Télécom SudParis
Année scolaire 2017/2018

Projet Développement Informatique
PRO 3600

MiniGamesClash

Membres : Lavergne Eric
Wieleba Krystian
Katsoulakis Christos
Jaouhari Youssef

Enseignant responsable : **Sutra Pierre**

TELECOM SudParis

INSTITUT Mines-Télécom

22 Mai 2018

Sommaire

1 Introduction

2 Cahier des charges

3 Développement

- 3.1 Analyse du problème et spécification fonctionnelle
- 3.2 Conception préliminaire
- 3.3 Conception détaillée
- 3.4 Tests
- 3.5 Intégration

4 Interprétation des résultats

5 Manuel utilisateur

6 Conclusion

7 Bibliographie

8 Annexes

- 8.1 Annexe 1 : Planning Prévisionnel
- 8.2 Annexe 2 : Code source

1 Introduction

Ce document présente le déroulement et le résultat de notre travail pour le module PRO 3600 dans lequel nous avons développé un jeu multijoueur sous Android. Dans une première partie, nous présentons le cahier des charges. Nous donnons ensuite les détails du développement en présentant les différentes phases telles que l'analyse du problème et les spécifications fonctionnelles, la conception préliminaire puis détaillée, le codage, les tests et l'intégration. Enfin nous présentons le manuel utilisateur pour faciliter la prise en main de notre application. Nous terminons alors par une conclusion avec des pistes d'améliorations.

Le développement de cette application a été réalisé par Christos Katsoulakis, Eric Lavergne, Krystian Wieleba et Youssef Jaouhari encadrés par M.Sutra.

2 Cahier des charges

Concept global

Le projet consiste au développement d'un jeu mobile sous Android opposant deux joueurs en ligne sur plusieurs mini-jeux rapides.

Le joueur s'identifie, ou participe de manière anonyme. Il trouve ensuite un adversaire en ligne grâce à l'un des deux modes proposés : défier un ami / défier un joueur au hasard. Ainsi le joueur pourra construire une liste d'amis en ajoutant l'identifiant d'un joueur.

Ils s'affrontent ensuite dans une série de mini-jeux différents dans un ordre aléatoire ou déterminé. La partie entre deux joueurs se termine lorsque l'un des deux joueurs atteint un certain score. Ces mini-jeux auront des graphismes et un gameplay simples et efficaces ; cela participera à une ambiance de jeu fun et décontractée et nous permettra de nous concentrer sur l'aspect multijoueur.

Mode Training

Un mode "Training" pourrait également être disponible ; le joueur y jouera seul au jeu de son choix (avec une optique de record ou contre l'ordinateur en fonction de la logique du jeu). Cela nous permettrait de commencer à implanter les graphismes et les fonctions de base de chaque jeu dans un premier temps et d'incorporer les difficultés du mode "Versus" dans un second temps.

Mini-jeux proposés

Les mini-jeux seront donc proposés en 2D parmis les suivants :

- Presser un bouton lorsqu'un compteur atteint zéro ou qu'un indicateur apparaît à l'écran
- Eclater sur l'écran les bulles de la couleur de son adversaire (qui se déplacent à la manière de bulles de savon donc) et pas les siennes
- Un morpion
- Une course d'escargots où il faut balayer l'écran pour faire avancer son escargot
- Un hockey
- Attraper un maximum de fruits qui tombent des cieux avec un personnage en bas de l'écran.
- Un mini-golf 2D en une manche.

Des extensions pourront être réalisées avec des mini-jeux supplémentaires, plus complexes, comme un combat de plateforme ou une course 2D.

Spécificités techniques

Cette application sera développée grâce au langage Java et au logiciel Android Studio. Ce dernier intègre en parallèle le langage xml pour l'aspect strictement graphique des interfaces de l'application. L'une des principales difficultés de sa programmation résidera dans la gestion du multijoueur en ligne, avec une transmission des données qui doit se faire quasiment en temps réel pour certains mini-jeux.

Aperçu de l'esprit du jeu

L'image ci-dessus (non contractuelle) tente de résumer l'esprit du jeu : d'un gameplay simple et amusant, les mini-jeux doivent s'enchaîner rapidement - un peu à la manière dont se déroulent les parties dans certaines applications de quiz à deux joueurs.

3 Développement

3.1 Analyse du problème et spécification fonctionnelle

Analyse du problème

Nous nous sommes posés plusieurs questions de manière à analyser et fixer plus précisément les objectifs et le cadre du sujet.

- Est-ce que les jeux sont proposés dans un ordre aléatoire ou sont choisis tour à tour par un des joueurs parmis une liste de trois ou quatre jeux (la liste des jeux changeant à chaque tour) ?

La seconde option peut apporter une ‘’tension’’ dans le sens où un joueur pourra sélectionner délibérément un jeu qui mettra en délicatesse son adversaire. Toutefois, ce mode de sélection ne présente un intérêt que si le nombre de mini-jeux proposé est bien supérieur au nombre de manches disputées, ce qui ne sera sans doute pas le cas. De plus la première option est moins compliquée à coder et peut apporter tout autant en tension dans le jeu. C’est pourquoi nous avons choisi une sélection des jeux au hasard.

- Combien de mini-jeux gagnants ?

Une série à un unique mini-jeu gagnant ne constitue pas à proprement parler une série et n’offre pas la possibilité de revanche immédiate. Par ailleurs, trois jeux gagnants peuvent entraîner la nécessité d’effectuer cinq manches (score 3-2) soit cinq jeux au minimum. Nous choisirons donc l’option intermédiaire de deux mini-jeux gagnants.

- A quel moment est-ce que le joueur s’identifie ?

Le joueur peut s’identifier au lancement de l’application ; une première interface d’identification est d’abord affichée dans le cas où l’application ne retient pas l’identifiant du joueur. Une alternative pourrait être de demander la connexion seulement après que l’utilisateur ait sélectionné le choix du mode multijoueurs, puisqu’elle n’est pas nécessaire pour le mode Training.

- Est-ce que le joueur possède un identifiant stable et la possibilité de construire sa liste d'amis ou bien il peut changer d'identifiant à chaque connexion ?

Créer une liste d’amis pour l’utilisateur implique de nombreux détails techniques : il faudrait stocker l’ensemble des joueurs dans une base de donnée - dont on ne connaît pas le possible emplacement. Peut-être toutefois qu’il existe un service (comme Google Play) qui permette d’utiliser de telles fonctionnalités en facilitant les choses. L’autre option est sans

doute moins complexe, mais elle a des désavantages certains : l'utilisateur doit contacter ses amis et leur demander leurs identifiant pour les défier. Après réflexion, elle nécessite aussi la présence d'une base de donnée, par exemple ne serait-ce que pour éviter d'enregistrer deux utilisateurs avec le même identifiant et pour les enregistrer temporairement. Pour ce qui est du mode adversaire aléatoire, le choix entre les deux options a peu de conséquences sur la réalisation.

- Est-ce que les joueurs pourront mettre une partie en pause / la quitter ?

La possibilité de faire pause dans une partie à deux joueurs, surtout où la plupart des mini-jeux sont à temps réel, est un cadeau empoisonné pour celui qui subit la pause. A prohiber donc d'autant plus que les parties sont conçues pour être courtes et jouées d'une traite. De la même manière, il vaut mieux ne pas laisser au joueur la possibilité de quitter la partie autrement qu'en écrasant l'application (de manière à ne pas faciliter la vie des mauvais joueurs). Ainsi, si un joueur est déconnecté la partie devrait se terminer en donnant la victoire au joueur restant.

- Un mode Training est-il nécessaire ?

La principale visée du jeu est le mode multijoueur mais le mode Training conserve un intérêt car il permettra aux joueurs de mieux appréhender certains jeux. Surtout, cela nous permettra dans un premier temps d'affronter uniquement la conception du jeu puis dans un second temps de faire face aux difficultés du mode multijoueur.

- Est-ce que les scores des parties précédentes sont stockés et affichés au joueur ?

Une fenêtre de statistiques avec le nombre de parties jouées par le joueur et son ratio victoire/défaite est envisageable. Cependant ceci n'est possible que si chaque joueur a un identifiant stable, car dans le cas contraire il semble difficile de stocker les données des joueurs.

- Y aura-t-il de la musique et des bruitages ?

Java fournit des bibliothèques qui devraient nous permettre d'incorporer des sons sans trop de difficultés. Ainsi, on pourrait implémenter une musique de fond différente selon le jeu, et en plus des bruitages lors des actions effectuées, que ce soit lors d'un jeu ou en parcourant le menu.

- Un onglet Réglages dans le menu est-il nécessaire ?

Une interface de réglages pourrait comporter la possibilité de couper le son de l'application, de changer la luminosité ? (pour que l'interface ne soit pas vide sinon pas nécessaire); autoriser les notifications comme une demande en amis ? (beaucoup d'applis ont cette option); pour les âmes sensible qui ne veulent pas se faire insulter à la fin d'une partie, masquer le chat ?
Un onglet “Compte” avec l'identifiant du joueur et l'option de le changer.

Un onglet “App” avec les informations de notre jeux telle que le cadre dans lequel nous l’avons développer, la version actuelle et le nom de ses créateurs. Nous pourrions penser cependant à faire cet onglet à part, hors des réglages.
Malgré tout, un onglet “Réglages” est vraiment optionnel et donc dans le cadre de ce projet, à priorité faible.

- Un chat entre les joueurs est-il nécessaire ?

Le chat possède plusieurs intérêts comme la possibilité d’envoyer des piques à son adversaire et ainsi d’augmenter la rivalité, ou au contraire de féliciter son opposant. De cette manière la tension et l’enjeu d’une manche en sortent renforcés. De plus cela pourrait permettre de programmer une revanche. Toutefois, dans le jeu que nous concevons, où les parties sont rapides et jouées d’une traite, on peut se demander ou le situer : seulement un échange à la fin qui se coupe ensuite ? Ou entre les parties avec n’importe qui ? Avec la possibilité de rejouer contre son adversaire en fin de partie, un chat rajouterai une interaction de plus qui pourrait mener à une revanche. Alors que un chat “permanent” permettrait de discuter avec ses amis. Cependant, cela représente trop de travail supplémentaire et n’est pas d’une utilité fondamentale ; nous nous concentrerons donc sur les autres fonctionnalités.

- Comment mettre deux joueurs en contact, avec le mode aléatoire ou avec le choix par identifiant ?

En considérant que le jeu ne se déroule pas au tour par tour mais en temps réel, la problématique de réunir deux joueurs au même moment se pose. Dans le cas du choix par identifiant, l’application peut afficher une fenêtre ou un onglet intermédiaire que le joueur défié peut accepter ou rejeter. Il faudra aussi veiller à transmettre l’échec de la requête au joueur défiant dans le cas d’un rejet ou d’une connexion handicapante. Il faudrait également autoriser le joueur à annuler la recherche d’adversaire afin de ne pas être coincé si aucun adversaire n’est trouvé. Dans le cas du mode aléatoire, il faut que deux joueurs se connectent pendant un faible intervalle de temps qui coïncide. L’application se charge de mettre en relation les deux joueurs, et la partie démarre automatiquement sur leur écran après le temps de recherche, avec le nom de l’adversaire affiché sur la première interface intermédiaire. Une fois encore, il existe sans doute des bibliothèques ou des modules pour faciliter ces opérations.

Spécification fonctionnelle

Les fonctionnalités choisies sont donc les suivantes :

- *S’identifier*
- *Jouer seul pour s’entraîner sur les jeux proposés (Mode Training)*
- *Défier un adversaire aléatoirement*

- *Défier un adversaire à l'aide de son identifiant*
- *Jouer contre l'adversaire défié*
- *Avoir connaissance des scores entre deux mini-jeux vs*
- *Disposer de musique et de bruitage pendant la partie*
- *Accéder à l'aide*

Tests de validation :

Beaucoup de cas d'erreurs seront bien évidemment spécifiques aux jeux.
Cependant, on peut déjà penser les tests suivants comme pertinents :

- Si un joueur lance l'application
  Résultat : une interface d'identification est ouverte ou l'utilisateur est automatiquement connecté
- Si un joueur entre un nom d'utilisateur déjà pris
  Résultat : message d'erreur "Nom d'utilisateur déjà pris, veuillez entrer un nom différent"
- Si un utilisateur entre un nom vide comme identifiant
  Résultat : message d'erreur "Veuillez entrer un nom d'utilisateur non vide".
- Si la recherche d'adversaire n'aboutit à rien au bout d'un certain temps
  Résultat : annulation de la recherche et message d'erreur "Aucun adversaire trouvé".
- SI le joueur adverse se déconnecte au cours de la partie
  Résultat : affichage du message d'erreur "Adversaire déconnecté" suivi d'un retour au menu
- Si une partie dure trop longtemps au cours d'un jeu (blocage du jeu)
  Résultat : égalité entre les joueurs et passage à l'interface intermédiaire à la fin d'un compteur

3.2 Conception préliminaire

Découpage en modules

Main : ce premier module correspond à la sempiternelle fonction principale avec initialisation, lancement de l'application et affichage du menu.

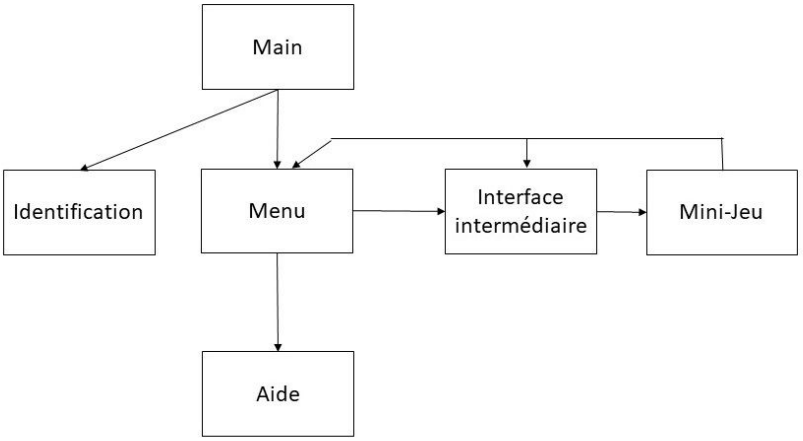
Menu : Cette première interface guide l'utilisateur vers les options qui s'offrent à lui : Jeu à deux joueurs en ligne, Jeu en mode Training, Liste d'amis, Aide, …

Aide : Il s'agit d'une interface qui expose le fonctionnement du jeu dans les grandes lignes.

Interface intermédiaire : Ce module affiche une interface avant / après chaque mini-jeux qui présente la situation (les scores respectifs des deux joueurs, l'identifiant du gagnant si la partie est terminée) et présente le mini-jeu qui va suivre (avec un message simple tel que "Eclatez toutes les boules rouges - et pas les bleues").
Avant un mini-jeu, cette interface attendra un clic du joueur pour disparaître et laisser la main au mini-jeu. Dans le cas de l'affichage de fin de partie, deux options seront affichées en bas de l'écran ; une option permettant aux joueurs de rejouer devrait être disponible - puisque les parties sont rapides, les joueurs voudront avoir leur revanche sur leurs amis sans repasser par le menu, l'autre permettra de revenir au menu.

Mini-jeu : Chaque mini-jeu représente un module à part entière. Il attend que les deux partis soient prêts (voir précédemment) pour démarrer. Il prend fin à la victoire d'un joueur et cède alors la place à l'interface intermédiaire.

Identification : Ce module affiché au lancement du jeu a pour but de permettre au joueur de créer son identifiant. Il fera également lien avec une base de donnée des joueurs inscrits.



Structures de données, stockage des information

Différentes informations devront être enregistrées de manière à ce que l'utilisateur puisse y accéder fréquemment.
Certaines pourront être stockées dans l'appareil mobile du joueur : ce serait par exemple le cas d'un possible enregistrement des scores des parties jouées, que l'utilisateur consulterait dans un onglet du menu.
C'est aussi le cas pour la liste des identifiants des amis du joueur, sous la forme d'une liste de caractères.
Une base de données sera utilisée pour lister les identifiants de tous les joueurs inscrits dans l'application. Elle regroupe donc leurs identifiants et leur adresse IP.

Tests d'intégration

Il faut faire un choix parmis plusieurs méthodes de test d'intégrations : top-down, up-down, sandwich, big-bang.
Les deux méthodes les plus employées sont les méthodes top-down et up-down.
Elles consistent toutes les deux à effectuer les tests de manière hiérarchique cependant dans un ordre différent. En effet, la méthode top-down consiste à tester les modules à partir du haut, et donc à partir du main, puis en redescendant. Alors que la méthode up-down, au contraire, consiste à partir des modules inférieurs et donc les différents mini-jeux pour ainsi remonter en conséquence.
Dans notre cas cela paraît plus cohérent de tout d'abord tester les différents jeux, et ensuite remonter vers le main. Et donc d'adopter un test d'intégration du type up-down.

3.3 Conception détaillée

Chaque jeu a tout d'abord été développé indépendamment des autres. Ainsi, l'implémentation a dû se faire en plusieurs étapes.

Les jeux ont d'abord été développés pour le mode "Training", permettant ainsi de pouvoir tester le bon fonctionnement du jeu en lui-même, sans la nécessité d'un partage de données entre joueurs.

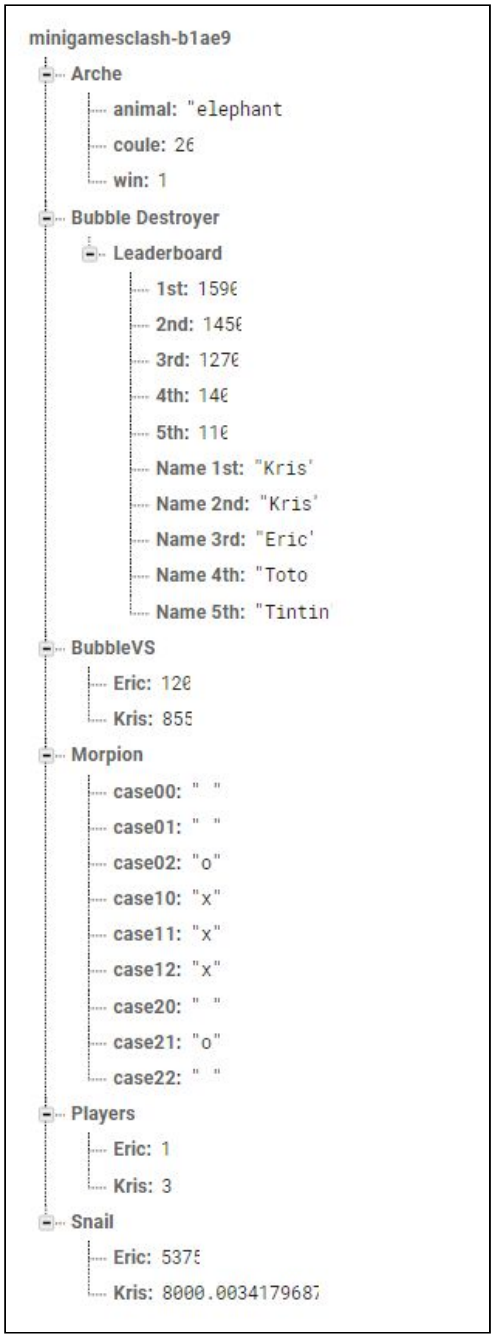
Pour passer à l'étape suivante, il nous fallait être capable de faire communiquer deux joueurs entre eux. Un service de partage de contenu s'est avéré nécessaire. Nous avons en l'occurrence utilisé Firebase. Ainsi se familiariser aux bases de données de cet outil fut une phase clé pour l'accomplissement de ce projet. Cet apprentissage s'est tout d'abord effectué avec l'implémentation d'un leaderboard pour le jeu "Bubble Destroyer". Cette première tâche a permis d'avoir une bonne connaissance des différentes méthodes utiles pour écrire et lire à partir de Firebase. Ainsi, dix "childs" différents ont été requis pour l'établissement du leaderboard des cinq meilleurs joueurs, ceux qui stockent les scores et ceux qui stockent les noms des joueurs. (**voir "Leaderboard")**

Cela nous a ensuite permis d'implémenter le multijoueur à chaque jeu, toujours séparément afin de pouvoir isoler les différents problèmes et erreurs rencontrées. Même si certaines méthodes dont le "listener" sont reprises régulièrement, chaque jeu a dû être intégré différemment à Firebase en fonction de ses spécificités : en tour par tour , à temps réel, par échange de score. Ainsi, "**BubbleVS"** possède un "child" respectif au joueur qui prend comme valeur le score obtenu. L'**"Arche"** a trois "childs", l'**animal** sélectionné par le joueur dont c'était le tour, le poids **"coule"** qu'il représente, et enfin **"win"** qui a pour rôle d'indiquer au joueur que son adversaire a fait couler l'arche, et donc sa victoire. Le **Morpion** a un "child" pour chaque case avec pour valeur un String représentant son état parmi { " ", "o", "x"}. Quant à **Snail**, le "child" du même nom que le joueur prend comme valeur la distance parcourue et donne la victoire au premier joueur ayant atteint le bout de son écran. Un listener est appliqué sur ces différentes références de la base de données Firebase, et provoque des actualisations de l'état du jeu diverses - dans le cas de l'arche et du morpion par exemple, cela débloque le joueur dont c'est le tour.

Après que chaque jeu ait été intégré au multijoueur et donc à Firebase, il a été nécessaire de créer un système d'identification pouvant faire suivre les deux joueurs respectifs lors des différents jeux. Le but initial de notre projet est en effet d'organiser un duel via une série de mini-jeux s'exécutant à la suite. Ce système d'identification se fait en entrant un nom. Ainsi, chaque joueur sera identifié par son "child" auquel on affecte une valeur initialement nulle servant de score. **Voir "Players".** La valeur de ce "child" sera alors incrémentée à chaque jeu remporté par le joueur et affiché dans la fenêtre "Interface". Cette identification a dû être implémentée à chaque jeu, c'est à dire, notamment devoir faire suivre le nom des joueurs entre les activités ainsi qu'une variable sous le nom "tour" prenant comme valeur "0" pour un joueur et "1" pour l'autre joueur, et servant pour les mini-jeux tour par tour à désigner celui qui ouvre la partie.

Une fois en possession de nos différents jeux fonctionnant en mode Training et en mode VS ainsi que nos éléments Règles et Crédits, il a fallu assembler le tout et créer des liens entre les différentes activités. Pour ce faire un Menu Principal permet d'accéder aux différentes

fonctionnalités de l'application (Training, VS, Règles, Crédits) puis un sous-menu dans le mode Training offre une vue sur l'ensemble des mini-jeux disponibles. Dans le sens inverse il y a par exemples un bouton Retour qui permet de remonter de Règles et Crédits vers le Menu Principal, ou des mécanismes de retour vers le Menu Principal une fois les mini-jeux du mode Training finis (par un bouton Retour ou de manière automatique).

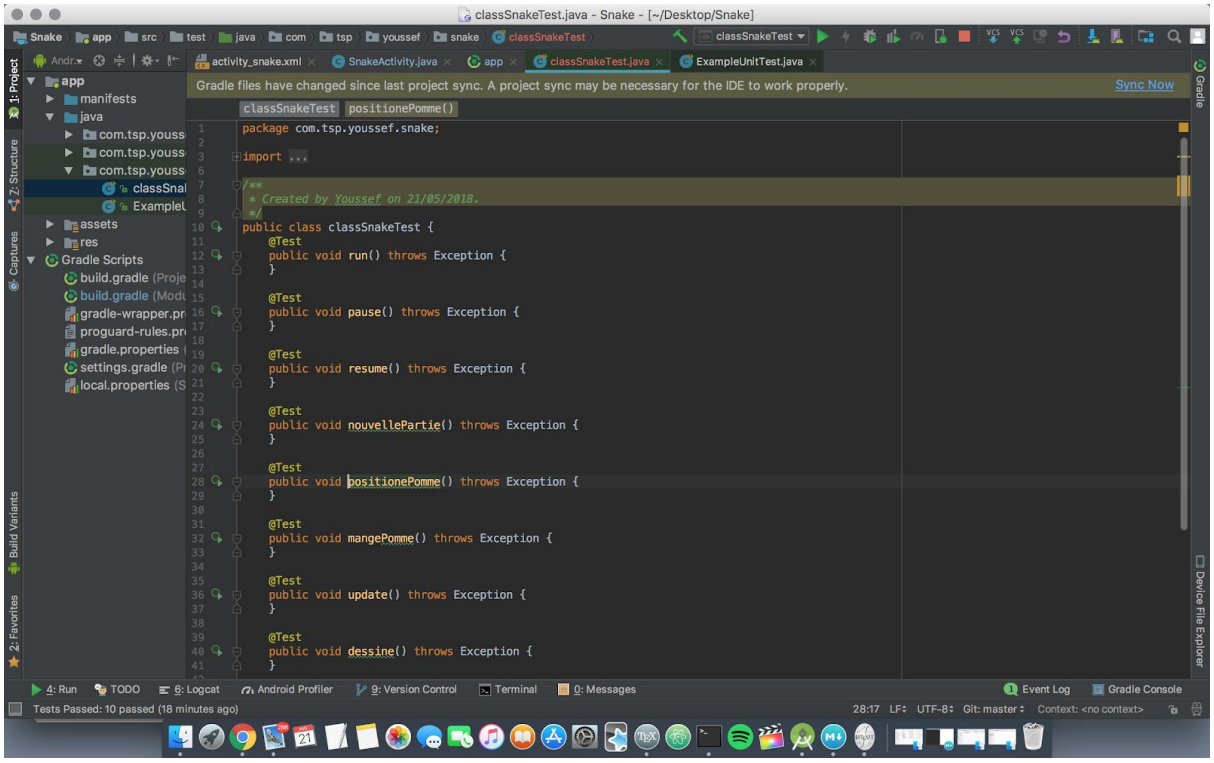


Nous sommes amenés, après avoir codé les fonctions et procédures nécessaires au fonctionnement de notre programme, à faire des tests sur celles-ci pour s’assurer de leur bon fonctionnement.

31 Tests unitaires

Un test unitaire a pour but de valider le bon fonctionnement d'une partie du programme (Fonction dans notre cas). Il vérifie l'adéquation des résultats obtenus par rapport aux résultats attendus. Dans cela, nous avons été amenés à utiliser JUnit directement sur Android Studio.

Nous avons décidé d'effectuer les tests sur des méthodes de jeux bien choisis de telle sorte que le test soit facile à réaliser. Une capture d'écran des méthodes de test est donnée ci-dessous.



Nous avons effectué des tests unitaires sur des méthodes hors celles du projet avec succès. Toutefois, nous nous sommes heurtés à la difficulté des méthodes que l'on a écrites et ont été incapables de les tester.

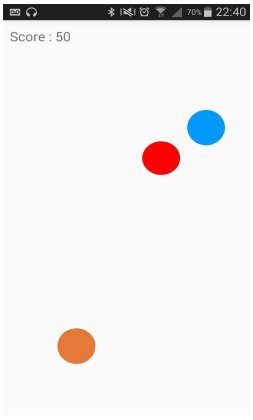
32 Tests d'intégration

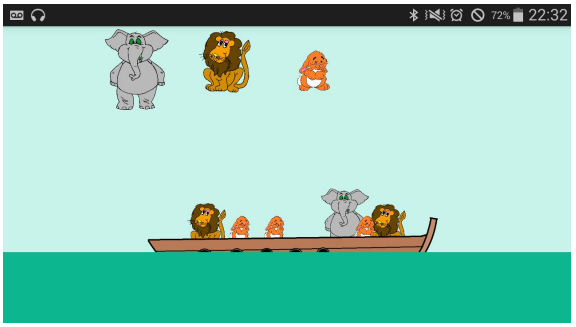
Les tests d'intégration ont pour objectif de détecter d'éventuelles erreurs après assemblage des différents modules de notre application.

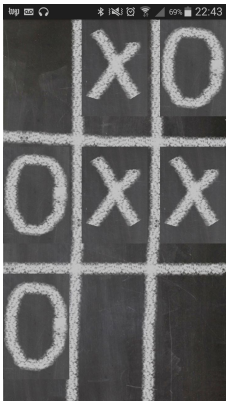
Dans notre cas, il nous a fallu vérifier que nos différentes classes fonctionnent bien dans l'ensemble et communiquent bien entre elles . Comme dit dans le premier livrable, il

s’agit d’effectuer un test d'intégration de type top-down (Test de jeux en premier puis test du main).

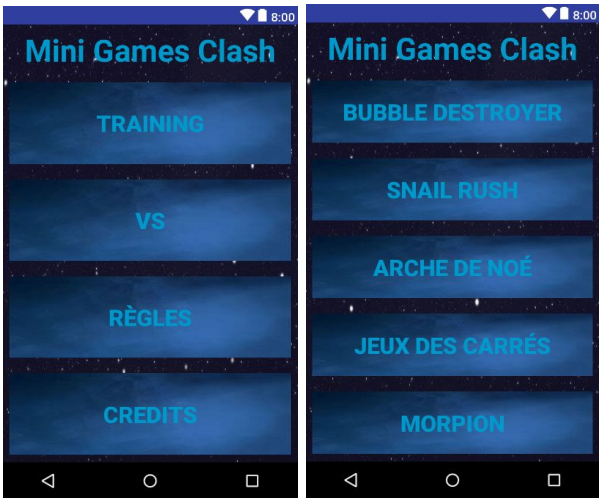
Ci dessous le test de jeux :







Ci-dessous le test du menu principal :



33 Tests de validation

Les tests de validation ont pour objectif de comparer résultats attendus et résultats effectifs pour valider ou non les spécifications. La comparaison est faite dans la section interprétations des résultats (Ci-dessous).

4 Interprétation des résultats

Les résultats obtenus après programmation des différents modules répondent à la plupart de nos attentes. Chaque mini-jeu fonctionne en mode training, ainsi qu'intégré à la suite des autres mini-jeux. (cf tests unitaires et d'intégration). Le modèle de conception préliminaire a été bien respecté. De plus, un leaderboard a été implémenté au jeu Bubble Destroyer.

Nous avons toutefois dû revoir certains projets initiaux secondaires par rapport au tronc du projet, qui est une série de mini-jeux à deux joueurs fonctionnelle. Nous avons donc mis de côté notamment l'implémentation de bruitages et celle d'une liste d'amis, non indispensables au fonctionnement global du jeu. En outre, l'onglet "Réglages" a été remplacé par deux onglets "Crédits" et "Règles".

L'ordre des jeux, qui initialement était voulu aléatoire afin de développer l'aspect spontané et imprévisible, a été repensé. En effet, on a jugé qu'il était plus pertinent de faire découvrir aux joueurs chaque jeu de manière à éviter que certains jeux ne se répètent et que d'autres ne soit pas joués au cours d'une même partie. Les conditions de victoire ont ainsi été changées : le gagnant n'est désigné seulement que lorsque les deux joueurs se sont affrontés sur l'ensemble des mini-jeux.

Le multijoueur a bien été implémenté au sein du jeu. Ainsi, en utilisant la base de données en temps réel de Firebase, deux joueurs communiquent entre eux lors d'une partie de MiniGamesClash. Cependant, tous les jeux n'ont pas encore été intégrés au VS : les derniers jeux n'ont pas pu être adaptés au multijoueur mais sont disponibles au mode Training. Ceci s'explique tout d'abord par la complexité de l'utilisation d'un nouvel outil tel que Firebase dont la maîtrise a demandé un temps conséquent. De plus, chaque jeu étant différent, une implémentation particulière à chaque jeu a été nécessaire augmentant les imprévus et erreurs. Ensuite, l'authentification a été repensée au cours du projet. Malgré une première tentative qui n'a pas porté suffisamment ses fruits, il s'est avéré qu'une simple identification par nom serait suffisante pour nos cas d'utilisations, c'est-à-dire faire communiquer deux joueurs. Ce qui au final se rapproche de la mise en relation aléatoire de deux joueurs pendant un intervalle de temps qui coïncide qu'on avait prévus initialement.

5 Manuel utilisateur

Cette notice a pour objectif de permettre une utilisation rapide, correcte et aisée de notre logiciel.

MiniGamesClash n'est pas encore disponible en version publique.

Ainsi, le jeu requiert tout d'abord un téléphone portable Android de version minimum 4.0.3, ainsi que Android Studio afin d'effectuer son installation. Sur ce logiciel, il suffit de compiler et lancer notre programme au moins une seule fois afin qu'il soit disponible par la suite sur le portable de l'utilisateur. De plus, le code du jeu est disponible en intégralité sur le dépôt Github suivant : https://github.com/KrystianWieleba/MiniGamesClash.git

Par conséquent, c'est aussi Android Studio qui est utilisé par les administrateurs et programmeurs du jeu. Cependant, pour assurer le bon fonctionnement du jeu, il faut également un accès à la base de donnée Firebase lié au projet.

Mode d'emploi

MiniGamesClash est un jeu multijoueurs dans lequel deux joueurs s'affrontent sur un ensemble de quatre mini-jeux. Ces mini-jeux et d'autres sont également disponibles dans un mode solo.

A l'ouverture de l'application, le joueur fait face au menu principal du jeu présenté ci-dessous.

Menu Principal

Le joueur a alors accès aux fonctionnalités suivantes : les modes de jeu ‘’Training’’ ou ‘’VS’, les règles des mini-jeux et les crédits du jeu.

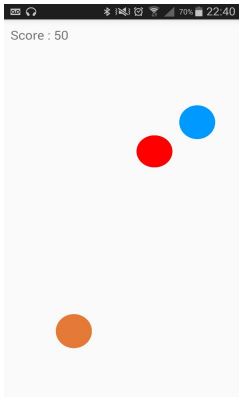
1) Training

Le mode de jeu ‘’Training’’ est le mode solo du jeu.
Là le joueur peut accéder à l’ensemble des mini-jeux sans connexion internet, il peut donc bénéficier du divertissement offert par le jeu à tout moment. Il peut de plus, sous réserve d’une connexion internet, chasser des records dans les différents mini-jeux via un historique centralisant les meilleurs scores tout joueur confondu.

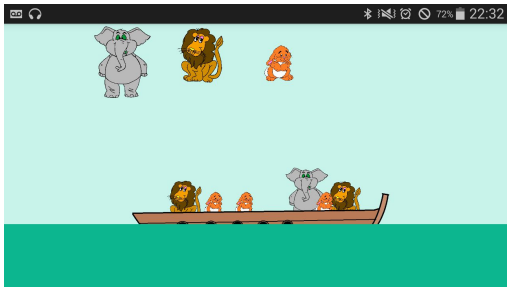
Après avoir sélectionné le bouton ‘’Training’’ le joueur accède à la liste des mini-jeux disponibles.

Après avoir sélectionné le mini-jeu souhaité, la partie se lance.

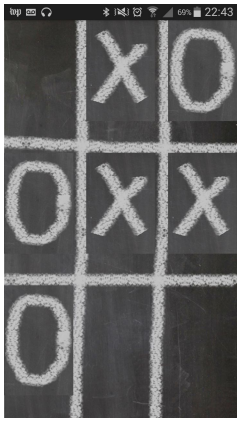
Voici le Bubble Destroyer :



Voici l'Arche de Noé :



Voici le Morpion :



Une fois le jeu fini, pour :

- Bubble Destroyer, une fois le jeu terminé un bouton permet de rejouer et un autre permet de revenir au Menu Principal. Un autre permet de comparer son score aux meilleurs scores.
- Snail Rush et Arche de Noé, une fois le jeu terminé un bouton permet de revenir au Menu Principal.
- le jeu des carrés et le morpion, le joueur gagnant est annoncé puis il y a un retour vers le Menu Principal.
- le snake, le joueur peut atteindre un score maximal de 50 points. Il pourra retourner au menu principal quand il voudra.

Les règles des jeux sont disponibles dans la section Règles.

2) VS

Le mode de jeu ‘’VS’’ est le mode de jeu dans lequel deux joueurs peuvent s’affronter dans quatre manches et à l’issue desquelles un vainqueur ou une égalité sont déclarés.
Après sélection du mode “VS”, le joueur s’identifie et patiente alors que son adversaire finisse son identification. Cette condition atteinte lance le premier jeu suivi des autres jeux séparés chacun par une interface récapitulatif pour les joueurs. Lorsque tous les jeux ont été joué, un vainqueur est désigné suivi d’un retour vers le menu.

La contrainte principale est que seulement deux joueurs peuvent jouer simultanément. Puisque la base de données Firebase utilisée est réinitialisée à chaque lancement du jeu, cela implique qu’aucun autre joueur ne doit lancer le jeu au cours d’une partie déjà en cours. De plus, il est déconseillé aux joueurs de quitter le jeu lors d’une partie en cours puisque cela bloquerait le joueur adverse. Cependant, il serait théoriquement possible de minimiser le jeu lors de la fenêtre “Interface” et y revenir dans un délais bref au désagrément du joueur adverse qui devra patienter le retour de son adversaire.

3) Règles

Dans cette section le principe des différents mini-jeux est exposé.
Un bouton Retour permet de revenir au Menu Principal.

Aides le serpent affamé à manger un maximum de pommes. Mais fais attention, tu ne devras point toucher les bords. Et si tu cherches à aller trop vite, tu finiras peut être par te manger toi-même !

4) Crédits

Dans cette section vous aurez accès aux crédits du jeu.
Un bouton Retour permet de revenir au Menu Principal.

6 Conclusion

Ce projet a été riche en enseignements car d'une part il nous a permis de découvrir les spécificités du développement informatique en nous familiarisant avec la gestion de projet Agile, l'échange de points de vue pour débloquer certaines situations, les outils de collaboration comme Git… D'autre part, il nous a permis d'accroître grandement notre culture informatique. Nous avons découvert en effet le fonctionnement et le développement d'applications Android et toutes ses spécificités comme l'interaction entre interface graphique et algorithmie, la notion d'activité et les communications entres elles. Nous avons également pu en apprendre davantage sur le développement de jeux. Enfin l'aspect multijoueur de notre application nous a permis de découvrir Firebase et des méthodes pour communiquer entre différents appareils.

Les améliorations restantes consistent tout d'abord à intégrer l'aspect "VS" au dernier jeu restant et à l'optimisation des différents jeux que ce soit au niveau algorithmique ou design, afin de permettre au joueur une meilleur expérience. L'amélioration principale reste toutefois l'expansion du multijoueur à plusieurs parties simultanées sans causer d'erreurs aux parties déjà en cours. Pour ce faire on pourrait penser à isoler dans la base de données chaque partie sous une "key" unique, ceci permettrait d'éviter les contraintes actuelles présentes. On peut aussi de manière plus efficace encore utiliser les fonctions d'authentifications de Firebase, qui permet notamment de se connecter via un compte Google, Facebook ou Twitter. Nous avions envisagé initialement cette dernière solution mais n'avons pas eu le temps pour l'implémenter avantageusement.

7 Bibliographie

Beginning Android Games, Mario Zechner.

Algorithmique et langage de programmation, Pierre Sutra et Gaël Thomas, Cours CSC 3101 à Télécom SudParis.

Créez des applications pour Android, Frédéric Espiau, https://openclassrooms.com/courses/creez-des-applications-pour-android.

Génie logiciel et gestion de projets informatiques, J. Paul Gibson, Cours PRO 3600 à Télécom SudParis.

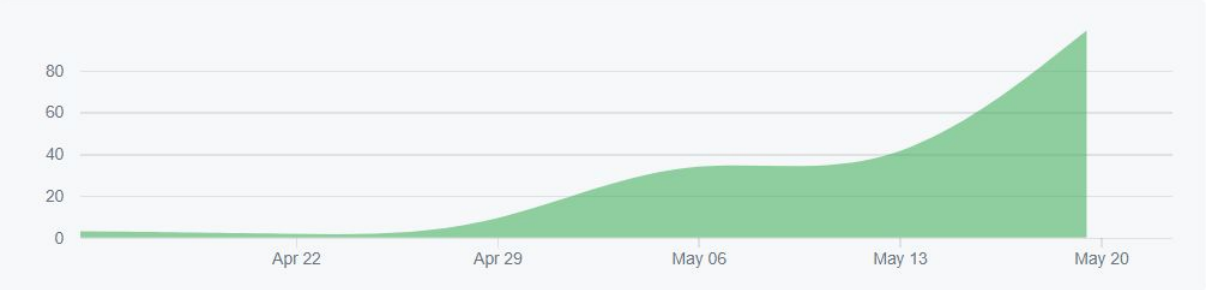
Get Started With Firebase, https://firebase.google.com/docs/android/setup.

Firebase API Reference, https://firebase.google.com/docs/reference/.

Android Studio : Documentation for app developpers, https://developer.android.com/docs/.

8 Annexes

Annexe 1 : Gestion de projet



L'outil Git nous a été primordial lors de ce projet malgré que son utilisation a été tardive dans le vie du projet. En effet, il a fallu se familiariser avec cet outil assez inconnu et compliqué à première vue par les membres du groupe. De plus, au début du projet chacun développait un mini-jeu de son côté afin de se familiariser avec le développement sous android. C'est seulement par la suite, lorsqu'on a dû regrouper les différentes parties puis continuer le développement en parallèle, que Git est vite devenu indispensable. On peut ainsi observer sur la courbe ci-dessus que le nombre de commits a gagné progressivement en densité jusqu'au délai final. Ce phénomène s'explique aussi par les commits moins fournis à la fin du projet, donc possiblement plus nombreux. (souvent des corrections d'erreurs sur quelques lignes)

Annexe 2 : Exemple de planning prévisionnel établi

	Krystian	Christos	Eric	Youssef
9 mai	Git : essayer de redéposer le dossier **sans les build** et cie. **Intégrer l'arche de Noé** au projet. Reprendre la première intégration de **firebase** au projet. *Objectifs : écrire son score dans la base de données et lire le score de l'adversaire*	**Transférer les documents du drive vers git** : déposer le pdf du premier livrable sur Github **Terminer le design du menu** en accord avec celui des mini-jeux et **l'intégrer** au projet.	Git : résoudre avec Krystian le **problème de partage** du projet sur Github.	Se familiariser à Git : cloner le dépôt, déposer le pdf du **premier** livrable sur Github et essayer de lancer le projet cloné.
11 mai	Réussir à **afficher les meilleurs scores** en mode Training. Ajout du nom des joueurs pour le leaderboard	Reprendre la version du morpion pour améliorer l'aspect visuel. (Morpion à deux joueurs sur un même écran)	Intégrer **firebase à l'arche** de Noé (écrire dans la bd l'animal ajouté, attendre le changement, lire et appliquer les modifications et permettre au joueur de jouer)	Reprise du snake pour améliorer le rendu final.
14 mai	**Créer l'activité d'identification** qui introduira le mode vs.	Terminer la version du morpion à développer ensuite en modes Training et Vs. Réfléchir à la conception du jeu des carrés	Intégrer **firebase au snail rush** (échanges en temps réel de la position)	**Intégrer le snake** au projet en mode training (lier avec intent + bouton de retour).