université de BORDEAUX

Master 1 Informatique

Projet Approche Objet - Rapport

Jeu du Labyrinthe

GIOVANNANGELI Loann, MAYOLINI Maxime, MOUHOUB Noureddine, RIOU Maxence

22 décembre 2017

Lien Github : https ://github.com/LoannGio/Labyrinthe

Chapitre 1

Fonctionnalités

1.1 Fonctionnalités implémentées

— génération d'un labyrinthe "imparfait" : on génère un labyrinthe parfait puis on supprime des murs de manière intelligente ;

— déplacements du joueur : Pacman avance de manière périodique (très courte) dans une direction. On modifie cette direction via les touches : Z,Q,S,D et les flèches UP, DOWN, RIGHT, LEFT ;

— ajout d'une porte de sortie du labyrinthe placée de manière aléatoire ;

— ajout d'un adversaire qui poursuit Pacman ;

— déplacements intelligents de l'adversaire : avance aussi de manière périodique en même temps que le joueur. Il suit toujours le chemin le plus court vers le joueur. On recalcule ce chemin à chaque déplacement ;

— gestion de fin de partie : on gagne la partie lorsque Pacman atteint la porte. On perd la partie si l'adversaire atteint Pacman ;

— ajout de bonbons bonus qui augmentent le score lorsque Pacman les récupère. Il existe 4 types de bonbons qui montent le score de 50, 100, 150 et 200 ;

— ajout d'indicateurs :

  — temps écoulé depuis le début de la partie ;

  — score : calculé en fonction du temps écoulé et des bonbons ramassés ;

  — niveau de la partie en cours ;

  — direction courante du Pacman ;

— plusieurs niveaux : on augmente la difficulté à chaque niveau en augmentant le nombre de murs (limite : labyrinthe parfait) et la vitesse du jeu (limite : 400ms entre deux déplacements) ;

— boutons Quitter et Rejouer (même niveau) lorsqu'on perd ;

— boutons Quitter et Continuer (niveau suivant) lorsqu'on gagne.

1.2 Fonctionnalités non-implémentées

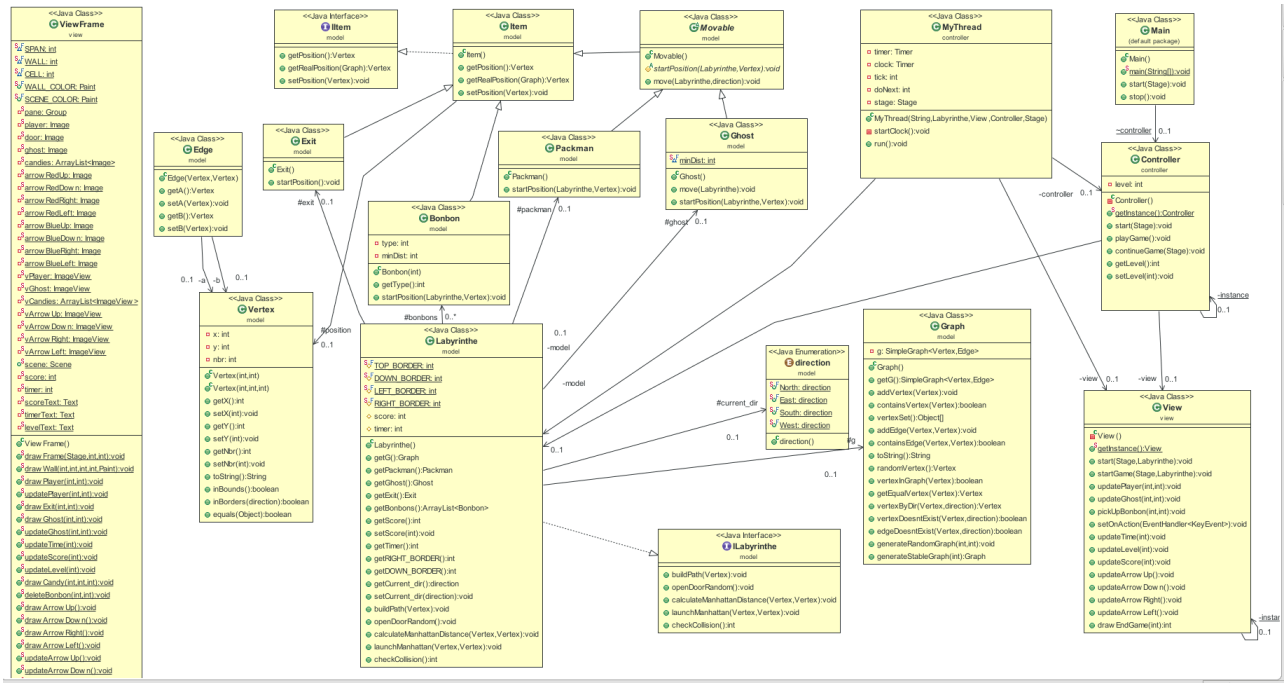
— interrupteur en jeu permettant d'ouvrir/fermer une porte ;

— mouvement "fluide" du Pacman et de l'adversaire ;

— tous les X niveaux, rajouter un Ghost poursuivant.

Chapitre 2

Conception



Nous avons utilisé un Design Pattern MVC (Model-View-Controller) de façon à décomposer le code de notre application en trois sous-parties distinctes qui communiquent entre elles.

La partie Model contient les données de l'application : graphe, sommets, arêtes, labyrinthe, Pacman, adversaire, bonbons. La partie Controller gère les événements utilisateur comme les déplacements du Pacman, les collisions, les boutons pour quitter ou rejouer une partie. La partie View comporte tout le code qui se réfère à l'affichage des sprites du jeu ainsi que les différents indicateurs.

Notre application utilise un seul Controller. Nous avons implémenté un Singleton sur cette classe pour interdire de multiples instanciations en même temps. Nous avons fait de même pour la classe View.

Des interfaces ont été utilisées pour les classes Labyrinthe et Item afin de décrire des fonctionnalités attendues côté utilisateur et donc à implémenter. La classe Movable est abstraite car son code est utilisé par le Pacman et l'adversaire.

Chapitre 3

Tests

Des tests unitaires ont été réalisés dans le projet afin d'assurer le bon fonctionnement de certaines méthodes importantes.

Voici la liste des tests implémentés :

- VertexTest :
  - testInBounds() : les sommets ajoutés sont bien à l'intérieur de labyrinthe ;
  - testInBorders() : les sommets peuvent se déplacer correctement par rapport aux éventuels murs les entourant ;
- LabyrintheTest :
  - testCheckCollision() : la méthode renvoie la bonne valeur en fonction de la collision correspondante (avec l'adversaire, avec la porte de sortie, avec les bonbons, pas de collision) ;
- GraphTest :
  - testContainsVertex() : le graphe du labyrinthe contient bien un certain sommet ou non ;
  - testContainsEdge() : la graphe du labyrinthe contient bien une certaine arête ou non ;
- BonbonTest :
  - testStartPosition() : les bonbons sont placés initialement de façon correcte, c'est-à-dire qu'ils ne sont pas au même endroit qu'un autre élément (adversaire, porte de sortie ou autre bonbon).

Chapitre 4

Répartition des tâches

Fonctionnalités	Développeurs
Génération du graphe du labyrinthe parfait	Noureddine, Loann, Maxime, Maxence
Affichage du labyrinthe en fonction du graphe	Maxime
Rendre le labyrinthe imparfait de manière intelligente	Noureddine, Maxence
Ajout porte de sortie	Maxime, Maxence, Noureddine
Ajout joueur Pacman	Loann, Maxence
Déplacement joueur Pacman	Loann, Noureddine
Gestion de la fin de partie	Loann, Maxime
Ajout d'un adversaire	Maxime, Maxence
Déplacement adversaire	Loann, Noureddine
Gestion des collisions	Loann, Maxence, Noureddine
Ajout des bonbons bonus	Loann, Noureddine
Ajout des indicateurs temps, score, niveau, direction	Maxence, Maxime
Tests unitaires	Maxime