Développement Mobile :
Jeu Snake

Sinama Richard, l3 Informatique
LE LIDEC Tristan, L3 Informatique

1er mai 2021

Résumé

Dans notre rapport, nous verrons pas de façon structuré les différentes parties abordé lors de la création de notre projet. D'une part les approches selon le language, la structure de notre projet et pour finir les différents problème que nous aillons rencontré durant notre création d'application mobile.

Table des matières

1 Introduction

Dans le cadre de l'UE Développement Mobile, nous avons eu à réaliser une application sur les plateformes Android et iOS. Ce projet était libre, nous avons choisi de réaliser un jeu. La première idée que nous avons eu a été de créer un jeu d'échec humain contre ordinateur. Ce projet était trop ambitieux pour nous. Nous avons donc opter pour un jeu plus simple, un Snake. Nous allons tout d'abord vous présenter la façon dont nous avons aborder ce projet. Puis nous détaillerons l'application sur la plateforme Android. Dans un troisième temps, nous verrons celle sous iOS. Enfin, nous examinerons les différents problème que nous avons rencontré durant ce projet.

2 Description générale de l'application

L'application est composé de plusieurs activité. Lorsque nous la lançons, nous arrivons sur le menu. Du menu, nous pouvons accéder au jeu, ou à la liste des meilleurs scores enregistrés. Lorsque nous appuyons sur le bouton *start* le jeu se lance, avec le dialogue de départ qui nous propose de jouer ou de revenir au menu. Si nous choisissons de jouer, alors le snake se mets à bouger. Nous le contrôlons en glissant un doigt sur l'écran. Une pomme est placée aléatoirement, le but est d'en manger le plus possible pour grandir et augmenter notre score. Un panneau stop est aussi placé sur la zone de jeu. Si nous entrons en collision avec, le jeu s'arrête et le dialogue apparait de nouveau. Sur le menu nous avons aussi un bouton qui permet d'accéder à la liste des meilleurs score enregistrés.

3 Approche du projet

Nous avons cherché des applications à réaliser qui nous permettent de mettre en pratique ce que nous avons appris durant l'UE. Tout d'abord, nous avons pensé à créer un jeu d'échec humain contre ordinateur. Cette application n'a pu voir le jour à cause d'un manque de temps, de conaissances et d'organisation. Nous avons donc cherché une application plus simple à programmer. L'application que nous avons choisi est un jeu Snake.

Nous avons développé deux applications, une sur Android et une sur IOS.

4 Application Android

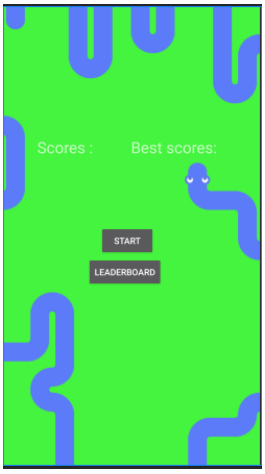

FIGURE 1 – Desing et blueprint du layout de l'activité *ListeBestScores*

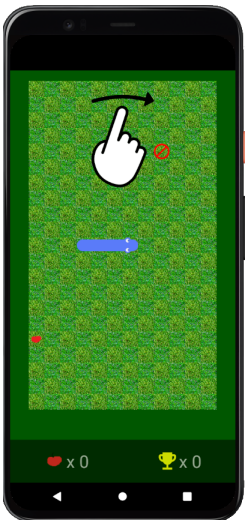

FIGURE 2 – Activité *ListeBestScores*

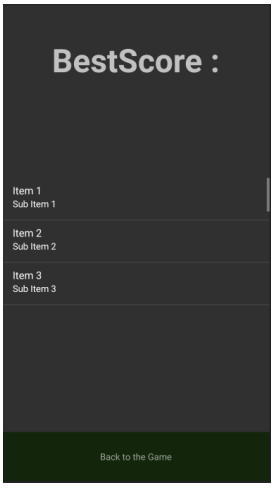

FIGURE 3 – Activité *ListeBestScores*

Pour chaque application nous avons réalisé des recherches sur le jeu que nous voulions créer. Ceci nous a permis de trouver des tutoriels[3]. Cependant nous n'avons pas réalisé la même application, nous avons décidé de modifier celle-ci, de lui ajouter notre touche et les conaissances que nous acquis durant l'UE de développement mobile.

4.1 Logique du jeu

Le jeu Snake est le suivant, nous avons un serpent (Snake) qui parcourt une zone de jeu et qui doit manger des pommes pour grandir. Les pommes mangées représentent le score du joueur. Si le snake sort de la zone de jeu ou qu'il entre en collision avec une partie de son corps, alors il y a Game Over. Nous avons ajouté un panneau stop qui fait mourrir le joueur s'il entre en collision avec. Ce panneau se déplace toutes les 5 secondes.
Pour réaliser la zone de jeu, nous avons construit un tableau qui $lignes \times colonnes$ textures d'herbe qui seront dessinées dans un Canvas. Nous avons utilisé deux textures différentes pour créer un damier.
Comme ce Canva représente notre zone de jeu, pour que le serpent meurent en sortant de la zone de jeu. Nous devons simplement chercher quand est ce que les coordonées de la tête sont supérieures, ou inférieures à celles des textures d'herbe qui se trouvent sur les bords. Pour ce faire, testons la position de la tête à chaque update. Si ses coordonnées sont inférieurs à celles de la première texture ou supérieures à celles de la dernière texture d'herbe, alors nous appelons la fonction gameOver, qui provoque l'arrêt du jeu et affiche votre score avec le meilleur score enregistré.
Durant la partie, si le serpent passe sur la case où est posée une pomme, alors le score s'incrémente de 1, si le score est supérieur au meilleur score, alors celui-ci est remplacé. Si le serpent entre en collision avec le panneau stop, de la même manière qu'avec une pomme, cette fois-ci le joueur meurt et la fonction gameOver est appelée.

4.2 La zone du jeu

Cette zone est construire avec des textures d'herbes. Nous avons deux textures d'herbes. Nous créeons un tableau qui contiendra les *lignes* $\times$ *colonnes* textures. Nous dessinons ensuite ces textures dans un Canvas.

4.3 Le menu

Pour notre menu, nous avons créé une nouvelle activité. A l'aide de différents boutons redirigeant vers les différentes activités que nous avons crée (Le jeu et Le tableau des scores), nous avons rendu le déplacement entre activité possible. Nous avons aussi afficher sur notre menu le dernier score obtenu ($\neq$0), ainsi que le meilleur score de la session.

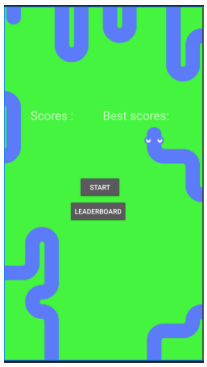

FIGURE 4 – Menu de l'application

4.4 La sauvegardes des meilleurs scores

Pour ce jeu nous ne sauvegardons que les meilleurs. Pour se faire nous utilisons deux systèmes de sauvegardes. Le premier, les shared preferences qui ne gardent en mémoire qu'une petite taille de données. Nous ne gardons ici que le dernier meilleur score enregistré. Nous voulons pouvoir réaliser un historique des meilleurs scores enregistrés. Ceux ci sont enregistrés dans un fichier texte. Nous stockons dans une ArrayList<String> tous ceux que nous alons stockés. Cette liste est mise à jour à chaque fin de partie, si le score est supérieur au meilleur score d'avant la partie (celui qui apparait durant la partie et qui est à battre), alors ce score est ajouté à notre liste. À chaque fois que la fonction gameOver est appelée, cette liste est écrite dans un fichier qui sera lu à la prochaine ouverture de l'application.

4.5 Affichage de la liste des meilleurs scores

Nous affichons la liste des meilleurs stockés dans une nouvelle activité, sous une forme de liste. Pour construire la liste dans cette activité, nous avons utilisé une vue $ListView$[1], il permet d'afficher une liste. Dans la classe de l'activité on crée un $ArrayAdapter$, il prends en argument le context de l'activité, le layout des cellules, et la liste à afficher. Ici c'est la liste des score qui est un attribut de la classe $GameView$.

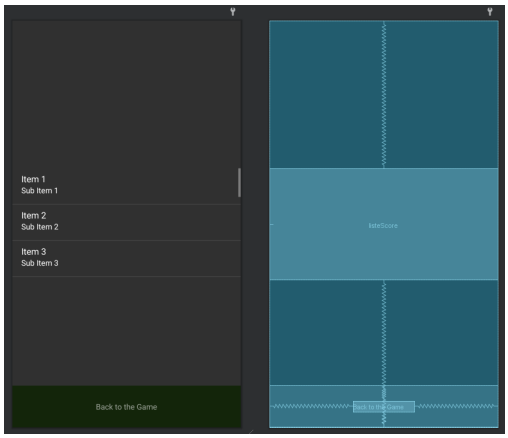

FIGURE 5 – Desing et blueprint du layout de l'activité *ListeBestScores*

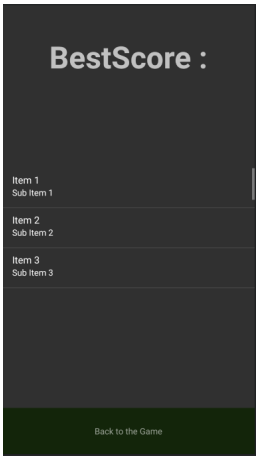

FIGURE 6 – Activité *ListeBestScores*

4.6 Passage entre les activités

Nous avons plusieurs activités dans cette application qui ont chacune un rôle. On démarre sur l'activité principale, le menu, sur ce menu, un bouton permet de lancer le jeu. L'appui sur le bouton *start* démarre une nouvelle activité grâce à un *Intent*. Ensuite on peut soit lancer le jeu, soit revenir au menu. Dans le menu, nous pouvons aussi accéder au tableau des meilleurs scores. C'est aussi une nouvelle activité. Dans cette activité nous affichons simplement la liste des scores qui ont battu les précédents meilleurs scores.
Pour changer d'activité, nosu avons utilisé deux méthodes. Nous avons placé un listener, soit sur un bouton, comme c'est le cas dans le menu, soit sur *RelativeLayout*.
Les listener placés sur les boutons du menu permettent de lancer un nouvelle activité, en l'occurence le jeu ou le tableau des meilleurs score.
Alors que les listener placé dans le dialogue du jeu, ou dans le tableau des scores, terminent l'activité en cours. Ils ne relancent pas l'activité principale. Ils font appels à la fonction *finish()* qui permet d'arrêter une activité.

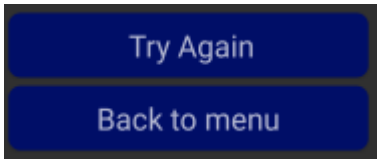

FIGURE 7 – *RelativeLayout* de changement d'activité

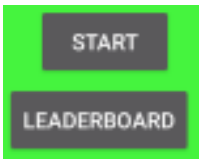

FIGURE 8 – *Buttons* de changement d'activités

4.7 Les Assets

Nous avons pu récupérer plusieurs assets via les tutoriels que nous avons suivi, tel que notre image de Serpent, ou encore des assets libre de droit comme nos texture d'herbe. Nous avons aussi crée certaines de nos assets : le menu de mort (très beau), la pomme et les assets sonore enregistré directement avec le micro de notre ordinateur (Maxime TRITON pour la voix de mort).

5 iOS

Pour la partie iOS, il est important de noter que notre équipe s'est avant tout pencher sur la création d'un jeu snake sur ce système d'exploitation. N'étant pas familier avec le langage *Swift* et la façon de créer une application sur Xcode, nous avons choisi de nous documenter, nous avons trouvé un tutoriels[2] que nous avons décidé de suivre.

5.1 Logique du jeu

Comme pour notre application sous Android, le but du jeu est de manger le plus de pommes pour faire grandir notre snake. Cependant ici nous n'avons pas pu implémenter de façon de mourir. Nous pouvons donc sortir de la zone de jeu et revenir dans celle-ci.
Nous nous déplaçons à l'aide des quatre boutons placés en bas.
Le snake est un tableau de *SnakeCell*. Une *SnakeCell* est une instance d'une classe qui possède les attributs *col* et *row*. Ce sont des *Integer* qui représente la colonne et la ligne dans laquelle se trouve cette partie du Snake.

5.2 La zone de jeu

La zone du jeu a été dessinée et non créée à l'aide de textures, nous avons une view dans laquelle nous avons dessiné $lignes \times colonnes$ carrés depuis une origine (x, y).

5.3 Menu

Nous avons commencer par créer une nouvelle UIViewController nous permettant d'obtenir un écran suplémentaire afin d'implémenter un menu à notre jeu déjà existant. Notre menu nous ramène comme pour notre application Android vers notre jeu à l'aide d'un bouton.

5.4 Passage entre les activités

Le déplacement entre activités (ici ViewController), vu traité à l'aide d'un bouton de redirection. La gestion de déplacement a été geré par *self.performSegue* qu'on utilise pour rediriger vers le ViewController désiré.

6 Contraintes

Nous avons rencontré différents contraintes et problèmes. Nous n'avons pas eu l'occasion de travailler suffisamment sur l'application IOS. Une de nos contrainte fut d'abord dû à notre emplois du temps. Ëtant donné que Tristan est en bilicence, nous avons eu quelques soucis à nous rejoindre au PTU afin de travailler sur la version iOS de notre projet (de même pour le couvre feux). La contrainte la plus importante fut le temps. Nous avons changer de sujet de projet assez proche de l'échéance, ce qui nous a obligé à nos focaliser sur le langage nous permettant de travailler dans les meilleurs conditions. Nous avons tout de même tenté de retranscrire une version non terminé de notre Application Android sous iOS.

7 Bugs

Notre projet Android, bien que fonctionnel ne peut toujours pas être considéré comme terminé. Nous avons encore certains bugs que nous n'avons tout simplement pas eu le temps de résoudre. L'activité Leaderboard fait crash l'application si elle est lancé avant qu'un partie soit joué (donc qu'il y ai une valeur ajouté par le joueur inscrit). La gestion de la rotation n'est pas effectué. Le menu de mort s'affiche alors qu'on lance pour la première fois le jeu.

8 Conclusion

Pour conclure, nous avons crée un jeu "Snake" fonctionnel sur Android, retenant les meilleurs scores et possédant plusieurs activités. Nous avons 2 versions de snake sur iOS une version retranscrite d'Android et une version basé sur un différent systeme.

Références

[1] Le Tutoriel de Android ListView https://devstory.net/10435/android-listview

[2] Snake on Swift iOS - YouTube https://www.youtube.com/playlist?list=PLhct0hTlSsNn_508kHtiH8ZRUfBe6EVgt

[3] Snake Game With Android Studio - YouTube https://www.youtube.com/playlist?list=PLx-Q7RE4RIUEyktbIoeRNtCryYP07OWDM