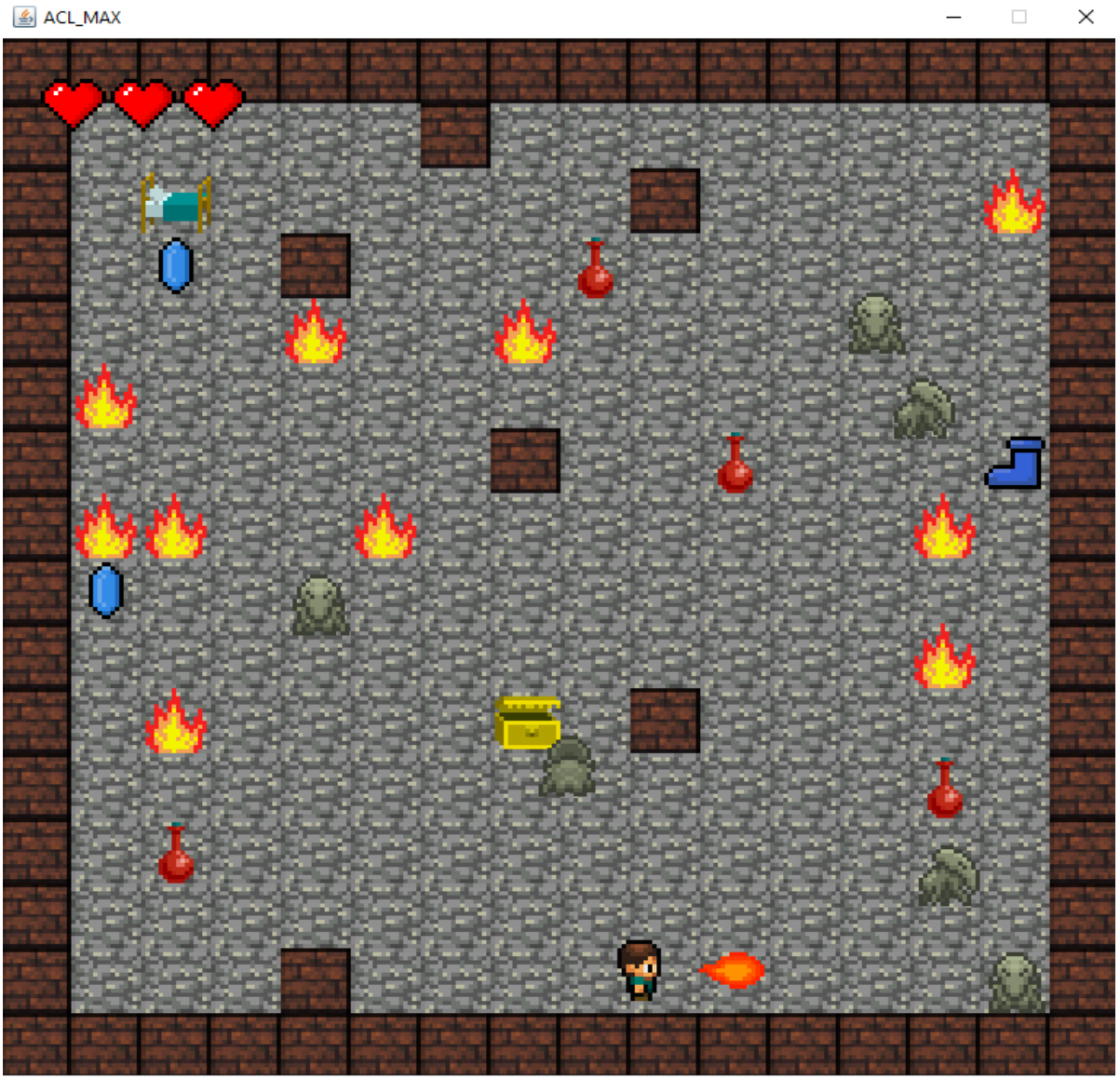
Document de conception

Projet ACL 2023 /2024

Réalisé par : ACL-2023-UL-MAX

« Balade dans un Labyrinthe »



Introduction :

Ce document présente la conception détaillée du projet **"Balade dans un labyrinthe",** réalisé dans le cadre du cours d'Analyse et Conception de Logiciels . Notre équipe, **ACL-2023-UL-MAX**, s'est engagée dans un processus de développement itératif inspiré de la méthodologie **Scrum**. Ce projet vise à développer un jeu mono-utilisateur avec une interface graphique, où le joueur dirige un personnage dans un labyrinthe à la recherche d'un trésor, tout en évitant les monstres de différents types.

Présentation du Projet :

Le projet a été structuré en plusieurs sprints, chacun focalisé sur l'ajout de nouvelles fonctionnalités et l'amélioration continue du jeu. Parmi les fonctionnalités clés, on retrouve la génération de labyrinthes, la présence de monstres, et différents éléments interactifs comme des trésors, des pièges, et des effets magiques. Chaque sprint débutait par la définition des fonctionnalités à intégrer et se concluait par la livraison d'une version fonctionnelle du jeu. L'élaboration des diagrammes UML d'analyse et de conception a joué un rôle crucial dans la planification et l'implémentation des différentes composantes du jeu.

Objectifs du Document de Conception :

Le document de conception vise à fournir une vue d'ensemble des fonctionnalités implémentées, ainsi qu'une représentation détaillée des structures sous-jacentes du logiciel à travers des diagrammes de classes et de séquences. Ce document sert également à illustrer l'organisation de notre projet, en détaillant la répartition des tâches et les responsabilités au sein de l'équipe pour chaque sprint. Notre objectif est de présenter un aperçu clair et structuré du développement du jeu, en mettant en lumière les décisions de conception et les stratégies d'implémentation adoptées tout au long du projet.

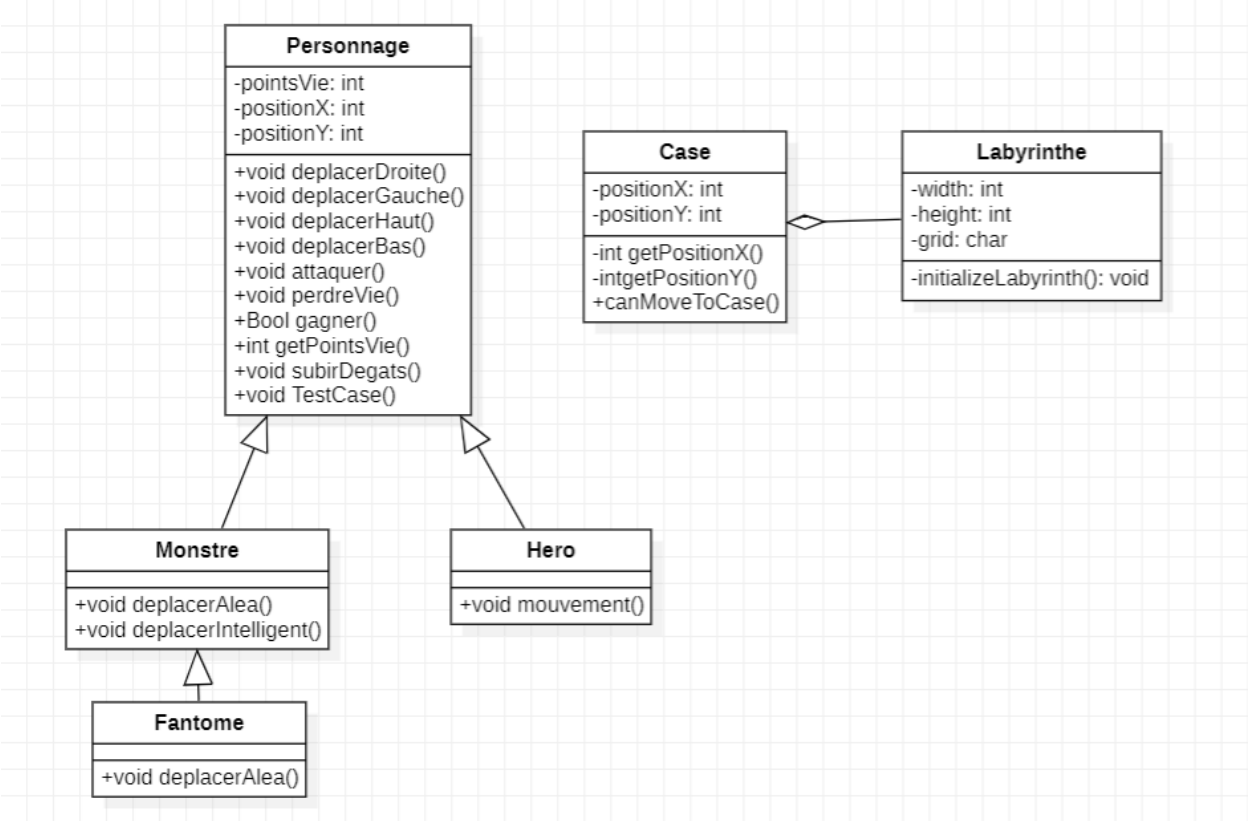
Transition vers la Mise en Œuvre : De la Conception à la Réalisation

Après avoir posé les bases de notre projet "Balade dans un labyrinthe" et défini les objectifs clairs de ce document de conception, nous nous tournons désormais vers la phase concrète de mise en œuvre. Dans cette section, nous allons explorer en détail le déroulement de nos sprints et une analyse des diagrammes UML – les diagrammes de classes et de séquences – qui ont joué un rôle essentiel dans la modélisation et la structuration de notre projet.

Sprint 1 :

- La mise en place les classes Personnage et Labyrinthe.
- la logique de déplacement et la génération des bordures du labyrinthe.
- Mettre en place la logique de déplacement du héros à l'intérieur du labyrinthe en version texte suite à une entrée par clavier `(Z , Q , S , D )`.
- Implémenter la génération du labyrinthe.

Diagramme de classe :



Sprint 2 :

- Génération aléatoire du labyrinthe, implémentation de la fonction **generateRandomly().**
- Créer des fonctions pour l'affichage graphique des éléments de jeu pour le héros et les cases**, repaint().** (Walid)
- Intégrer la construction automatique en utilisant **Maven** et la tester.(Walid)
- Implémenter l'algorithme du mouvement du joueur.() (Othmen)
- Ajouter l'affichage des points de vie de l'héro
- Ajouter les interactions entre le héros et les différentes cases du labyrinthe. (Moemen)
- Implémenter le mouvement du monstre aléatoire. (Khaoula)

Diagramme de classe :

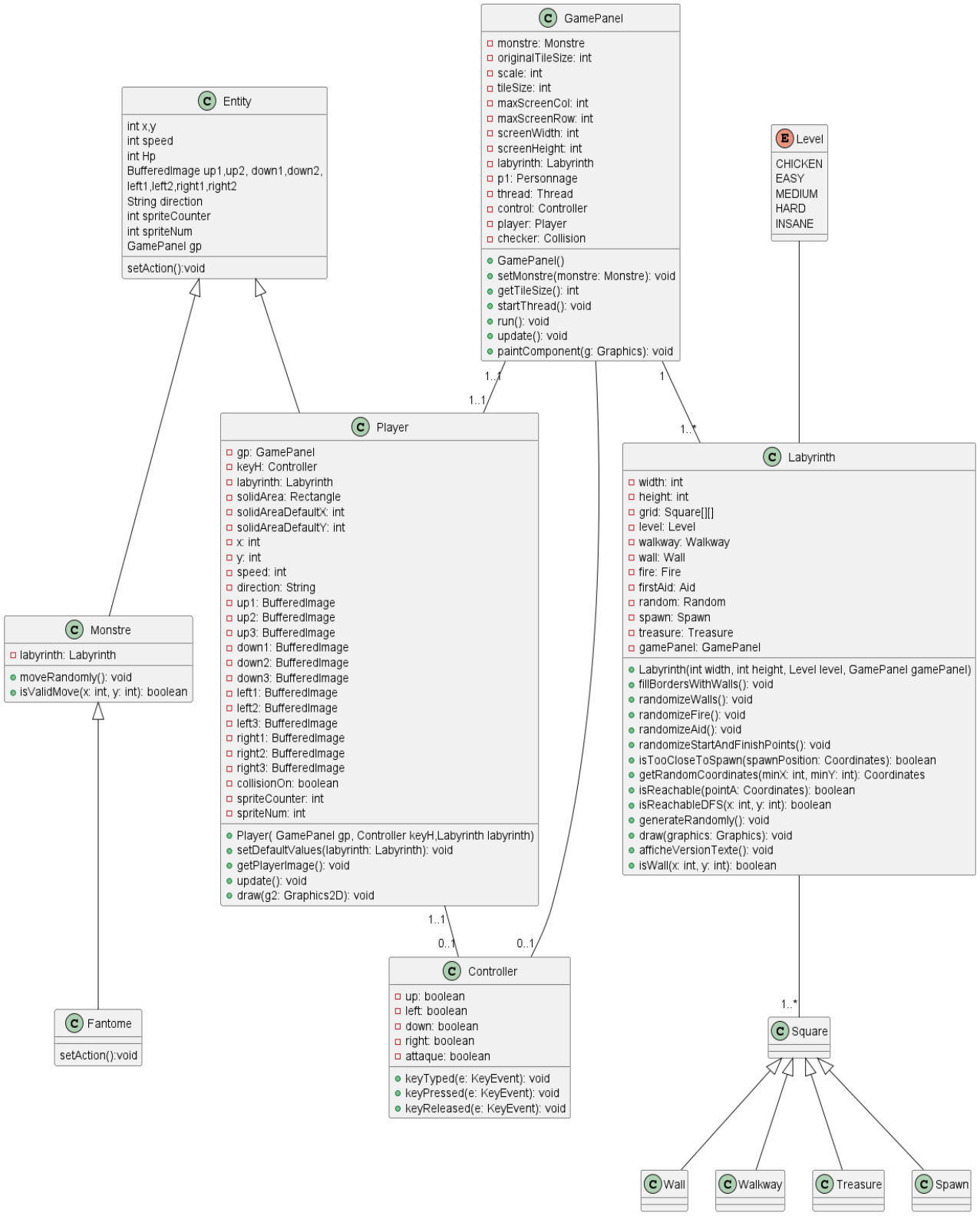
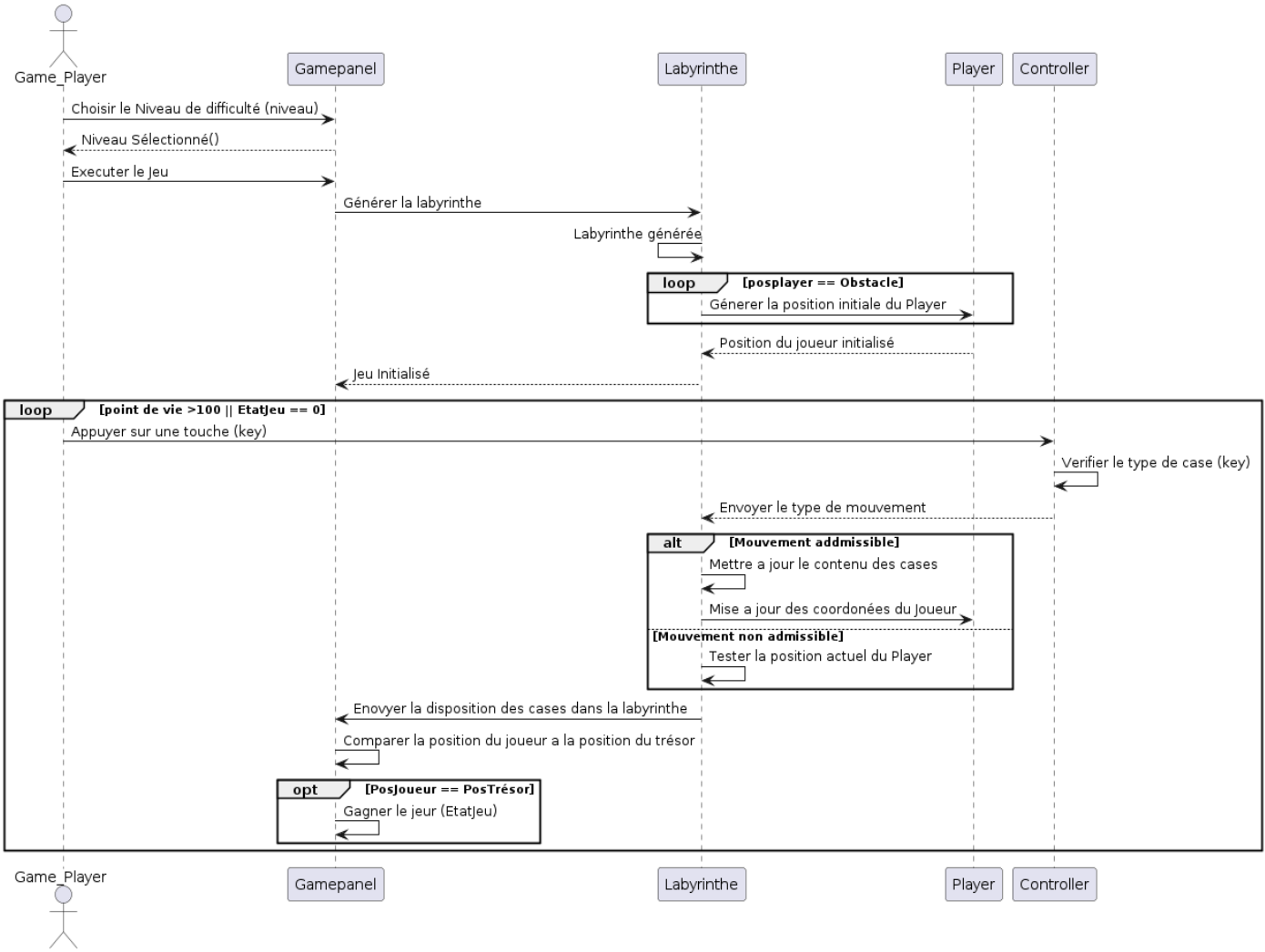


Diagramme de séquence générale du sprint 2 :

Le diagramme de séquence du sprint 2 capture les interactions essentielles entre le joueur, le contrôleur, le panneau de jeu et le labyrinthe pendant l'exécution des fonctionnalités.



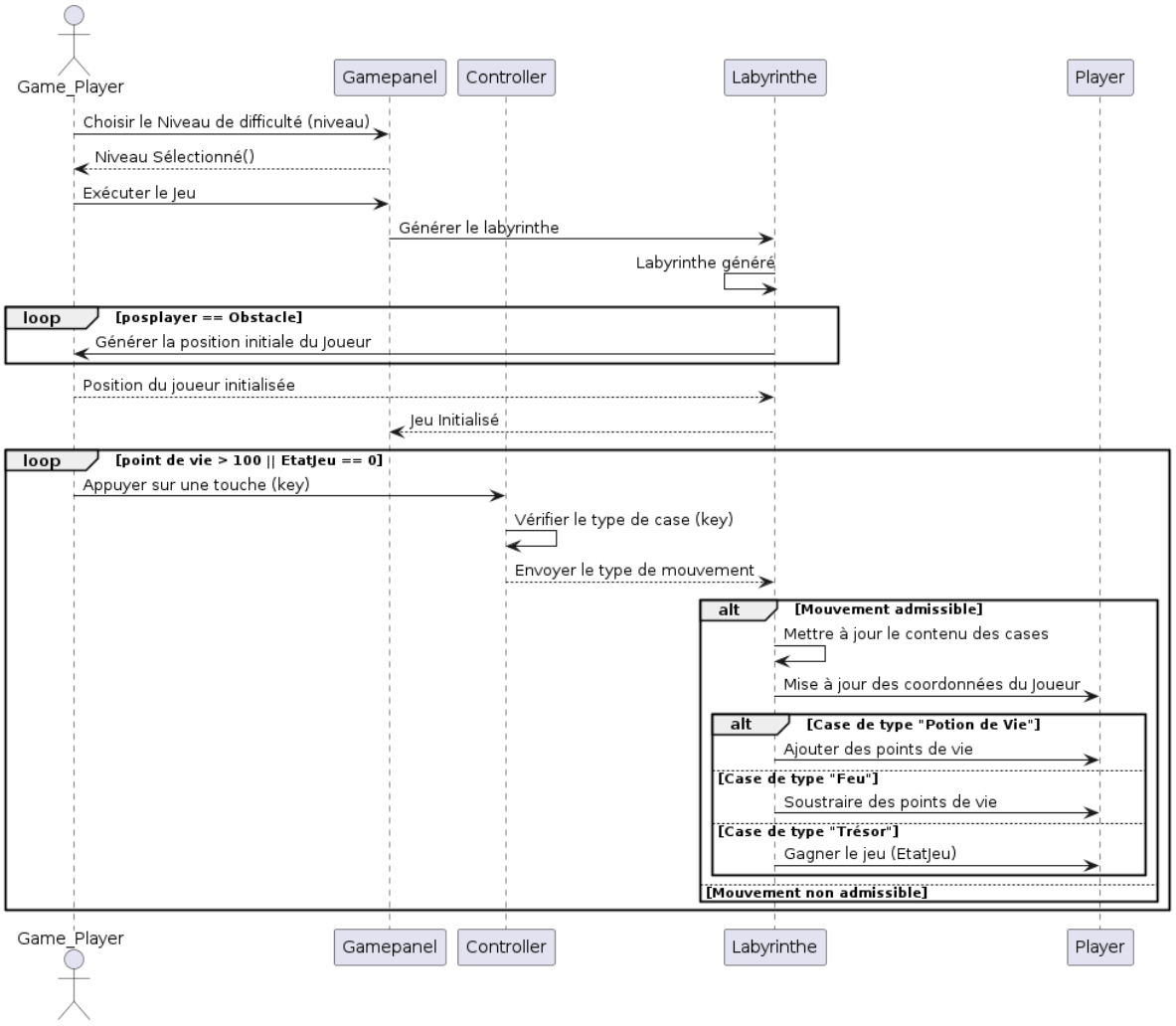
1. **Choix du Niveau de Difficulté :**
    - Le joueur (Gamer) interagit avec le panneau de jeu (Gamepanel) pour choisir le niveau de difficulté du jeu.
2. **Initialisation du Jeu :**
    - Le panneau de jeu (Gamepanel) communique avec le labyrinthe pour générer aléatoirement la disposition du labyrinthe.
    - Le joueur (Gamer) obtient sa position initiale dans le labyrinthe, évitant les obstacles si nécessaire.
3. **Boucle Principale du Jeu :**
    - Tant que le joueur a plus de 100 points de vie et n'a pas atteint la fin du jeu, la boucle principale du jeu est maintenue.
4. **Interaction du Joueur :**

- Le joueur (Gamer) appuie sur une touche, et le contrôleur (Controller) vérifie le type de mouvement associé à cette touche.

5. **Mouvement Admissible :**
   - Si le mouvement est admissible, le labyrinthe est mis à jour en conséquence, les coordonnées du joueur sont ajustées, et la disposition des cases est envoyée au panneau de jeu (Gamepanel).
6. **Vérification des Conditions de Fin du Jeu :**
   - Le panneau de jeu (Gamepanel) compare la position du joueur à la position du trésor.
   - Si les positions coïncident, le joueur gagne le jeu.
7. **Fin de la Boucle Principale :**
   - La boucle principale continue jusqu'à ce que le joueur n'ait plus de points de vie ou atteigne la fin du jeu.

Diagramme de séquence jeu détaillé :

Le diagramme de séquence jeu détaillé du sprint 2 apporte des ajouts significatifs, notamment la gestion détaillée des interactions avec les cases du labyrinthe, et des conditions spécifiques de fin du jeu.



1. **Génération de Position Initiale du Joueur :**
    - Dans cette version détaillée, le diagramme spécifie explicitement la génération de la position initiale du joueur par le labyrinthe, en évitant les obstacles.
2. **Gestion des Cases Spécifiques :**
    - Le diagramme détaille la gestion spécifique des différentes cases du labyrinthe en fonction du type de mouvement effectué par le joueur. Il inclut la mise à jour des coordonnées du joueur, l'ajout/soustraction de points de vie en fonction du type de case (Potion de Vie, Feu), et la condition de gain du jeu en atteignant un trésor.
3. **Conditions de Fin du Jeu :**
    - La section sur les conditions de fin du jeu est plus détaillée. Elle montre comment les actions du joueur influent sur l'état du jeu, déclenchant la victoire (en atteignant un trésor) ou la défaite (en perdant tous les points de vie).

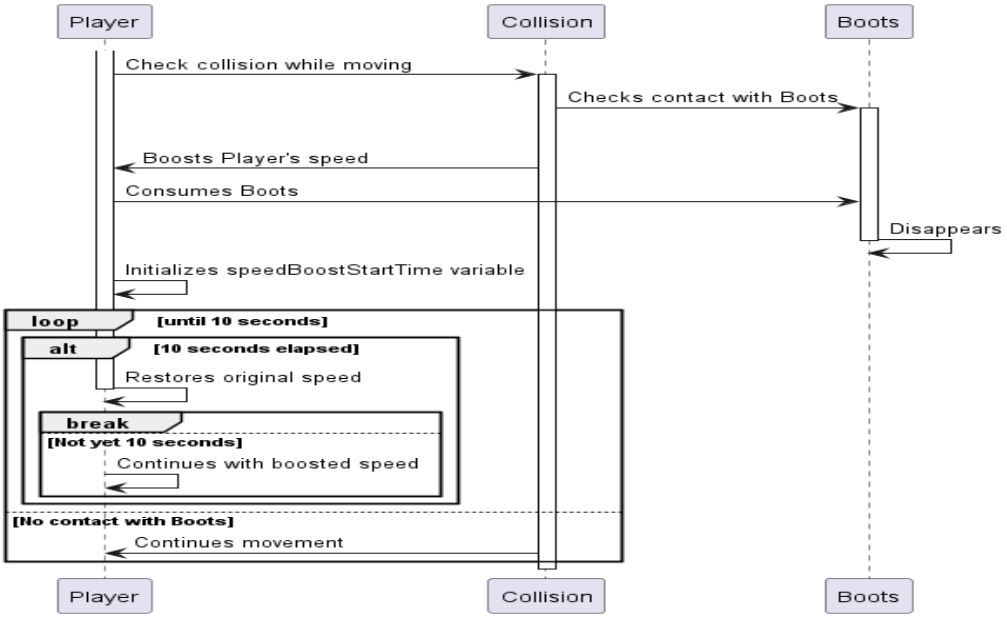
Sprint 3 :

- Implémentation des dégâts sur le personnage lorsqu'il est en contact avec le monstre.
- Implémentation de l'attaque rapprochée du héros contre le monstre.
- Implémentation de l'attaque magique (émission de la boule de feu).
- Amélioration du système de transition entre les niveaux.
- Intégration d'effets sonores lors du passage par des cases spéciales : feu, potion.
- Intégration d'une musique d'ambiance.
- Génération des monstres en fonction du niveau de difficulté.
- Implémentation de la classe *user interface* assurant les fonctionnalités : Retry, Play, Again, Pause.
- Implémentation de tests unitaires pour la classe Labyrinth.
- Implémentation de l'attaque à distance (avec une hache).
- Interaction entre le monstre et les attaques magiques.
- Interaction entre le monstre et les attaques à distance.
- Implémentation du pouvoir d'accélération en prenant les *Boots* pendant une durée limitée.
- Interaction entre le monstre et les attaques à distance (Projectiles). *(Othemen)*
- Ajout du monstre fantôme. *(Walid)*

Diagramme Accélération Héro :

Il illustre la manière dont une interaction spécifique (en l'occurrence, la collision avec les bottes) influence le comportement du joueur dans le jeu. Il s'agit d'une fonctionnalité spécifique introduite dans le sprint 3, pour offrir au joueur des éléments de gameplay tels que des bonus temporaires de vitesse. Ce diagramme aide

à comprendre comment ces interactions sont gérées et comment elles influent sur le déroulement du jeu dans le contexte du sprint 3.

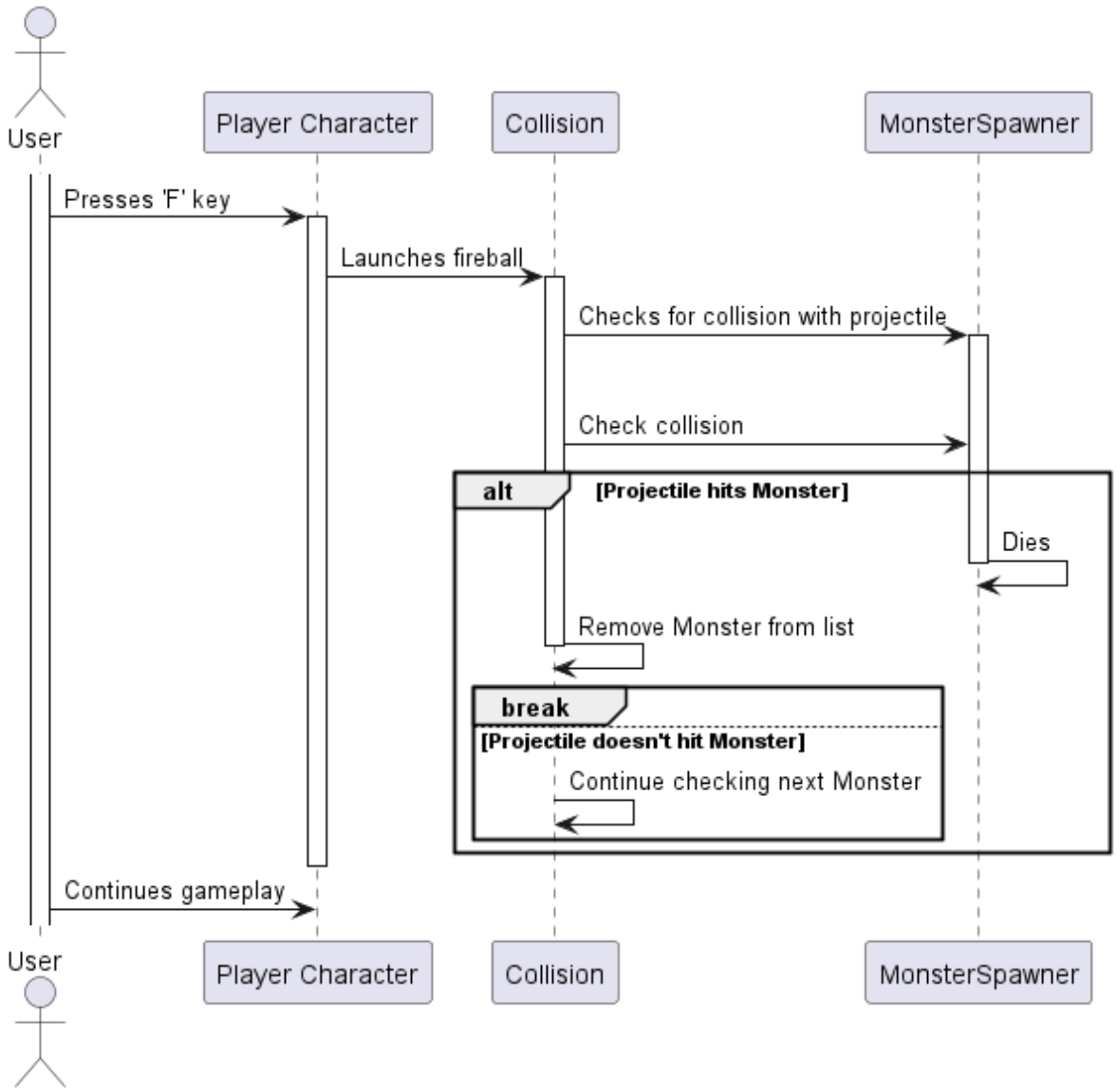


1. **Collision et Interaction avec les Bottes :**
    - Lorsque le joueur (P) se déplace, le système vérifie les collisions avec d'autres éléments, dont les bottes (B).
2. **Vérification de la Présence des Bottes :**
    - Le composant de collision (C) vérifie si le joueur est en contact avec les bottes.
3. **Gestion de l'Interaction :**
    - Si le joueur entre en contact avec les bottes, plusieurs actions sont déclenchées :
        - Le joueur bénéficie d'une vitesse accrue.
        - Les bottes disparaissent, indiquant qu'elles ont été consommées.
        - Le temps de début du bonus de vitesse est enregistré.
4. **Gestion de la Durée du Bonus de Vitesse :**
    - Un boucle est initiée, simulant la durée du bonus de vitesse (dans cet exemple, 10 secondes). Pendant cette période, le joueur continue à se déplacer à une vitesse accrue.
    - Si les 10 secondes s'écoulent, la vitesse du joueur est restaurée à sa valeur d'origine, et la séquence est désactivée.
    - Sinon, le joueur continue à se déplacer avec une vitesse accrue.
5. **Fin de l'Interaction :**

- Si le joueur n'est pas en contact avec les bottes, la séquence se termine sans effet.

Diagramme Attaque magique :

Ce diagramme illustre le processus d'attaque à distance par le joueur à l'aide d'une boule de feu dans le cadre du sprint 3

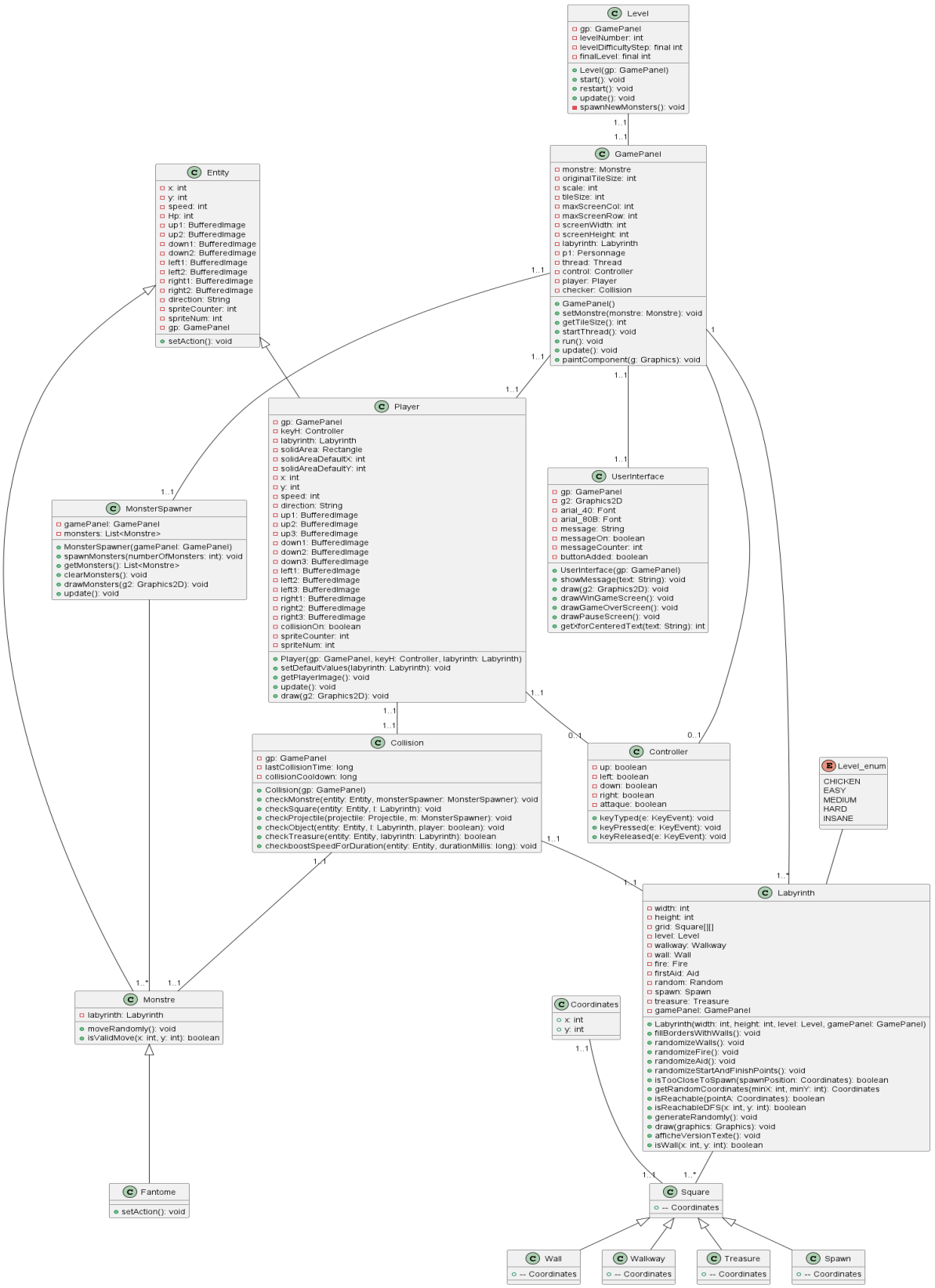


1. **Activation de l'Action du Joueur :**
   - L'utilisateur (représenté par U) presse la touche 'F' pour initier l'action de lancer une boule de feu.
2. **Vérification de Collision avec un Projectile :**
   - Le joueur (P) active la séquence d'attaque, gérée par le composant Collision (C), qui vérifie la collision entre la boule de feu et les monstres générés par le MonsterSpawner (M).
3. **Itération sur les Monstres :**

    - La séquence itère sur chaque monstre présent dans la zone du jeu.
4. **Vérification de Collision avec un Monstre :**
    - Pour chaque monstre, la collision avec la boule de feu est vérifiée.
5. **Réaction en Cas de Collision avec un Monstre :**
    - Si la boule de feu frappe un monstre, le monstre meurt, est retiré de la liste, et la séquence se termine.
6. **Continuation de l'Interaction :**
    - Après les vérifications de collision, la séquence se termine, et le joueur peut continuer à jouer.

Ce mécanisme permet au joueur d'attaquer les monstres à distance en lançant des boules de feu, introduisant ainsi une dimension stratégique et offrant une nouvelle approche tactique dans la progression du jeu. La séquence se termine de manière transparente, permettant au joueur de reprendre le gameplay sans interruption majeure.

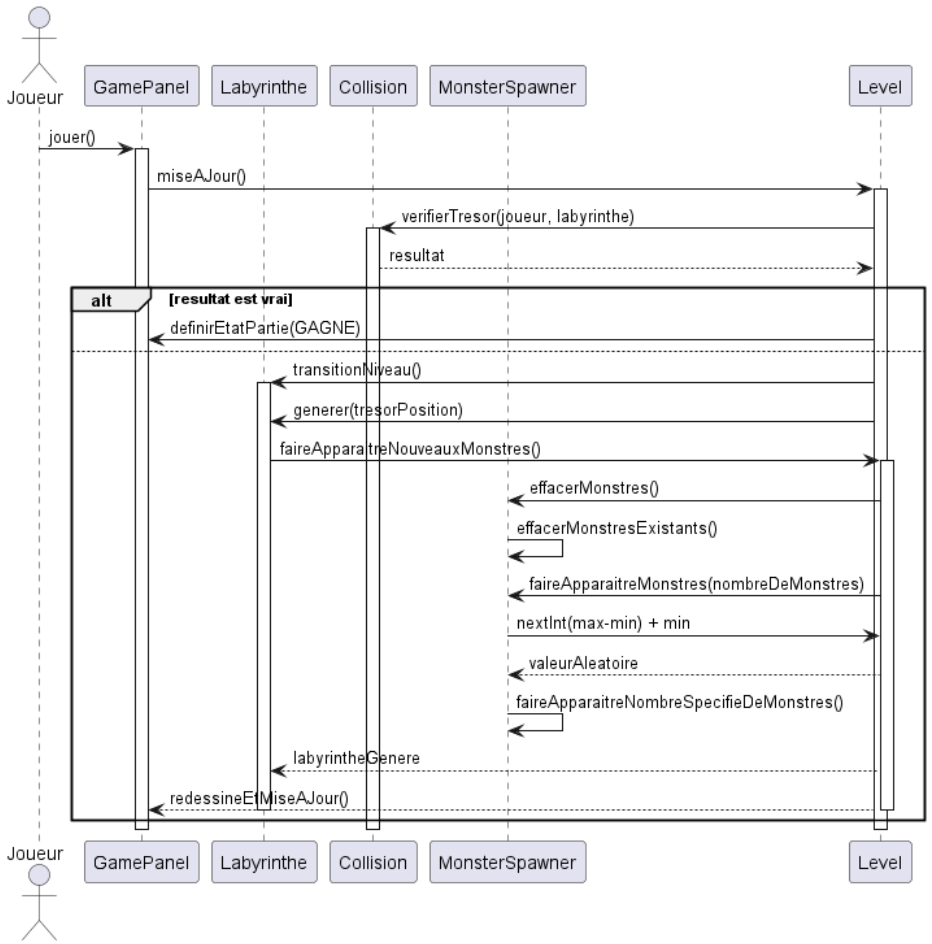
Diagramme de classe :



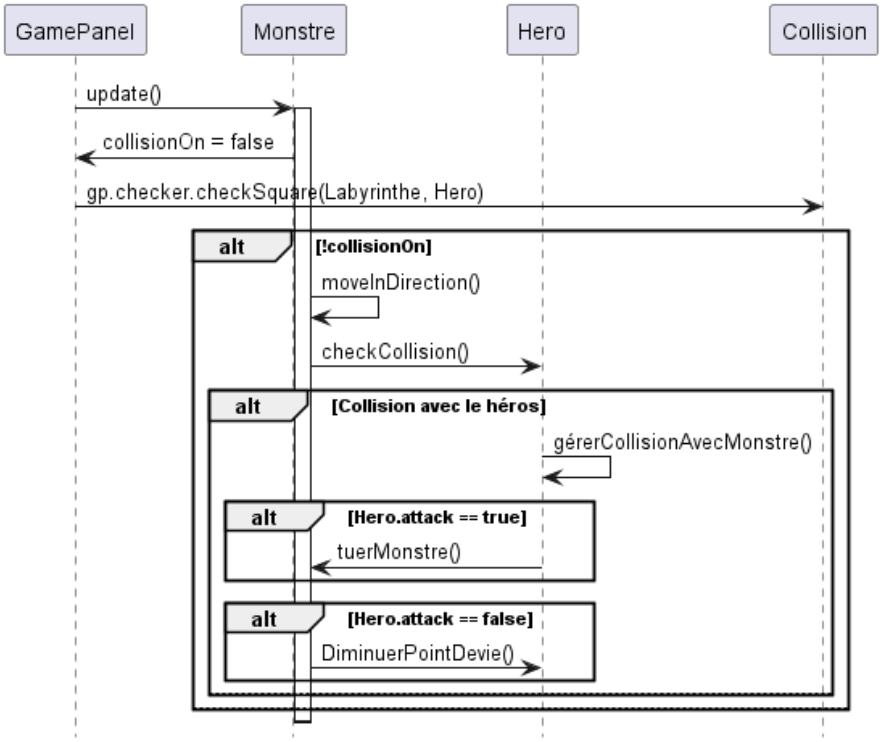
Sprint 4 :

- Implémentation des attaques contre le monstre fantôme. *(Walid)*
- Implémentation de la capacité magique du joueur (Mana : système de points magiques). *(Othemen)*
- Sélection du joueur au début du jeu. *(Khaoula)*
- Implémentation d'un système de calcul du score. *(Khaoula)*
- Implémentation du menu au début du jeu (Start Menu). *(Moemen)*
- Implémentation de tests unitaires pour la classe *Monster*. *(Moemen)*
- Implémentation de tests unitaires pour la classe *Player*. *(Walid)*

- **Diagramme de séquence de l'avancement de niveau:**



- **Diagramme de séquence de l’interaction hero et Monstre**



Organisation du Travail et Gestion de Projet :

Réunions :

Des réunions régulière ont été organisées pour chaque sprint pour discuter de l'avancement du projet, résoudre les problèmes , planifier les fonctionnalités pour les sprints à venir et pour maintenir une communication fluide au sein de l'équipe et un suivi régulier.

Outils Utilisés :

Les réunions se déroulaient virtuellement via Discord ou Microsoft Teams.

Git a joué un rôle primordial pour le partage de documents, la communication asynchrone, et le suivi des modifications du code source.

Division des Tâches par Sprint :

À chaque sprint, les tâches ont été divisées en fonction des fonctionnalités spécifiques à développer :

- **Définition des Fonctionnalités :** Au début de chaque sprint, nous identifions les fonctionnalités clés à développer ou améliorer.
- **Attribution des Tâches :** Chaque fonctionnalité est décomposée en tâches distinctes, attribuées aux membres de l'équipe .
- **Suivi Continu :** L'avancement des tâches est régulièrement suivi pour assurer une progression alignée avec les objectifs du sprint.

Cette méthode a permis une gestion efficace des ressources et une progression cohérente vers les buts du projet.