Université de Limoges
Faculté des Sciences et Techniques Année Universitaire 2016 - 2017

Master Informatique, semestre 1

Java Avancée

Rapport de Projet

version du 18 octobre 2017

Auteur

Nicot Fabien
Le Gac Pierre

Responsable : M. Conchon Emmanuel

Introduction

Pour ce projet nous nous sommes réparties les taches de la manière suivante, Pierre s'est occupé de la parties jeux du projet, de la structure du projet ainsi que la partie SQLlite. Tant dis que Fabien s'est occupé de la partie des scores ainsi que du rapport.

Le projet a été testé sur l'API d'android 19 avec un émulateur de nexus 5 avec une résolution de 1080*1920 et un émulateur 3.4 WGVGA avec une résolution de 240x432.

Pour accéder à l'application il faut rentrer un des identifiants et mots de passe suivants :

- Login : **testeur** Mot de passe : **test**
- Login : **developpeur** Mot de passe : **developpe**
- Login : **joueur** Mot de passe : **joue**

1 Mode Classique

1.1 Fonctionnement du jeu

Le bingo que nous avons réalisé permet de jouer sur une grille 3 par 3, remplis avec des chiffres aléatoire allant de 1 à 100, le joueur se voit initialiser son compte de 500 limcoins. Ensuite pour jouer il faut que le joueur tape sur le numéro qui apparaît sous la grille pour pouvoir valider la case, le but est d'avoir taper sur tout les numéros de sa grille.

Une fois la première grille remplis le joueur passe au niveau suivant avec une nouvelle grille, de plus le vitesse est plus rapide plus le niveau augmente afin de rajouter de la difficulté. A chaque niveau qu'il gagne, il augmente son compte de limcoins de 500.
Au niveau 1, le temps entre deux tirages est de 5s. A chaque fois qu'un niveau est passé le temps entre deux tirages diminue de 0.5s. Le temps entre deux tirages minimum est de 1s pour permettre quand même au joueur de pouvoir continuer de jouer car avec un temps inférieur à 1s il est quasi impossible de finir un niveau.

Pour gagner le joueur ne devra pas taper sur un nombre différent que celui qui est affiché en dessous de la grille. De plus si il n'a pas saisi un nombre qui était sur sa grille alors que la valeur a déjà été tiré, le joueur a perdu et il le découvre une fois que tout les nombres ont été tirés.
Cependant grâce au limcoins gagnés il peut reprendre le jeu au niveau où il était en dépensant la somme de 250 limcoins, quand le joueur n'a plus de limcoins et qu'il perd, il perd définitivement la partie. Il peut recommencer du début ou arriver sur l'écran des scores.

1.2 La réalisation du jeu (ActivityGame)

L'ActivityGame est une activity qui comporte beaucoup de fonctions et de variables puisqu'il faut pouvoir gérer plusieurs types de données pour les grilles, qu'il faut récupérer

des données d'un service et qu'il faut également prévoir les interactions avec le joueur.

Le service qui s'occupe de réaliser les tirages est le BingoService. Il est exécuté par un thread crée dans l'ActivityGame et possède quelques variables permettant d'attendre entre chaque tirage ou encore de savoir dans quel mode de tirage on se trouve (classique, image ou mots).

Quand le joueur a perdu on lui propose soit de recommencer (on recrée un ActivityGame par un intent) ou on le redirige vers l'écran des scores. Dans les deux cas on ajoute son score actuel à la liste des scores.

2 Mode Choix

2.1 Les images

Pour les images nous avons modifié le gridAdapter pour afficher des imagesView ou des textView en fonction du mode choisi. Les images sont stockées dans le dossier app/src/main/res/raw et ont toutes une taille de 32x32 pixels. Si on choisit l'activation du service de tirage, alors les images qui seront utilisés pour remplir la grille et pour les tirages seront aléatoires. Sinon la grille est remplit à l'aide de 9 images prédéterminé dans le même dossier.

Dans ActivityGame on affiche chaque image de la grille ainsi que l'image tirée. Entre le service et l'activity on fait uniquement passer les identifiants des images et on les utilise pour l'affichage. Ainsi on réutilise déjà les structures de données que l'on avait crée pour les grilles de nombres.

2.2 Les textes

Pour ce mode nous utilisons deux fichiers : fruits.txt si le service est activé et android.txt si le service est désactivé. Ces deux fichiers contiennent un mot par ligne mais la lecture des fichiers a été conçu pour que le fichier puisse contenir plusieurs mots par ligne.

Dans ActivityGame on affiche chaque mot de la grille ainsi que le mot tiré. Contrairement aux deux autres modes (nombre et images) on a du revoir les structures de données que l'on utilisait. Donc lorsque le service doit tirés des mots aléatoires il le fait en passant une chaîne de caractères à l'activity.

2.3 Service Actif/Inactif

Dans le mode choix on doit pouvoir activer et désactiver le service de tirage. Lorsque le service est actif on récupère les données de la grille et des tirages de la même manière qu'en mode classique.
Par contre lorsque le service est désactivé, ce sont les neuf premières valeurs d'un fichier (si on choisit le mode fichier) ou les neuf images prédéterminées.

3 Persistance des données

3.1 Affichage score

Sur cette activity nous affichons tous les scores réalisés par les joueurs dans le mode classique du bingo ou dans le mode au choix lorsque le service est activé. Toutefois nous ne nous occupons pas du mode au choix lorsque le service est désactivé car le tirage est indépendant de l'application.

Pour afficher les scores nous avons donc choisi de remplir une gridView. Pour cela nous avons choisi de créer une nouvelle classe Score pour pouvoir un score à un joueur. Elle contient 2 variables, une instance de la classe joueur ainsi qu'un entier signifiant le niveau atteint pas un joueur.

Puis à l'aide d'un GridAdapter (c'est le GridAdapterScore qui est utilisé) nous remplissions la GridView avec tout les scores qui ont été réalisés.

3.2 SQLite

Afin de garder les scores et les joueurs de l'application nous avons décidé d'utiliser SQLite. En faisant quelques recherches sur l'utilisation de SQLite avec Android nous sommes parvenus à stocker les données que nous souhaitions de l'application. Ainsi nous avons deux tables :

- une table Joueur qui contient le pseudo (clé primaire) et le mot de passe d'un joueur
- une table Score qui contient un id (clé primaire) le pseudo du Joueur et le niveau atteint

Pour utiliser SQLite nous avons crée plusieurs classes :

- DatabaseHandler qui hérite de SQLiteOpenHelper et qui permet de faire les requêtes SQL
- DAOBase qui est une classe abstraite dans laquelle on lie un objet qui souhaite réaliser des requêtes SQL au DatabaseHandler
- JoueurDAO qui hérite de DAOBase et qui permet de réaliser les opérations SQL nécessaires (à savoir la sélection)
- ScoreDAO qui hérite de DAOBase et qui permet de réaliser les opérations SQL nécessaires (à savoir la sélection et l'ajout)

Conclusion

Durant le déroulement de se projet nous avons rencontré plusieurs problèmes, notamment en ce qui concerne le temps de réalisation du projet. Avec les nombreux projets en cours dont le projet de moteur 3D et le projet d'algorithmique et programmation avancée nous avons perdu du temps qui aurait pu être utilisé pour ce projet durant la première semaine.

Également certaines fonctionnalités comme la possibilité de pouvoir choisir le dossier d'images, la possibilité de pouvoir remplir la grille comme on le souhaite ou encore la possibilité de pouvoir prendre une capture de l'écran et de l'envoyer par mail, n'ont pas pu être réalisés.

On a aussi été confronté à un problème pour les images tirés aléatoirement. Au départ nous avions décidé de récupérer dans le dossier drawables toutes les images. Ainsi on avait regardé si cela était possible et à l'aide de la classe Field on pouvait récupérer tout les fichiers du dossier. Seulement ce dossier contenait également des fichiers autre que des fichiers images ainsi que des images transparent et à l'aide de la classe Field on ne pouvait pas savoir quel était le type de fichier (du moins nous n'avons pas trouvé comment faire).