Prep’ISIMA 2 :

Rapport du projet – Application Android

Sommaire

[I. Présentation du projet 3](#_Toc512117464)

[II. Cahier des charges 3](#_Toc512117465)

[A. Liste des fonctionnalités à implémenter 3](#_Toc512117466)

[B. Time line de la difficulté 4](#_Toc512117467)

[III. Etapes de développement et explications techniques 6](#_Toc512117468)

[Projet de base – 15 février 2018 6](#_Toc512117469)

[En quête d’amélioration des collisions – 27 février 2018 7](#_Toc512117470)

[Les collisions encore et toujours – 27 février 2018 7](#_Toc512117471)

[Les collisions : dernière étape – 1er mars 2018 8](#_Toc512117472)

[L’apparition du menu – 9 mars 2018 8](#_Toc512117473)

[Et la physique ? – 9 mars 2018 8](#_Toc512117474)

[Toujours plus de réalisme physique – 9 mars 2018 9](#_Toc512117475)

[IV. Problèmes et difficultés rencontrées 9](#_Toc512117476)

[V. Apports personnels du projet 9](#_Toc512117477)

[VI. Idées d’amélioration 10](#_Toc512117478)

# Présentation du projet

Notre projet consiste en le développement d’un jeu destiné aux smartphones et plus précisément à ceux qui utilisent le système d’exploitation Android de Google.

Notre jeu peut être décrit succinctement car il repose sur des mécaniques de *Gameplay* intuitives. Effectivement, le joueur est matérialisé dans le jeu par un personnage ou une balle qui le représente dans l’univers virtuel et dont le but est de survivre le plus longtemps dans un espace confiné par des bordures. Le contrôle de l’avatar est permis par l’utilisation de l’accéléromètre du téléphone. Evidemment, des ennemis surgissent des quatre côtés de l’écran pour compliquer la tâche et accentuer la difficulté. Par ailleurs, cette dernière s’accroît de manière constante en fonction de la progression du joueur notamment par l’augmentation du nombre d’ennemis ou de leur vitesse.

L’accès au jeu est permis par un menu qui offre également la possibilité de consulter l’historique des meilleurs scores du joueur, gérés grâce à une base de données.

Le développement du jeu aura été effectué grâce à l’IDE *Android Studio* en API Android KitKat 4.4 car cette dernière proposait l’ensemble des fonctionnalités nécessaires au développement tout en permettant de cibler un maximum de terminaux Android (le nombre se chiffrant approximativement à 90% des téléphones du marché utilisant ce système).

# Cahier des charges

## Liste des fonctionnalités à implémenter

*La liste suivante ordonne les diverses fonctionnalités et caractéristiques que devra implémenter et respecter notre projet. A noter que certaines ne seront malheureusement qu’annexes que l’on pourra potentiellement ajouter au projet. La liste les trie des plus faciles et triviales aux plus complexes à implémenter. On veillera d’ailleurs à respecter l’ordre de développement défini dans la mesure du possible.*

* Réussir à bouger la balle grâce à l’accéléromètre présent dans le smartphone.
* Ajouter des ennemis comme des fusées.
* Programmer un système de collision le plus réaliste et optimal possible.
* Gérer les scores du joueur en local via la présence d’une base de données.
* Ajouter un menu principal basique pour permettre à l’utilisateur la consultation des scores.
* Améliorer la physique du jeu en rendant les déplacements de la balle plus réalistes (notamment en ajoutant une inertie).
* Ajouter des bordures sur les côtés de l’écran pour améliorer l’esthétique et signaler le danger au joueur.
* Augmentation de la difficulté suivant le temps (accroissement du nombre de fusées qui apparaissent et de leur vitesse, ajout d’autres ennemis comme des lasers). Se référer à la « time line de la difficulté » pour plus de détails relativement à l’augmentation de la difficulté suivant la progression du joueur.
* Inclure une musique et des bruitages notamment lors des collisions pour améliorer l’ambiance sonore.
* Améliorer le design du jeu (les sprites, le background, l’aspect graphique en général etc.)
* Rajouter un système de vies.
* Remplacer le « Timer » par un score augmentant au fil du temps.
* Connexion à une base de données distante pour le classement des meilleurs scores.
* Rajouter des bonus comme des points supplémentaires, un bouclier temporaire, un ralentisseur de déplacement des fusées, des vies additionnelles qui traversent l’interface graphique et que le joueur peut attraper, …
* Rajouter un malus qui inverse les commandes pendant une durée déterminée.
* Rajouter des crédits que l’on obtient à chaque fin de partie, puis, ajouter une boutique qui permet d’acheter des bonus temporaires, des skins pour la bille, un nouveau thème (tunnels ou murs), etc.

## Time line de la difficulté

*Le tableau en page suivante donne une idée de la time line de la difficulté c’est-à-dire l’augmentation de cette dernière en fonction de l’avancement du joueur dans la partie. A noter qu’il peut évidemment être sujet à des modifications pour notamment des raisons d’équilibrage.*

On définira plusieurs types d’ennemis :

* Ennemis de classe A (fusées par exemple) se déplaçant de manière horizontale ou verticale d’un bord vers son opposé à vitesse constante.
* Ennemis légèrement plus dangereux de classe B (couleur distincte pour marquer cette différence) se déplaçant à vitesse croissante d’un bord vers son opposé.
* Ennemis de classe C se déplaçant en courbe à vitesse constante d’un bord vers un autre.

Les types suivants sont des idées d’améliorations optionnelles :

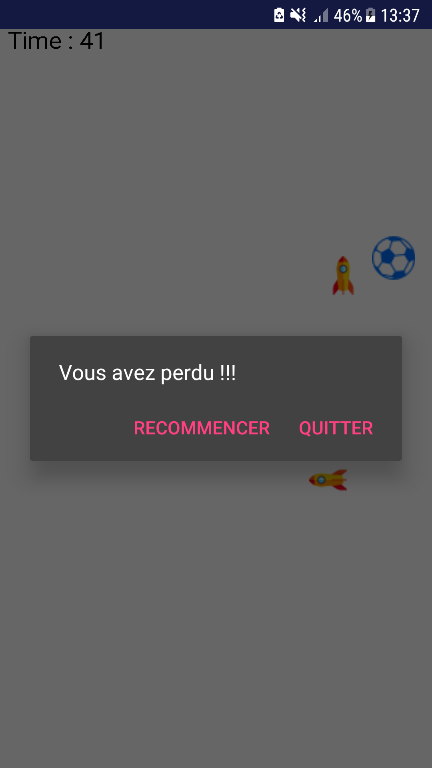
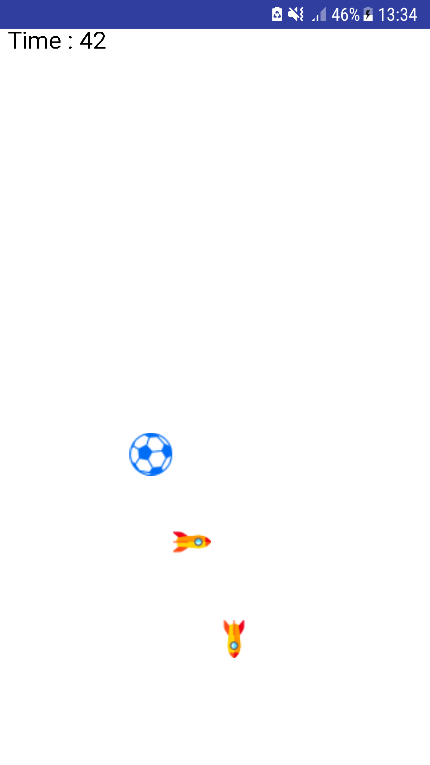
* Ennemis à « tête chercheuse » de classe D partant d’un bord et se dirigeant vers le joueur. A noter que ces derniers disposent évidemment d’une inertie et peuvent-être esquivées par le joueur.
* Ennemis non ponctuels de classe E (comme des lasers) qui réduisent la zone de jeu.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SCORE DU JOUEUR | LOCALISATION DES APPARITIONS | FREQUENCE D’APPARITION | | |
| fA | fB | fC |
| 0 | Les ennemis A n’apparaissent que d’un bord fixé arbitrairement ou déterminé de manière aléatoire. | 0.5 | - | - |
| 10 | Les ennemis A apparaissent en plus depuis le bord opposé. | 1 | - | - |
| 20 | Les ennemis A apparaissent depuis tous les bords. | 1 | - | - |
| 30 | - | 1.5 | - | - |
| 40 | - | 2 | - |  |
| 50 | Des ennemis B apparaissent depuis tous les bords. | 2 | 0.5 |  |
| 60 | - | 2 | 1 |  |
| 70 | - | 2 | 1.5 |  |
| 80 | - | 3 | 2 |  |
| 90 | Des ennemis C apparaissent depuis un bord arbitrairement ou déterminé de manière aléatoire. | 3 | 2 | 1 |
| 100 | Des ennemis C apparaissent depuis tous les bords. | 4 | 3 | 2 |
| 120 + (20) | Les fréquences augmentent toutes de manière proportionnelles. | += 1 | += 0.75 | += 0.5 |
| - | - | - | - | - |

# Etapes de développement et explications techniques

Cette partie du compte-rendu retrace l’ensemble des étapes de développement, c’est-à-dire comment le projet a effectivement été développé tout le long de sa production. Evidemment, cette ramification résume l’avancement dans l’ordre chronologique de sa réalisation.

### Projet de base – 15 février 2018

Les deux premières exigences du cahier des charges sont remplies. Une balle est présente au centre de l’écran et peut se mouvoir grâce à l’accéléromètre du téléphone. En outre, des fusées apparaissent sur les côtés comme prévu. Le jeu est déjà jouable mais demeure néanmoins à l’état de prototype car la difficulté est effectivement constante.

Nous avons, par ailleurs, tenté de coder un système de collisions mais celui-ci présente des bugs et est dans les faits totalement inefficace. Cela est dû à un mauvais calcul au niveau des coordonnées

Par ailleurs le jeu offre la possibilité, non souhaitée, d’être joué en affichage de type « paysage ». Ce léger problème se doit d’être corrigé rapidement.

Figure 1 : Première image du jeu en plein fonctionnement

Figure 2 : Illustration du problème de collision. La fusée à gauche de la balle ne devrait pas entraîner un Game Over

Enfin, une fenêtre telle celle présente en *figure 1* apparaît de manière aléatoire en début de partie et indique au joueur qu’il vient de perdre la partie avant même que cette dernière ne démarre. Ce bug du « recommencer » demeurera d’ailleurs été récurrent car il reviendra à plusieurs reprises même après avoir été corrigé jusqu’à disparaître définitivement.

### En quête d’amélioration des collisions – 27 février 2018

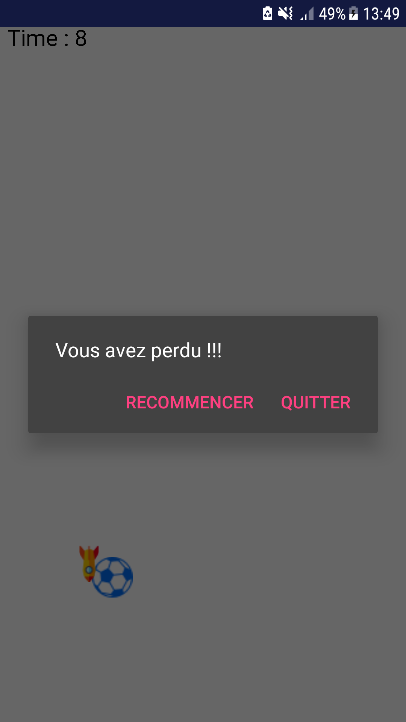
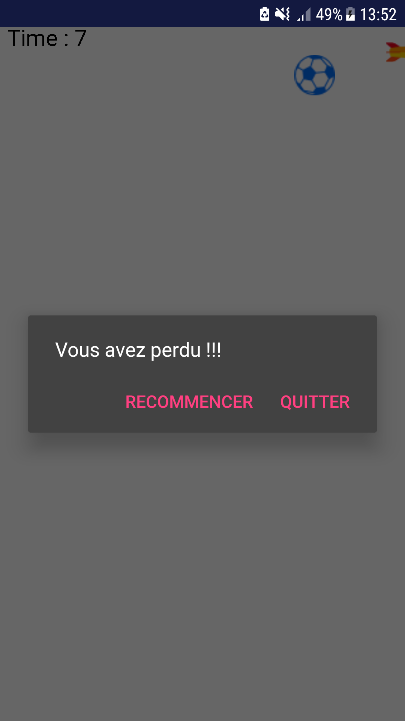
Les collisions ont été nettement améliorées même si certains problèmes subsistent notamment à cause de la rotation des fusées qui advient lorsque l’on souhaite que ces dernières soient orientées de manière horizontale. La position « de base » étant « orientée vers le haut » et, le sprite étant rectangulaire, les coordonnées ne sont pas modifiées en conséquence lors d’une rotation de 90° ce qui provoque évidemment un problème au niveau des collisions.

Figure 3 : Collision problématique à cause d'une rotation à 90° vers la droite

Figure 4 : Collision normale. La rotation à 180° n'affecte pas les coordonnées

Enfin, malgré sa correction antérieure aux modifications des collisions, le « bug du recommencer » est revenu par la suite.

### G:\Sauvegarde du 21-04-18 projet Java + syst exploitation\screenshots_projetisima\da2ab3a\Screenshot_20180420-135942.pngLes collisions encore et toujours – 27 février 2018

Une nouvelle fois, les collisions ont été améliorées mais celles-ci présentent encore un dernier problème. En effet, les coins des fusées ne génèrent pas de collision ce qui ne provoque pas de Game Over comme escompté. « Le bug du recommencer » est toujours présent.

Figure : Collision anormale dans le coin inférieur droit de la fusée

### G:\Sauvegarde du 21-04-18 projet Java + syst exploitation\screenshots_projetisima\87aa2ff\Screenshot_20180420-140833-2.pngLes collisions : dernière étape – 1er mars 2018

Tous les problèmes de collision sont résolus grâce à l’utilisation de bounding boxes qui permettent de grandement simplifier le calcul et d’optimiser la tâche pour le processeur.

### L’apparition du menu – 9 mars 2018

Deux nouvelles activités Android implémentées offrent l’affichage d’un menu principal basique et d’un tableau des scores. Ces deux interfaces graphiques utilisateur sont d’ailleurs disposées grâce à deux fichiers XML. Concernant les scores, l’utilisateur est invité en fin de partie à sauvegarder son exploit ; libre ensuite à lui d’en faire le choix. En ce qui concerne le menu principal, ce dernier est constitué linéairement de haut en bas, d’un bouton permettant d’accéder à l’activité des scores, d’un champ de saisie pour le pseudonyme de l’utilisateur et enfin d’un bouton jouer pour lancer une partie.

Figure : La bounding box permet des résultats très satisfaisants concernant les collisions. Elle est ici mise en valeur en vert sur la fusée ci-dessus.

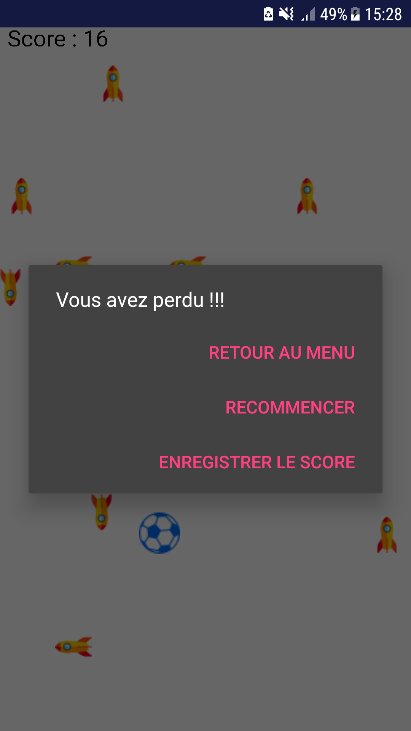
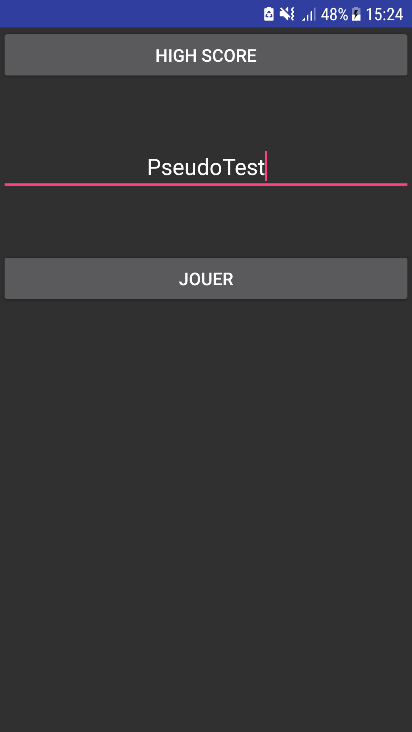
Le « bug du recommencer » a été corrigé et il est désormais impossible d’afficher l’application en format paysage.

Figure : Tableau des scores

Figure : Changement de la fenêtre de fin de partie

Figure : Le menu principal de l'application

### Et la physique ? – 9 mars 2018

Les déplacements de la balle sont plus réalistes d’un point de vue physique grâce à l’ajout d’une composante servant à représenter l’inertie.

Après plusieurs recherches, nous avons convenus qu’il était catégoriquement mieux de créer nous même un système physique se rapprochant au maximum de la réalité tout en étant basé sur des faits empiriques plutôt que d’implémenter un réel moteur physique exact au sens mathématique. La complexité d’une telle solution aurait été effectivement beaucoup trop chronophage pour notre projet.

### Toujours plus de réalisme physique – 9 mars 2018

L’ajout d’une composante visant à modéliser la dispersion de l’inertie, c’est-à-dire la manière que cette dernière a de s’atténuer avec le temps par la faute des frottements, nous a permis d’atteindre des résultats très satisfaisants. Les coefficients ainsi créés peuvent amenés à être modifier pour parvenir à mimer la réalité de la meilleure manière possible.

# Problèmes et difficultés rencontrées

Au cours du développement de l’application, la rencontre de problèmes et entraves a été inévitable. De manière attendue, plusieurs bugs sont apparus au cours de l’implémentation des différentes fonctionnalités prévues dans le cahier des charges qui souffraient parfois d’un problème de compatibilité avec les diverses fonctionnalités développées précédemment.

Par exemple, suivant l’ajout d’un système physique pour le déplacement de la balle, afin de renforcer le réalisme du jeu, un bug apparaissait au lancement de la première partie et entraînait un crash. Il nous aura fallu plusieurs jours avant de trouver l’origine du problème qui consistait effectivement en la mauvaise initialisation d’une variable.

Par la suite, la recherche d’un système de collision convenable n’aura pas été une mince affaire car elle aura nécessité l’apparition de plusieurs fonctions mathématiques ainsi que l’inclusion de bounding boxes pour optimiser la collision entre formes géométriques complexes (on se rappellera notamment de la collision entre la balle du joueur et les différentes fusées).

# Apports personnels du projet

Nous concernant, ce projet nous aura offert la possibilité de renforcer nos capacités en langage Java ainsi que de nous familiariser avec les API Android que nous avons eu l’occasion d’utiliser. Mener un projet en rapport avec la technologie embraquée aura été très intéressant et enrichissant car nous sommes tous deux très intéressés par cette discipline. Par ailleurs, nous aurons pu nous rendre compte de l’utilité et de la puissance d’Android Studio qui reste une référence pour le développement mobile en offrant énormément de raccourcis et options pour faciliter les étapes d’implémentation.

M. Wagner aura été un très bon tuteur car il nous aura aiguillés dans la création de l’application en insistant sur les points importants et tout en nous laissant la liberté d’implémenter toutes les fonctionnalités selon notre envie et suivant un ordre modulable. Avant le début de la programmation au sens propre du terme, Il nous a suggéré la technologie Django côté serveur pour faire la gestion de la base de données. L’on aura malheureusement pas pu mettre en œuvre cette technologie par faute d’un manque de temps.

# Idées d’amélioration

Nous avons par exemple pensé à rajouter une page d’inscription et de connexion. L’utilisation devra alors créer un pseudo et un mot de passe pour pouvoir se connecter au jeu. Ces informations seraient stockées dans une base de donnée externe. La technologie utilisée serait soit un script python avec Django au niveau du serveur, soit un script PHP. Ce script renverrait les données en JSON et l’application utiliserait la librairie RETROFIT pour récupérer les données et ensuite les traitées.

Nous avons aussi pensé à rajouter des skins de balle et des fonds d’écran que le joueur pourrait choisir. Ces skins changeraient la façon dont les fusées se comportent ainsi que le fond par exemple.

Nous aimerions aussi rajouter des bonus et des malus qui s’afficheraient sur l’écran à certains moments dans la partie et permettraient d’avoir un jeu encore plus intéressant.

Si nous avions eu plus de temps, nous aurions développé un système d’achat de skins et de fonds d’écran et aurions aussi mis l’application sur le Play Store.