Rapport de projet

Groupe 23

RENAULT Alexis

LEVEQUE Quentin

RENARD Vincent

Encadrants :

TODINCA Ioan

PEREZ Anthony

Informations :

Langage : Java 8, MySQL

Bibliothèque Graphique : JavaFX

Sujet : ColorSwitch Classique

Structure des classes :

Nous avons essayé de faire au mieux pour avoir une structure MVC.

Le Main est constitué de ces 3 éléments et les appelle.

Notre implémentation du MVC ne comporte pas de lien direct entre la vue et le modèle, les informations dont a besoin la vue passent par le Contrôleur, dont le seul but est de faire le lien entre le Modèle et la Vue.

Modèle :

Ensuite, notre modèle encapsule toutes les données non graphiques, de manière à séparer toutes les composantes graphiques du modèle de manière à pouvoir utiliser d'autres bibliothèques graphiques.

Rapport de projet

La classe phare du modèle est la classe Game, cette dernière comprend plusieurs champs :

- Path : Le chemin que suivra la Balle, constitué d'Obstacles, eux-mêmes constitués de Shapes

- Universe : Un ensemble de lois que suivra la Balle, l'interface graphique, et les interactions entre différents éléments

- Ball : La balle représentant le joueur principal, que l'on peut faire sauter.

Vue :

La vue, comprend des implémentations graphiques de ces classes et différentes classes pour afficher les différentes pages, telles que le Menu, le Score, ou le jeu en lui-même.

En particulier, nous avons modelLaw, modelObstacles, modelShapes, modelBall qui deviennent UseLaw, Obstacles, Shapes, Player dans la vue, qui sont composés du modèle correspondant et de composants graphiques pour les afficher.

View appelle d'autres classes commençant par View pour voir les différentes pages.

ViewGameManagement sert à convertir une Game, classe du modèle, en Scene pour JavaFX

Elle utilise donc ViewPath qui convertit notre Path du modèle en composantes graphiques, et ViewTimer, composée d'un AnimationTimer, classe de JavaFX dont la méthode handle s'appelle à chaque frame. On utilise cette classe pour appliquer les Law du modèle en useLaw, qui est une interface, cela nous permet d'appliquer différents traitements que l'on peut choisir, à la manière du Design Pattern Strategy.

En théorie, cela nous permet de détecter les collisions, avoir une gravité plus fluide, différents modes de jeux et de déplacements de la balle, et différentes manières de regarder la Scene.

Dans la pratique nous avons des problèmes de performances en fonction du nombre d'obstacles.

Rapport de projet

Fonctionnalités principales:

- Un Color Switch qui fonctionne plus ou moins comme attendu

- Un Menu Principal

- Un système de Gestion des Scores via une Base de Donnée

- Plusieurs niveaux / Modes de jeux et la possibilité d'en rajouter d'autres facilement

- Un système de génération d'obstacles aléatoires

- Implémentation d'une musique et de sons

Difficultés rencontrées :

- Nous avions du mal à savoir comment implémenter ce projet avec une structure MVC
- Difficulté de trouver l'outil nécessaire afin d'implémenter les collisions
- La maîtrise des composantes graphiques et des performances de l'application n'était pas une tache facile et nous a conduit à manquer de temps à la fin de projet, ce qui cause le rendu un peu brouillon en fonction de nos ambitions
- Le travail en groupe de 3 n'a pas toujours été évident, il y a eu quelques désaccords lors du déroulement du projet, mais pas au point de faire éclater le groupe
- Nous avons manqué de temps pour implémenter tout ce que l'on voulait

Commentaires :

- Nous avons dû revenir à une ancienne version du saut du joueur et de la gravité à cause de performance de l'application. La nouvelle implémentation est visible dans le mode Gravity.
- Nous comprenons totalement l'utilité de MVC mais cela nous a forcé à dupliquer beaucoup de nos classes, ce qui peut poser des problèmes de maintenance.
- Nous sommes un peu déçus, car nous aurions aimé implémenter d'autres fonctionnalités, comme une IA, développer une petite interface en Jeu et surtout régler notre problème de performance qui gâche un peu le travail fait en amont
- Le problème de faire un projet avec une BDD, c'est qu'il faut un serveur. La possibilité de se connecter à un serveur arbitraire n'est pas tout à fait implémentée
- La Javadoc est fournie en Annexe, et un .jar est inclus