

|  |  |
| --- | --- |
|  | |
| 07.09.2017 | **Projet développement mobile - Roleplay**  **INF3dlm-a** |
|  |  |
|  | * Gabriel Griesser * Axel Bento da Silva |
|  |  |
|  | A destination de :   * Rizzotti Aïcha |

Abstract

Ce rapport décrit la réalisation du projet du cours Développement Mobile. Ce dernier consiste à créer un programme de notre choix, en Java, tournant sur un smartphone Android.

Une fois le groupe créé, nous avons réfléchi sur le type d’application que l’on pourrait faire. Étant les deux passionnés de jeu de rôle, nous nous sommes donc penchés sur un jeu de ce style.

Ce jeu est une application narrative dans laquelle l’utilisateur choisi sa propre destination en répondant aux questions.

Ce rapport décrira entre autres les recherches et les explications nécessaires à la bonne compréhension, la conception du programme, l’implémentation, le développement du jeu ainsi que l’utilisation de ce dernier.

Table des matières

[1. Introduction 4](#_Toc504568257)

[2. Analyse //A COMPLETER 4](#_Toc504568258)

[3. Conception 4](#_Toc504568259)

[3.1. Cahier des charges 5](#_Toc504568260)

[3.2. Diagramme de Gantt 5](#_Toc504568261)

[3.3. Scénario 5](#_Toc504568262)

[3.4. Modélisation 7](#_Toc504568263)

[3.4.1. Diagramme UML 7](#_Toc504568264)

[3.4.2. Convention de codage 7](#_Toc504568265)

[3.4.3. Use Case 8](#_Toc504568266)

[4. Implémentation //A COMPLETER 9](#_Toc504568267)

[4.1. Logiciel et bibliothèque 9](#_Toc504568268)

[4.1.1. Eclipse Java 9](#_Toc504568269)

[4.1.2. Swing 9](#_Toc504568270)

[4.2. Projets et classes 9](#_Toc504568271)

[5. Utilisation du jeu //A COMPLETER 9](#_Toc504568272)

[6. Problèmes rencontrés //A COMPLETER ? 10](#_Toc504568273)

[6.1. Incompréhension du stockage de données 10](#_Toc504568274)

[6.2. Accès bloqué aux données 10](#_Toc504568275)

[6.3. Problème lié aux émulateurs 10](#_Toc504568276)

[7. Améliorations futures 10](#_Toc504568277)

[8. Conclusion 11](#_Toc504568278)

[9. Sources 11](#_Toc504568279)

[10. Annexes 11](#_Toc504568280)

# Introduction

Pendant le 1er semestre de la dernière année, les étudiants de la HE-ARC. Si la création d’un nouveau programme ainsi que le langage sont choisis, le projet, lui, n’est imposé.

Le principe de base de l’application était de raconter une histoire à l’utilisateur, en le mettant à la place du héros, il doit faire des choix qui modifieront son chemin tout au long de l’histoire.

Ces différents choix permettent d’aborder l’histoire du héro sous un autre angle et permet donc d’augmenter la durée de vie de l’application.

Ce type de jeu narratif est un classique des jeux pc, mais n’étant plus à la mode lors de l’explosion des smartphones, il n’en existe que très peu sur le marché du mobile.

L’application est écrite en Java spécialisé développement Android. Parmi des choix de spécificités imposés, nous avons choisi les sauvegardes persistantes ainsi que l’utilisation de l’accéléromètre.

# Analyse //A COMPLETER

Parmi les nombreuses possibilités offertes par Java pour la réalisation d’un projet en 2D, nous avons choisi d’utiliser la librairie *Swing*.

Avant de se lancer dans le projet, nous avons réfléchi à l’implémentation. Comment atteindre l’objectif sans se perdre en chemin. Nous avons donc fait le cahier des charges et les spécifications du projet pour avoir une idée optimale du déroulement de notre jeu. Ces derniers expliquent la façon dont notre projet verra le jour, les objectifs principaux et secondaire, et les détails de ces derniers. Avant d’attaquer le projet, nous nous sommes mis d’accord sur une convention de codage, sur l’utilisation de certaines méthodes et manières d’aborder notre projet.

Ensuite, nous avons dessiné les différents schémas nécessaires pour une meilleure réussite du projet. Ces schémas sont le diagramme UML, le use case et le diagramme de séquence.

# Conception

Cette partie est consacrée à la conception du projet. Premièrement, nous avons rédigé le cahier des charges et avons discuté des spécificités entre nous. Ensuite, après s’être réparti les différentes tâches du projet, nous nous sommes intéressés aux différents points chauds, puis avons réalisé un diagramme de Gantt pour avoir une meilleure visibilité du projet.

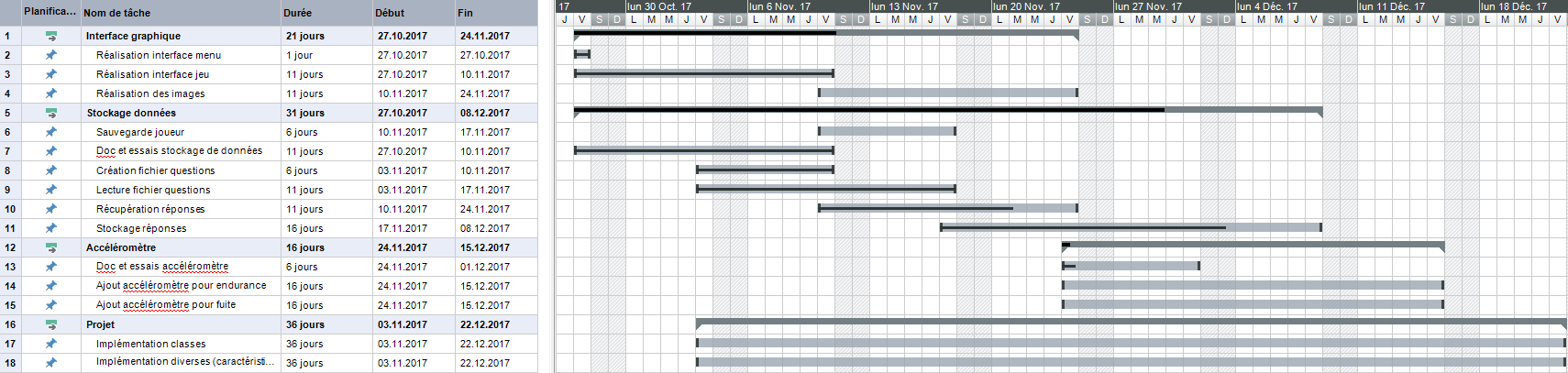
## Cahier des charges

Le cahier des charges défini clairement les objectifs à atteindre. Ces derniers sont détaillés dans la spécification du projet. Ces deux documents nous ont permis d’obtenir une meilleure définition du but, des objectifs et des détails concernant le projet.

Dans le cahier des charges, nous avons défini nos objectifs principaux, secondaires, nos points chauds, le déroulement de la partie ainsi que quelques informations supplémentaires. Il servira à quiconque le lit, d’avoir une idée globale du projet et des objectifs à atteindre. Nous avons pu y intégrer les objectifs principaux accompagnés de quelques objectifs optionnels (améliorations). Les différents points chauds (que nous avons défini ensemble) y sont également notés afin de mieux se rendre compte de la difficulté du projet.

## Diagramme de Gantt

Le diagramme de Gantt est une manière de décrire, d’attribuer et de visualiser les différentes tâches nécessaires au bon déroulement au projet. Ce diagramme est à prendre avec des pincettes car les limites de temps données n’ont pas toujours été respectées.



## Scénario

Chaque partie de l’histoire est stockée dans un fichier séparé. Ce dernier est défini ainsi :



Figure , fichier de scénario

Au début du fichier se trouve le texte narratif, puis les lignes débutant par un « / » donne les nœuds vers les fichiers suivants. Entre les deux « / » se trouve le nom du fichier vers lequel la réponse pointe. La fin de la ligne est le choix qui sera présenté à l’utilisateur.

Il existe également des fichiers où le texte du scénario est remplacé par un mot-clé commençant par « \* », ces nœuds définissent des actions du héros comme se battre ou prendre la fuite.

Il existe un dernier type de fichier qui ne possède pas de nœuds vers lesquels se dirigé, puisqu’il s’agit des nœuds dans lesquels le héros meurt ou termine son histoire.

L’ensemble des fichiers de scénario représente le premier chapitre de l’histoire et est représenté par cet arbre :



Une histoire se suit par rapport aux lettres qui compose les fichiers de scénario, en sachant que A1 est le fichier de base de toutes les histoires. Par exemple

*A1, A2, A3, A4, A5, A6, A7, A*8 composent une histoire*. A1, A2, B1, B2, B3, B4* en font une autre.

Les fichiers commençant par D (*death*) indiquent que le personnage meurt s’il atteint un de ces nœuds.

## Modélisation

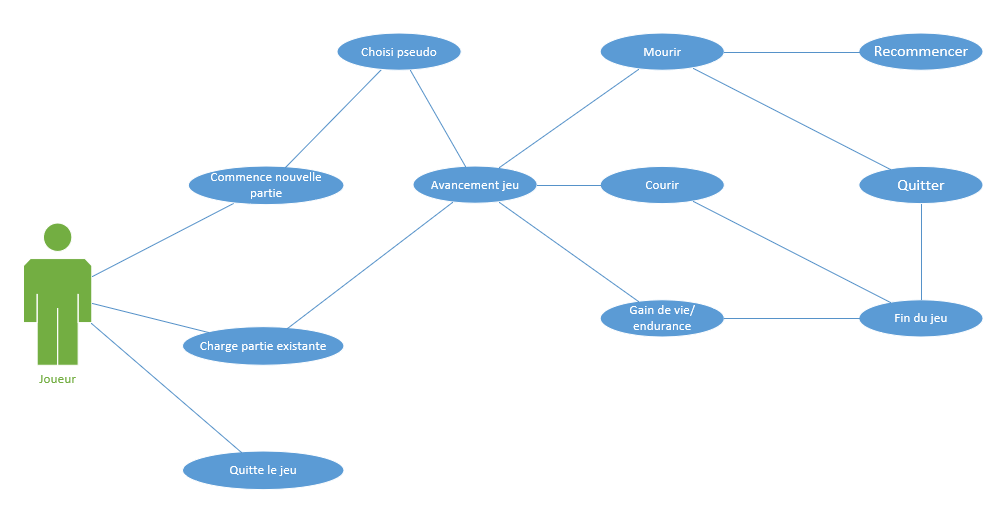
### Diagramme UML

Le diagramme UML, bien que simple, nous permet d’avoir une vision globale du déroulement de l’application. 

### Convention de codage

* Les noms des classes sont en anglais, commencent par une majuscule et peuvent contenir une ou plusieurs majuscule 🡪 *FileHandler*
* Les noms des attributs et objets commencent par une minuscule et peuvent contenir une majuscule. Ils sont écrits en anglais 🡪 *attributeChanged*
* Les noms des méthodes commencent par une minuscule et peuvent contenir une ou plusieurs majuscules. Ils sont écrits en anglais 🡪 *savePlayer()*

### Use Case

Le *use case*, ou diagramme de cas d’utilisation, défini la manière d’utiliser le programme et les différentes actions possibles.

# Implémentation //A COMPLETER

Cette partie du rapport explique l’implémentation du projet. Nous détaillerons donc la façon d’atteindre le but principal de ce projet, les logiciels et bibliothèque principaux utilisés, puis les différentes parties du programme, leurs spécificités et le rôle de chacune.

## Logiciel et bibliothèque

### Eclipse Java

Eclipse Java est un logiciel réputé pour sa fiabilité. Son but est de développer un environnement de production de logiciels libre, utilisant principalement le langage Java. Il produit et fournit des outils pour la réalisation de logiciels, intégrant Framework, librairies, etc. et fournit également un atelier de génie logiciel, recouvrant modélisation, conception, test, etc.

### Swing

Swing est une API faisant partie de la bibliothèque *Java Foundation Classes* (JFC). Elle permet de créer des interfaces graphiques, jongler entre ces dernières, faires différentes animations sur les objets et utilise le principe MVC. Swing supporte le graphisme 2D, permet de faire des animations et affichages graphiques standard. De plus, les composants de Swing sont *légers*. Ils ne requièrent pas d’allocation de ressources. Nous avons utilisé Swing plutôt que *JavaFX* car notre projet, qui est en 2D, ne demandait que peu d’animation et pouvait être réalisé avec cette librairie.

## Projets et classes

Comme décrit dans la partie [3.3.1](#_Diagramme_UML), notre programme est divisé en 3 parties. Ces dernières ont chacune leur rôle.

La partie *goldminercreator* contient le code pour l’éditeur de niveau. C’est ici que l’utilisateur pourra créer, modifier et sauvegarder un niveau. Il pourra ensuite l’utiliser avec la partie *goldminergame*. Ce bloc contient le code nécessaire au lancement et déroulement du jeu. La dernière partie du projet, *goldminercommon* contient tous les éléments et leurs propriétés utilisés dans les 2 premières parties. Les objets (*diamond, bomb, etc.*), les propriétés de ces objets et les parseurs XML pour les niveaux sont dans ce dernier projet. Il nous a semblé cohérent de regrouper ces éléments à part car ils sont tous utilisés par *goldminercreator* **et** par *goldminergame*.

# Utilisation du jeu //A COMPLETER

# Problèmes rencontrés //A COMPLETER ?

Durant le déroulement du projet, nous avons rencontrés quelques problèmes, qui, pour la plupart, ont été résolus.

## Incompréhension du stockage de données

Une erreur que nous avons fait lors de la conception, a été de ne pas spécifié le type de stockage de données, un membre du groupe voulais utiliser *l’internal storage* et l’autre pensais que l’objectif demandé était de stocker les données sur un support externe (*sdCard*), après avoir rencontré le problème, nous nous sommes mis d’accord sur l’utilisation de fichier interne.

## Accès bloqué aux données

Un problème rencontré durant la création du *FileHandler* (gestionnaire de sauvegarde et de lecture des fichiers), est que nous n’avions pas accès au dossier data de l’émulateur à travers « *Android Device Monitor* », ce qui nous posait des problèmes de débogages, notamment pour savoir si les fichiers de sauvegardes étaient créés.

Après quelques recherches nous avons compris qu’il s’agissait d’un problème de droit de lecture/écriture, que nous avons tenté de modifier, ce qui nécessitait plusieurs opérations notamment le routage de l’appareil, cette direction nous a fait perdre beaucoup de temps, pour au final ne pas apporter de résultats concrets.

Nous avons heureusement trouvé une autre solution temporaire, qui était de ne plus utiliser l’appareil de test fournit par l’école, mais d’utiliser un émulateur avec un niveau d’API supérieur ou égal à 23, qui n’ont pas ces problèmes de droits.

Ce qui nous à mener au problème suivant.

## Problème lié aux émulateurs

Nous avons eu de nombreux problèmes liés aux émulateurs, crash au démarrage, freeze alors que l’application n’est même pas lancée, les *error* 0x502 que nous n’avons pas pu corriger. Tous ces problèmes ont passablement ralenti, la partie sauvegarde et chargement de fichier.

# Améliorations futures //A COMPLETER

# Conclusion

…

# Sources

Documentation java : <https://docs.oracle.com/javase/7/docs/api/>

Documentation Swing : <https://www.jmdoudoux.fr/java/dej/chap-swing.htm>

Diverses aides : <https://stackoverflow.com/>

# Annexes

Ci-joint :

* Cahier des charges
* Spécification du projet
* Diagramme UML des classes
* Diagramme de séquence
* Planning
* Cas d’utilisation
* Code source du jeu
* Exécutables sous formes de .jar