SOMMAIRE

INTRODUCTION

Ce rapport vise à présenter en détail les aspects clés de notre jeu d'aventure *Journey to the Lost Kingdom*, de la phase d'analyse la première phase de conception. Nous y présenterons de nombreux diagrammes accompagnés d'explications comme demandé dans le sujet.

Le cahier des charges définit les attentes en termes de gameplay, de graphismes, de monde du jeu et d'interface utilisateur, ainsi que les exigences techniques en termes de langage de programmation et autres.

Le document d'analyse fournit une vision des cas d'utilisation et des diagrammes de séquence système, offrant une compréhension claire des interactions entre les différents composants du jeu.

De même, le document de conception présente l'architecture logique du jeu, la description de l'incrément choisi et la conception détaillée à travers des diagrammes de classes logicielles et des diagrammes de séquence système.

Le manuel utilisateur a été élaboré pour guider les joueurs à travers le processus de compilation, d'exécution et de jeu du *Journey to the Lost Kingdom*, garantissant une expérience sans heurts et gratifiante.

Le bilan sur les outils de modélisation permet de comprendre comment les diagrammes UML ont été réalisés.

I. CAHIER DES CHARGES

Au vu des spécifications et exigences données dans la description du projet, nous avons choisi d'implémenter **Journey to the Lost Kingdom**, un jeu très passionnant qui se présente dans les sections suivantes.

1. Spécifications du jeu

1.1. Gameplay

Le jeu sera un jeu d'action-aventure avec une vue de dessus, similaire à celui du premier Zelda. Les joueurs contrôlent un personnage principal dans un monde ouvert, explorant des donjons composés de salle, combattant des ennemis et résolvant des énigmes. Des éléments de RPG seront intégrés, permettant aux joueurs d'améliorer les capacités de leur personnage et de collecter des objets. Les joueurs devront collecter des objets clés pour progresser dans le jeu, déverrouillant de nouvelles zones et des capacités spéciales. Le jeu offrira une variété d'armes et d'objets que le joueur peut utiliser pour combattre les ennemis et résoudre des énigmes. L'objectif final est de trouver la *couronne sacrée.*

1.2. Graphismes et audio

Les graphismes seront dans un style rétro (style 8 bit). Une bande sonore sera implémentée pour accompagner l'action du jeu et renforcer l'ambiance.

1.3. Monde du jeu

Le jeu se déroule dans un vaste monde ouvert composé de divers environnements tels que des forêts, des montagnes, des déserts et des donjons. Des mécanismes de voyage rapide seront disponibles pour permettre aux joueurs de se déplacer rapidement entre les zones déjà explorées.
Le monde sera rempli de quêtes, de PNJ (personnages non-joueurs) et d'ennemis variés pour maintenir l'intérêt des joueurs. Il sera possible d'échanger et de vendre des objets avec des PNJ marchands. Des secrets et des trésors seront disséminés dans le monde du jeu pour encourager l'exploration.

1.4. Interface utilisateur

L'interface utilisateur sera conviviale et intuitive, fournissant des informations claires sur la santé du personnage, l'inventaire et les objectifs de quête. Un menu d'inventaire intuitif permettra aux joueurs de gérer leurs objets, leurs armes. Un mini-carte sera présent pour permettre au joueur de se repérer dans son environnement. Cette carte mettra en valeur les points d'intérêt (Objectif de quête, marchand, donjon, …). Des mécanismes de sauvegarde seront inclus pour conserver sa progression.

2. Exigences techniques

2.1. Plateformes

Notre jeu sera développé pour être joué sur PC en particulier sur les PC de l'Ensimag.

2.2. Moteur de jeu

Le jeu sera développé à l'aide de la bibliothèque graphique Swing (et AWT)

2.3. Langage de programmation

Journey to the Lost Kingdom sera développé en langage Java.

3. Livrables

Pendant la phase de développement du jeu:

- Un incrément jouable sera à présenter.
- Un manuel utilisateur correspondant à l'incrément.

À la fin du développement, le livrable comprendra :

- Le jeu complet et fonctionnel, prêt à être distribué.
- Le manuel d'utilisation.
- Tous les actifs de jeu, y compris les graphismes, la musique et les effets sonores.

4. Conclusion

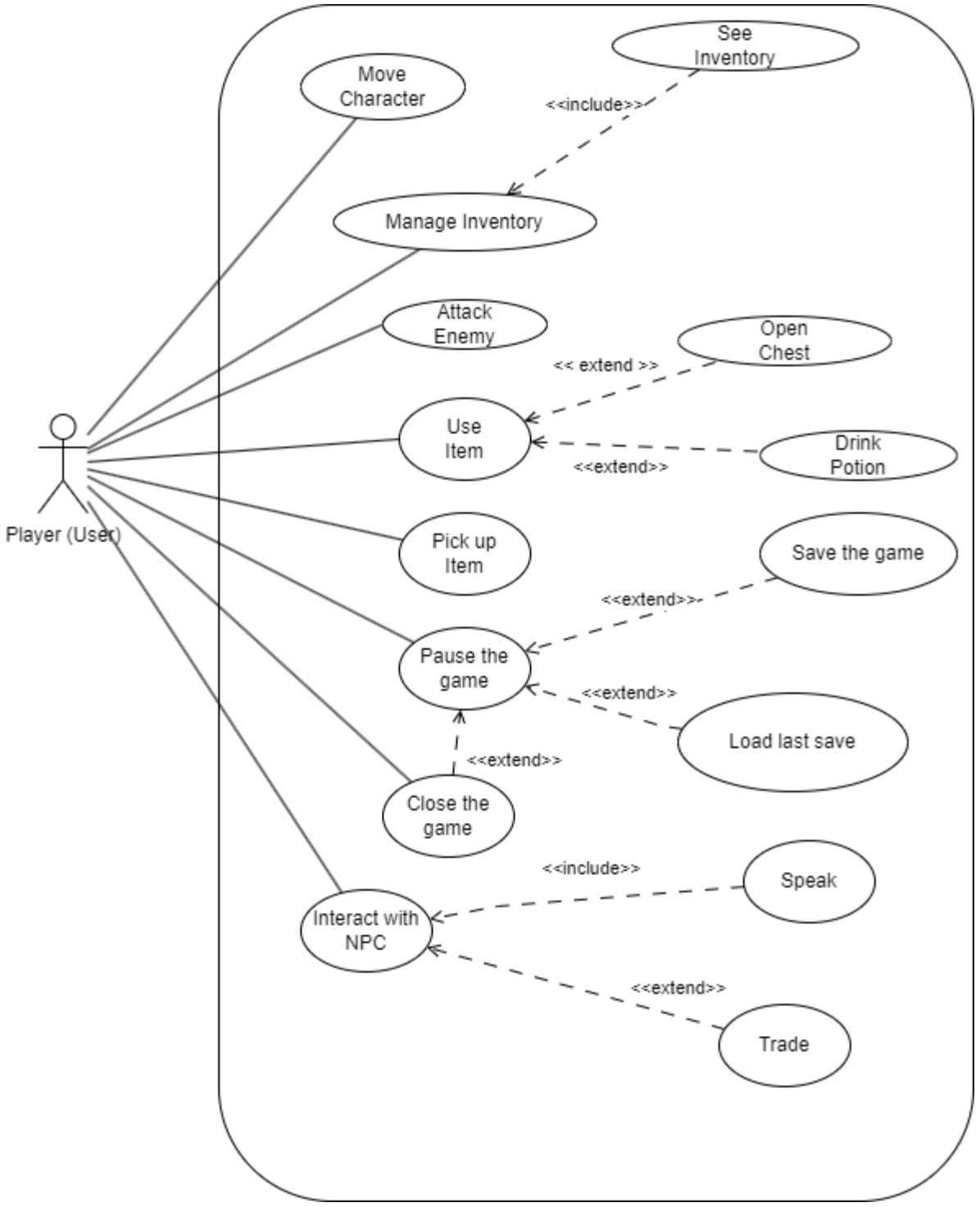
Ce cahier des charges définit les spécifications et les exigences pour le développement du jeu ***Journey to the Lost Kingdom***.

II. DOCUMENT D'ANALYSE

1. Diagramme de cas d'utilisation

1.1. Diagramme de cas d'utilisation

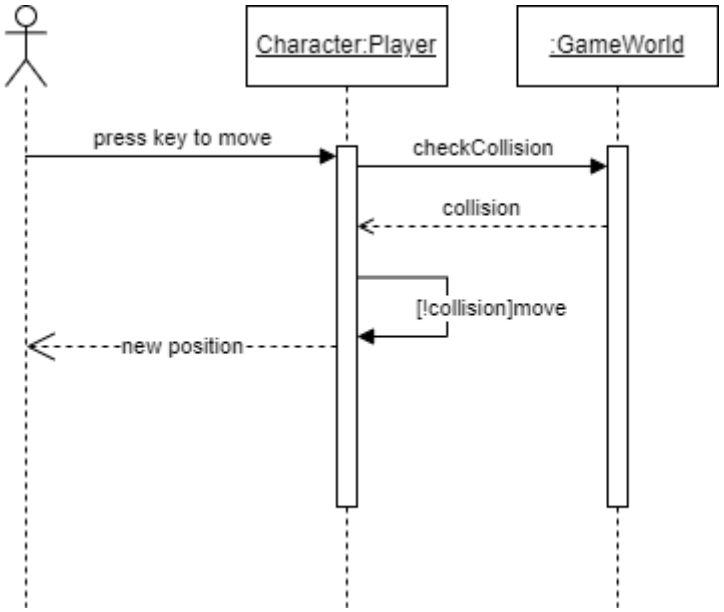
Nous avons élaboré le diagramme de cas d'utilisation suivant pour notre système de jeu.



1.2. Diagrammes de séquence système

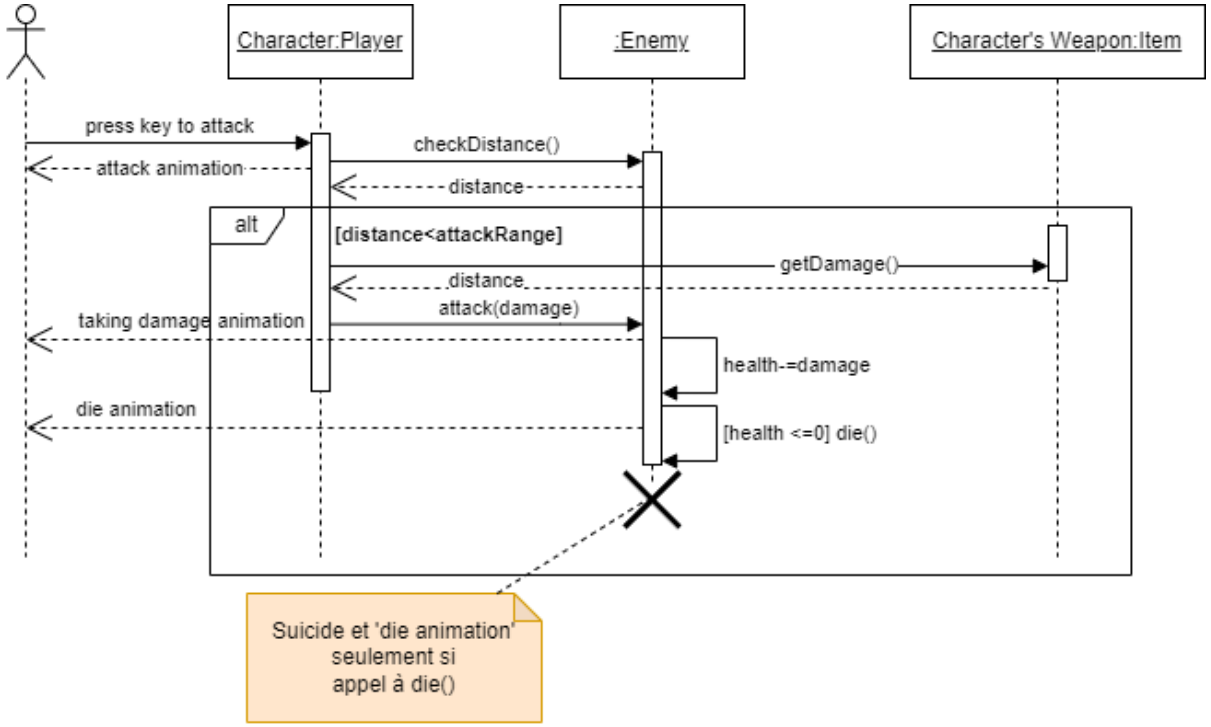
Nous présentons ici les diagrammes de séquence système de notre jeu. Chaque diagramme de séquence est accompagné d'une explication.

1.2.1. Diagramme de séquence du cas d'utilisation Move Character



Pour déplacer le personnage l'utilisateur appuie sur une touche du clavier, cette touche est transmise au personnage, celui-ci vérifie s' il y a collision avec l'environnement du jeu. Si il n'y a pas de collision, le personnage se déplace. La position du personnage est ensuite envoyée à l'utilisateur.

1.2.2. Diagramme de séquence du cas d'utilisation Attack Enemy



Ce diagramme représente le scénario où le personnage attaque un ennemi. Pour ce faire l'utilisateur presse une touche, que l'attaque réussisse ou non l'animation d'attaque est affichée (éventuel coup d'épée dans le vide). Le Personnage vérifie la distance avec l'ennemie (checkDistance), si cette distance est inférieur au rayon d'attaque du personnage:

Le personnage récupère le la valeur des dégâts causée par son arme(épée ou autre) puis attaque l'ennemi. Celui ci perd les PV correspondant (l'animation de dégât reçu est affichée sur l'ennemie), si c'est PV atteigne zéro l'ennemie 'meurt' (et l'animation de mort est affiché sur l'ennemie)

1.2.3. Diagramme de séquence du cas d'utilisation Pick up Item

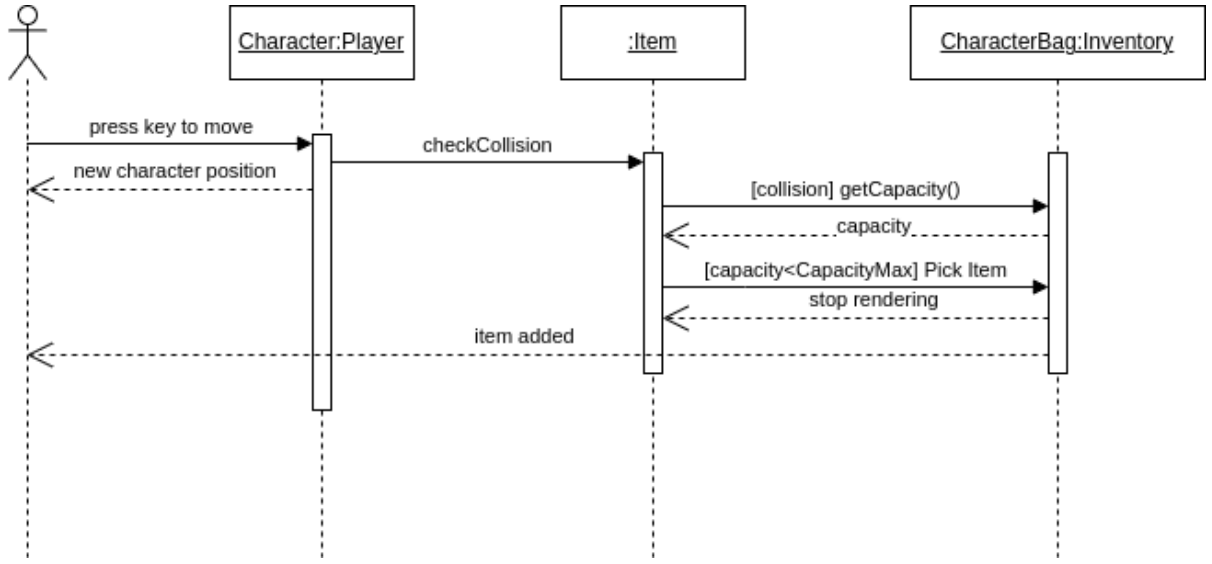
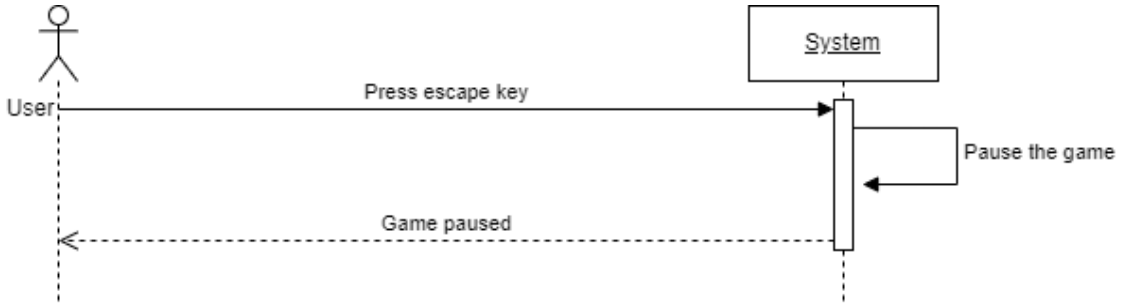


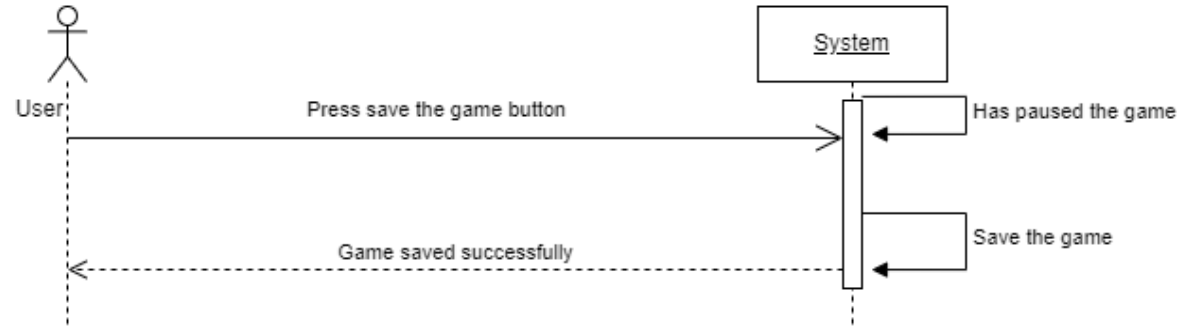
Diagramme décrivant l'action de ramasser un item, si le personnage entre en collision avec l'item, il l'ajoute à son inventaire (si il y a de la place) et arrête de le l'afficher.

1.2.4. Diagramme de séquence du cas d'utilisation Pause the game



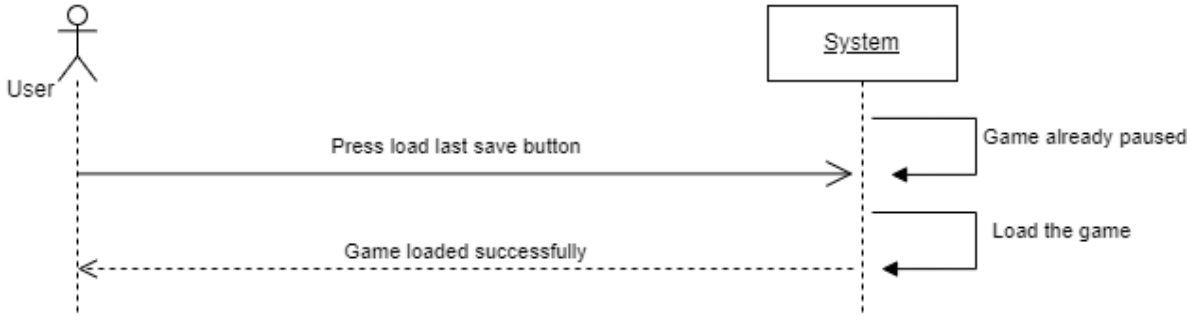
Lorsque l'utilisateur appuie sur echap le jeu se met en pause.

1.2.5. Diagramme de séquence du cas d'utilisation Save the game



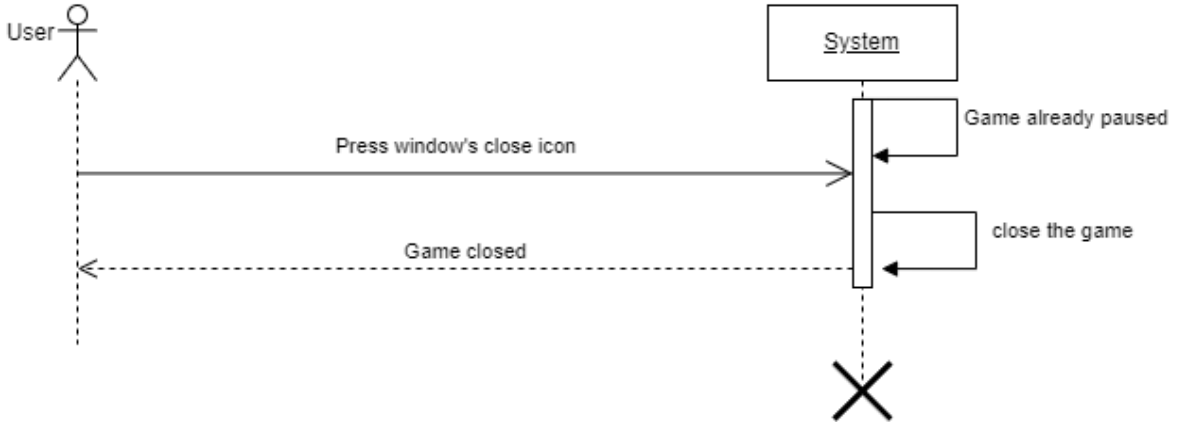
Lorsque le jeu est en pause l'utilisateur peut sauvegarder la partie.

1.2.6. Diagramme de séquence du cas d'utilisation Load the game



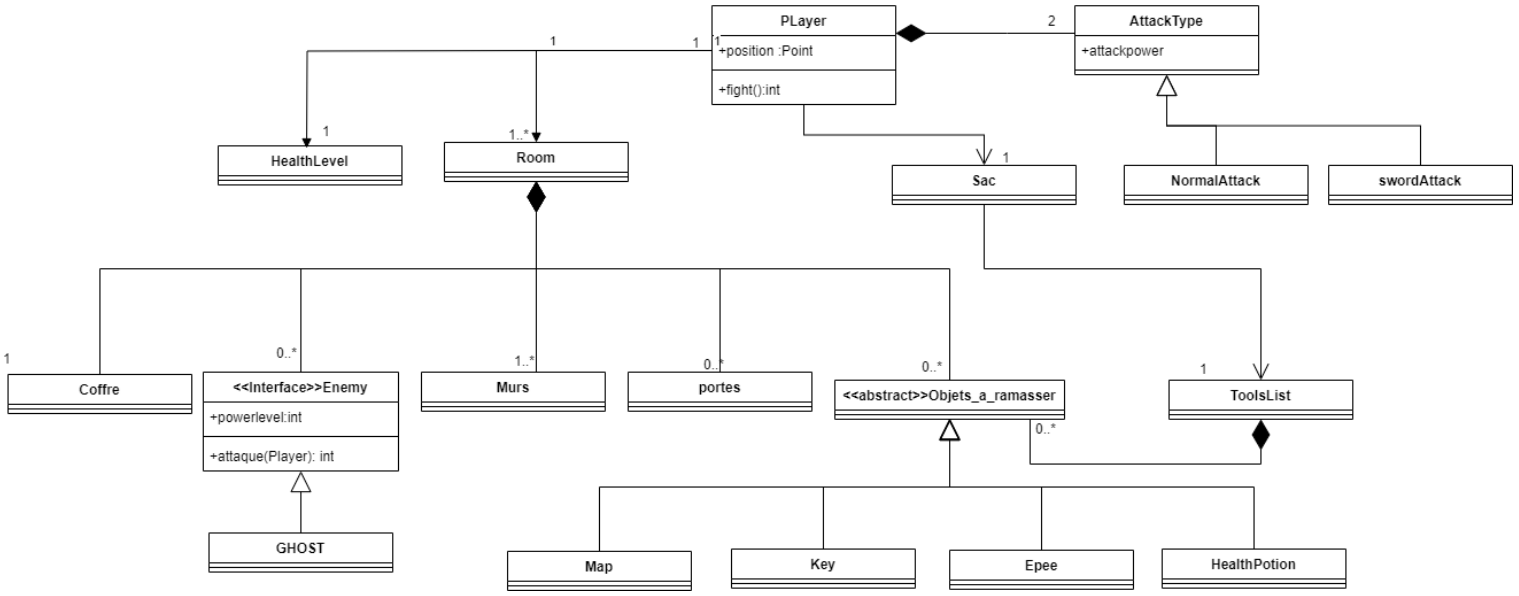
Lorsque le jeu est en pause l'utilisateur peut charger la dernière partie sauvegarder.

1.2.7. Diagramme de séquence du cas d'utilisation Close the game



Lorsque le jeu est en pause l'utilisateur peut fermer le jeu comme on ferme tous les fenêtres.

2. Diagramme de classe d’analyse



Au début de la phase d'analyse, il est courant de concevoir un diagramme de classe conceptuel pour identifier les principaux composants de votre programme de jeu. Ce diagramme offre une première vue d'ensemble des éléments clés et de leurs relations initiales. Il met en évidence les entités fondamentales telles que les salles où le jeu démarre, les caractéristiques essentielles des joueurs, comme leur sac et leur santé, ainsi que les interactions préliminaires entre ces composants.
Parmi les classes les plus importantes on retrouve par exemple :

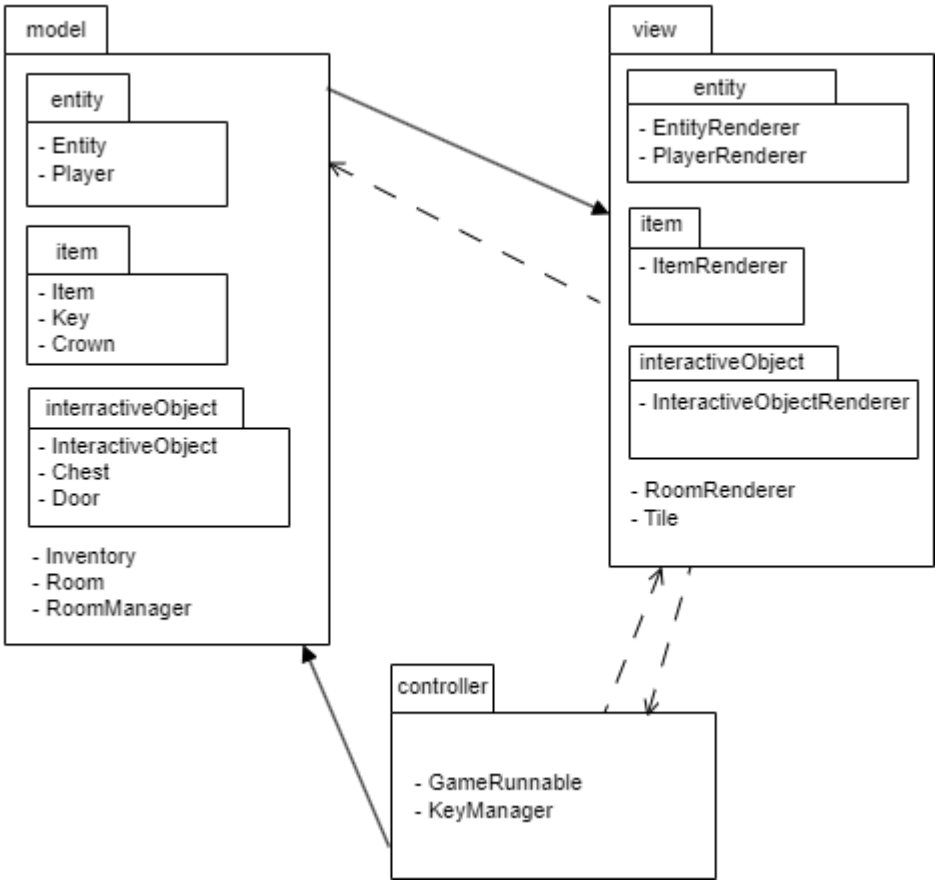
Room (Salle) : Représente une salle dans le jeu où l'aventure commence. Cette classe peut comprendre des attributs décrivant la salle, comme son nom ou sa description, ainsi que des méthodes pour interagir avec celle-ci, comme entrer ou quitter la salle.

Player (Joueur) : Représente le personnage principal du jeu. Cette classe peut inclure des attributs tels que le sac à dos du joueur pour stocker des objets trouvés dans le jeu, ainsi que sa santé pour suivre son état de vie. Des méthodes pourraient être définies pour gérer les actions du joueur, telles que ramasser un objet, utiliser un objet du sac ou subir des dégâts.

III. DOCUMENT DE CONCEPTION

1. Architecture Logique du Jeu

L'architecture choisie est celle du ***M**odel **V**iew **C**ontroller* (**MVC**). Le code est donc découpé en 3 paquetages principaux : model, view et controller.



Dans le paquetage ***model*** on retrouve les classes principales du projet regrouper en différent sous-paquetage. Dans le paquetage ***view*** on retrouve les classes *Renderer* qui réalise le rendu graphique des différentes classes de ***model*** associé (chargement de la texture selon le statut, sélection du *sprite*,...). Dans le paquetage ***controller*** on retrouve le class *KeyManager* qui récupère les entrées clavier passer par le joueur. La classe *GameRunnable* informe aux modèles de mettre à jour leurs statuts à intervalle régulier.
Il existe aussi un paquetage ***main***,non représenté ici, qui contient simplement l'exécutable main.

2. Description de l'incrément choisi

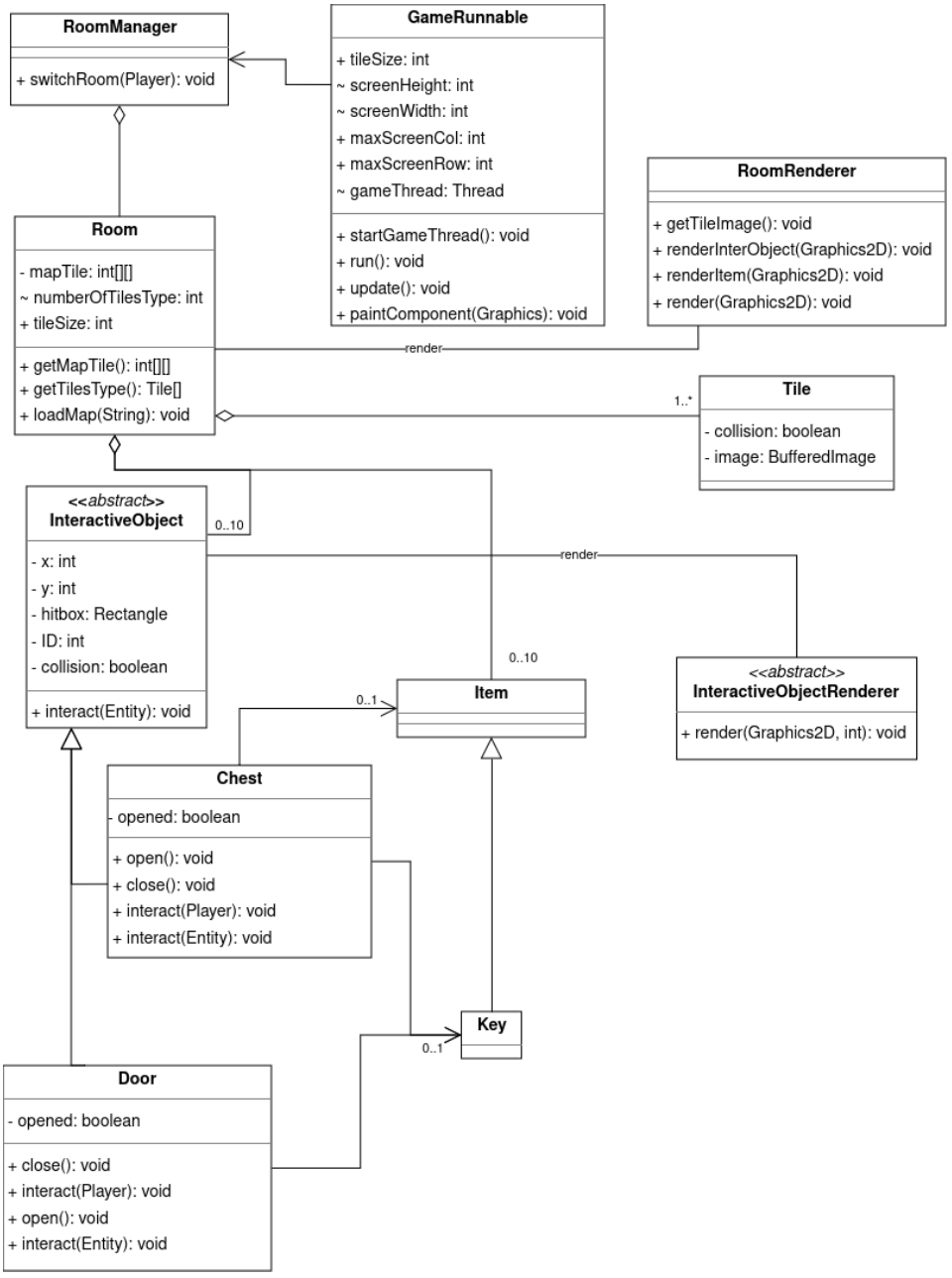
Ce livrable contient le premier incrément du jeu, seules certaines fonctionnalités primaires ont été implémentées mais le jeu dispose déjà d'une interface graphique.
Il est déjà possible de contrôler le personnage jouable à travers la carte qui est seulement composée de deux salles. Il est possible de ramasser des items et d'interagir avec deux types d'objets (les coffres et les portes). Seuls deux items sont implémentés (la clé et la couronne). Le passage d'une salle à l'autre se fait de façon sommaire.

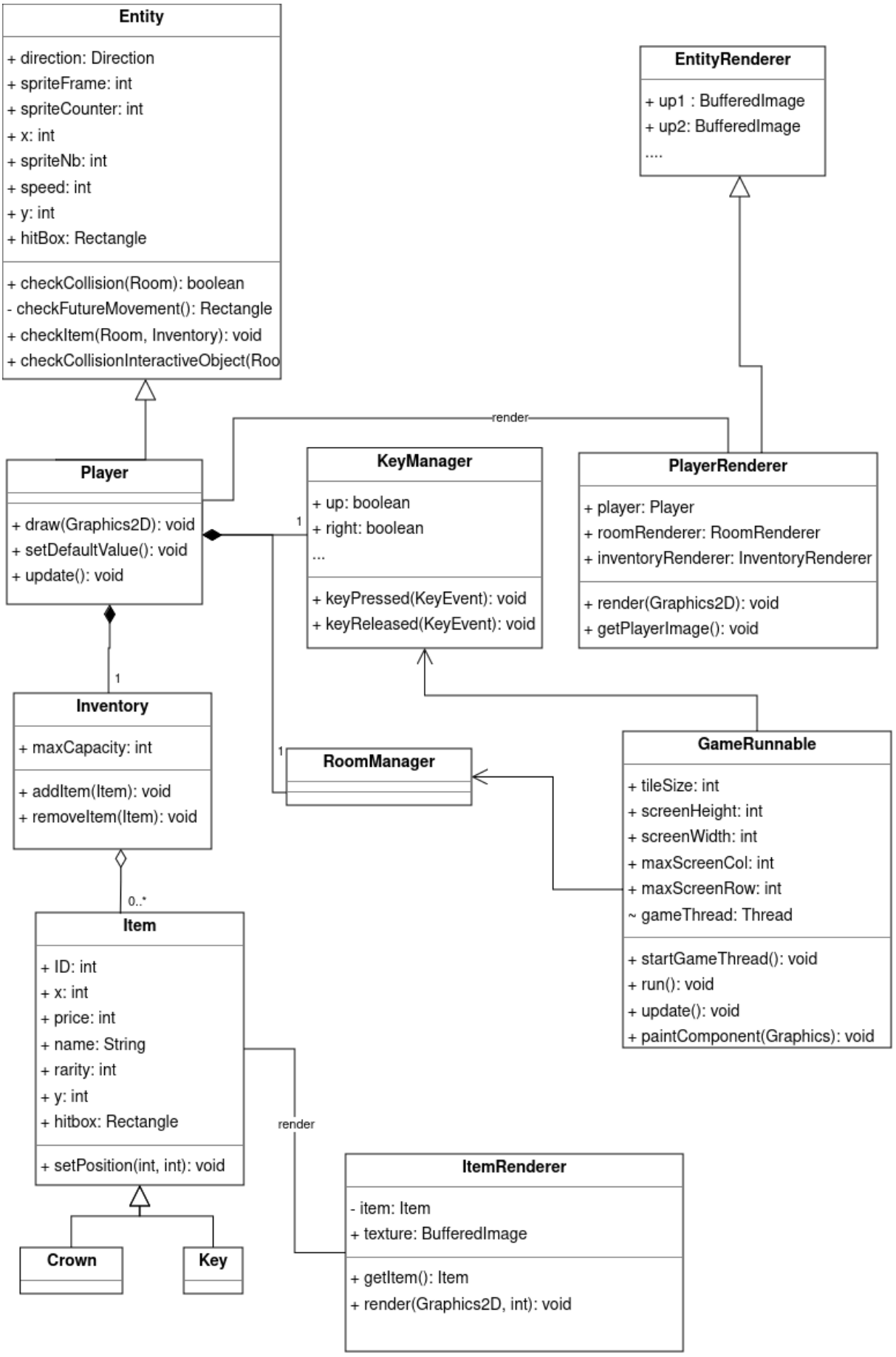
Il n’y a pas de gestion de la “caméra”, ainsi les salles ne peuvent être plus grandes que la taille de la fenêtre au risque que le personnage sorte de la fenêtre.
Il n’y a pas encore d’écran titre, d’écran de pose ni d’HUD hormis une vision sommaire de l’inventaire.

3. Conception détaillée

3.1. Diagramme de classes logicielles

Une analyse profonde de notre incrément choisi nous a permis d’élaborer le diagramme de classe logicielle suivante. Le diagramme étant trop important pour tenir sur une seule page, nous avons choisi de le séparer sur deux pages par souci de lisibilité. Ainsi plusieurs classes peuvent apparaître sur les deux parties du diagramme pour en faciliter la lecture. Seules les méthodes et attributs pertinents sont présents dans les classes. Les multiplicités de 1 ne sont pas forcément représentées.




Entity
+ direction: Direction
+ spriteFrame: int
+ spriteCounter: int
+ x: int
+ spriteNb: int
+ speed: int
+ y: int
+ hitBox: Rectangle
+ checkCollision(Room): boolean
- checkFutureMovement(): Rectangle
+ checkItem(Room, Inventory): void
+ checkCollisionInteractiveObject(Roo
EntityRenderer
+ up1 : BufferedImage
+ up2: BufferedImage
....
render
Player
+ draw(Graphics2D): void
+ setDefaultValue(): void
+ update(): void
KeyManager
+ up: boolean
+ right: boolean
...
+ keyPressed(KeyEvent): void
+ keyReleased(KeyEvent): void
PlayerRenderer
+ player: Player
+ roomRenderer: RoomRenderer
+ inventoryRenderer: InventoryRenderer
+ render(Graphics2D): void
+ getPlayerImage(): void
1
1
Inventory
+ maxCapacity: int
+ addItem(Item): void
+ removeItem(Item): void
1
RoomManager
GameRunnable
+ tileSize: int
+ screenHeight: int
+ screenWidth: int
+ maxScreenCol: int
+ maxScreenRow: int
~ gameThread: Thread
+ startGameThread(): void
+ run(): void
+ update(): void
+ paintComponent(Graphics): void
0..*
Item
+ ID: int
+ x: int
+ price: int
+ name: String
+ rarity: int
+ y: int
+ hitbox: Rectangle
+ setPosition(int, int): void
render
ItemRenderer
- item: Item
+ texture: BufferedImage
+ getItem(): Item
+ render(Graphics2D, int): void
Crown
Key

3.2. Diagrammes de séquence système

Le diagramme de cas d'utilisation présenté en section 1.1 se décrit à travers les diagrammes de séquence système suivants donnant l'enchaînement chronologique des actions pour chacun des cas d'utilisation présentés.

3.2.1. Diagrammes de séquence système du cas d'utilisation Move Character

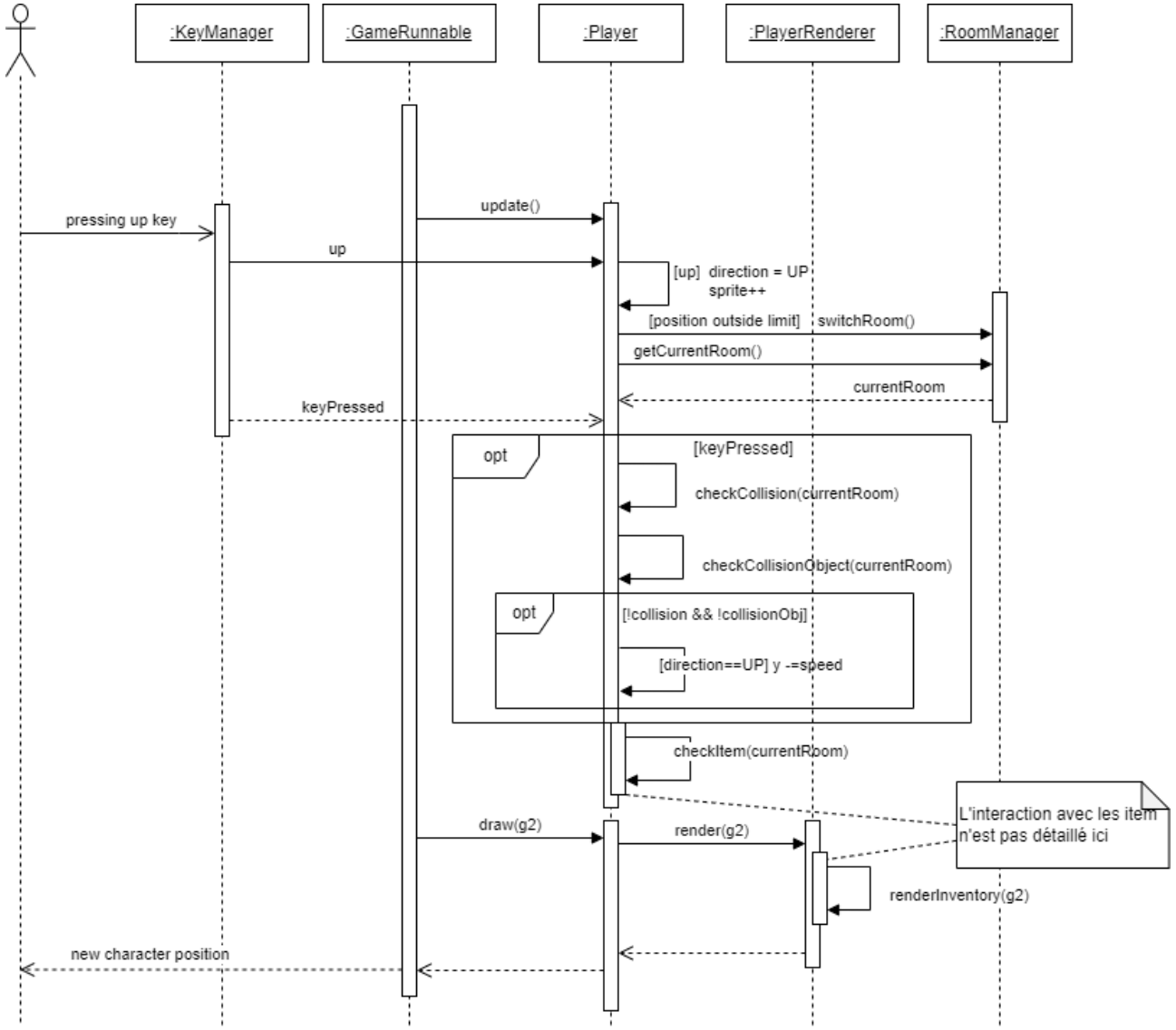


Diagramme représentant toute la séquence d'appel lors du déplacement du personnage alors que l'on presse la touche up (flèche directionnelle de haut).
A des temps réguliers, **GammeRunnable** met à jour la position du personnage. Celui- ci récupère la touche actuellement appuyée par le joueur grâce à l'intermédiaire du KeyManager (ici la touche est up). On vérifie qu'on ne sort pas des limites de la pièce. Si une touche est pressée (le cas ici) Player vérifie la collision avec les mur puis les objet

interactif présent dans la pièce. Si il n'y a pas de collision la position est mise à jour.Les vérifications pour la collision avec les Item n'est pas détaillées ici.
Par la suite **GammeRunnable** demande l'affichage du joueur, c'est là que **PlayerRenderer** interview (package view)

3.2.2. Diagrammes de séquence système détail de l'appel à checkItem

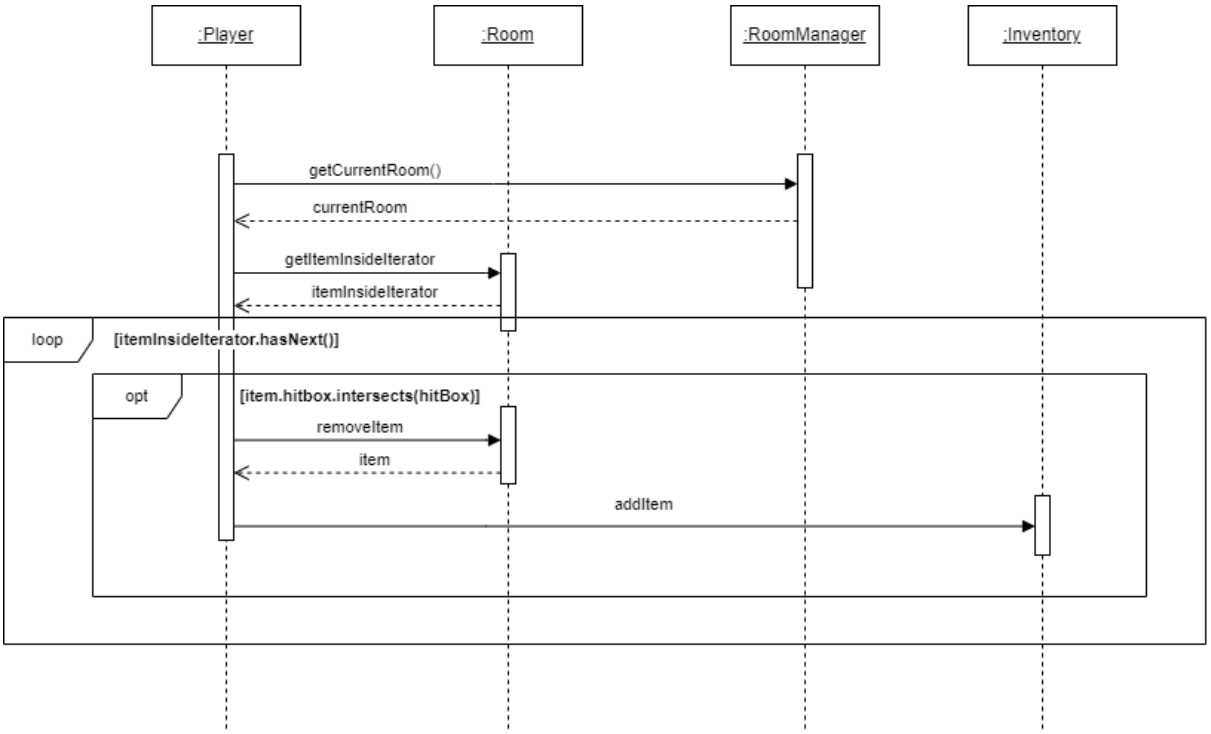


Diagramme décrivant les appelle effectuer lors du ramassage d'un item par le **Player.**

IV. MANUEL UTILISATEUR

Toute réalisation informatique ayant abouti à la livraison d'un produit fini doit nécessairement contenir un support technique destiné aux utilisateurs finaux afin de leur faciliter la prise en main de leur nouvelle solution. Notre projet réalisé nommé *Journey to the Lost Kingdom* ne restera pas en marge de cette bonne pratique. Nous détaillons ci-dessous les manipulations nécessaires pour jouer le jeu.

1. Compilation et exécution

1.1. Exécution en ligne de commande dans un terminal

Pour **compiler le jeu**, déplacez-vous dans le dossier *tp_acol* racine puis exécuter successivement les commandes :

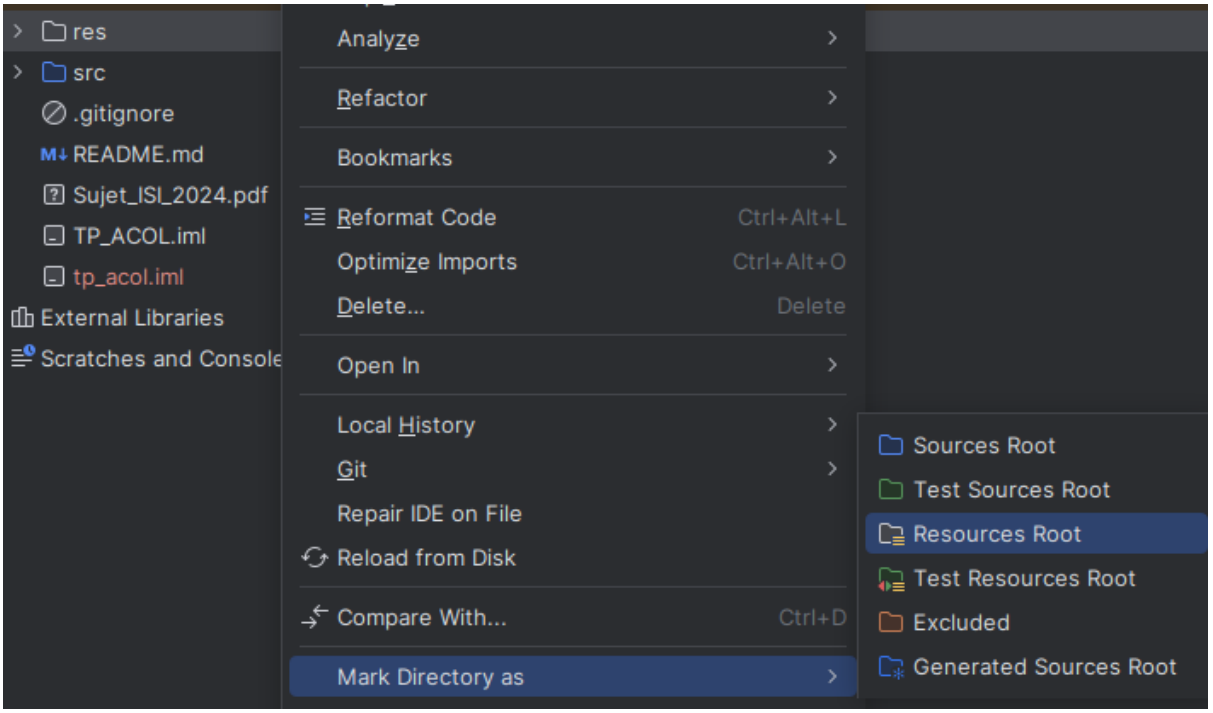
javac -d out $(find src -name "*.java")
cp -R res/* out/

Une fois le jeu compilé, il ne vous reste plus qu'à l'exécuter en tapant :

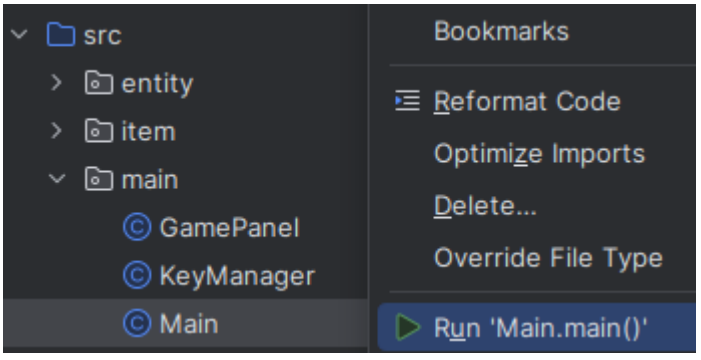
java -cp out main.Main

1.2. Exécution dans IntelliJ

Amoureux (se) de l'éditeur de code IntelliJ ? Alors, on vous facilite la tâche via nos outils et raccourcis mis à vos dispositions.
La première des choses à faire consiste à marquer le dossier **res** comme *Ressource Root.*
Pour cela, clic droit sur ***/res ->Mark Directory as ->Ressources Root***



Lancez le jeu en effectuant un clic droit sur ***/src/main/Main ->Run main.Main()*** comme le montre l'image ci-dessous.



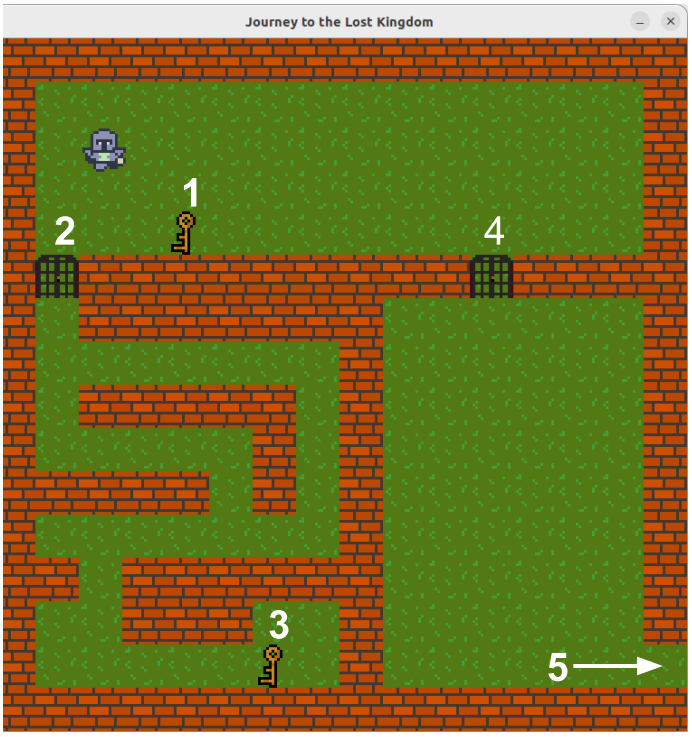
2. Jouer au Journey to the Lost Kingdom

Voici ci-dessous l'interface de jeu de notre jeu. Il se joue de la manière suivante :

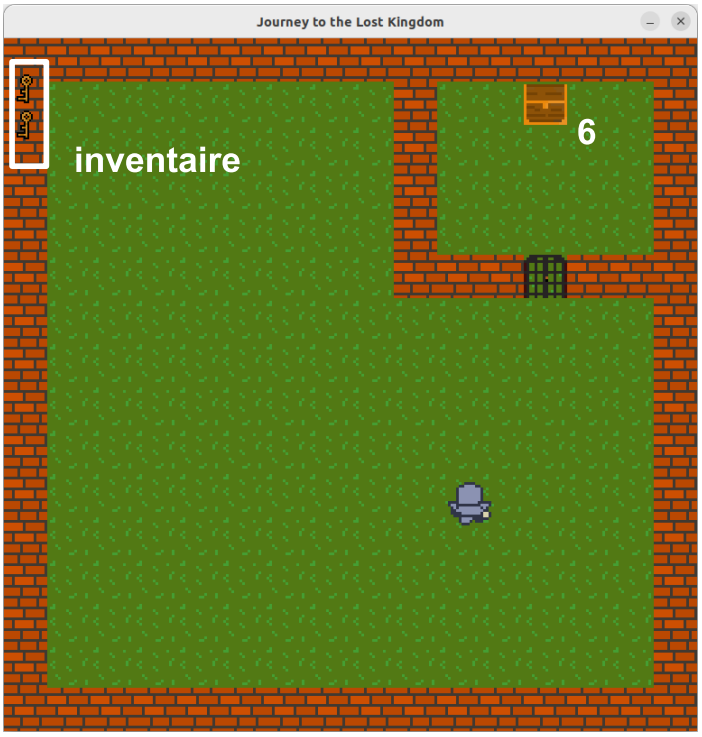
- Pour vous déplacer, utilisez les **touches directionnelles** du clavier ou les **touches Z** (monter), **Q** (aller à gauche), **S** (descendre)**, D** (aller à droite).
- Pour ouvrir les **portes** (2 & 4) et les **coffres**,vous devez trouver la clé correspondante.

- Pour ramasser une **clé** (1 & 3), vous devez simplement marcher dessus et pour ouvrir une porte ou un coffre vous devez vous en approcher avec la bonne clé dans votre inventaire.

L'objectif est de récupérer la couronne dans le **coffre** (6) de la deuxième salle.



Comment résoudre la première salle



Fin du jeu après

V. BILAN SUR LES OUTILS DE MODÉLISATION

Pour modéliser les différents diagrammes UML, nous avons utilisé l'application web ***draw.io*** de *diagrams.net*. Certains diagrammes étant trop grands nous avons parfois dû les couper sur plusieurs pages pour faciliter la lisibilité.