android/app/Activity.html#finish()
[2] http://developer.android.com/reference/android/app/Activity.html
[3] http://developer.android.com/reference/android/app/Activity.html#onActivityResult%28int,%20int,%20android.content.Intent%29

lancée.

Tester. Que se passe-t-il lorsqu'on tourne la tablette (ou l'émulateur) lorsqu'on est dans l'activité principale ? Nous verrons dans un prochain cours comment y remédier.

À vous de jouer

1. Faire en sorte que, lorsque l'utilisateur sélectionne « Settings » dans le menu (d'une des deux activités), cela lance une nouvelle activité affichant que l'on se trouve dans les réglages. Le code exécuté lorsque l'utilisateur sélectionne un choix dans le menu est la méthode `onOptionsItemSelected` de l'activité à partir de laquelle on a ouvert le menu.
2. Paramétrer l'interface de cette nouvelle activité pour pouvoir choisir la taille du texte de l'activité depuis laquelle les réglages ont été lancés. Pour cela, mettre trois boutons : petit, moyen, grand.
3. Faire en sorte que cette activité termine lorsque l'utilisateur appuie sur un des boutons, en donnant l'information du bouton choisi à l'activité parente.
4. Faire en sorte que l'activité parente récupère l'information et en tienne compte en modifiant la taille de ses zones de texte.
5. Partager le menu entre les deux activités `TD3Activity` et `LevelActivity`, en partageant à la fois le « layout » et l'action à effectuer lorsque l'utilisateur appuie sur « Settings ». Pour ce dernier aspect, on pourra créer une classe Java auxiliaire qui ne soit pas une activité.
6. Au lieu de trois boutons simples, utiliser des boutons « radio », après avoir regardé la documentation des classes RadioGroup [4] (apparaissant dans la liste des « Containers » de la palette de l'interface) et RadioButton [5]. Faire en sorte que l'activité de paramètres se lance avec le bon bouton coché au départ.

Remarque : nous verrons par la suite comment faire un menu de paramètres qui retient les choix de l'utilisateur entre deux lancements de l'activité.

[4] http://developer.android.com/reference/android/widget/RadioGroup.html
[5] http://developer.android.com/reference/android/widget/RadioButton.html