III • Sudoku

Ce TP implémente un Sudoku très simple. Il y a de multiples façons de l'implémenter :

- Avec des layouts : `TableLayout`, `GridLayout`, `GridView`, `RelativeLayout`, `AbsoluteLayout`
- Avec un `Canvas` (à la manière du serpent)
- En OpenGL

Dans ce TP, on va l'implémenter avec une `GridView`.

1. Créer un nouveau projet, avec Android 5, compatible Android 4.0, en utilisant le modèle « blank activity »
   Q1 : Décrire en quelques lignes l'architecture du projet « Snake » vu en TP1.
2. Dans le layout principal, ajouter une `GridView`, qui doit avoir :
   a. 9 colonnes
   b. Un fond noir
   Q2 : comment avez-vous spécifié la couleur ? Pourquoi ne spécifie-t-on que le nombre de colonnes et pas le nombre de lignes ?
3. Ajoutez un fichier `res/layout/item_sudoku.xml` avec :
   ```
   <?xml version="1.0" encoding="utf-8"?>
   <TextView xmlns:android="http://schemas.android.com/apk/res/android" />
   ```
   Q3 : vu qu'ils ont été définis en XML, comment va-t-on accéder à la `GridView` d'une part et à item_sudoku d'autre part, dans le code des activity ?
4. Dans l'activity principale, utilisez un `ArrayAdapter` pour ajouter 81 cases de type « `item_sudoku` » à la `GridView`, contenant chacune un nombre de 1 à 81
   Q4 : à quoi sert l'adapter ? Quelle est la différence entre un `GridLayout` et une `GridView` ?
5. Modifiez la grid et l'item pour :
   a. Ne pas avoir de marge
   b. Avoir des cases blanches et des bordures noires
   c. Des numéros centrés dans les cases
   d. permettre à l'utilisateur de saisir un et un seul chiffre par case
   Q5 : expliquez chacune vos modifications
6. Créer deux classes `SquareGridView` et `SquareEditText` qui étendent respectivement `GridView` et `EditText`
   a. Dans le fichier layout XML, remplacer `<EditText` par `<votre.package.java.SquareEditText`, idem pour la grid
   b. Pour chacune des deux classes, surcharger la méthode « `onMeasure` », pour avoir une grille carrée avec des cases carrées, quelle que soit la rotation du téléphone
   Q6 : comment avez-vous procédé dans chacune ces deux méthodes ?
7. Créer un `SudokuAdapter` qui étend `ArrayAdapter` et la remplace dans votre activity
   a. Qui prend en constructeur et stocke une `SudokuGrid` (pour le test, récupérer une grille exemple avec `Sudoku.getExampleGrid`)
   b. Qui surcharge la méthode `getView` pour ajouter à chaque widget de saisie un « `TextChangedListener` » qui imprime dans le log les coordonnées de la case lorsque l'utilisateur effectue un changement

Q7 : à quoi la méthode `getView` sert-elle ?

8. Ajouter un bouton « Générer », en plus de la grille

   Q8 : comment avez-vous géré le layout lors du passage portrait-paysage ?

9. Faire en sorte qu'un clic sur le bouton régénère la grille (utiliser `Sudoku.generateValidGrid`)

   Q9 : comment avez-vous fait en sorte que l'application ne reste pas bloquée pendant la génération ?

10. Lors de la modification d'une case :
    a. Mettre à jour la `SudokuGrid` avec la nouvelle valeur entrée par l'utilisateur
    b. Mettre la case modifiée en rouge si l'entrée est en conflit avec une autre case (utiliser `Sudoku.checkValue`)

       Q10 : comment faire pour que les anciennes valeurs ne soient pas prises en compte dans la vérification de la grille, après une regénération de grille ? pour que les autres cases en conflit soient aussi passées en rouge ?

Question bonus :

Pour que ça ressemble vraiment à un sudoku, il faut que certaines lignes soient plus épaisses pour couper la grille en neuf secteurs. Comment faire ça *proprement* ?

Notation :

- Chaque question vaut 1 point (y compris la question bonus) : 10 points (+1)
- Application résultante : 10 point x % complétude
  - Qualité du code : 5 points
  - Bon fonctionnement de l'application : 5 points

Informations générales :

- Le projet doit être réalisé en binôme
- Envoyé par mail avant le **11 décembre à minuit** *(cachet du premier SMTP de confiance)*.
  - olivier.mathieu@keyconsulting.fr
- Le mail doit contenir un zip avec :
  - L'application compilée et packagée
  - Le code source :
    - qui correspond à l'application compilée ;
    - qui compile sans erreur et sans modification.
  - Un rapport au format PDF avec les réponses aux 10 questions.

Une réutilisation non documentée comme telle d'un code source trouvé sur Internet ➔ 0/20