École Polytechnique de Montréal,
Québec, Canada

INF8405 - Informatique mobile

Travail Pratique N.1

Application de jeu pour Android

Auteurs :
Franck Brazier 1815797
Abbas Omidali 1759476
Farshad Tir 1769679

Soumis à :
Fabien BERQUEZ

POLYTECHNIQUE
MONTRÉAL

LE GÉNIE
EN PREMIÈRE CLASSE

15 février 2017

1 Introduction

Le but de de ce TP est de créer une application pour le jeu match-3. Nous avons voulu pour la phase de jeu proposer un modèle Modèle-Vue-Contrôleur ce qui nous semblait le plus approprié.

2 Présentation technique

2.1 Activités

Notre application possède deux types d'activités, des activités où ne sont présent que des boutons pour pouvoir accéder à l'activité de jeu et les activités de jeu et de règles qui sont les activités finales. Ces activités intermédiaires sont :

- Main Activity
  Cette activité est l'activité première de l'application et permet d'accéder à la page de choix des niveaux, aux règles du jeu ou le quitter
- StartActivity
  Cette activité permet d'effectuer le choix du niveau auquel l'utilisateur veut jouer, l'utilisateur est bloqué si le niveau précédent n'est pas réussi

Les activités dites finales sont :

- RulesActivity
  Cette activité comprend uniquement les règles du jeu (compris dans des *TextViews*) sans autre intéraction avec l'utilisateur
- GameActivity
  L'activité la plus importante de l'application qui est la *vue* du jeu, elle récupère la bonne grille depuis un fichier xml grâce à un extra dans la création de l'activité et initialise le contrôleur de jeu.

Chaque classe hérite de la classe BaseActivity qui implémente les fonctions communes à chaque activité, par exemple celles appelées lorsque l'on appuie sur un bouton de la bar d'application située en haut de l'écran.

2.2 Classes

Notre application possède 5 classes essentielles pour le fonctionnement du jeu. il y a deux classes qui implémentent des Listeners. (*OnTouch* et *OnDrag*).

L'*OnTouchListener* permet de faire disparaître l'élement lorsque l'on appuie dessus et qu'on comment à le déplacer. L'*OnDragListener* a surtout pour but de détecter les éléments que l'utilisateur veut échanger et d'*envoyer* cette requête au contrôleur. Les autres classes sont Circle, Level et LevelController.

2.2.1 Level

L'instance de cette classe contient tous les caractéristiques (nombre de lignes, colonnes, score à atteindre,...) d'un niveau de jeu.

2.2.2 LevelController

Certains de ses attributs sont statique pour pouvoir être accessible tout le long de l'exécution comme le niveau maximal débloqué. LevelController est responsable de la grille, de permettre un échange valide, de supprimer les éléments alignés et les remplacer.

2.2.3 Circle

Circle est la représentation d'un élément, ses attributs sont donc son positionnement, sa couleur, nous avons aussi rajouté la position cible de l'élément (par exemple lors d'un échange orchestré par l'utilisateur, la position cible sera celle de l'autre élément, cette position peut ne pas être valable pour un échange)

3 Difficultés particulières

Nous avons éprouvé des difficultés essentiellement pour le contrôleur, spécialement pour créer un état temporaire pour voir s'il y a des alignements et si possible annuler cet état et être dans le précédent sans laisser de trace de cet état.
Nous avons perdu du temps pour pouvoir changer la taille des éléments, cependant nous avons trouvé une solution qui n'est pas optimale en mettant la taille dans des fichier xml en fonction de la taille de l'écran. Notre problème venait du fait que nos éléments héritait du boutons et qu'il était difficile de pouvoir modifier leur taille, nous avons donc fait hérité de *TextView* nos éléments

4 Critiques et suggestions

Nous avons identifié quelques points faibles dans notre application. Voici la liste :

- Nous avons pas réussi à faire en sorte que la grille et les éléments s'épousent parfaitement avec la taille de l'écran
- Nous avons essayé après coup de créer un mode pour voir les différentes étapes dans un même coup ou échange.
- Des artefacts peuvent apparaître dans le jeu (des alignements qui devraient disparaître) ou un élément qui reste dans son état *à enlever* identifiable par sa transparence.
- Nous aurions aimer pouvoir créer des animations.
- Des textes plus plaisant à lire. Le but serait d'avoir un meilleur agencement des *TextViews*.
- Une amélioration serait de pouvoir créer des niveaux aléatoirement avec leur difficultés.
- Nous avons utilisé nos propres smartphones pour les tests, nous avons seulement pu essayer d'obtenir un aperçu sur les plus grands appareil grâce à la fonctionnalité *Preview* d' *Android Studio*

5 Autres

Pour résoudre nos différents problèmes ou manque de connaissances en Android nous avons consulté plusieurs sites, la liste **non-exhaustive** de ces sites sont présent dans le fichier *src/doc/sources*. Ces sites ont inspiré notre code et des parties de celui-ci sont directement issues de ces sites.