Rapport



**Java & LibGDX**

Présenté à :

**RIZZOTTI Aïcha**

**PASIN Marcelo**

Étudiants :

**VAUCHER Joël**

**OMBANG NDO Charles**

**RODRIGUES LOURENÇO Daniel**

Abstract

Ce rapport présente notre projet académique P2 réalisé durant l’HES d’été.

Table des matières

1. Introduction 3

2. Conception 3

2.1. Les Uses Cases 3

2.2. Diagrammes de séquence 3

2.3. Schéma de classes 5

2.4. Architecture de logicielle 6

3. Outils de conceptions et développement 7

4. Tiled map 7

5. LibGdx 9

5.1. Diagramme de classe global de Libgdx 9

5.2. PlayScreen, World & JdcGame 9

5.3. Sprite 10

5.4. InteractiveTileObject 10

5.5. Collision & Mask 10

5.6. Son 10

6. Problèmes rencontrés et solutions 11

7. Améliorations possibles 11

8. Conclusion 12

9. Bibliographie 13

# Introduction

Dans la continuité des projets académiques de l’école, cette session d’été représentant la phase d’intégration nous donne l’opportunité de concrétiser un projet non imposé et avec un langage de programmation choisi par chaque équipe. Nous avons choisi d’utiliser le langage Java. Notre choix ne signifie pas une peur d’aller vers des sentiers battus mais, plutôt de prendre un langage déjà connu pour assurer le délai de production du projet.

Dans les lignes qui suivent, nous allons détailler les différentes phases de dévéloppement, les outils utilisés, l’architecture de developpment et les problèmes rencontrés. Une conclusion nous permettra de tirer des enseignements de ce travail fait en trois semaines.

# Conception

Ce chapitre présente le génie logiciel de notre projet. Il présente les différentes fonctionnalités offertes par le jeu dans un use case, des diagrammes de séquences et de classes décrivant l’enchainement de certaines actions et les objets de notre jeu.

## Les Uses Cases

Le diagramme des cas d’utilisations métier du projet est représenté ci-dessous.

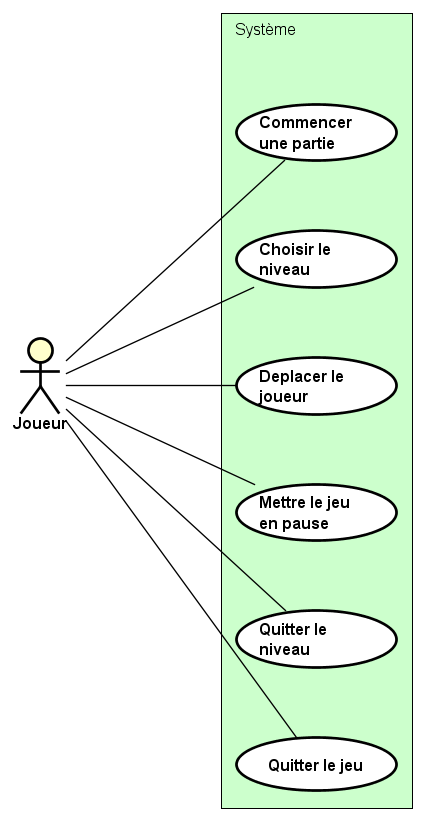


Figure 1 - Diagramme des cas d'utilisation

## Diagrammes de séquence

Le diagramme d’Uses cases ci-dessous montre les différentes fonctionnalités que le logiciel doit offrir à ses utilisateurs.

### Séquence dans le menu

Ce schéma présente le déroulement des interactions lorsque l’utilisateur est dans le menu principal. Le déroulement des interactions dans les sous-menus est le même donc un seul diagramme est présenté.

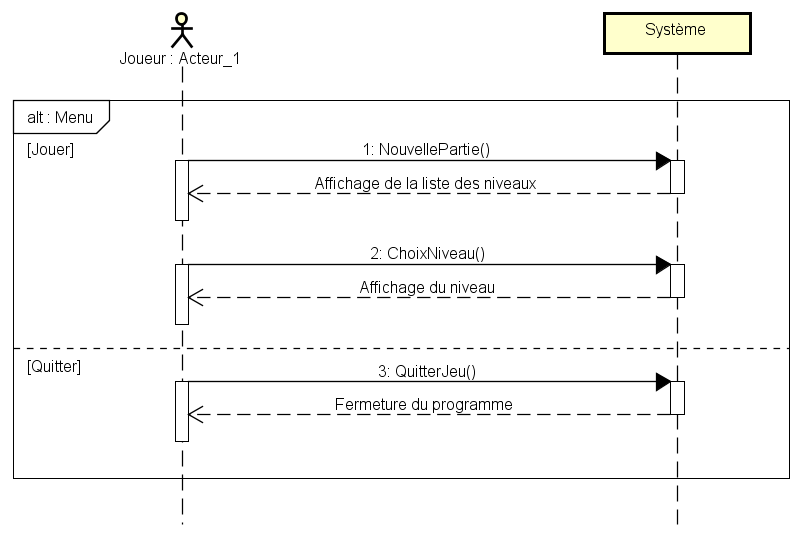


Figure 2 - Diagramme de séquence dans le menu

### Séquence durant le jeu

Voici le diagramme de séquence lors du déroulement d’une partie.

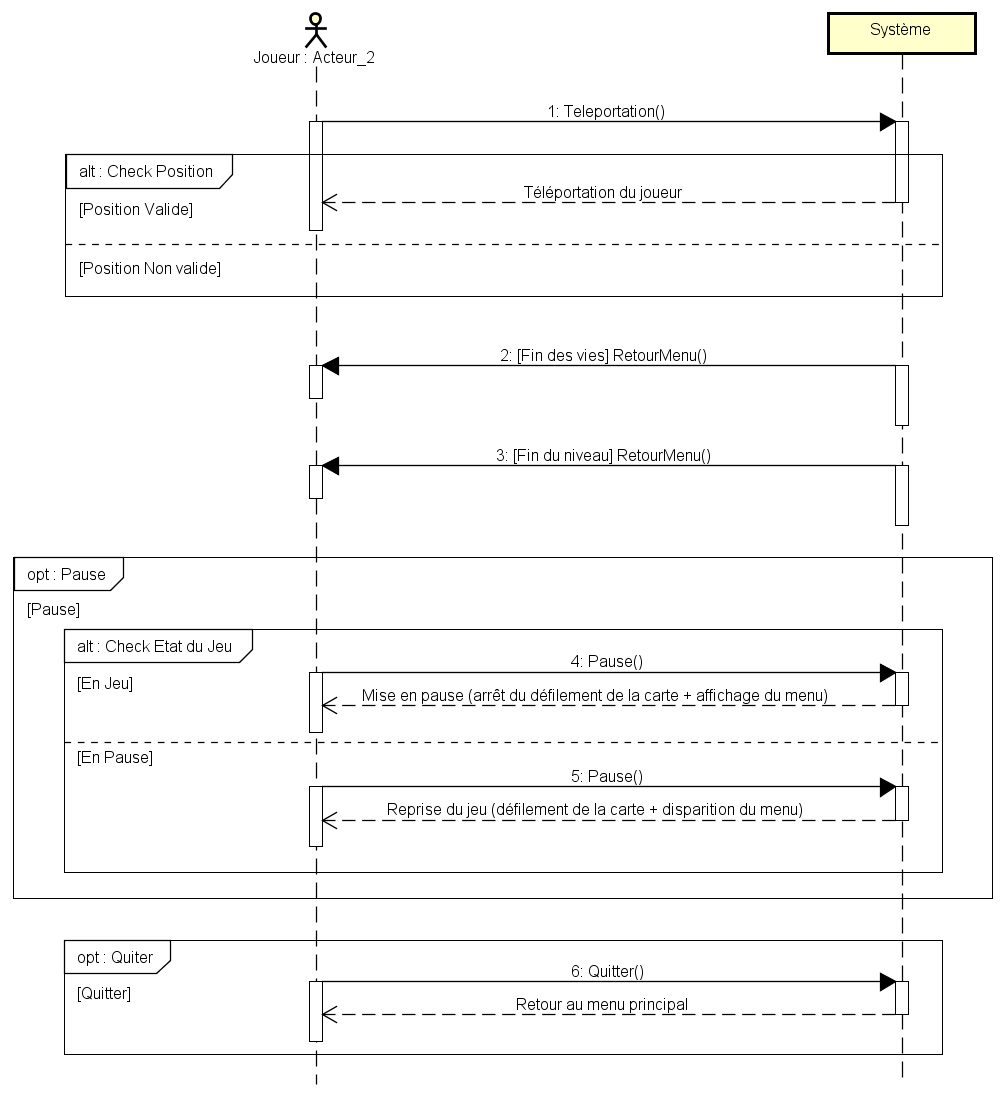


Figure 3 - Diagramme de séquence durant le jeu

## Schéma de classes

Voici le schéma de classes du projet. La version de ce schéma correspond au projet dans son état actuel. Le schéma prévu initialement se trouve en annexe.

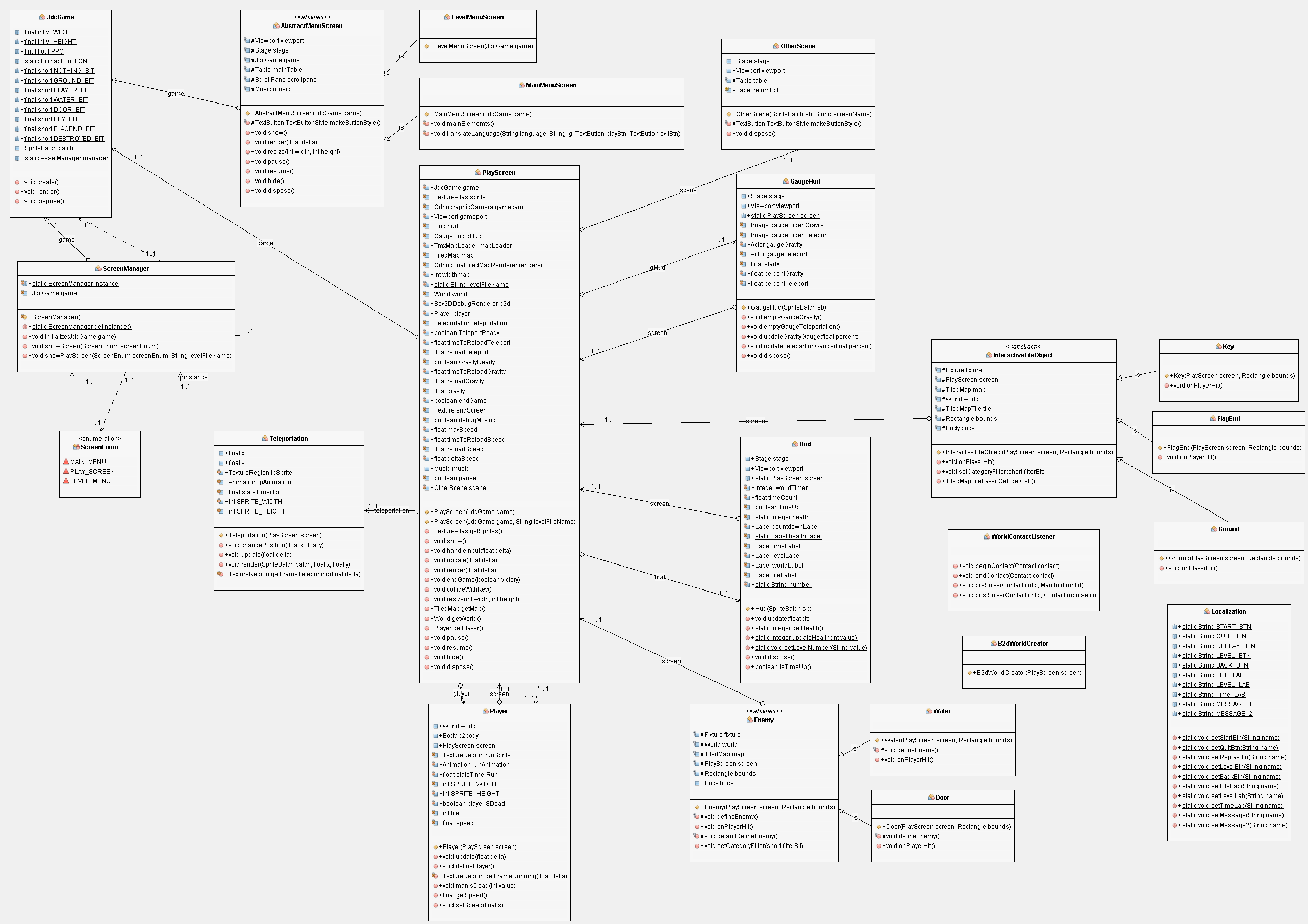


Figure 4 - Schéma de classes

## Architecture de logicielle

# Outils de conceptions et développement

La mise en place de ce projet a nécessité des outils assez simples d’utilisation et surtout, des outils utilisés quotidiennement pour notre formation.

Tableau 1 - Liste des outils utilisés

|  |  |
| --- | --- |
| Nom | Description |
| NetBeans | C’est l’IDE retenue pour la programmation de notre logiciel. |
| LibGdx | FrameWork pour les jeux 2D/3D en Java |
| GitHub | Comme outils de collaboration. |
| Astah | Pour les diagrammes de modélisation utilisés. |

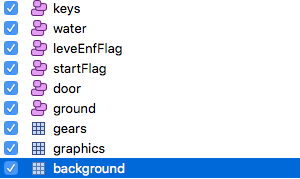
Nous avons également pu compter sur un tutoriel expliquant la base de LibGdx :

<https://www.youtube.com/playlist?list=PLZm85UZQLd2SXQzsF-a0-pPF6IWDDdrXt> (Aureli, 2015)

# Tiled map

TiledMap est un logiciel gratuit permettant de construire les maps. C’est ce dernier que nous avons utilisé pour construire nos différents niveaux.

Chaque niveau est composé de 9 calques parmis lesquels 3 calques contenant les tiles (gears, graphics et background) et 6 calques définissant les objets entrant en collision avec le personnage, ce sont les calques avec l’icône rose sur la capture ci-dessous.



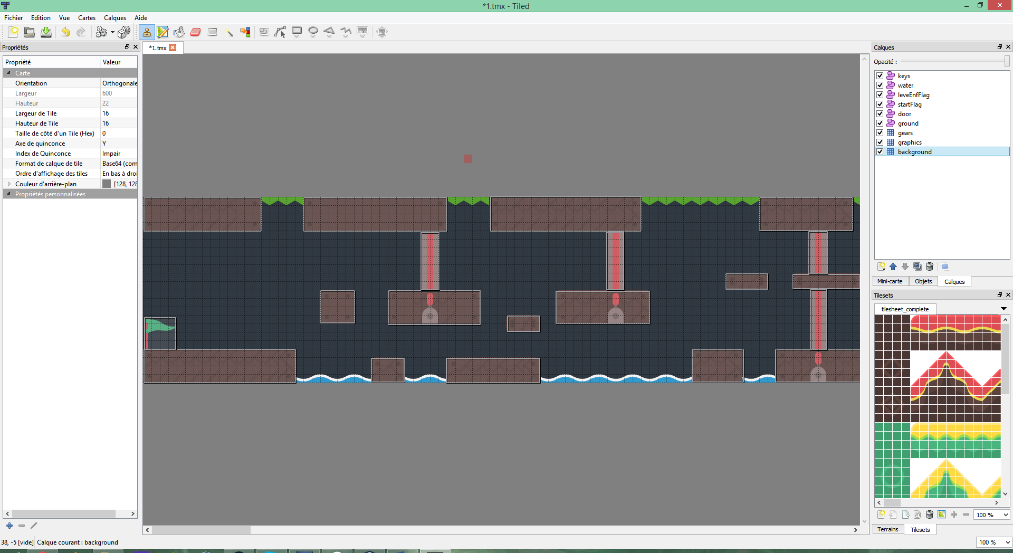


Figure 5 - Interface de Tiled Map

Le fichier en sortie de TileMap est un fichier dont l’extension est tmx. Ce fichier n’est rien d’autre qu’un simple fichier xml décrivant la totalité de la map et de ses différentes propriétés.

La classe **TmxMapLoader** permet de récupérer aisement la map dans notre code java et seulement en quelques lignes.

Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 15.02.10.png

La map chargée est stockée dans un objet de type **TiledMap**

Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 15.03.49.png

Et finalement on charge la map en stockant dans l’objet **map**.

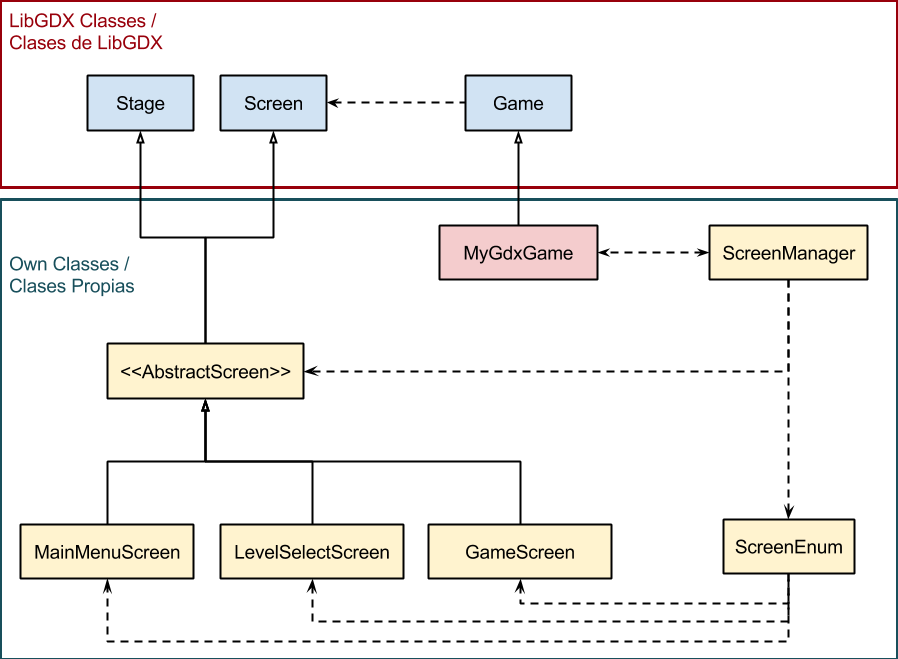
Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 15.04.07.png

Finalement la map est rendu code l’indique le code ci-dessous.

Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 15.08.08.png

# LibGdx

## Diagramme de classe global de Libgdx



La figure ci-dessus présente globalement les classes composants le framework libgdx. L’architecture logicielle de notre jeu est basée sur celle du schéma ci-dessus.

## PlayScreen, World & JdcGame

Nous avons suivi la structure proposée par le tutoriel qui conseillait d’utiliser les classes de com.badlogic.gdx Screen, World et Game.

Screen représente notre application est sert de « main ». Elle crée les objets world et game ainsi que viewport qui est la fenêtre qui apparait à l’écran et gamecam qui est la position logique de ce qui apparaitra dans le viewport.

World contient les objets qui ont une apparence dans le jeu ou une interaction avec (comme des senseurs). Il ne faut pas le confondre avec la gamecam qui est juste la proportion du monde qu’on affiche à l’écran. Chaque corps qui est ajouté dans le monde doit posséder un Body contient des coordonnées formant un Polygon ainsi qu’un type (StaticBody ou DynamicBody, KinematicBody n’a pas été utilisé).

Game qu’on a appelé JdcGame pour Joel-Daniel-Charles-Game contient les paramètres du jeu et ces ressources. On y trouve les constantes de proportions, les sons ainsi que les bits de mask.

## Sprite

Un Sprite est une image contenant toutes les différents frames d’une animation.

C:\Joel_Vaucher\INF2-b\HesETE-p2-Vaucher\hesETE-jdcGame\core\assets\sprites\Run_sprite.png

Figure 6 - Exemple de Sprites

LibGdx possède une classe Sprite afin de simplifier leurs intégrations au jeu.

## InteractiveTileObject

Parmi les objets qu’on a créé dans le monde, certain devront effectuer certaines actions au contact du joueur (mort, bonus).

Le tutoriel proposait une classe abstraite laissant une méthode virtuelle pour que les différentes spécifications fassent eux-mêmes les conséquences d’une collision au joueur.

Nous avons fait deux classes abstraites Enemy et InteractiveTileObject qui fonctionne de la même manière et pour les mêmes raisons, mais dont la famille d’objet est différente et dont les extensions futures pourraient être problématique s’il n’y avait pas deux classes.

## Collision & Mask

Une gestion des collisions assez courante dans les IDE de 2D/3D est la gestion par calque. Cela signifie qu’on définit que certains objets font partie d’un groupe (ou plusieurs) et qu’il ne peut entrer en collision qu’avec seulement certains autres groupes.

## Son

La gestion du son dans notre projet est facilité par la puissance du framework libgdx, lequel offre des outils simples et efficace pour implanter des sons dans le jeu.

La classe AssetManager permet de charger les éléments comme les textures, les polices, la map, les sons et la musique. En déclarant un objet de type AssetManager, on charge les différents sons et musiques utiles à notre jeu.

Dans notre cas c’est dans la classe **JdcGame** que l’on déclare l’object AssetManager en static.

Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 13.09.35.png

L’utilisation est ensuite simple, il suffit de faire appel grace à la méthode get(..) du son qui nous interesse, Ici c’est un exemple dans la classe **PlayScreen**, où l’on fait jouer le son produit par la téléportation du personnage.

Macintosh HD:Users:charlesombangndo:Desktop:Capture d’écran 2016-09-08 à 13.11.25.png

# Problèmes rencontrés et solutions

Le développement de ce jeu a nécessité de la documentation pour comprendre le fonctionnement et les principes du framework libgdx. N’ayant aucune connaissance sur ce framework avant le début de ce projet, nous avons plus ou moins rencontré des problèmes liés à une mauvaise compréhension de libgdx et son utilisation approximative. Il était par exemple difficile de trouver les bonnes proportions, de manipuler la caméra du framework et d’interagir avec précision sur les objets de nos maps de level. Les solutions à nos différentes préoccupations résidaient dans des forums, des tutoriels et surtout dans la documentation du framework.

# Améliorations possibles

N’étant pas des game designers accomplis, il est possible d’apporter des améliorations pertinentes à ce jeu. Que ce soit sur le plan esthétique que techinique, le programme peut évoluer et atteindre un niveau bien plus important.

* **Sauver le status du jeu**: On peut rajouter une fonctionnalité permettant de sauvegarder l’état du jeu en fin de chauqe étape. Dans ce cas, le joueur peut à tout moment revoir ses statistiques de jeu.
* **Des levels à plusieurs à plusieurs étapes**: dans le style mario, terminé un level correspondrait à teminer chaque étape (exemple 5) composant le level. Cela rajouterai du challenge au jeu, avec la satisfaction et l’objectif de terminer tous les niveaux
* **Portes animées**: Pour ajouter du stresse au jeu, animer certaines portes. Celles-ci s’ouvreront allors et se refermeront suivant un laps de temps.
* **Des obstacles en mouvement**: toujours dans l’optique d’augmenter le challenge, on pourrait intégrer des obstacles se déplaçant dans un sens contraire à celui du protagoniste et d’aucuns prenant des directions aléatoires, le but étant de déstabiliser le joueur.
* **Une option de paramétrage :** cette option permettrait à l’utilisateur de configurer le jeu à ses préférences, d’avoir le contrôle de l’interface. Le user pourrait donc dans ce cas controler le son, son volume, il pourrait agir sur la luminosité, le sytle soit la couleur des composants et même choisir un personnage dans une liste plus fournie et plusieurs autres paramètres personnalisables.
* **Un sytème de ranking :** Un sytème de ranking pourrait rendre le jeu passionnant. En effet classer les joueurs attise générlement des challenges, cela eveille l’esprit de compétition. Alors, dans la même lignée que le premier point, sauvegarder le satus du jeu permettrai aux joueurs de connaître le meilleur dans chaque level et tout simplement le meilleur au score dans tout le jeu.

# Conclusion

Au terme de ce challenge, on peut estimer que nos objectifs ont été atteint. Nous avons bel et bien un projet fonctionnel fait dans les trois semaines qui nous ont été données. Globalement comme souvent dailleurs, la collaboration au sein de l’équipe était bien organisée et la gestion des tâches respectée. La prise en main d’un nouveau framework et la mise en place d’un projet avec ce dernier nous donne une certaine satisfaction quand on tient compte du temps imparti. Le résultat des expériences désormais acquise grace aux précédents projets est implacable. On remarque de bons comportements adoptés lors du codage, des réflexes dans la recherche des la documentation dans le cas où des informations nous manque, une utilisation des outils de collaborations comme GitHub de plus en plus maitrisée. LibGdx nous a sacrément surpris grace à sa bibliothèque Box2D qui gère de façon élégante tout les aspects Physiques (science) que ce soit la gravité, les collisions, cette librairie simplifie le travail dans ce domaine qui peut parfois être difficile à mettre en place, comme c’était le cas dans le cadre du projet du premier semestre.

Neuchâtel, le 9 septembre 2016

Vaucher Joël

Ombang Ndo Charles

Rodrigues Lourenço Daniel

# Bibliographie