

INF8405– Informatique Mobile

TP1 - Application de jeu pour Android

Groupe 01

Cloutier, Frédéric

Lam, Ba Samson 1671028

Vuong, Sylvester

Soumis à : Berquez, Fabien

15 Février 2017

## Introduction

Dans les années passées, nous avons constaté une croissance importante dans le développement des applications mobiles. Ce progrès fut marqué par la naissance des téléphones intelligents. Il serait donc intéressant d’étudier sur la technologie des téléphones mobiles. Plus précisément, nous allons nous intéresser dans le développement des applications mobiles.

Dans le cadre du travail pratique, nous devons concevoir une application mobile à titre éducatif. Le but du travail est d’aider à l’apprentissage de base pour les étudiants. Les étudiants doivent se familiariser aux outils de travail et ainsi qu’à l’environnement des terminaux mobiles.

À cette fin, nous concevons un jeu intitulé “Match-3”. Le jeu en question possède une grille contenant des formes simples ayant des couleurs choisis aléatoirement. Le joueur doit rassembler au moins trois objets de même couleur par un simple geste de *swipe*. Par la suite, les objets connectés sont alors détruits et les objets au-dessus de ses derniers prennent leurs places. De nouvel objets apparaissent dans les emplacements disponibles. La partie est gagnée lorsque le joueur ait atteint le minimum de points requis dans un nombre d’exécution donné selon le niveau de difficulté.

Pour pouvoir accomplir la tâche donnée, nous devons tout d’abord analyser et conceptualiser le jeu. Donc, dans ce rapport, nous allons décrire toutes les aspects techniques concernées. Par la suite, nous énumérons les difficultés rencontrés lors de la conception du jeu. Enfin, nous élaborons une analyse critique et quelques suggestions seront émis au lecteur.

## Présentation Technique

En premier lieu, nous allons présenter la structure du projet. :

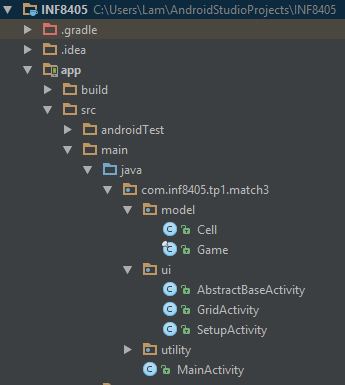


Fig. 1: Structure du projet dans Android Studio

L’activité d’entré est celui du MainActivity. C’est elle que nous avons émis le intent-filter comme *MAIN* et *LAUNCHER* (*action* et *category* respectivement). C’est aussi elle qui donne accès à l’activité du menu de sélection.

L’activité SetupActivity détient tous les boutons pour le choix des niveaux. Ces boutons ont une particularité: elle détermine le niveau auquel le joueur jouera. Dans le cadre du travail, quatre niveaux ont été établis. Le joueur ne peut accéder qu’au niveau qu’il a auparavant débloqué. Par exemple, il ne peut pas jouer le niveau 3 s’il n’a jamais terminé le niveau 2. Les niveaux bloqués sont alors moins opaques que ses contreparties. Voici un petit example d’une simple interface graphique:



Fig. 2: Setup Activity

Comme illustré dans la figure 2, nous apercevons que seulement le niveau 1 est jouable. Le niveau 2, 3 et 4 ne sont disponible lorsque le joueur ait vaincu et atteint les requis.

Ensuite, nous avons l’activité GridActivity qui batit le jeu au complet. Il utilise en fait le layout de type gridLayout. Nous avons choisé ce type de layout pour profiter de ses caractéristiques de grid tels que les rowspec et colspec. Il est donc impératif que lors de la création du GridActivity (à l’appel de *onCreate()*), GridActivity reçoit et analyse le contenu (*extras*) du Intent. Il doit obtenir les dimensions de la table pour qu’il puisse concevoir cette dernière.

D’ailleurs, cette activité produit dynamiquement toutes les cellules (boutons circulaires) selon le niveau donné. Il est donc un gestionnaire des interactions entre le jeu et le joueur. En fait, elle interagit avec ses enfants (cellules), la classe *Game* et le joueur.

La classe *Game* détient toute la logique du jeu.Elle détient aussi les informations du joueur telles que les statistiques de la partie actuelle et le niveau de jeu atteint par exemple. De plus, elle utilise une référence vers la grille (GridLayout) qui lui permet d’effectuer des recherches approfondies lorsqu’elle lui semble opportun.

La classe *Cell*, qui hérite de la classe *Button,* est utilisée par GridActivity et Game pour instancier les cellules. Nous avons créé cette dernière pour ajouter plus de contenus importants pour le jeu. Par exemple, nous avons créé des membres privés (attributs) tels que les voisins de la cellule (*top, bottom, right, left)*. Lors d’une recherche d’un *match*, nous analysons chacun des voisins récursivement jusqu’à temps qu’aucun match adjacent soit possible.

Notre algorithme est le suivant:

1. Le joueur effectue un onTouch (event down)
   1. Nous enregistrons les positions x et y
   2. Une cellule est sélectionnée
2. Le joueur relâche sont doigt (event up)
   1. Nous enregistrons à nouveau les deuxièmes positions x et y
   2. Nous effectuons des calculs pour savoir si le geste s’agissait d’une mouvement de swipe vers une direction précise (gauche, droite, haut ou bas)
   3. Dans le cas valide, nous récuperons la deuxième cellule selon l’orientation du mouvement
3. Une récursion est effectuée:
   1. Nous effectuons une échange de position entre les deux cellules
   2. Nous parcourons chacun des voisins des deux cellules sélectionnées et nous comparons ces derniers avec la cellule sélectionnée (la première des deux ensuite la deuxième lorsque son tour viendra)
   3. S’il y a un match, nous la marquons comme cellule visitée et *matchée*
   4. Sinon, elle est simplement marquée (pour éviter qu’on la revisite plus tard)
   5. Les cellules *matchée* sont alors à son tour analysée (revoir l’étape 3 a)) jusqu’au point qu’aucune autre suite possible
4. Suite à un match (lorsque l’analyse des cellules est terminé), nous devons enlever les cellules marquées et en générer des nouvelles
   1. En utilisant la méthode swapBtn(), nous interchangeons les cellules marquées avec leur voisin du haut jusqu’à temps que son nouveau voisin soit null (il détient une position à la première ligne). En effectuant ainsi, l’ordre des cellules restantes demeure et ces cellules prennent les bonnes positions
   2. Ensuite, nous éliminons (removeView) les cellules marquées
   3. Par la suite, de nouvelles cellules aléatoires sont générées en utilisant les indexes des cellules éliminées (ces indexes sont récupérée avant l’élimination)
   4. Enfin, nous effectuons une mise à jour des cellules concernée (le voisinage)
5. Vérification des *combos*
   1. Toutes les nouvelles cellules seront marquées comme cellules combos
   2. Les cellules combos vont alors passer à leur tour la procédure au point 3 et 4
   3. Si un match est trouvé, les règles de bonus sont alors appliquées

Enfin, nous avons une dernière classe nommée AbstractBaseActivity. Cette classe contient toutes les méthodes de bases et ainsi que les attributs de base. Les classes MainActivity, SetupActivity et GridActivity héritentn d’AbstractBaseActivity.

## Difficultés

### Difficultés personnelles

L’une des plus grandes difficultés est la familiarisation du monde Android. En fait, aucun d’entre nous ait conçu une application auparavant. Nous étions donc confrontés à effectuer beaucoup de recherche pour comprendre le fonctionnement de nos outils.

### Difficultés techniques

Parmi les difficultés rencontrées, nous avons eu beaucoup de souci lors du développement du jeu. Par exemple, lors de la conception de la table, nous avons utilisé le TableLayout dans un premier temps. Le problème avec cette solution réside dans le fait que nous devons toujours revenir au 2e parent puisque le TableLayout utilise souvent des TableRows. Les cellules résident donc dans les TableRows. Par contre, cette solution permet adéquatement de générer les nouvelles cellules puisque nous pourrions orienter les TableRows verticalement et ainsi que ses éléments. Nous avons aussi tenté d’utiliser le GridView, qui est un type *adapter*. Le problème avec ce dernier est l’accès aux éléments. Nous avons eu beaucoup de misère à chercher la bonne documentation pour accèder aux éléments dans un adapter. En recherchant les types de layout, nous avons abouti au GridLayout, qui nous semble être le meilleur compromis.

Une autre difficulté rencontrées est celle de la conception de notre algorithme. Initialement, notre algorithme utilisait beaucoup de ressources et il n’était pas lisible. On pouvait se perdre facilement et de plus le débogage était fastidieux et pénible. Il y a aussi des problèmes de récursion: le match-3 semblerait fonctionner, mais des problèmes de sorties de la récursivité prenaient place. Nous tombons dans une *loop* infinie.

Une dernière difficulté technique à noter est le manque de test. Nous avons basé presque entièrement nos tests en utilisant nos cellulaires. Il faut prendre en considération les différents appareils mobiles tels que les tablettes, les différents types de téléphone et etc.

## Critiques et suggestions

Somme toute, nous avons trouvé que le travail était très pertinent en général. Les spécifications étaient précis et simple à comprendre.

Toutefois, nous trouvons qu’un guide technique pourrait s’avérer très utile pour les débutants. Ce guide pourrait guider un peu les initiés à mieux développer l’application.

Du côté technique, nous trouvons que le jeu est un peu trop simpliste. Il pourrait intégrer d’autre fonctionnalités telles que l’ajout du graphisme ou bien l’ajout des animations. Il pourrait aussi augmenter le niveau de difficulté en intégrant des éléments de temps. L’ajout de son pourrait augmenter l’expérience d’utilisateur. Une base de donnée pourrait aussi être pertinent si le joueur décide de sauvegarder sa partie et quitter le jeu.

Nous trouvons d’ailleurs que la charge du travail était un peu lourde. Un quatrième membre d’équipe pourrait faciliter la travail, quoique cela s’applique seulement pour les débutants.

## Annexe

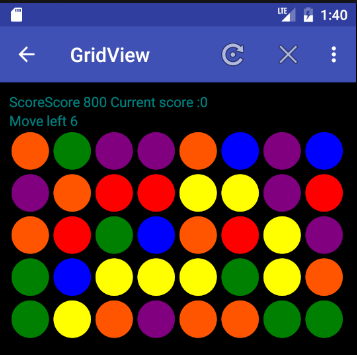


Fig. 3: Mode portrait

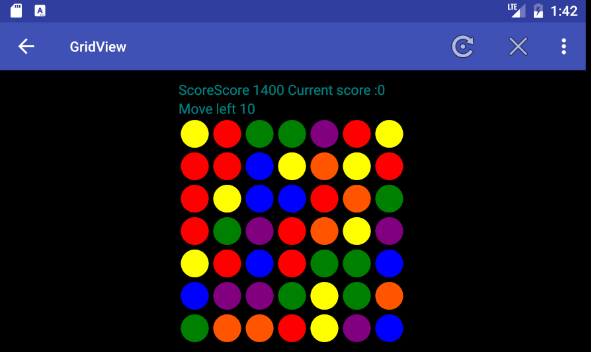


Fig. 4: Mode *landscape*



Fig. 5: Affichage dynamique des gains de points (le +++100 qui *fade out* graduellement)



Fig. 6: Dialogue de victoire

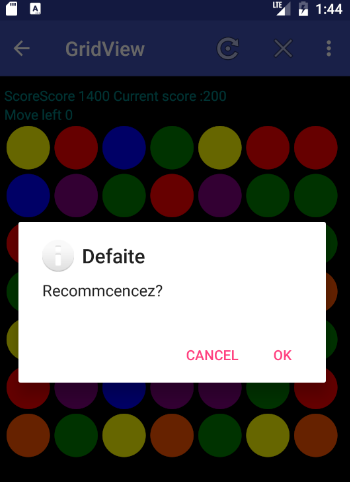


Fig. 7: Dialogue de défaite

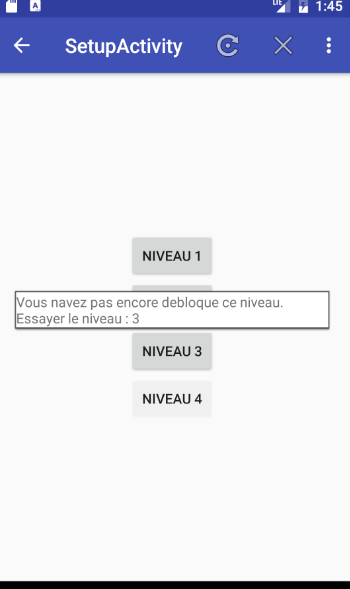


Fig. 8: Niveau interdit

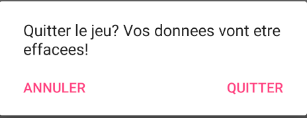


Fig. 9: Dialogue pour quitter le jeu