École Polytechnique de Montréal

INF8405 - Informatique Mobile

Travail Pratique N°1 : Application de jeu pour Android

Étienne Ducharme, 1533453

Laurence Lafontaine, 1529965

Martin Lachance, 1528918

Soumis à Éric Fafolahan

12 mars 2012

Résumé

Ce travail avait pour premier but de nous familiariser avec le développement d'applications pour terminal mobile. Dans un même sens, les buts étaient aussi de nous familiariser également avec le système d'exploitation Android et ses outils de développement reliés, puis avec les caractéristiques particulières des terminaux mobiles tels que l'écran tactile. Plus particulièrement, nous avons dû réaliser un jeu du style *Bejeweled* avec plusieurs menus et avec la conservation des meilleurs scores.

Introduction

Depuis le début de l'informatique, les logiciels développés ont comme plate-forme cible les ordinateurs. Un tel développement est plutôt simpliste par rapport aux entrées à gérer, car il n'a que la souris et le clavier comme vecteur d'entrée. Par contre, le développement des logiciels pour ordinateurs fixes ou portables a perdu quelque peu sa popularité lorsque la popularité des téléphones mobiles a été plus importante. En effet, depuis quelques années, les appareils mobiles sont très utilisés et sont adoptés dans la vie d'une majorité de gens. Par conséquent, les applications mobiles sont de plus en plus en demande.

Ce type de développement apporte de nouveaux défis aux développeurs puisque les entrées d'un logiciel sont multiples. En effet, il y a les entrées faites par l'utilisateur avec ses doigts en touchant l'écran. Celles-ci sont multiples: les doigts sur l'écran, l'inclinaison de l'appareil, sa direction, sa position géographique, la vitesse à laquelle l'appareil est soumis, et ainsi de suite.

Bien que nous étions familiers avec le développement d'un logiciel pour ordinateur, le développement mobile était assez inconnu pour notre équipe. Nous avons conçu une application qui est l'adaptation d'un jeu connu sur mobile, et ceci dans le but de nous familiariser avec le développement d'applications mobiles.

Notre méthodologie lors du développement de notre application sera présentée. Par la suite, il y a aura des tests d'exécution pour démontrer notre application puis une analyse sur nos difficultés rencontrées suivra.

Méthodologie

Répartition des travaux

Pour commencer, nous avons fait des schémas en équipe de l'interface usager et des activités de l'application. Par la suite, nous avons chacun développé notre activité assignée pour que la navigation soit fonctionnelle. Finalement, nous avons divisé le travail pour la zone de jeu elle-même. Une personne était principalement responsable de la surface de dessin qui se met à jour et les animations du jeu. Une autre personne s'est occupée de la logique de jeu pour valider des combinaisons, détruire les éléments formant des combinaisons et ainsi de suite. Puis, l'autre personne s'est occupée de calculer les scores, le temps écoulé, les déplacements restants et sauvegarder le résultat final du joueur.

Phases de réalisations

Notre projet a suivi quatre phases de réalisation: analyse, conception, programmation et validation.

Analyse

Le but de cette phase est de produire une analyse des besoins du client. Ces derniers nous ont été transmis par un document de spécification qui comportent des requis fonctionnels et non fonctionnels. En faisant une bonne synthèse de ces éléments, nous nous assurons de produire une application qui est celle demandée. Nous avons fait plusieurs lectures du document de spécifications pour bien relever les éléments et pour ne rien oublier. C'est également dans cette phase que nous avons réfléchi pour ajouter quelques éléments supplémentaires pour distinguer notre application de celle des autres équipes. Entre autres, nous avons eu l'idée de jouer avec des légumes plutôt que des bijoux ce qui fait que notre application est un peu plus unique.

Conception

Dans cette phase, nous avons commencé avec les requis établis précédemment puis nous avons réfléchis à comment les implémenter. C'est durant la conception que nous produisons des artefacts tels des diagrammes de classes et de séquences. Nous avons fait un diagramme de séquences de l'utilisateur en planifiant quelles activités nous allons créer puis nous avons établi quelles seraient les interactions entre chacune des activités. Nous avons aussi fait quelques squelettes de programmation pour définir dans quelles activités seraient exécutées quelles tâches. Cette tâche s'est faite entièrement en équipe.

Programmation

Cette phase est celle où nous implémentons le logiciel avec les besoins définis et établis dans les phases précédentes. Pour cette phase, nous avons travaillé séparément, mais avec des rencontres à quelques reprises. Le travail des trois équipiers a bien été séparé en trois tiers les plus égaux possible. Nous avons bien traversé cette phase, car notre analyse et notre conception n'ont pas été négligées.

Validation

Lorsque la programmation a été terminée, nous avons procédé à la validation de notre application. Celle-ci consistait à tester le comportement de l'application et s'assurer qu'un comportement douteux ne survenait pas. La validation a été relativement agréable puisque notre application est un jeu assez amusant. Si nous trouvions un comportement indésirable, toute l'équipe en était informée puis la personne responsable de la fonctionnalité touchée s'occupait de résoudre le bogue.

Structure de l'application

Les fichiers de code java se divisent en trois catégories différentes :

- Les activités
    - Les activités de l'application, soit le menu principal (*MainMenu*), la page de scores (*HighScores*) et la page de jeu (*Game*).
- Les vues personnalisées
    - Les implémentations de vue personnalisées, soit notre canvas dynamique sur lequel s'affiche la planche de jeu (*GameView* et *GameViewThread*).
- Autres
    - D'autres classes relatives aux paramètres (Settings), une interface (*MoveListener*) et des classes reliées à l'affichage et la gestion du jeu (*VeggieGrid* et *Veggie*).

Nous avons les trois activités suivantes :

- *MainMenu*
    - Ceci est simplement le menu principal de l'application qui est affiché au démarrage. Il contient quatre boutons: jouer en mode chronométré, jouer en mode *Blitz* avec un nombre fixe de permutations permises, voir les scores et quitter. Il y a également un interrupteur pour activer ou désactiver les effets

sonores.

- *HighScores*
    - Cette activité présente la liste des meilleurs scores avec les noms des joueurs. Les boutons affichés diffèrent si on provient du menu principal ou d'une partie. Si l'on provient d'une partie, on présente des boutons pour rejouer et changer de mode de jeu.

- *Game*
    - Ceci est évidemment l'activité où se déroule le jeu. On présente en haut à gauche un bouton pour recommencer la partie et en haut à droite un bouton pour quitter la partie courante (avec confirmation). Au centre supérieur, on affiche les statistiques du jeu tel que le mode de jeu, le score, le nombre de chaînes réalisées et selon le mode de jeu, soit le temps restant ou le nombre de déplacements restants. Dans le bas, nous avons la grille de jeu. La vue de la grille de jeu est un *GameView*. Nous expliquerons plus loin qu'est-ce que le *GameView*. Ce qui est important de retenir, c'est que l'activité Game s'enregistre comme *MoveListener* auprès du *GameView* et une fonction *onSwipe* est appelée lorsque le joueur effectue un geste pour déplacer un item. Ainsi, toute la logique de jeu est située dans l'activité Game (pause, reprendre, timer, partie terminée, joueur sons, etc.).

Voici une description des classes liées à des vues :

- *GameView*
    - C'est la vue (le canvas) dans lequel on va dessiner la grille de légumes. C'est un *SurfaceView* Android qu'on a modifié pour qu'il détecte les gestes de l'utilisateur et affiche automatique tous les légumes contenus dans sa référence vers une *VeggieGrid*. Il utilise également un *GameViewThread* pour redessiner périodiquement.

- *GameViewThread*
    - C'est un simple thread qui reçoit un *GameView* et le force à redessiner selon un délai spécifié.

Voici une description des autres classes restantes :

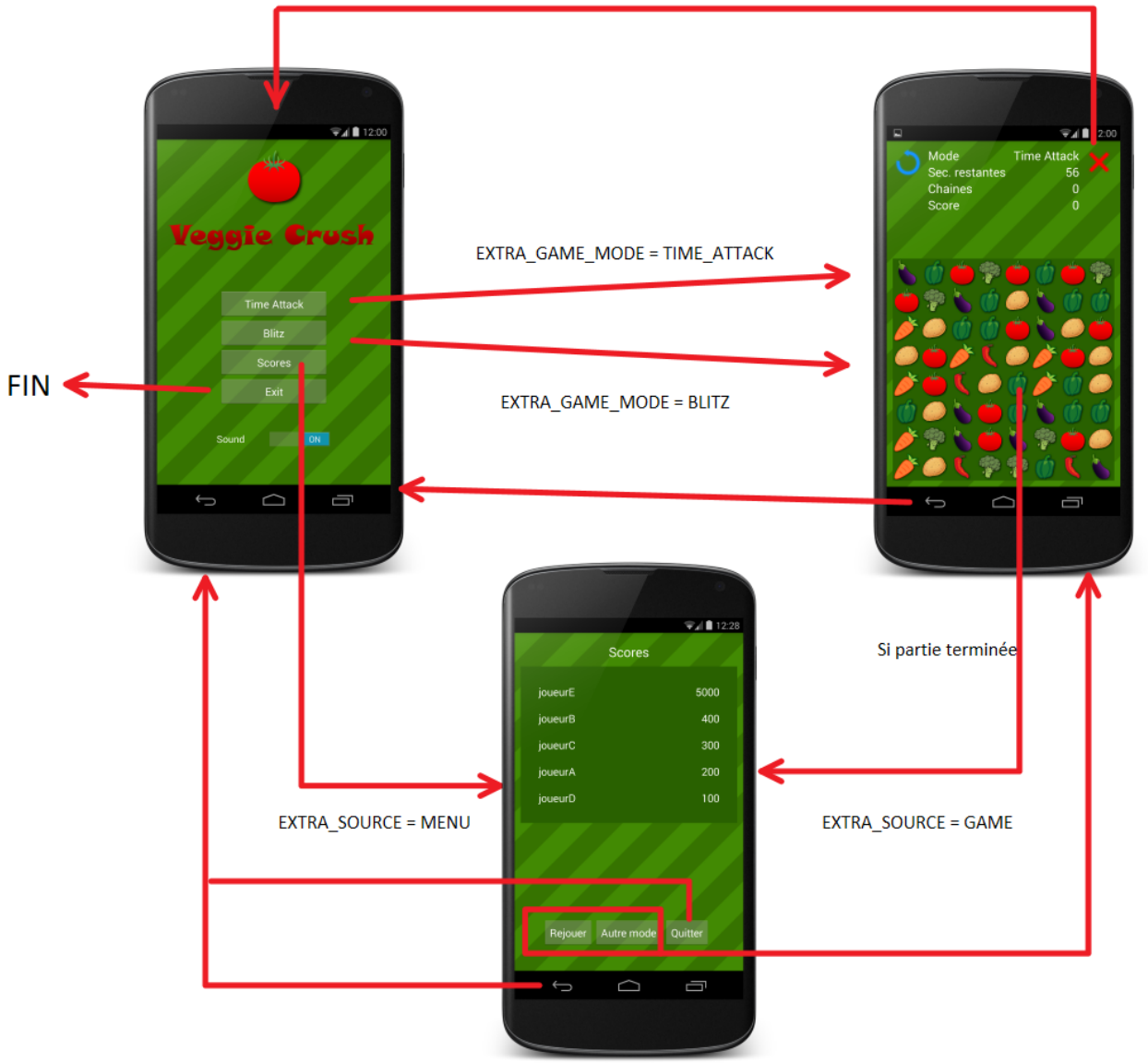
- *Veggie*
    - C’est un simple objet qui est positionné dans la grille de jeu. Il comporte un *bitmap*, un rectangle d’affichage, une sorte de légume (énumération) puis un offset d’affichage (utilisé pour l’animation lorsqu’on interchange deux éléments dans la grille).

- *VeggieGrid*
    - C’est un objet qui contient un tableau de *Veggie*. La grille a une certaine taille et un nombre de rangées et colonnes (8 dans le cas de notre jeu). La classe contient de nombreuses méthodes pour afficher le tableau sur un canvas, réinitialiser la grille, inverser deux légumes de place, vérifier les chaînes présentes, etc.

- *Settings*
    - C’est une classe statique qui contient plusieurs clés, préférences, énumérations et ressources *bitmap* générales à l’application.

Séquence d'exécution

Une séquence d'exécution est présentée dans la figure ci-haut. Les chaînes de caractères affichées en haut des flèches représentent les arguments passés dans la navigation entre les activités concernées.



Difficultés rencontrées

La première difficulté rencontrée fut de choisir la vue à utiliser pour afficher la grille de jeu. Nous avons fait des recherches pour comparer les options possibles (*GridLayout*/*GridView* ou une *SurfaceView* avec canvas). Nous avons finalement opté pour une *SurfaceView* avec canvas qui permet de dessiner de manière plus libre (avec position en pixels) et ainsi faire des animations. Nous avons toutefois dû également utiliser un fil d'exécution pour redessiner le canvas périodiquement.

La seule autre difficulté mineure rencontrée fut pour la gestion du flot d'exécution, plus particulièrement pour le bouton "retour" de Android (en bas à gauche). Après quelques recherches sur les sites d'information du SDK de Google et des expérimentations, nous avons réussi à obtenir le comportement désiré. Le problème était qu'on n'appelait pas toujours *finish()* sur l'activité de la partie qui était terminée ou en quittant l'écran des résultats.

Tests d'exécution

Dans cette section, des exécutions de l'application seront présentées, appuyées par des figures diverses qui sont des captures d'écran.

La page d'accueil qui est démontrée à la figure 1 possède les boutons *Time Attack* et *Blitz* et ces derniers mènent à la figure 2. Ensuite, en demandant et validant le nom du joueur, le bouton *Time Attack* mène à la figure 3 tandis que le bouton *Blitz* mène à la figure 8. Le bouton *Scores* navigue vers la figure 12 et le bouton *Exit* ferme l'application et revient à la page d'accueil Android.

Figure 1 - Page d'accueil

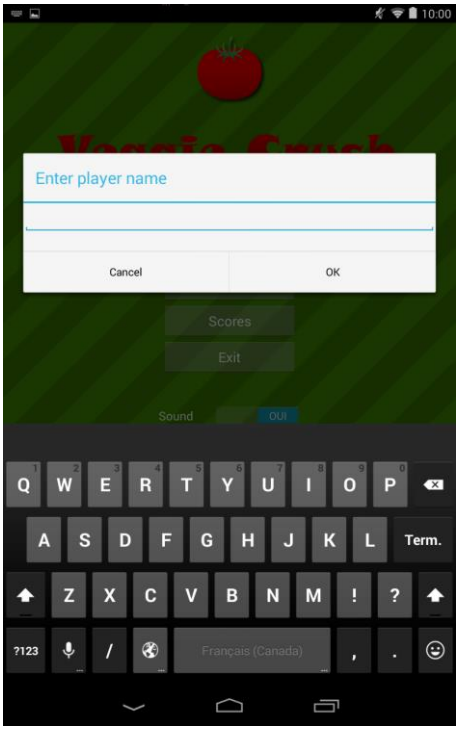


Figure 2 - Entrez le nom d'utilisateur

Figure 3 - Jeu en mode *Time Attack*

En cliquant sur le bouton *Reset* en haut à gauche, le dialogue présenté à la figure 4 apparaît. En annulant le dialogue, la partie en cours recommence et en acceptant le dialogue une nouvelle partie apparaît comme à la figure 3. En cliquant sur le *X* en haut à droite, le dialogue présenté à la figure 5 apparaît. En annulant l'action, la partie en cours reprend tandis que si l'action est acceptée, la partie est quittée et le menu d'accueil est présenté comme à la figure 1.

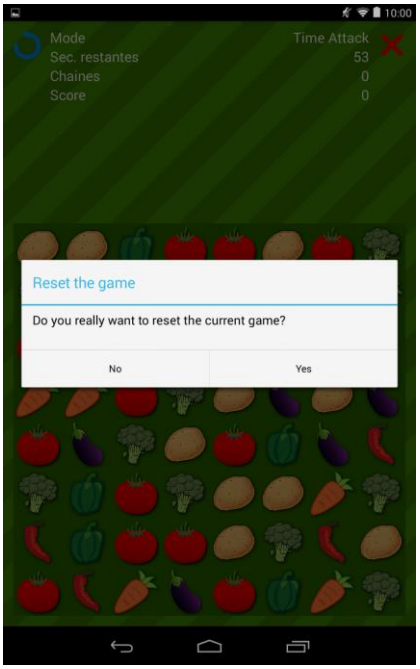


Figure 4 - Confirmation de la réinitialisation

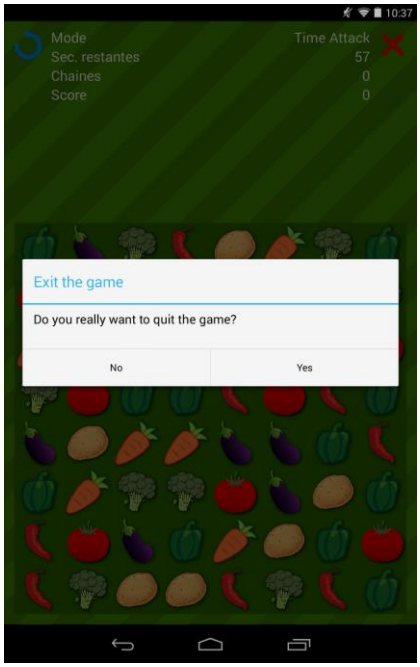


Figure 5 - Confirmation de l'arrêt

Quand le temps est écoulé complètement, un dialogue s'ouvre et confirme le score que le joueur a obtenu comme présenté à la figure 6. Quand le joueur accepte le dialogue, la page contenant les scores est affichée. Comme le présente la figure 7, les scores s'affichent en ordre des meilleurs pointeurs et chaque joueur n'apparaît qu'une fois. Au bas de la figure 7, il y a trois boutons, le premier réinitialise une partie dans le même mode de jeu, le deuxième recommence une partie avec le mode différent et le dernier bouton revient au menu d'accueil de la figure 1.

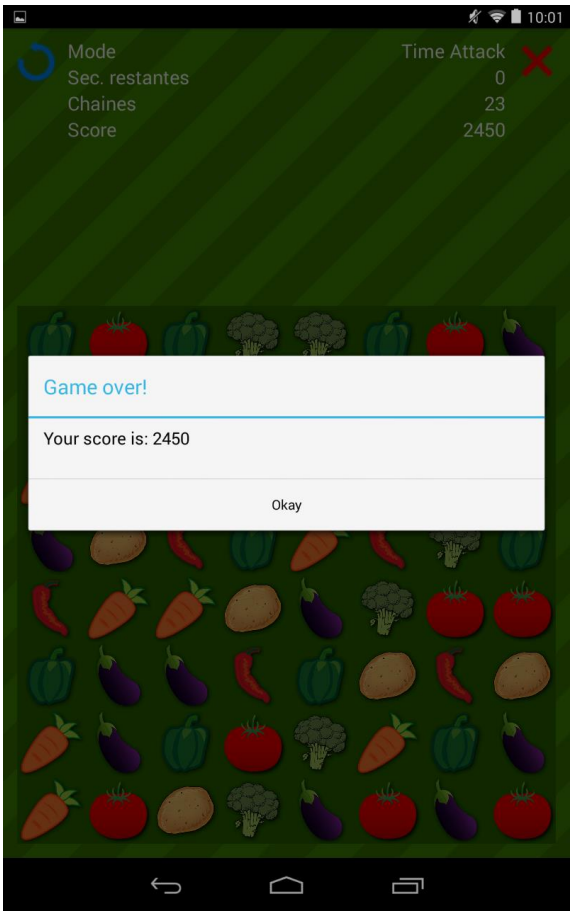


Figure 6 - Fin de la partie

Figure 7 - Présentation des scores

Les prochaines figures illustrent le mode *Blitz*, elles ont le même fonctionnement que les figures dans le mode *Time Attack*. La différence est la fin de la partie qui est déterminée par l'écoulement des dix chaînes de déplacement. La figure 8 correspond à la figure 3, la figure 9 à la figure 4, la figure 10 à la figure 5 et la figure 11 à la figure 6. Après la figure 11, la page des scores est affichée telle qu'expliquée précédemment dans la section du mode *Time Attack*.

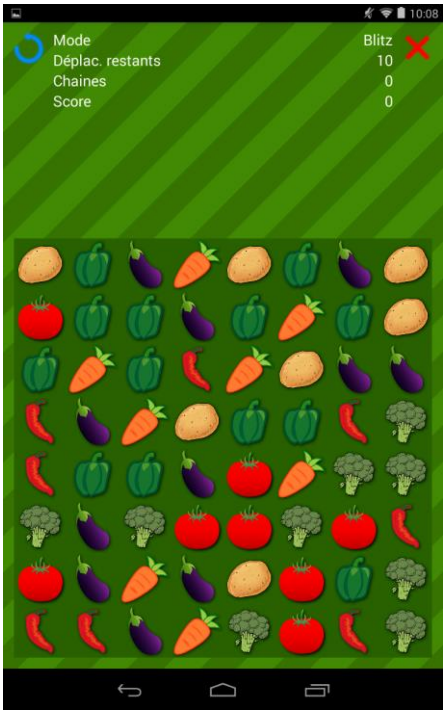


Figure 8 - Jeu en mode Blitz

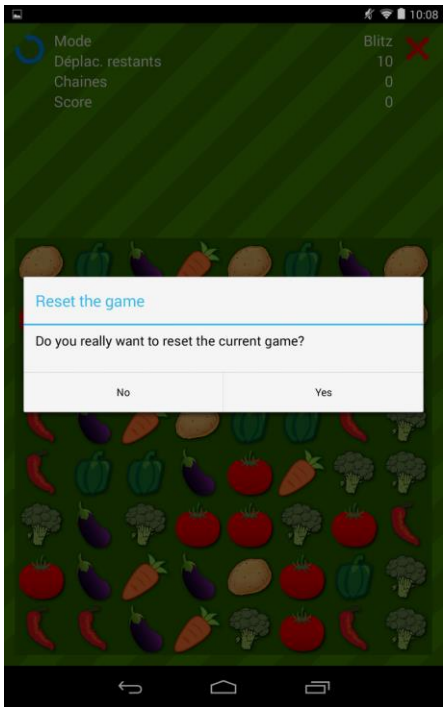


Figure 9 - Confirmation de la réinitialisation

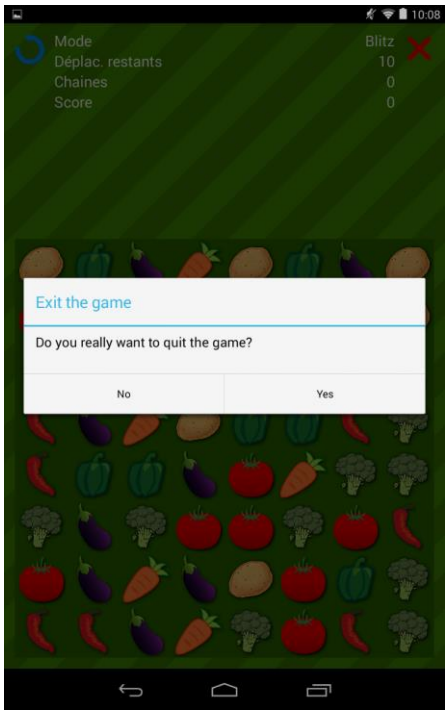


Figure 10 - Confirmation de l'arrêt

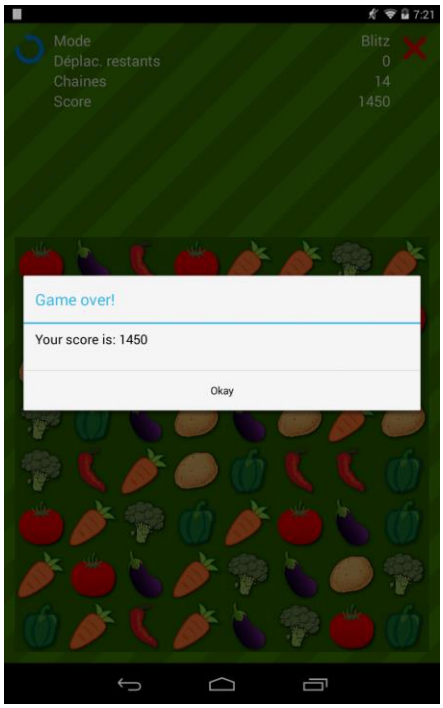


Figure 11 - Fin de la partie

Figure 12 - Affichage des scores

Conclusion

Dans ce laboratoire, nous nous sommes familiarisés avec le développement d'une application mobile. Ce genre de développement est particulier puisqu'il vise une plate-forme spécifique qui possède beaucoup de capteurs et de vecteurs d'entrées. De plus, nous nous sommes familiarisés avec l'API Android pour avoir un niveau élevé d'abstraction lors du développement. En plus de ces familiarisations, nous avons appris que le développement d'une application qui peut devenir très populaire est à notre portée, au niveau technique. Par exemple, le jeu *Candy Crush Saga*™ est un jeu extrêmement lucratif qui consiste en partie en notre jeu.

Des améliorations possibles pour notre application pourraient être par rapport à des éléments qui augmentent le plaisir de l'utilisateur. En effet, nous pourrions ajouter des éléments (des légumes dans notre cas) spéciaux qui auraient des effets lorsqu'elles sont détruites, par exemple détruire une colonne entière. Également, l'utilisateur pourrait obtenir des objets spéciaux lorsque certains légumes qui en contiennent sont détruits. Par exemple, en collectant

un outil agricole, l'utilisateur pourrait l'utiliser au moment de son choix pour éliminer certains éléments.

Par contre, ces idées d'améliorations ne verront pas le jour par manque de temps. Pour améliorer, sans trop changer, notre jeu actuel, nous aurions ajouté une étape de vérification des chaînes possibles. Ainsi, si le joueur est bloqué, l'application lui annoncera et il pourra recommencer. De plus, après plusieurs secondes d'inactivité de l'utilisateur, une des chaînes possibles pourrait s'illuminer pour le guider vers la poursuite du jeu. Cette fonctionnalité ne s'implémente pas trivialement, mais elle augmente de beaucoup la jouabilité et nous croyons qu'elle aurait pu être dans les requis de l'application.