Rapport Final

du

Projet de L3

GAME FACTORY

Realise par :

Simon ARNOULT

Wissame MEKHILEF

Vincent RENARD

Table des matières

1 Introduction

Blabla sur ce que l'on a voulue faire blabla bla

2 Présentation globale

2.1 Diagram classe

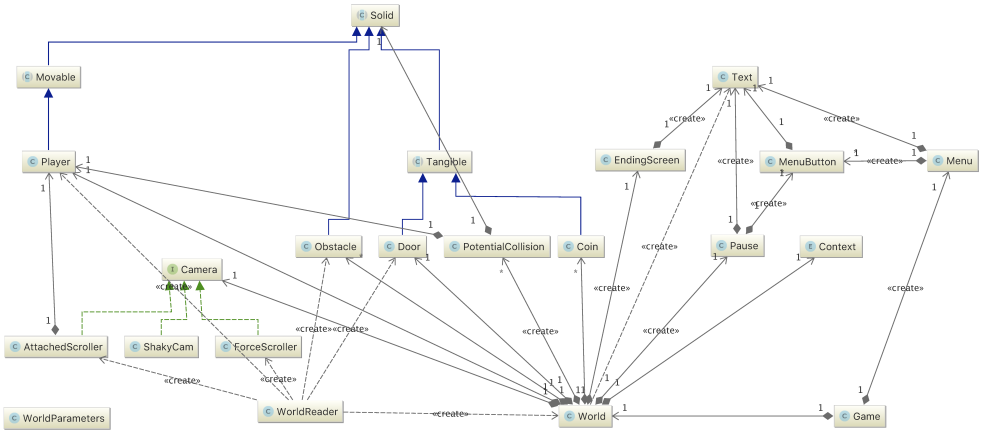

FIGURE 1 – Le diagramme de classe UML

2.2 Diagramme des boucles de jeu

2.2.1 Les contextes

2.2.2 La boucle update

2.2.3 La boucle render

2.3 Architecture MVC

2.3.1 Affichage des données

2.3.2 Interpretation des entrées clavier

3 Une organisation sans faille

+ blabla sur les deadline durant le projet

3.1 Description des taches

3.1.1 Recherche d'un framework + suivi d'un tutoriel

Nous avons choisi d'utiliser la librairie Lightweight Java Game Library (LWJGL) afin de gérer l'affiche graphique de notre jeu. Nous avons ensuite suivi le premier tiers du tutoriel " Jeu 2D avec LWJGL " sur la chaîne YouTube " Tuto Programmation (Marccspro) " afin de nous familiariser avec cette librairie.

3.1.2 Gestion d'un mouvement de caméra simple et autonome

Dans l'optique de pouvoir évoluer dans un monde plus grand que la fenêtre du jeu, nous avons implémenté un scrolling horizontal à vitesse fixée.

3.1.3 Gestion de l'interaction entre le joueur et la fenêtre

À ce stade du projet, nous avons créé un dépôt Git afin de mettre en commun notre travail, l'objectif à court terme étant de gérer simultanément la physique du joueur et le scrolling de la fenêtre, ainsi que les collisions entre les bords de cette dernière et le joueur.

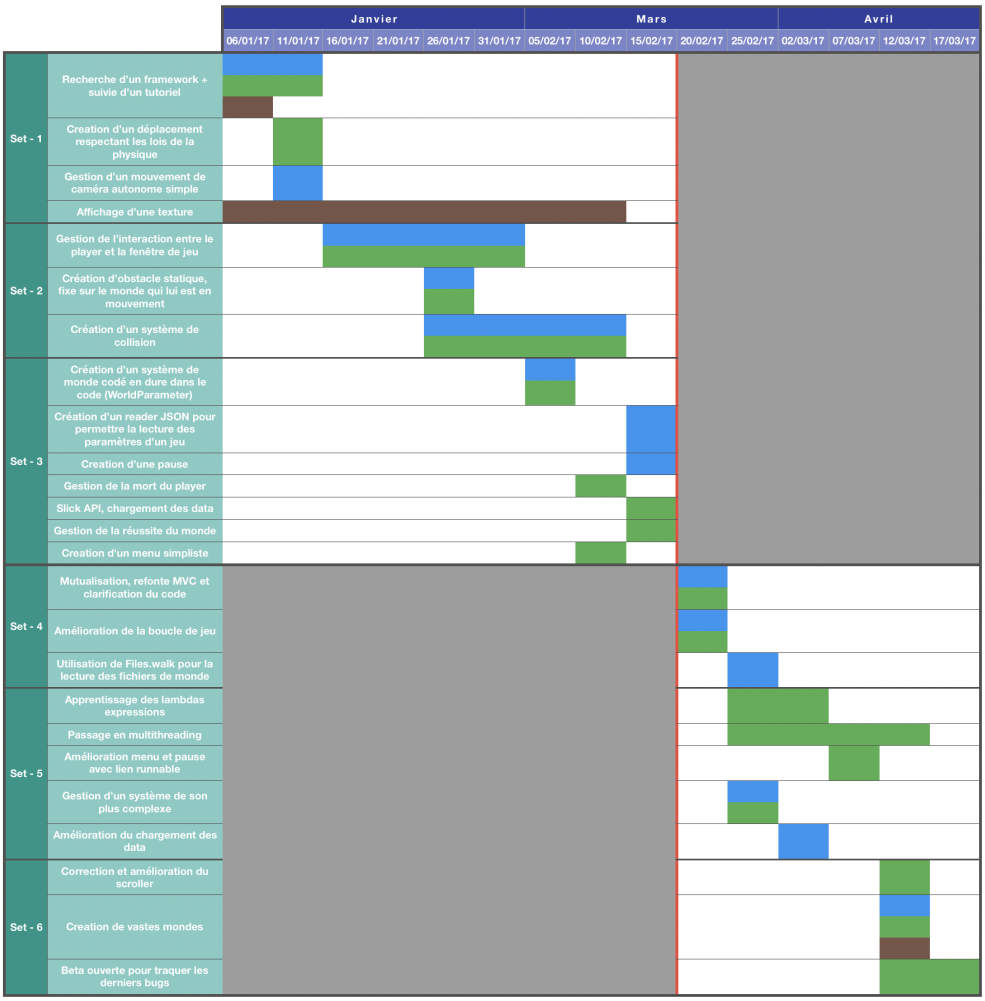

FIGURE 2 – diagram de gantt

3.1.4 Gestion des collisions

Après avoir implémenté des obstacles, nous avons décidé de gérer les collisions entre le joueur et ces derniers. Cette tâche a été le premier défi technique que nous avons rencontré.
Pour résoudre ce problème, nous avons implémenté des objets de type PotentialCollision. Ce sont des couples qui associe au joueur un obstacle du monde. À chaque update du jeu, chaque PotentialCollision est interrogé afin de savoir si le joueur est en contact avec un obstacle, et le cas échéant, sur quel bord de ce dernier la collision a lieu.
Ainsi, dans le cas où le joueur bloqué par un obstacle qui l'empêche d'avancer vers la droite, il est inutile de vérifier sur s'il est en contact avec le bord gauche d'un autre obstacle.

3.1.5 Création d'un parser JSON

Afin de pouvoir choisir entre plusieurs mondes au lancement du jeu, nous avons décidé de stocker les données des différents niveaux dans des fichiers JSON qui seraient lu au moment où l'utilisateur choisit le monde auquel il veut jouer.
Chaque fichier contient les toutes les informations nécessaires à l'instanciation du monde souhaité, telles que les textures à afficher, la musique à jouer, le placement des obstacles et de la sortie, le type de caméra, ou encore la puissance de la gravité qui s'applique au joueur.
Ce système nous a permis par la suite de créer des mondes variés et de s'affranchir des contraintes liées au fait de stocker les données des différents mondes directement dans le code du jeu.

4 Quelques defis techniques

4.1 Gestion des collisions

4.2 Le lambda calcul, multithreade

Table des figures