Rapport Final
Projet TOB
CraftSurvive

Groupe E51 : BAUD Alexandre / CHENG Jimmy / COVET Florent / DARY Jean-Léo / LAGRANGE Antoine / Gouneau Joceran / Pigneux Alice

Sommaire

1. **Description du projet :**

Le projet est un jeu où l'utilisateur est dans un monde ouvert 2D généré aléatoirement. Il doit pouvoir se déplacer et interagir dans ce monde, ainsi que sauvegarder son avancement.

2. **Fonctionnalités**

Fonctionnalité	Description	Statut
Armes	Permet de combattre les ennemis	Fait, itération 1
Ennemis	Des ennemis qui se déplace et attaque le Joueur lorsqu'il est à proximité	Fait, iteration 3
Se déplacer dans l'environnement		Fait, itération 2
Inventaire	Inventaire dans lequel le joueur peut stocker des objets tels que des armes, des bouclier ou des ressources	Fait, itération 3
Ramasser des objets / Coffres	Tout est dans le titre	Non fait. Les coffres ne sont pas géré il y a juste l'affichage
Craft	Permettre au Joueur de récupérer des ressources sur la map	Partiel, le joueur peut casser des décors mais pas récupérer les ressources
Combats	Permettre les combats entre le Joueur et les ennemis	Fait, itération 3
Menu	Une interface pour gérer le jeu ainsi que les sauvegardes	Fait, itération 3

S’équiper d’une arme et d’une armure	Le joueur peut s’équiper d’une arme et d’une armure qu’il a dans son inventaire elle lui ajouter	Fait, itération 3
Santé et XP	Pouvoir être informé de sa vie, et de son XP	Fait, itération 3

3. Sous-systèmes

Nous avons décidé de séparer l’application en sous-système. Il y a Personnage qui s’occupe de tout ce qui est lié aux Personnages que ce soit le joueur mais aussi les ennemis. Ensuite il y a Environnement qui gère tout l’environnement du jeux. Et pour finir IHM qui va gérer l’interface graphique et l’interaction entre le jeux et l’utilisateur. On regroupe ensuite tous ces sous-systèmes dans la classes IHM Craft qui va gérer l’affichage du Jeux. La fonction main se situe finalement dans la classe CraftSurvive et utilise une temporisation qui est gérée dans la classe Attente. Vous pourrez donc voir dans la partie suivantes les diagrammes de classes des différents sous-systèmes.

4. Diagramme de classe

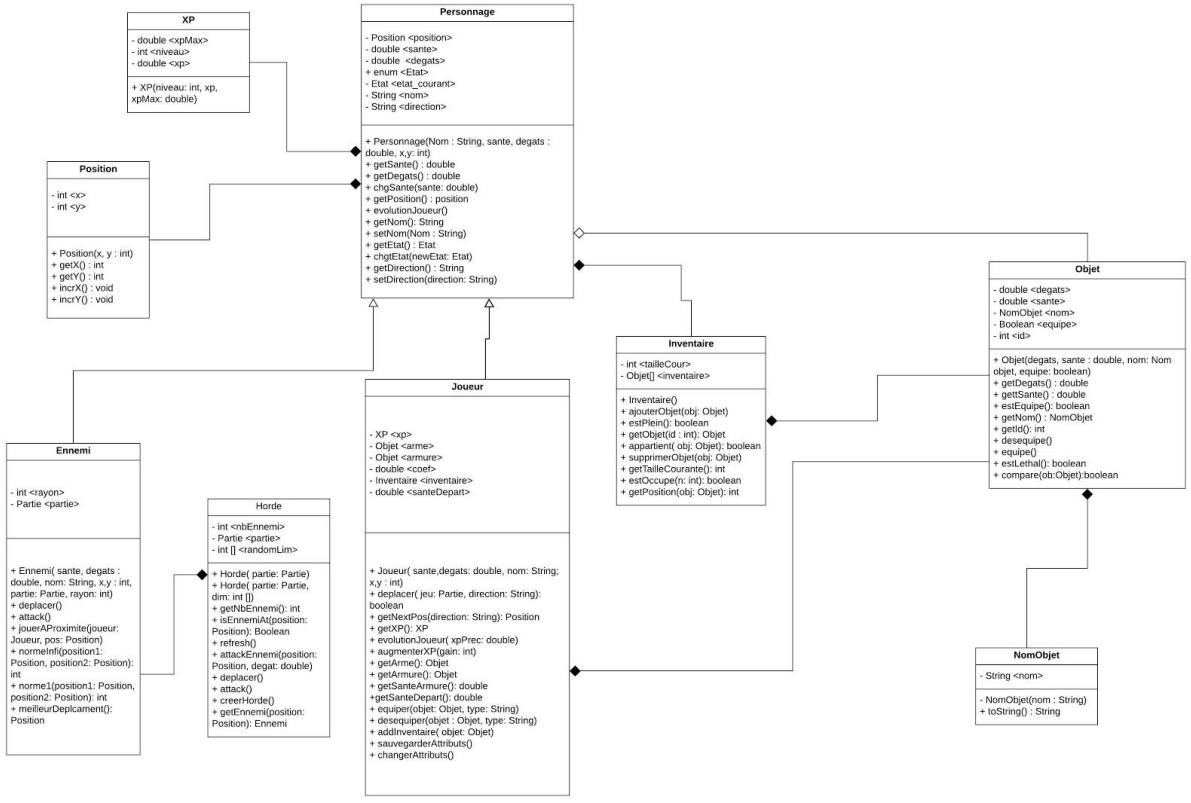

Diagramme de classe du sous-systèmes Personnage

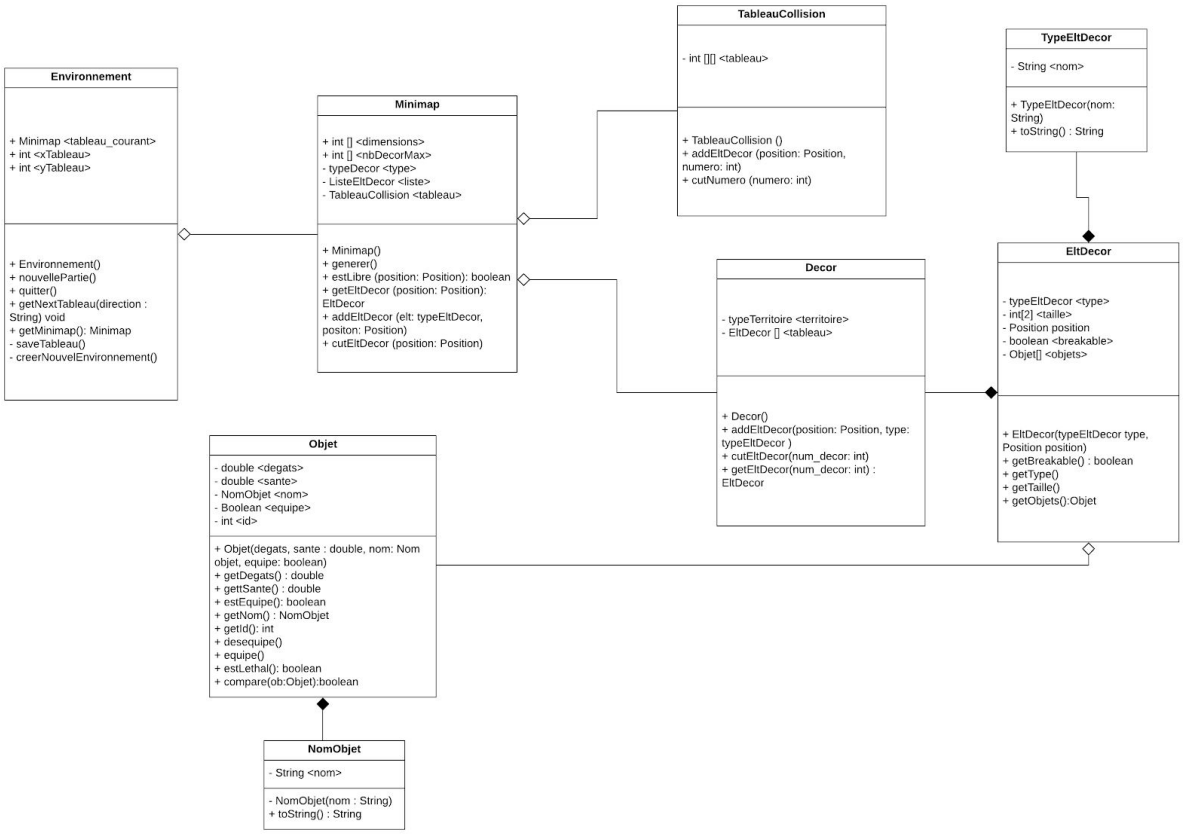

Diagramme de classe du sous-système Environnement

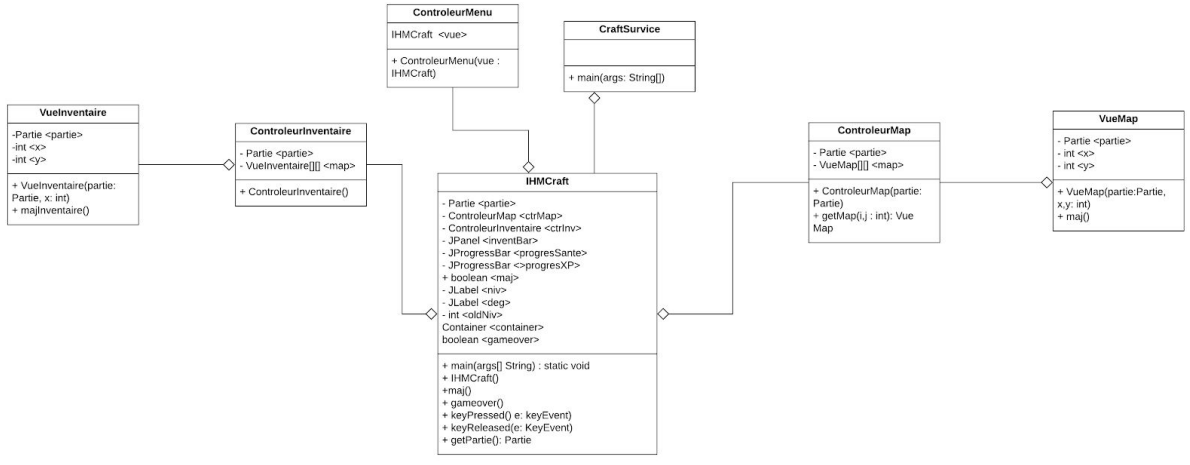

Diagramme de classe du sous-système IHM

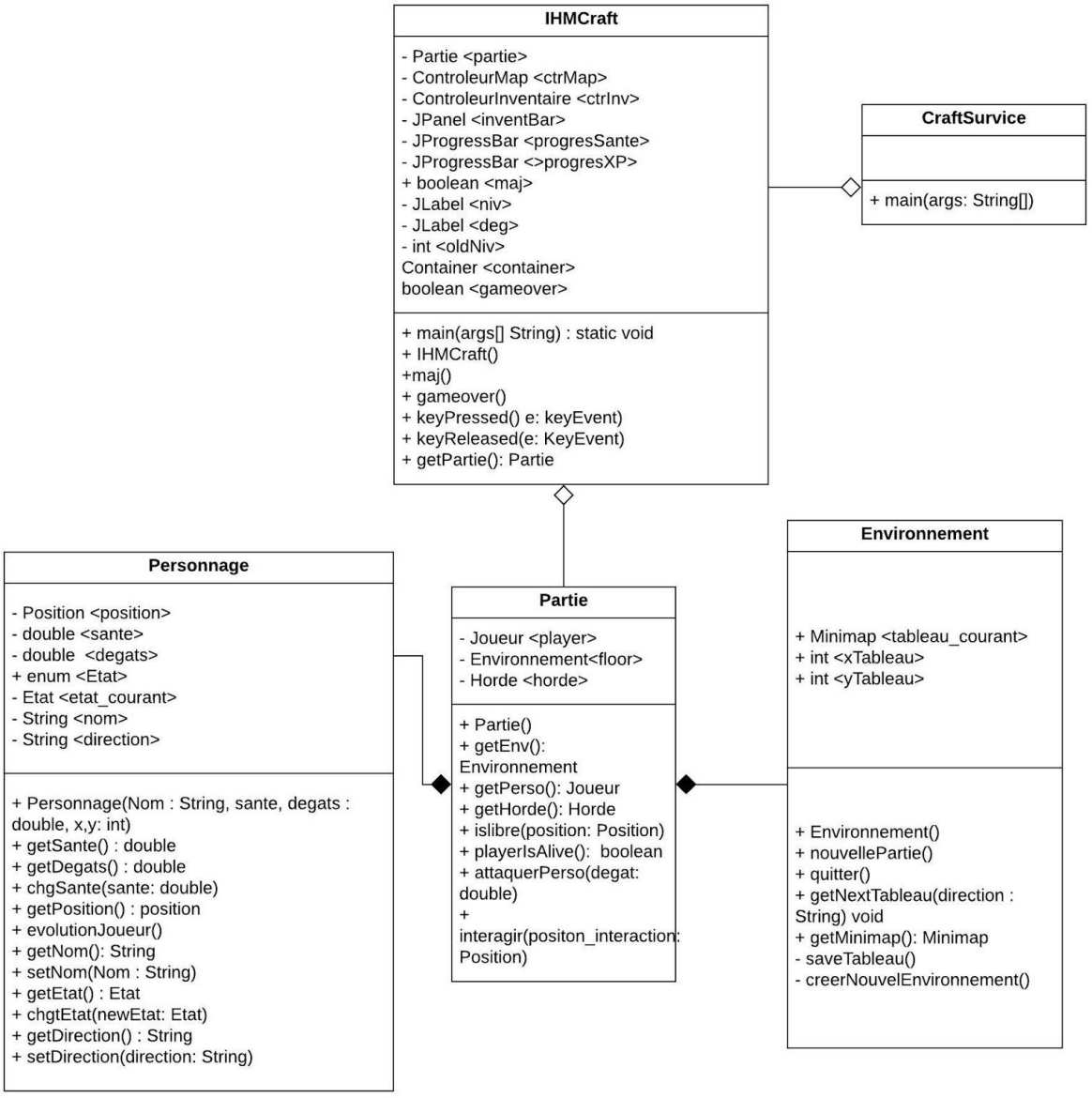

Diagramme de classe du projet

Vous pouvez retrouver le diagramme UML complé dans le dossier livrables du svn fichier UML_CraftSurvive.pdf.

5. Choix, problèmes et solutions

Nous n'avons pas eu de gros problèmes lors de l'implémentation de notre projet.
En ce qui concerne la gestion de l'environnement, et plus particulièrement l'implémentation d'une Minimap, celle-ci avait été imaginé au début comme un tableau 2D d'éléments de décor, chaque case étant occupée par l'élément présent à cette position. Le problème de cette solution apparaît lorsque l'on prend en compte des éléments qui n'occupent pas qu'une seule case (comme par exemple un rocher de taille 2x2) ; on ne peut alors pas savoir, simplement en lisant la valeur d'une case

du tableau, si celle-ci est recouverte ou non par un objet de grande taille situé dans une case voisine.
La solution choisie a alors été celle de séparer les deux types d'informations que doit fournir la Minimap dans deux objets distincts :
- les informations relatives aux éléments de décor sont gérés par un objet de type Décor qui est un inventaire de tous les éléments contenus dans la Minimap, chacun ayant un numéro qui lui est attribué et l'information de la position d'un élément est désormais un attribut de cet élément ;
- les informations relatives à l'espace occupé par ces différents éléments sont gérées par un tableau 2D d'entiers contenant en chaque case le numéro relatif à l'élément qui la recouvre si tel est le cas, ou 0 si cette case est vide.
Une exception CollisionException a enfin été créée afin de gérer les collisions lorsque, par exemple, lors de l'ajout d'un élément de décor, celui-ci se superpose à un autre ou sors de la minimap.

Concernant l'interface graphique, nous avons décidé de la faire en Swing, on a suivi le modèle MVC qui permet de bien distinguer la vue, le contrôle et le modèle ainsi nous pouvions plus facilement repérer d'où provenaient les problèmes.

Au niveau des personnages (i.e. Joueur et Ennemi) nous avons décidé de passer d'un attribut comportement avec deux comportements différents à une classe abstraite personnage dont héritent Ennemi et Joueur afin de factoriser certaines fonctions communes. Nous avons décidé lors de l'itération 3 de faire cela car il y avait trop de différences entre Ennemi et Joueur. Avec cette nouvelle architecture, il est possible de contrôler le Joueur et l'Ennemi de façon très différentes avec un Joueur contrôlé par l'utilisateur et l'Ennemi contrôlé par une IA;

Un autre choix a été fait au niveau de l'introduction des ennemis dans la partie, la création de Horde en découle, cela nous permet de mettre plusieurs ennemis sur un seul tableau ou de n'en mettre aucun. Ainsi chaque ennemi est géré dans cette classe.

Un problème que nous avons rencontré est le fait que nous ne gérons pas la temporisation des actions. Au départ nous avions pensé utiliser simplement une boucle while avec un sleep cependant cela bloquait totalement l'application. Pour résoudre cela après avoir fait des recherches, nous avons trouvé la notion de thread qui nous a permis de créer la classe attente et donc de gérer le problème de temporisation.

6. Organisation de l'équipe

Pour commencer l'implémentation, nous avons scindé l'équipe en deux afin que le premier groupe de trois personnes composé d'Antoine, Florent et Jimmy travaille sur la partie Personnages et que le second composé d'Alexandre, Alice, Jean-Léo et Joceran travaille sur la partie environnement. Par la suite nous nous sommes partagé les tâches sur l'implémentation de l'interface. Par ailleurs, la mise en place des méthodes agiles a été difficile compte tenu de la situation actuelle. Cependant nous avons essayé de faire des réunion régulière 1 à 2 fois par semaine afin de constater l'avancement de chacun et de pouvoir ajuster les objectifs en fonction de celui-ci. Durant la dernière itération, la répartition n'a pas pu vraiment se faire étant donné que chaque membre n'a pas mis la même implication dans le projet, il a donc était compliqué de mettre les méthodes agiles.