Equipe 3 : ETCAETERA Projet ACL

Conception

1. Fonctionnalités implémentées et organisations des tâches

Le planning et l'oganisation du projet a été fait sur le site : https://trello.com/invite/b/1p8QvJyv/eff51eb7ecd651f6885c1cde3ed2a46e/tableau-agile

On y retrouve l'équivalent des backlogs product et sprint ainsi que toutes les informations relatives au projet.

Sprint 1 (04/11 au 22/11)

Début du projet "Labyrinthe", on doit mettre en place les premières classes du projet. Codage en java des fonctionnalités de base du projet

Fonctionnalités : Il nous faut un plateau (pour l'instant on se concentre sur un plateau par défaut qui n'a rien dessus d'autre que le héros), un héros qui peut se déplacer sur le plateau (ne pas en sortir, positionné au hasard sur le plateau).

- Classes à programmer : Héros, Labyrinthe, Trésor, Principale

Héros : Il faut que le héros puisse être ajouté à un endroit quelconque du plateau de jeu. Il est représenté par ses coordonnées cartésiennes. On y trouve la méthode déplacement qui permet de faire déplacer le héros en ligne et en diagonale.

Labyrinthe : On doit créer la classe Labyrinthe qui aura les délimitations qu'on veut tout ça en coordonnées cartésiennes. Il n'est pas possible de dépasser les limites du labyrinthe.

Trésor : Classe simple qui détermine la position sur le plateau du trésor. Le trésor a une taille (1,1) sur le plateau. Principale : Classe qui permet de lancer le jeu, elle utilise toutes les autres classes programmées. Elle doit pouvoir positionner le héros et le trésor sur le plateau du labyrinthe et faire les mouvements.

Interface : Classe abstraite qui est appelée par Principale pour faire l'interface.

Compte-Rendu Sprint 1

Le programme est opérationnel et fonctionne en ligne de commande. Les classes ont été changées pour correspondre à nos attentes :

Héros : Le héros est représenté par ses coordonnées sur la carte (posX et posY). On lui attribue des getters/setters.

Labyrinthe : classe qui définit les limites du labyrinthe et permet un affichage dans la ligne de commande.

Trésor : Représenté par ses coordonnées sur la carte, on lui ajoute la méthode ouvrir() qui se déclenche quand le héros l'atteint et met fin au jeu avec un message.

Mur : Met en place des murs sur la carte à des coordonnées précises, on ne peut pas traverser les murs.

Main : Classe principale du projet, elle lance le programme : crée un héros, un trésor, initialise le labyrinthe ainsi que les murs si besoin. Elle lance une invite de commande qui permet de jouer. L'utilisateur est invité à se déplacer dans le labyrinthe et à atteindre le trésor pour finir la partie. Maven : Tout le code est utilisable par intermédiaire du l'outil de gestion Maven.

Sprint 2 (22/11 – 14/12)

Rappel, il y a un Trello qui récapitule de manière plus visuelle ce qui est à faire et qui nous sert de base de travail sur le lien : https://trello.com/invite/b/1p8QvJyv/eff51eb7ecd651f6885c1cde3ed2a46e/tableau-agile

L'objectif de ce sprint est la mise en place d'une interface graphique de base pour notre jeu ainsi que l'amélioration des classes déjà existantes pour avoir de plus amples capacités.

Tous les objets qui font partis du labyrinthe sont définis par leurs coordonnées sur la carte. Il revient donc à créer une classe Entite dont tous ces éléments héritent qui définit chaque objet par ses coordonnées.

Tresor, Heros, Mur héritent de Entite.

Nous avons choisi de garder le moteur graphique qui nous était fourni, nous allons donc prendre le temps de bien le comprendre afin de mettre en place notre interface graphique. Chaque classe déjà codée devra recevoir une amélioration, qui sera au moins une couleur unique permettant de les identifier sur la carte 2D.

Ce qui est à faire : Niveau d'importance de 1 à 5

Comprendre le template - 5 **(Tous)** : Il est impératif que nous ayons tous bien compris comment fonctionnait le moteur fourni pour mieux comprendre comment faire la partie graphique de notre projet.

Classe Entité **(Clément)** - 4 : Mettre en place la classe Entite dont héritent les autres classes définies par leurs coordonnées. Cela allège les notations de tous les objets mis en place.

Classe Trésor **(Thibaut)** - 3 : Mise en place du trésor sous la forme d'un point coloré de couleur jaune et d'une animation de victoire dans la méthode ouvrir() déjà créée.

Classe Héros (Clément) - 3 : Le héros doit être repéré sur le plan comme un point coloré (bleu) que l'on voit se déplacer. Classe Labyrinthe (Elko) – 3 : Les contours du labyrinthe sont des rebords marrons comme les murs. Il faut travailler à faire en sorte de pouvoir générer le labyrinthe grâce à un fichier .txt.

Classe Mur (Thibaut) - 3 : Les murs sont des entités de couleur marron qui sont bien visible sur la carte.

Fonction Main (Yassine) - 3 : Les améliorations des classes doivent se répercuter dans la fonction main.

Recherche de fichiers images (Yassine) - 2 : Travail plus secondaire qui consiste à préparer des images qui serviront pour la suite lors de la prochaine phase du projet, il faut un répertoire d'images (url,image) qui correspondent à tout ce qui est contenu dans le labyrinthe (mur, héros,sol,trésor).

Compte-Rendu Sprint 2

Nous avons utilisé le moteur de jeu fourni pour avoir une version opérationnelle similaire à celle proposée en v1 mais avec une interface graphique.

Compréhension du moteur de jeu : Fait

Classe Entite : Fait

Classe Trésor : Fait

Classe Heros : Fait

Classe Labyrinthe : Fait sans fichier txt (à faire sprint3)

Classe Mur : Fait mais à améliorer

Main : Faite

Recherche de fichiers image : Fait dans le dossier

Graphism icons (à compléter sprint3)

Ajout de la fonction canMove() qui détermine si le personnage peut se déplacer dans une direction ou non. Ajout de nouvelles méthodes de dessin pour chaque objet du jeu.

À améliorer : Nous avons une dynamique de travail trop irrégulière qui pourrait être améliorée en utilisant les 3 semaines de sprint de manière plus efficace, nous en aurons besoin pour la suite. Il y a des points à retravailler sur ce qui a été fait pour être utilisé dans un contexte plus global.

Bons points : - Charge de travail répartie - Les délais sont respectés et le travail avance comme il faut

Sprint 3 (15/12 – 06/01)

L'objectif de ce sprint est l'amélioration de fonctionnalités déjà existantes avec l'ajout de fichiers images pour représenter nos objets, l'ajout de nouvelles fonctionnalités qui seront utiles comme les monstres qui seront utilisées ou des statistiques pour les personnages qui peuvent attaquer. Sera aussi mis en place des tests J-Unit pour faire des vérifications de fonctionnement. À faire :

- (Yassine) Ajout d'une classe Personnage qui hérite de Entite et dont la classe Heros hérite. Cette classe permet de centraliser la récupération d'information comme les points de vie, la vitesse et d'autres statistiques des personnages du jeu.

- (Yassine) Ajout d'une classe Monstre qui hérite de Personnage. Se déplace de manière aléatoire sur le plateau de jeu à chaque tour (un tour = un mouvement du héros). Les monstres suivent pour l'instant les mêmes règles que le héros (ne peuvent pas traverser les murs). Si le monstre est sur la même case que le héros, le héros meurt (fin du jeu).

- (Elko) Tests unitaires pour vérifier automatiquement le bon fonctionnement du programme.
- (Thibaut) Les éléments graphiques récupèrent une texture d'après des images qui leurs sont transmis : Les personnages ont plusieurs textures mimant un déplacement, texture trésor, texture mur/parois pour l'instant.

- (Clément) Utilisation d'un fichier txt pour générer le labyrinthe : c'est un fichier qui est fait avec les caractères qu'on a utilisé pour former le terrain ('-' pour un mur h le héros t le trésor, etc.)

- (Clément) Ajout Piège : un piège est une case cachée du plateau qui fait des dégâts au héros quand il marche dessus.

- (Thibaut) Mur : Faire en sorte de mieux gérer le fait d'utiliser plusieurs Murs. Objectif de séance : Avoir formé le fichier .txt du labyrinthe (non implémenté) et avoir mis le monstre (immobile) sur le terrain.

Compte-Rendu Sprint 3

Classe Personnage : Fait Classe Monstre : Fait Textures : Fait sauf pour le Monstre et le Héros qui doivent être faits en mouvements Fichier texte pour générer le labyrinthe : Fait Tests Junit : Fait Mieux gérer les murs : Appliqués à la classe Labyrinthe avec le fichier texte : Fait Case piège : Faite (invisible 3 cases au-dessus du trésor)

Tout ce qui était prévu a été fait. Il ne reste que peu de temps pour terminer le jeu mais il est déjà fonctionnel et il n'y a que des petites choses à rajouter qui pourront être fait durant les quelques jours restants.

Sprint 4 (07/01 – 10/01)

L'objectif de ce sprint est de terminer les quelques détails qu'il resterait à faire sur notre programme ainsi que de préparer la présentation qui sera faite la séance prochaine.

(Clément) Case Téléportation qui téléporte le héros à un point prévu sur le labyrinthe. S'implémente de la même façon que la case piège, c'est une case invisible qui ne prend pas surprise et le déplace.

(Clément) Mettre en place plusieurs labyrinthes : 3 fichiers texte avec des niveau de difficulté:

(Thibaut) Héros et monstre avec une texture graphique en mouvement pour caractériser les déplacements de ces entités. Il faut que pour chaque déplacement on puisse voir qu'il y a une impression de mouvement.

(Yassine) Mise en place du fantôme : monstre particulier qui peut se déplacer à travers les murs. Fonctionnement semblable au monstre de base, peut se déplacer vers le héros si possible.

(Elko) Utilisation de la fonction attack() pour que le personnage puisse attaquer les monstres et se faire attaquer par eux aussi, perdre de hp,...

(Elko) Tests Junit pour confirmer ce qui aura été ajouté.

(Tous) Préparation de la présentation. Powerpoint, distribution des temps de parole, etc.

Objectif de séance : Avoir terminé la case Téléportation ainsi que les textures graphiques. Avoir distribué les temps de parole et les taches pour le Powerpoint.

2. Conception et diagrammes

Diagramme de classes de l'application Maze version 1.1 (sprint 1)

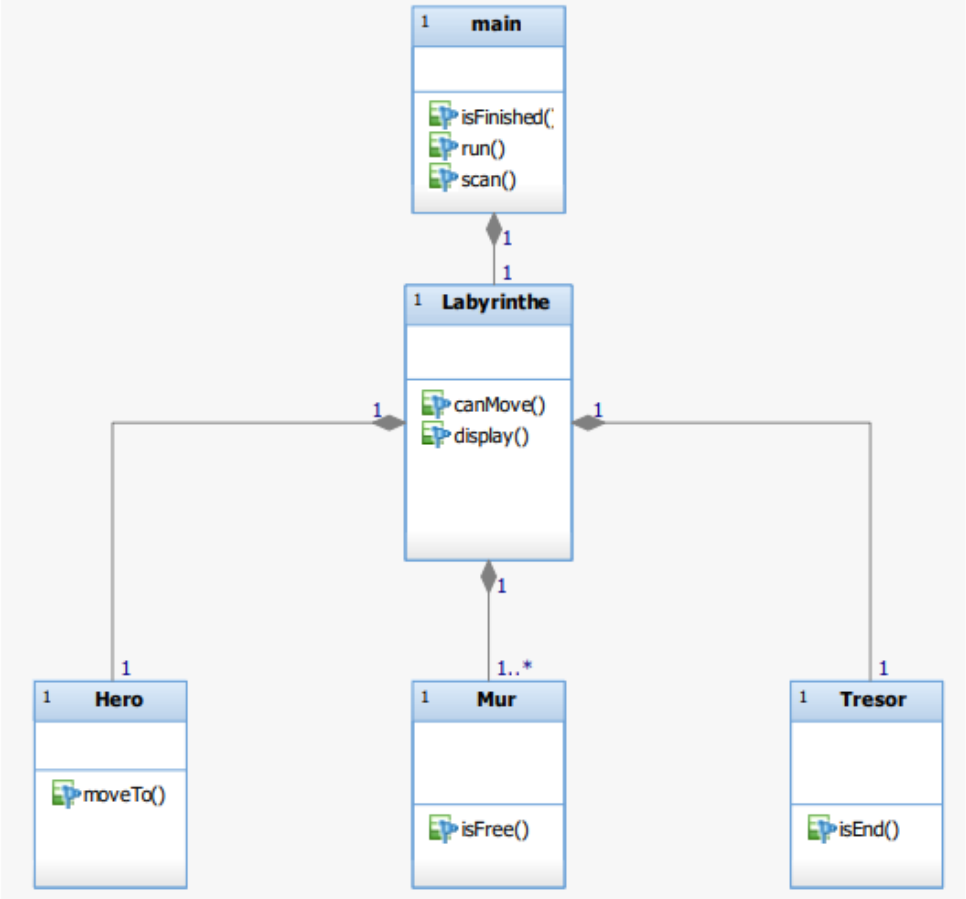

Diagramme de séquences de l'application Maze version 1.1 (sprint 1)

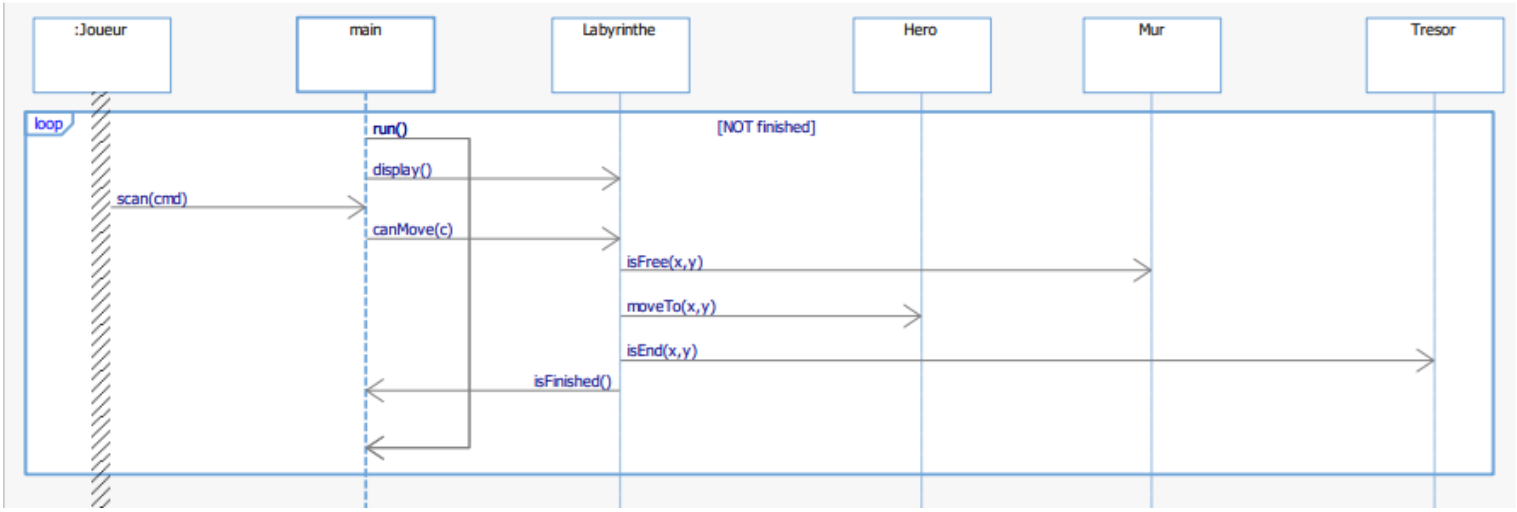

Diagramme de classes de l'application Maze version 2.1 (sprint 2)

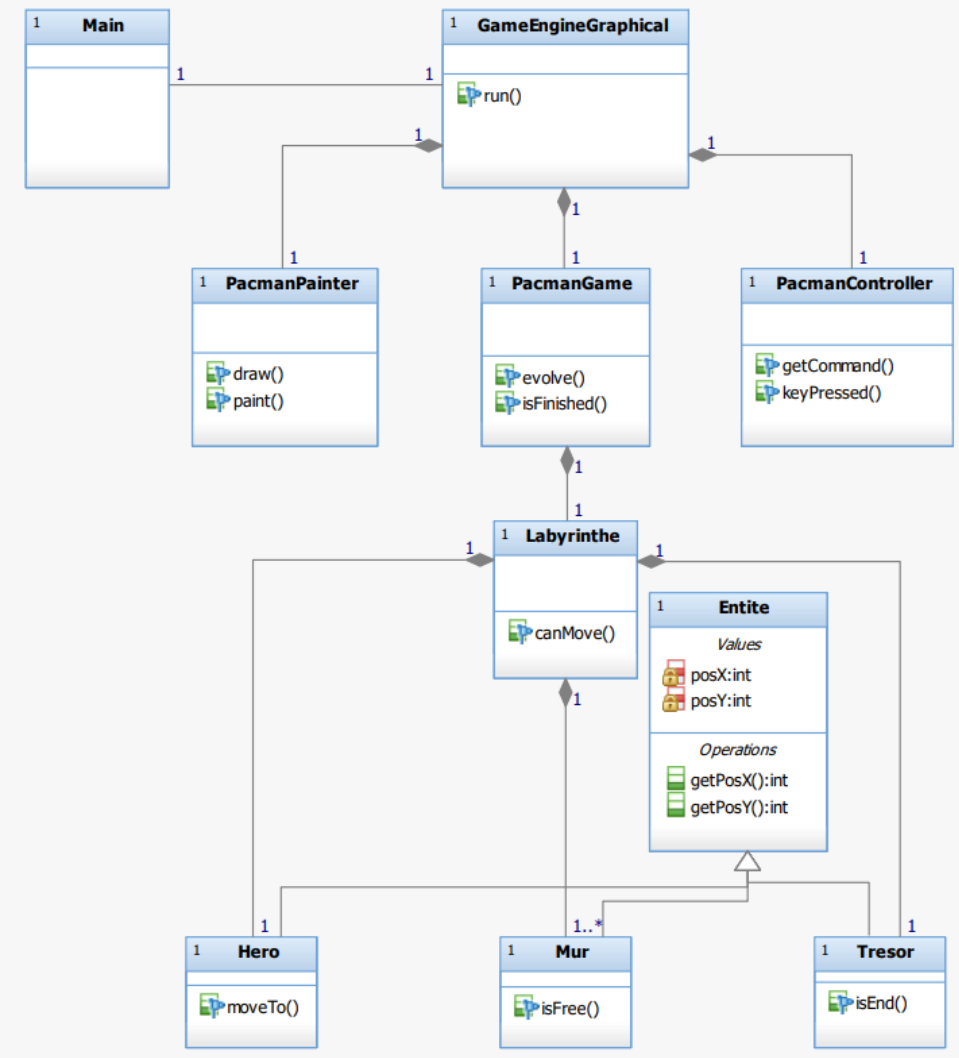

Diagramme de séquences de l'application Maze version 2.1 (sprint 2)

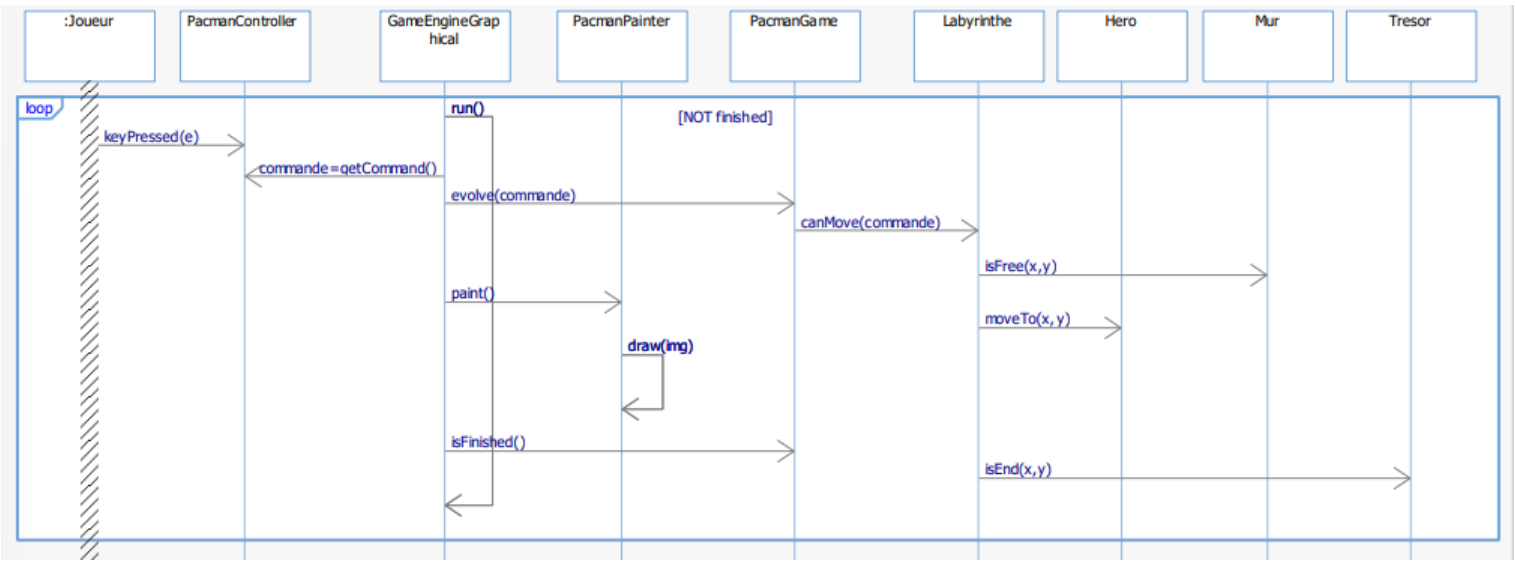

Diagramme de classes de l'application Maze versin 3.1 (sprint 3)

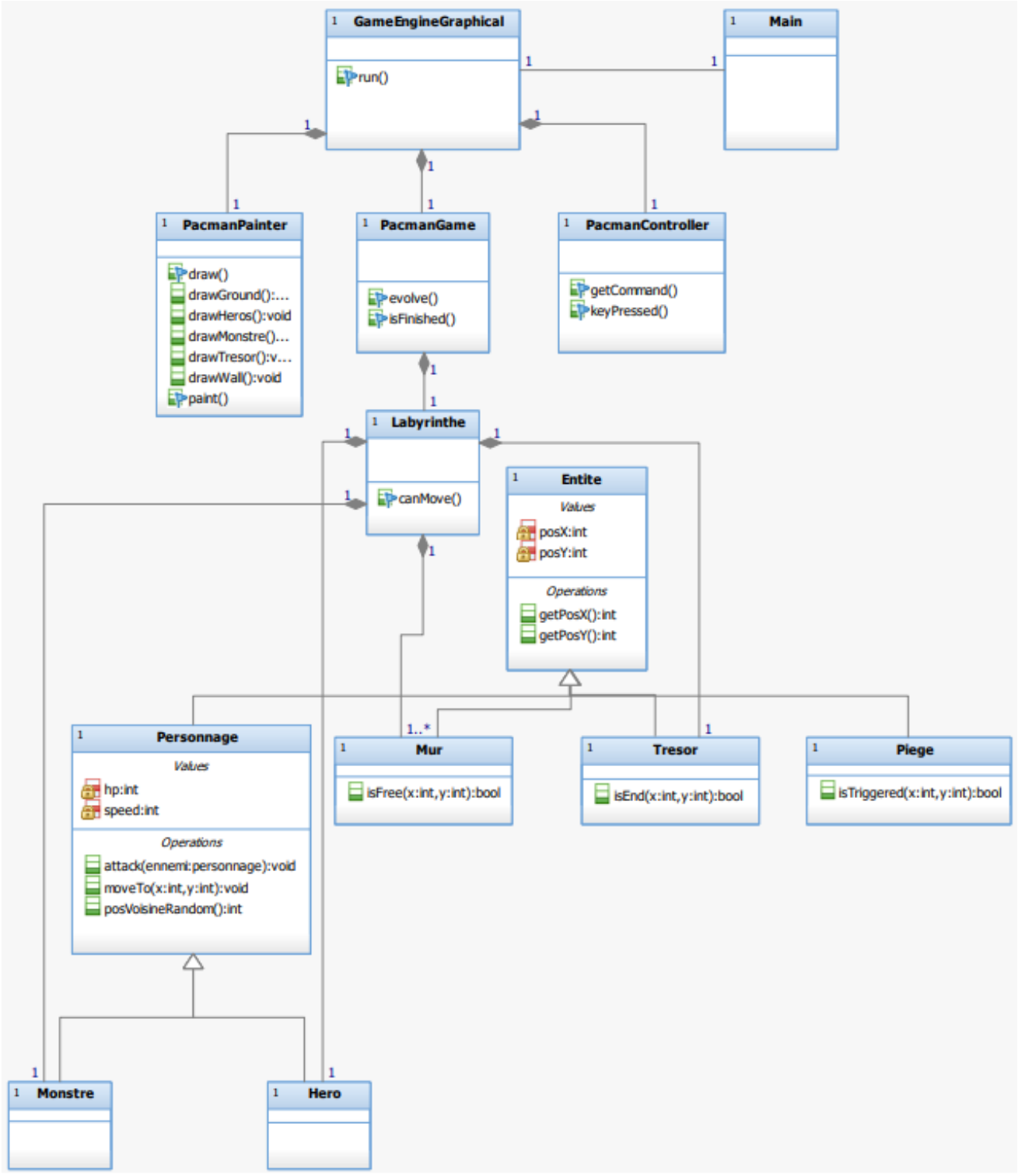

Diagramme de séquences de l'application Maze version 3.1 (sprint 3)

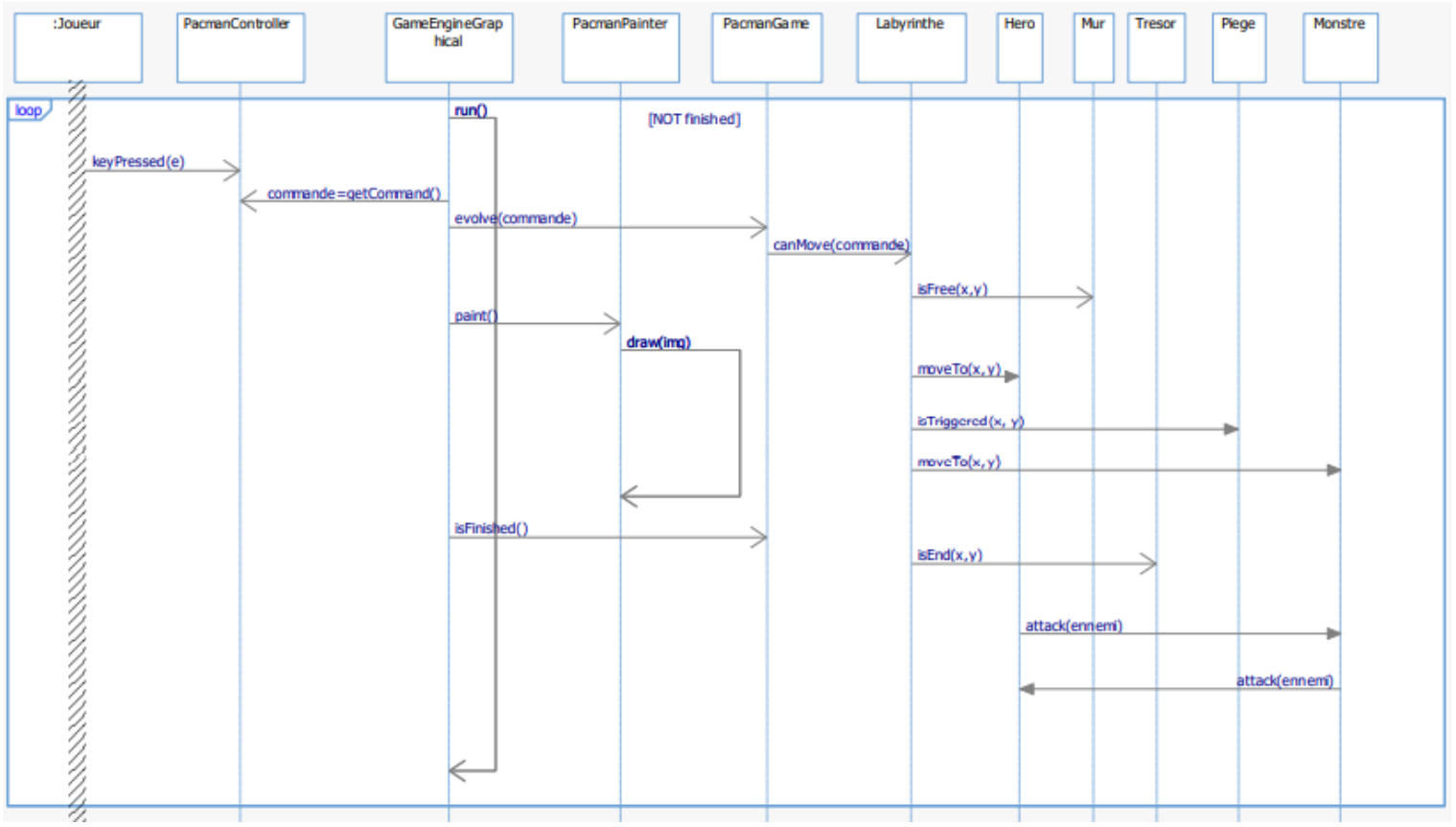

Diagramme de classes de l'application Maze version 4.1 (sprint final)

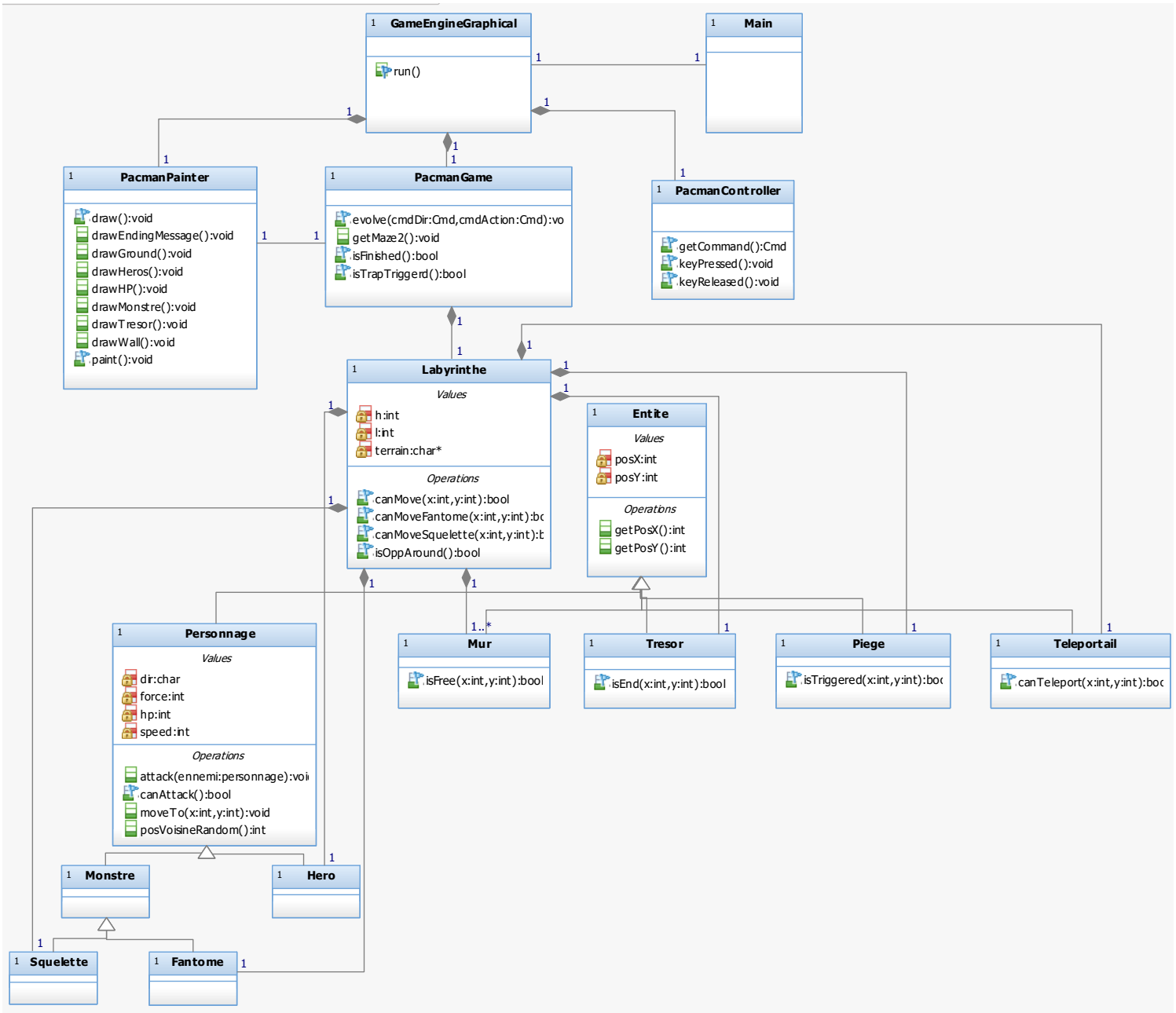

Diagramme de séquences de l'application Maze version 4.1 (sprint final)