Auteurs : Raphaël BACKLAND & Gaël PAQUETTE

Date de Création : 21/04/2024

Dernière Modification : 10/06/2024

Rapport de développement

Cette page est dédié à l'explication de l'architecture du projet `BACKLAND_PAQUETTE_BabaIsYou` et aux améliorations, corrections apportées depuis la soutenance bêta.

SOMMAIRE

INTRODUCTION

L'objectif de ce projet consistait à développer le jeu *Baba Is You* en utilisant la programmation orienté objet pour le cours de *Programmation Orienté Object*. Pour répondre au besoin, nous avons utilisé le language JAVA avec `jdk-21` ainsi que les bonnes utilisons de la programmation orientée objet. Ce projet nous a permit de monter en compétences sur la programmation orienté objet mais aussi l'utilisation d'un backend, d'un frontend et des méthodes simples devOps en JAVA. Au cours de ce rapport, nous verrons l'architecture de notre projet, le backend du projet, le frontend du projet ainsi ques les améliorations et corrections apportées depuis la soutenance beta.

ARCHITECTURE DU PROJET

Pour ce projet nous avons décidé d'utiliser une architecture MVC (Model - View - Controler) pour simplifier la lecture du code mais aussi bien différencier le backend du projet avec l'interaction utilisateur (Controler) et l'affichage du projet (View). Ainsi, le code du projet se répartie sous 3 package différents. Le premier `fr.esiee.babaisyou.controler` comporte les classes dédié à l'interaction avec l'utilisateur avec notamment la classe `Main.java`. Le deuxième package `fr.esiee.babaisyou.model` contient le backend du projet avec toutes les classes dédié aux règles du jeu, la gestion de l'aire de jeu, la gestion des objects est des mots. Le troisième package `fr.esiee.babaisyou.view` comporte le frontend du projet avec l'utilisation de librairie `Zen6` qui permet de gerer l'affichage graphique du jeu à l'écran.

BACKEND DU PROJET

Plusieurs classes composent le package `fr.esiee.babaisyou.model` pour le back-end du jeu. Dans un premier temps, nous avons la classe centrale du jeu `Gameboard` permettant l'initialisation de la grille de jeu et gérant les différents déplacements du joueur et le poussage des blocs. Pour gérer les déplacements, on utilise une énumération `Direction` pour éviter de les réécrire sous forme de chaîne de caractères. Dans un second temps, nous avons une interface `Square` représentant l'élement du jeu, on peut contrôler ses coordonnées, connaître son statut (de quel type est-il ?). Nous avons ainsi, les classes `Name`, `Object`, `Operator` et `Property` qui sont implémentées par cette interface. Enfin, il y a également la classe `Rule` permettant le contrôle des règles du jeu, c'est avec cette classe que la méthode détecte sur le plateau si un ensemble Nom-Opérateur-Nom ou Nom-Opérateur-Propriété existe. Elle permet également surtout de vérifier que le joueur reste toujours en règle par rapport aux ensembles formés, s'il existe toujours, s'il a gagné ou s'il a perdu.

FRONTEND DU PROJET

Pour cette partie du projet, nous avons utilisé la librairie `Zen6` qui permet une gestion des events clavier et souris créer par l'utilisateur mais aussi tout l'affichage graphique du jeu. La plus grosse partie du travail reposait sur le chargement des images en mémoire pour ensuite les afficher à l'écran en fonction du niveau chargé. Pour cette partie du travail, nous sommes parties sur l'utilisation de `.gif` qui permettait de donnée vie au élément du jeu pour respecter les demandes du client. Ainsi, nous avons deux classes pour gérer le chargement de toutes les images du jeu que ce soit les blocs ou les images de fin de partie (celle-ci sont fixes, nous utilisons donc des `.png` pour celles-ci). Les deux classes de chargement des images `ImagesLoader` et `ImagesEndLoader` sont similaires et reposent sur la même technique de chargement des images. C'est-à-dire, la lecteur du chemin d'accès à l'image, la lecture et le chargement de l'image. Puis l'insertion de l'image dans une Map pour retouver facilement celle-ci à partir de son nom. Dans la classe `ImagesLoader` nous utilisons deux Map pour les objects et les mots, et dans la classe `ImagesEndLoard` nous utilisons une seule map pour charger les images de fin de niveaux. Ensuite, nous utilisons une classe pour afficher le plateau de jeu sur l'ecran. Cette classe prend donc en paramètre la classe `GameBoard` pour afficher le jeu. Pour finir, le frontEnd du projet s'étend aussi dans le package `fr.esiee.babaisyou` avec la classe `EventGame` qui s'occupe de toute la gestion des events clavier créer par l'utilisateur.

AMELIORATION ET CORRECTIONS APPORTEES DEPUIS LA SOUTENANCE BETA

Lors de la soutenance bêta, nous étions arrivés à la moitié de la réalisation de notre projet, nous avons montré à l'enseignant l'ensemble des classes qui composent le package `fr.esiee.babaisyou.model`. L'ensemble des classes étaient pour la plupart correctes. Cependant, après discussion avec l'enseignant, nous avons remarqué que la classe `GameBoard` de ce package, classe la plus importante du jeu, ne serait pas compatible avec la suite du projet, en effet, cette classe reposait sur un tableau qui permettait d'associer un élement à chaque coordonnées. Au

début du projet, nous avons choisi cette structure pour démarrer le jeu sans y avoir fait assez de réflexion, c'est par la suite que nous avons su que chaque indice du tableau du jeu pouvait contenir plusieurs blocs. Ainsi, nous avons dû restructurer l'ensemble de cette classe pour l'adapter à la nouvelle structure qui répondra cette fois-ci à nos attentes : une map.

CONCLUSION

Ce projet nous à permit de monter en compétences sur la programmation orienté objet avec JAVA ainsi qu'une montée en compétence sur GitHub. en effet, pour ce projet nous avons utilisé un repository GitHub qui se trouve la l'adresse suivante `https://github.com/Gael-Paquette/BACKLAND_PAQUETTE_BabaIsYou`. En utilisant un gestionnaire de code comme GitHub, cela nous a permit de bien nous répartir le travail mais aussi de garder un code tout le temps à jour avec l'utilisation du principe de `branch` et de `merge`.