RENDU PROJET ACOL

BOUIHI Hamza - DORKENOO Eliott - LOURS Maxence

Sommaire :

Cahier des charges - Jeu d'aventures

1. Introduction

Ce projet se repose sur l'ensemble du cours d'ACOL, l'accent est donc mis sur la conception plutôt qu'au développement en tant que tel, même si l'un découle de l'autre. L'objectif va être de résoudre les difficultés de conception en amont pour ne pas avoir à résoudre ces problèmes lors de la phase de programmation. On essaiera donc d'être le plus clair possible dans tous les documents rendus pour que n'importe quel utilisateur ou personne voulant plus de détails sur le jeu puisse trouver toutes les informations qu'il souhaite. Ce cahier des charges a donc pour objectif de relier la partie conception avec ce que le joueur pourra véritablement faire lorsqu'il lancera la version finale de ce projet.

2. Objectifs du projet

Objectifs du jeu d'aventures :

- Jeu d'aventures joueur unique. Ce dernier contrôle un personnage, qui doit venir à bout de ses ennemis pour découvrir le monde qui l'entoure. A l'issue de son voyage, le personnage doit trouver un trésor caché. Durant son périple, il trouvera différents objets, qui l'aideront peut-être à trouver ce dont il a toujours rêvé, le fameux trésor.

3. Environnement du jeu

- Le jeu se joue via le lancement du programme principal sur ordinateur. Il nécessite d'avoir à disposition un clavier et une souris. Ceci permet les différentes interactions du joueur avec le système.
- Décor 2D type plateformes.
- Personnages du jeu: Ennemis, il faut s'en débarrasser pour pouvoir avancer. Mais attention, si vous n'en venez pas à bout, ils se chargeront de vous tuer…
- Différents paramètres peuvent être modifiés pour adapter le rendu visuel/audio meilleur pour l'utilisateur.
- Sauvegarde par niveau, progression dans le jeu selon le nombre de salles atteintes.

4. Exigences techniques

Voici les exigences techniques essentielles de notre projet :

- Langage de programmation Java.
- Pas de base de données, le projet est assez petit pour être bien plus rapide à exécuter sans.

5. Fonctionnalités principales à implémenter

- déplacement de personnages
- système d'objets et d'objets utilisables (possédés) par le personnage.
- récupérer un objet
- monstre à battre
- système de combat
- découvrir un trésor
- changement de salle

6. Interface utilisateur

Pour faire une interface graphique nous allons devoir utiliser une API Java déjà reconnue pour mener à bien notre développement. Nous avons opté pour JFrame car c'est une solution déjà vu en cours et bien documentée.

7. Livrables

Livrables finaux :

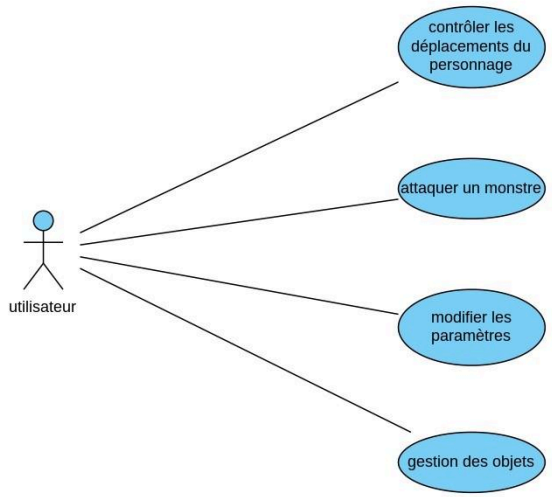
- Code source.
- Cahier des charges.
- Document d'analyse.
- Document de conception.

- Jeu fonctionnel.
- Manuel utilisateur

Document d’analyse

Cas d’utilisation

L’objectif de ce cas d’utilisation est de préciser les différents cas que l’utilisateur va rencontrer durant la partie.
Pour illustrer ceci, nous commencerons par expliciter le lien entre les acteurs du système du jeu:



L’utilisateur contrôle grâce au clavier interactions du joueur avec le monde qui l’entoure:

Déplacement joueur:

Déplacer le joueur permet à l'utilisateur d'avancer dans le jeu et d'interagir avec les éléments placés dans le décor.

Actions de déplacement:

- diriger le joueur vers la gauche ou vers la droite
- sauter
- rester immobile (mais il peut tout de même bouger, dans le sens où par exemple, il peut être immobile en sautant, mais la gravité le fait redescendre).

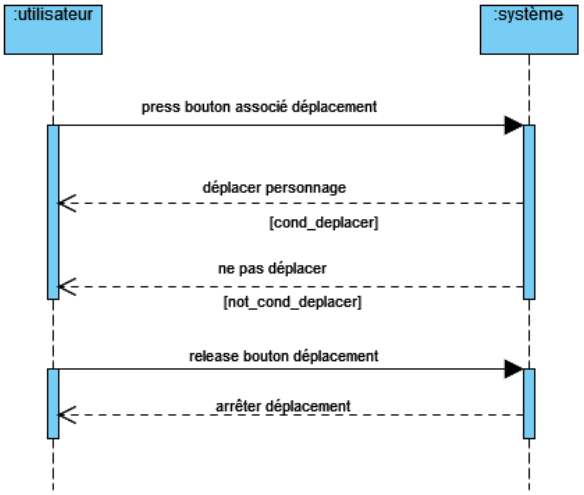


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'une commande de déplacement

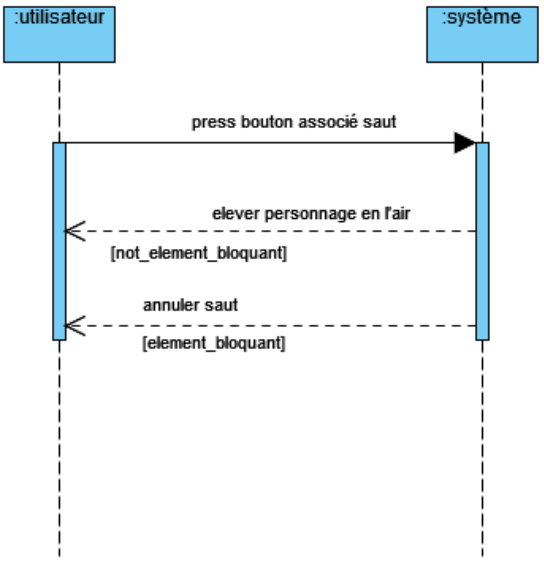


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'une commande de saut

Exemple de scénario:

- L'utilisateur appuie sur la commande permettant de se déplacer vers la droite. Le système répond par le déplacement du personnage dans cette direction. Tant que la touche n'est pas relâchée, le joueur continue son déplacement. Si il rencontre un obstacle, un mur par exemple, son déplacement s'arrête. De même, si la touche est relâchée, le joueur s'arrête.
- L'utilisateur lance la commande de saut. Ceci fait décoller du sol sur lequel le joueur se trouve. La hauteur du saut est fixe, il suffit d'appuyer une fois sur le bouton. Pendant le saut, l'utilisateur peut lancer les commandes de déplacement latéral. Si un plafond, un obstacle, empêche le saut d'être réalisé, ce dernier s'interrompt et fait descendre le joueur.

Attaquer individu:

Dans le monde du personnage, se trouvent des monstres. Ces derniers occupent différentes positions dans les salles, empêchant le joueur d'accéder à certaines ressources. L'utilisateur doit donc arriver à bout des monstres en utilisant la fonctionnalité d'attaque.

Attaque : lance une attaque dans la direction du personnage.

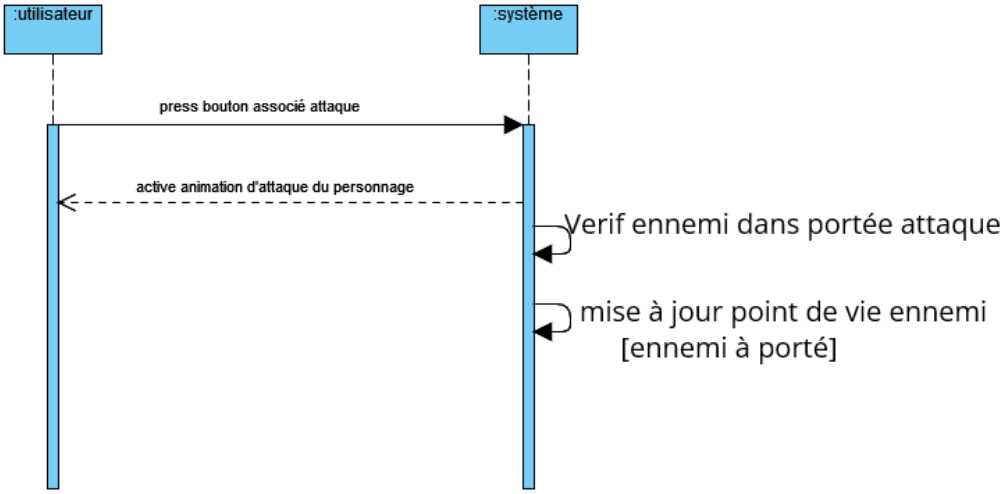


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'une commande d'attaque

Exemple de scénario:

- L'utilisateur lance la commande de saut. Pendant que le personnage est en l'air, il utilise la commande d'attaque. Le système active alors l'animation d'attaque pour que le personnage effectue le mouvement.

- L'utilisateur lance une commande d'attaque. Si le personnage ne se trouve pas à portée d'un monstre, alors le système active l'animation, mais rien ne se passe. Sinon, le système inflige des dégâts au monstre dans la range, et l'élimine (le fait disparaître de la salle).

Ouvrir les paramètres:

Pour un meilleur confort de l'utilisateur, ce dernier peut effectuer des modifications de l'interface utilisateur du jeu.

Modification du niveau de jeu:

- Paramètre permettant de modifier la difficulté du niveau de jeu. Cela correspond aux points de vie accordés au personnage principal. Moins il y en a, et plus le jeu sera compliqué.
- Paramètre permettant de valider ou non l'accès à une nouvelle salle. Cela correspond à définir une nouvelle condition, qui est qu'il faut éliminer tous les monstres de la salle actuelle afin d'accéder à la nouvelle salle.

Modification de l'affichage:

- Paramètre permettant de changer la couleur du personnage.
- Paramètre permettant de changer la taille de la fenêtre dans laquelle se trouve le personnage.

Activer le son:

- Paramètre qui permet d'activer le son ou non.

Contrôle de l'état dans le jeu:

- Une commande permet de sauvegarder le jeu, dans la salle dans laquelle se trouve le personnage au moment de la commande.
- Recommencer. Cette commande restart le jeu depuis le début.
- Quitter. Quitte le jeu. Sans sauvegarde, le jeu reprendra depuis le début au prochain lancement.
- Reprendre. Permet de reprendre le jeu au moment où l'utilisateur a lancé la commande ouvrant les paramètres (ce qui a mis le jeu en pause).

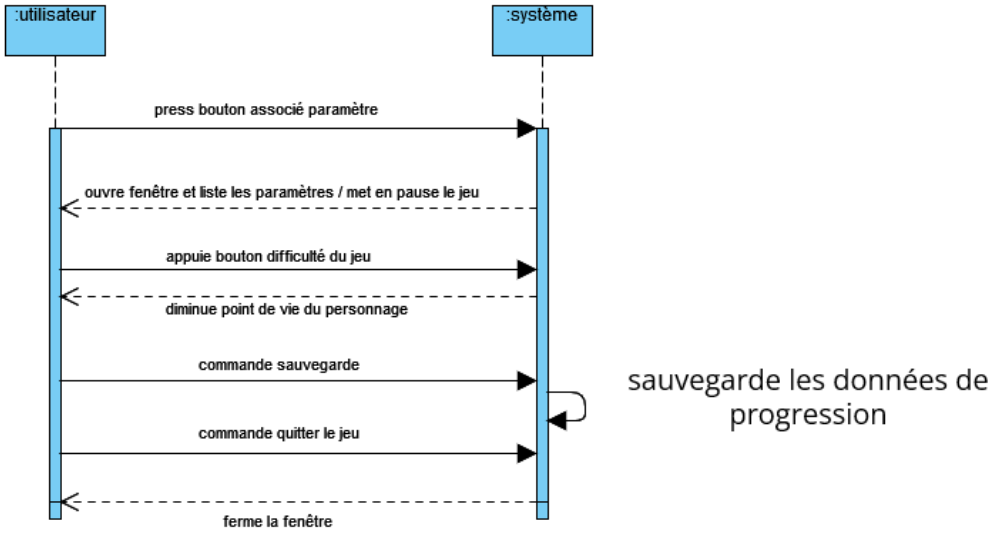


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'un enchaînement de commandes paramètres

Exemple de scénario:

- L'utilisateur lance la commande qui ouvre les paramètres. Ce qui met en pause le jeu. L'utilisateur appuie sur la touche qui change la couleur
- du personnage. Le système met à jour la couleur. Ensuite, il lance la commande pour l'audio. Le système active la musique du jeu. Enfin, il appuie sur la touche "reprendre". Le système quitte les paramètres, actualise les données du personnage, et relance le jeu dans l'état où il a été arrêté, avec une nouvelle couleur et de la musique.
- L'utilisateur lance la commande qui ouvre les paramètres. Il lance la commande sauvegarde, puis quitte l'application. Plus tard, il relance le jeu. Le système à ce moment met à jour les informations de la dernière sauvegarde, puis relance le jeu à l'endroit de la sauvegarde.

Gestion des objets:

Pour remporter la partie, l'utilisateur doit ramasser des objets. Parmi eux, se trouvent des clés, des objets inutiles, et le trésor, qui lui permet de mettre fin au jeu.

Gestion des objets:

- Ramasser objet. A portée d'un objet, l'utilisateur fait récupérer l'objet à son personnage. La commande ne peut aboutir si un monstre est à portée d'attaque du personnage.
- Déposer un objet. Si le personnage est sur un terrain vide (aucun objet à portée), il peut abandonner le dernier objet qu'il a retiré. La commande échoue également si un monstre est à portée, ou bien si aucun objet n'a été retiré.

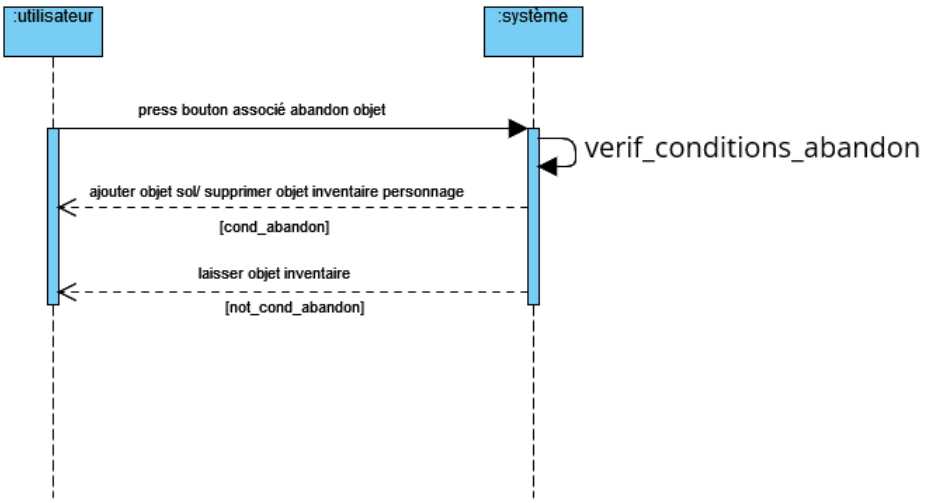


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'une commande qui permet de ramasser l'objet

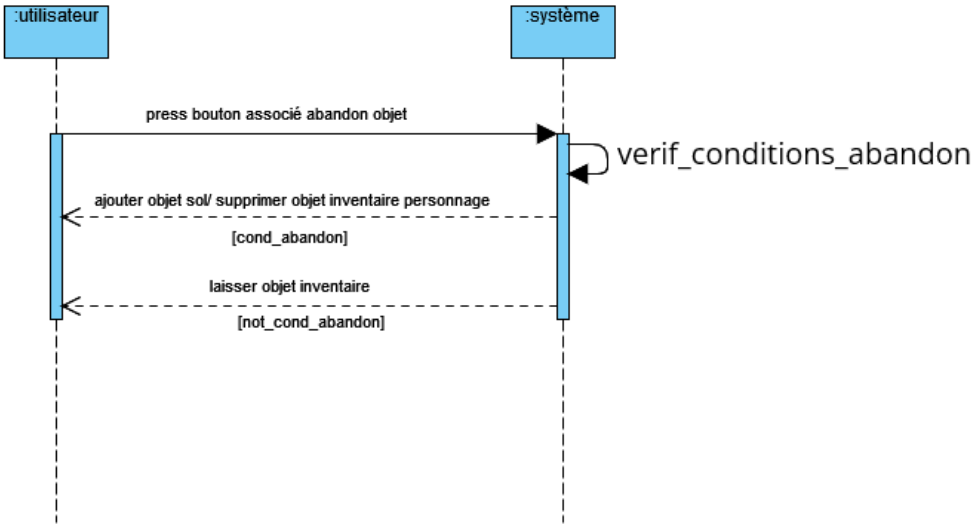
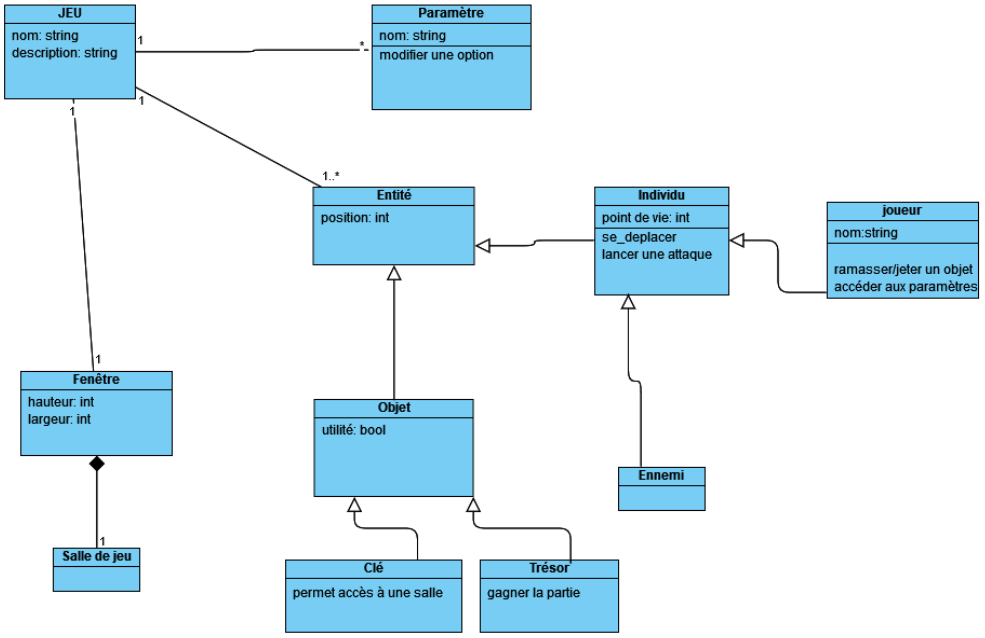


diagramme de séquence d'analyse qui représente un échange entre utilisateur et système lors d'une commande de rejet objet

Exemple de scénario:

- L'utilisateur entre la commande pour ramasser un objet. Si un objet est à portée, le système retire l'objet de la salle, et l'enregistre dans la liste des objets du personnage. L'utilisateur déplace le personnage, puis rentre la commande pour déposer. Le système vérifie que l'opération est possible, mais un objet est déjà présent sur le sol, donc l'objet n'est pas déposé, et reste dans l'inventaire du joueur.
- L'utilisateur entre la commande pour ramasser un objet. Or un monstre est dans la zone de combat et attaque le personnage. L'opération est annulée, l'objet reste dans l'inventaire.

Diagramme de classe d'analyse:



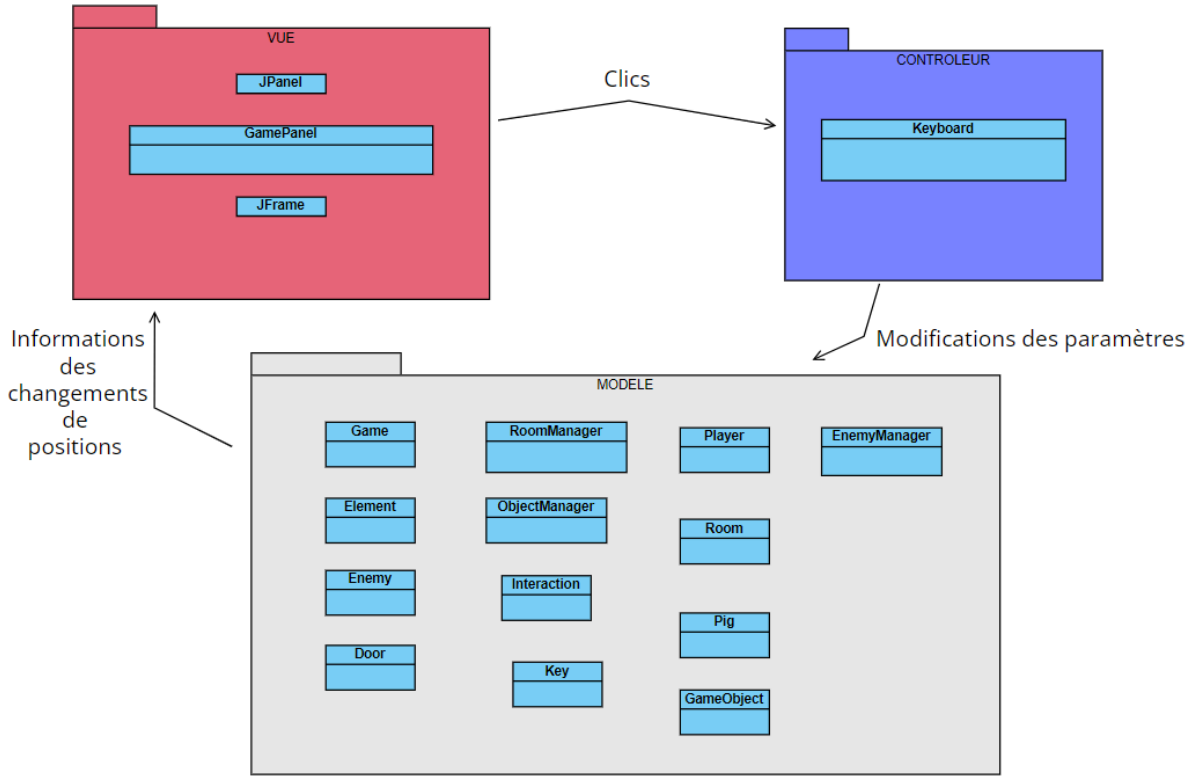
Ce diagramme met en relation les différents éléments qui composent notre jeu. Il permet d'éclairer l'utilisateur la structure mise en place pour l'élaboration du jeu d'aventure.
Pour faire la transition avec la partie conception, il faut savoir que la structure globale subira des modifications dans l'architecture logicielle afin de respecter au mieux le modèle choisi. C'est pourquoi vous verrez dans les diagrammes de classe logicielles certaines adaptations qui nous ont permis de mieux implémenter notre jeu d'aventure.
Quelques points sur ce diagramme :

- Les actions d'attaque sont communes aux individus, tandis que le joueur peut ramasser des objets et accéder aux paramètres.
- La fenêtre de jeu contient une salle de jeu à n'importe quel moment de la partie.
- Les clés et trésors sont des objets, qui peuvent être utilisés ou non. Par exemple, certaines clés ne serviront peut être à rien.

- La classe paramètre représente une option pouvant être modifiée à partir du menu des paramètres. Cela signifie que à un jeu est associé autant de paramètres que l'on souhaite, mais qui sera bien évidemment limité lors de la conception.
- Pour passer d'une salle à une autre, il faut avoir au préalable trouvé la clé qui permettra d'ouvrir la porte pour accéder à la prochaine salle.
- Le trésor, s' il est trouvé, lui permettra de remporter la partie et de mettre fin au jeu.

Document de Conception

3.A. Architecture logique du logiciel :



Voici l'architecture logique du logiciel. Nous avons tout d'abord le paquetage Vue qui se charge de l'affichage avec des classes comme GamePanel qui aura cet objectif, il mettra ensuite en écoute les contrôleurs pour les appeler au moment adéquat. Le paquetage contrôleur est uniquement composé du clavier car il n'y a pas d'interactions possibles autrement que par l'intermédiaire du clavier. Pour ce qui est de la classe modèle nous avons pensé à beaucoup de classes détaillées ensuite dans le diagramme de classe pour réaliser l'ensemble des fonctionnalités que nous souhaitons mettre en place. On peut notamment parler de game qui se chargera de faire l'appel des fonctions d'update du modèle et d'envoyer les informations de

changement de position à la vue au travers de ses liens avec gamePanel. Interaction joue également un rôle clef dans l'orchestration des interactions entre tous les composants.

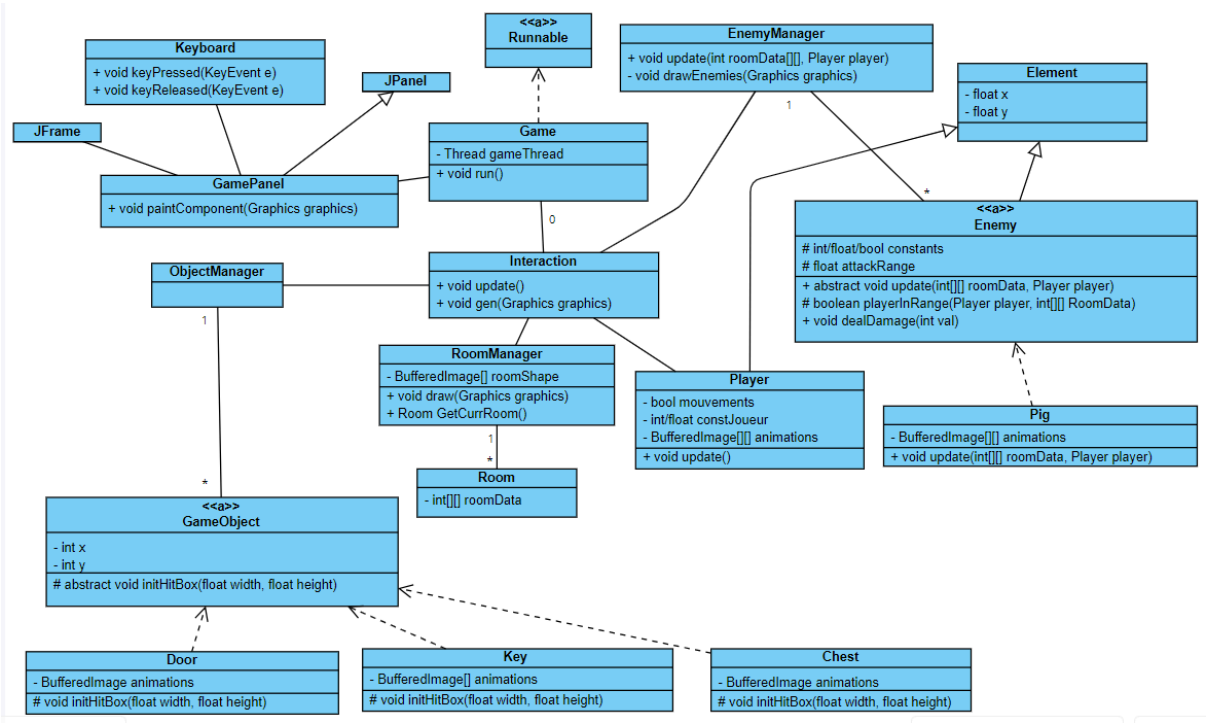
Pour nous, une structure MVC nous semble être la plus cohérente pour répondre aux exigences de ce problème, les missions à remplir pour chaque paquetage sont claires et précises et le code va être extrêmement lisible pour n'importe quelle nouvelle personne qui voudrait ajouter des modifications ou comprendre le fonctionnement en détail.

Nous pouvons aussi expliquer que ce choix de conception n'est pas anodin non plus, nous avons fait de nombreuses recherches en amont pour comprendre comment réaliser un jeu vidéo dans les meilleures conditions pour optimiser notre diagramme de classe. Par exemple, nous avons mis en place une technique classique de génération des salles avec des fichiers de 26 pixel par 14 où la valeur RGB à une signification importante pour la création des rooms. Nous avons fait le choix d'utiliser la valeur rouge R pour la parcelle de mur utilisé (fichier Room.png), la valeur G pour le type d'ennemi et la valeur B pour les objets à placer.

3.B. Description de l'incrément choisi

Pour notre projet nous avons décidé de commencer par réaliser l'ensemble des fonctionnalités à l'exception de la modification des paramètres et d'un système d'inventaire ou il pourrait posséder/relâcher des objets. On se contentera pour une première version de faire une fenêtre de jeu fonctionnelle avec des monstres, des objets et des déplacements.

Diagramme de classes logicielles :



Ce diagramme de classe montre une vision globale de l'implémentation que nous envisageons pour notre code. Nous ne pouvons pas détailler l'ensemble des fonctions que nous allons réaliser ni l'ensemble des paramètres mais on a essayé de montrer plusieurs choses importantes. On remarque bien la structure MVC comme présentée au-dessus avec une séparation entre les différents rôles. On a également décidé de regrouper les objets et les entités dans des classes bien héritées pour une meilleure lisibilité et une bonne utilisation des principes de l'encapsulation. La classe Interaction est au cœur de notre modèle, c'est elle qui va orchestrer tout le processus en ayant une vision globale.

On remarquera à quel point il est facile d'ajouter de nouveaux objets/ennemis avec notre structure qui repose sur des classes plus générales dans l'ensemble du code.

Une bonne structure sera donc respectée ce qui permettra une maintenabilité du code à long terme très simple. Notre première version aura donc un grand intérêt car elle créera des bases solides pour le développement d'un jeu bien plus gros. Si une nouvelle équipe veut travailler sur ce projet, il leur sera alors très simple de l'adapter à leurs besoins.

Pour rentrer un peu plus dans le détails de certaines fonctionnalités voici des diagrammes de séquences détaillés.

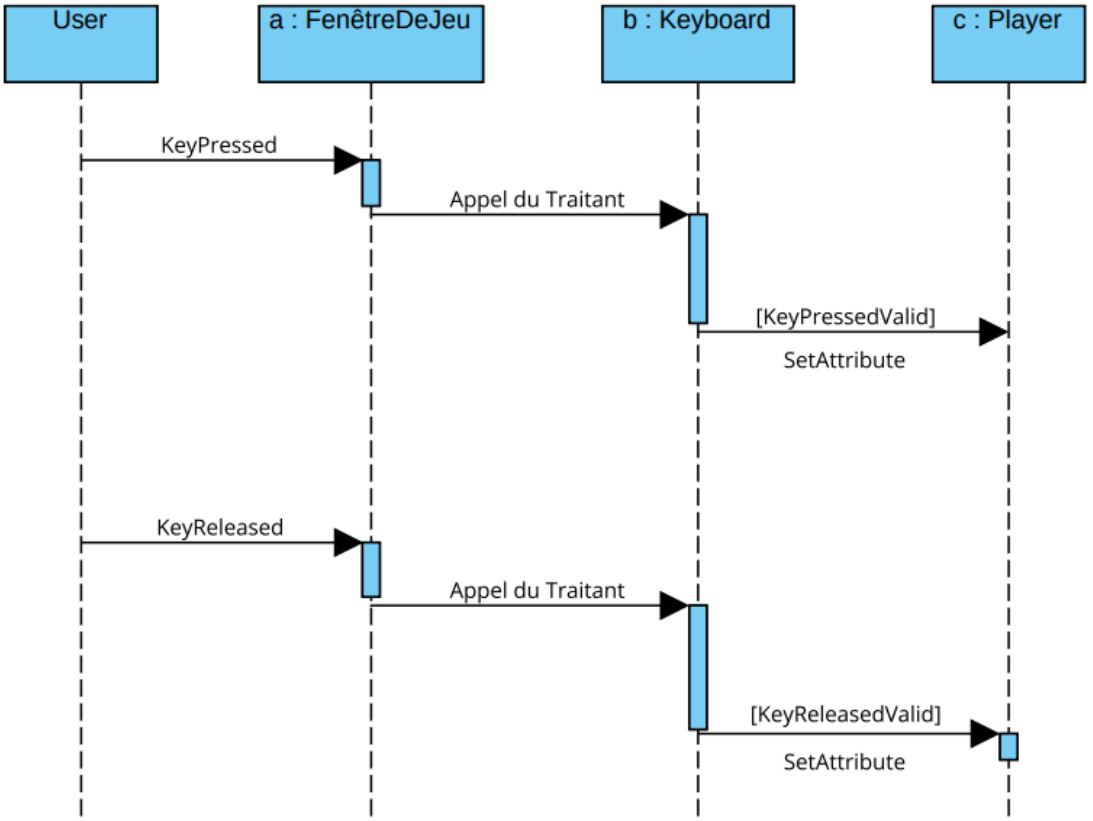


Figure 1 : Diagramme de séquence pour le déplacement joueur

Ce diagramme permet de comprendre plus en détail comment nous avons conçu la réalisation d'un déplacement joueur. La fenêtre de jeu se mettra sur écoute du clavier pour lui envoyer la touche et que la classe Keyboard puisse la traiter. Ensuite si la touche est valide, on change les attributs du joueurs pour qu'il réalise l'action voulu (si elle est permise mais ce n'est pas cette partie du programme qui se charge de générer le déplacement effectif du joueur, il se fait lui tout les x millisecondes en fonction des attributs qui ont été set par l'intermédiaire du clavier de l'utilisateur).

Pour y voir encore plus claire détaillons un cas précis de saut par l'utilisateur :

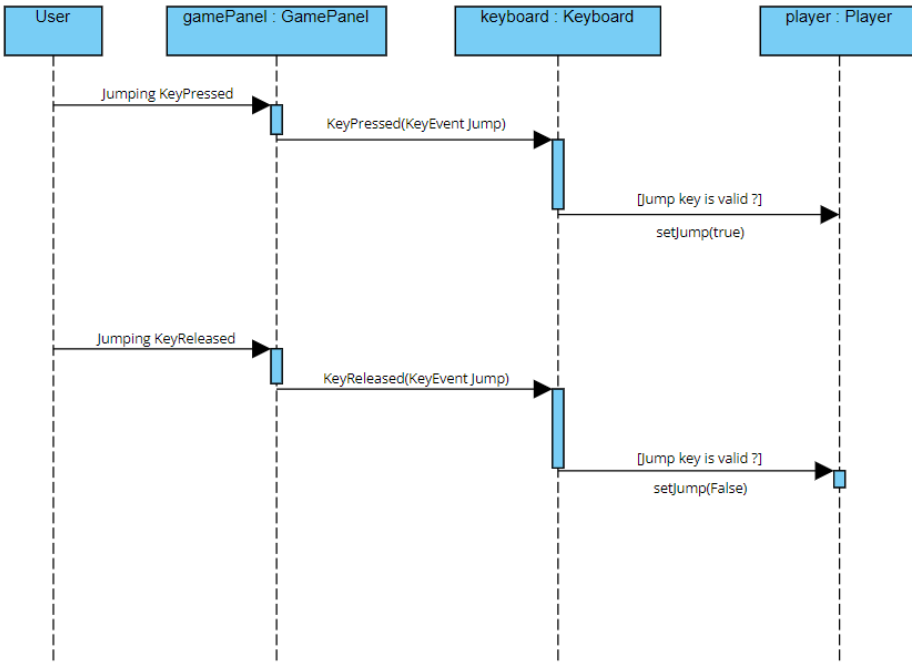


Figure 2 : Diagramme de séquence réel pour un saut

Ce diagramme met en pratique ce que nous allons réaliser par la suite pour gérer les déplacements du joueur.Nous allons maintenant voir plus en détail comment on ramasse des objets dans notre jeu :

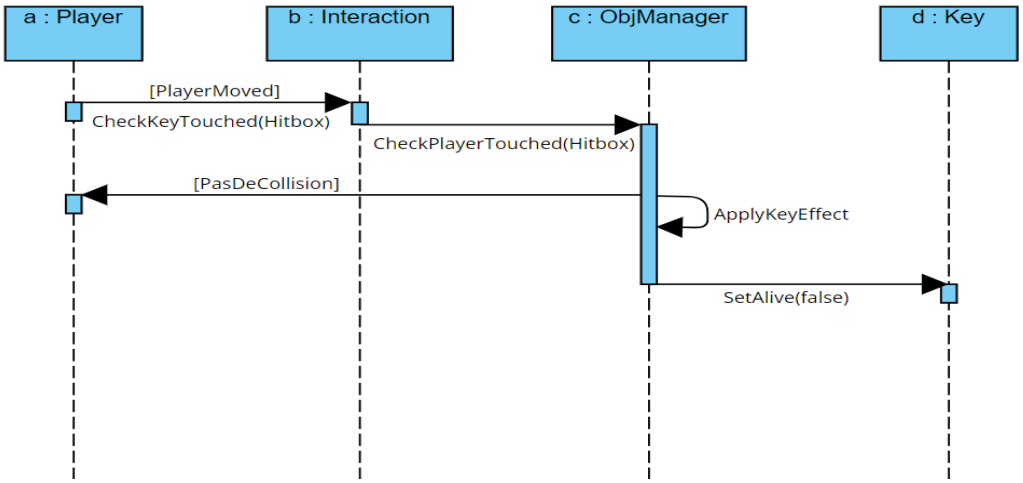


Figure 3 : Diagramme de séquence ramasser objet

Nous remarquons que nous regardons si on touche un objet uniquement lorsque le joueur bouge, de plus si il touche effectivement l'objet on va appliquer les effets de la clef, en l'occurrence notre jeu retient le nombre de clef récupéré pour permettre un évènement spécial quand on les a toutes. Pour finir on doit faire disparaître l'objet du jeu c'est pourquoi on setAlive(false) pour ne plus le dessiner.

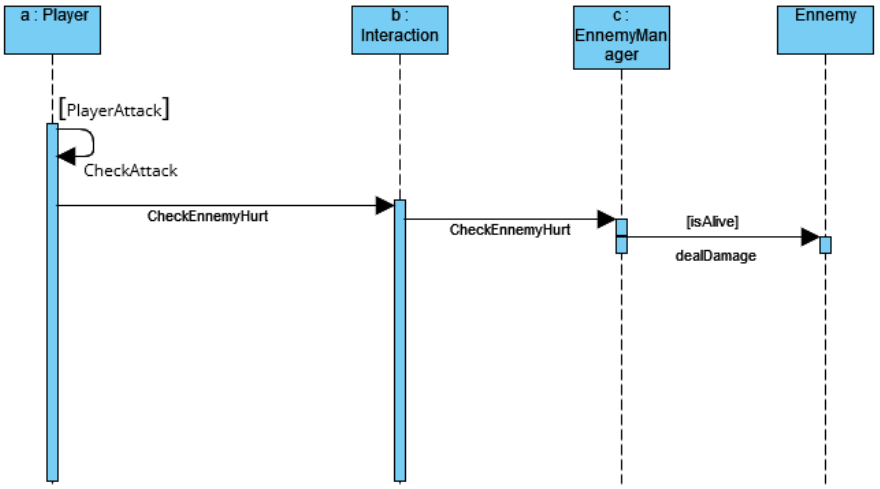


Figure 4: diagramme de séquence attaquer ennemi

Ce diagramme système représente un scénario dans lequel la touche d'attaque est lancée. Après avoir effectué la vérification si l'attaque peut être lancée, on envoie à la classe interaction les informations de hitbox. En effet, dans le cas où un ennemi est à distance d'attaque, la classe interaction envoie les informations à l'ennemi manager, qui, après avoir vérifié que le monstre est vivant, lui inflige les dégâts d'attaque.

Manuel Utilisateur

Dans cette partie nous allons préciser le but du jeu, l'ensemble des actions possibles par l'utilisateur et comment les réaliser.

But du jeu :

Le but du jeu est de collecter toutes les clés de chaque salle afin de débloquer une porte menant à la salle suivante. Dans la dernière salle, un trésor est caché, et votre objectif est de l'atteindre pour remporter la partie. Toutefois, vous devrez réussir cela en évitant tous les obstacles et en éliminant les monstres qui tentent de vous tuer sur votre chemin.

Actions possibles :

Actions de déplacement :

- se déplacer vers la droite : flèche de droite
- se déplacer vers la gauche : flèche de gauche
- sauter : barre espace

Actions d'interactions :

- attaquer : bouton z
- interagir avec les objets: Se rapprocher suffisamment de l'objet pour que l'interaction se fasse automatiquement.

Comment compiler et jouer au jeu :

Pour se faire rien de plus simple, cloner le dépôt, aller à la racine du projet et taper la commande suivante :

```
make play
```

Vous devriez voir le jeu se lancer directement et voir votre personnage atterrir dans le premier niveau.

Bilan sur les outils de modélisation

Dans cette partie nous allons discuter des choix des outils de modélisation effectuées et des difficultés rencontrées.

Au début nous avons commencé à travailler avec Modelio, une solution proposée sur Chamilo. Après une tentative de prise en main nous arrivions à faire quelques graphiques mais cela nous prenait beaucoup de temps. Nous avons alors pensé qu'il serait plus efficace de chercher une autre solution pour nous aider à faire des graphiques propres et facilement.

Nous avons ensuite découvert Visual Paradigm Online, cette solution nous a paru toute suite plus ergonomique et compréhensible, il y a des packaging déjà fait pour les différents types de diagrammes commun tel que les diagrammes de classes ou les diagrammes de séquence. Cela rend la réalisation de diagrammes bien plus fluides et faciles. Nous avons donc réalisé l'énorme majorité de nos diagrammes dessus.

Nous recommandons cette solution pour toutes les personnes qui n'ont pas déjà utilisé d'outils de dessins de diagrammes au préalable car c'est très simple d'accès et très rapide à prendre en main. Le résultat est également très joli, ce qui en fait une solution de qualité.

Merci pour votre lecture, en espérant que le jeu vous plaise !