

# **RAPPORT DE PROJET DE PROGRAMMATION 2024**

## **SUJET DE SARAH REBOULLET: SHOOTER**

# PLAN

- I. Architecture des classes
- II. Initialisation du jeu
  - a) Plateau
  - b) Factory
- III. Fonctionnalités primaires
  - a) Cases
  - b) Joueur
  - c) Ennemi
  - d) Armes
  - e) Niveaux
- IV. Fonctionnalités secondaires du jeu
  - a) Editing et mode créatif
  - b) Sauvegarde
  - c) Autres
- V. Déroulement du jeu
- VI. Pour finir: améliorations, organisation et retours
  - a) Organisation
  - b) Défis et difficultés
  - c) Améliorations
  - d) Retour
- VII. Conclusion

# Introduction

Notre projet de programmation est intitulé "Shooter en vue de dessus". Ce projet a pour objectif de créer un jeu immersif où le joueur doit affronter des ennemis dans une série d'arènes, avec une vue de dessus (birdview) pour une perspective stratégique unique.

Nous avons choisi le thème "Home Invasion". L'histoire met en scène Sam, notre personnage principal, qui se trouve dans une maison envahie par des intrus. Sam doit éliminer tous les ennemis en parcourant les différentes pièces de la maison pour gagner. L'invasion commence lorsque Sam revient dans son village natal, des années après la mort de ses parents et découvre que sa maison a été infiltrée par un groupe de malfaiteurs. Armé de son courage, Sam doit se frayer un chemin à travers les intrus pour sécuriser chaque pièce de la maison, retrouver ses biens et protéger ce qui lui est cher. Les joueurs devront utiliser leur intelligence et leurs réflexes pour naviguer dans les environnements complexes, gérer leurs ressources (munitions) et survivre aux attaques incessantes des ennemis.

Le développement de ce projet a été réalisé en Java, en utilisant les bibliothèques Swing et AWT pour les éléments de l'interface graphique. Swing nous a permis de créer des interfaces utilisateur riches et interactives, tandis qu'AWT nous a offert des outils supplémentaires pour gérer les événements et les composants graphiques de manière efficace.

Dans les sections suivantes, nous détaillerons l'architecture du jeu, le processus d'initialisation, les fonctionnalités principales et le déroulement d'une partie. Ces informations vous donneront une compréhension approfondie de la structure et du fonctionnement de notre jeu, ainsi que des défis que nous avons relevés et des solutions que nous avons mises en œuvre pour créer une expérience de jeu captivante et fluide.

# I. Architecture de classes

Notre jeu est organisé avec un modèle MVC avec des classes model, view et controls. Nos classes sont catégorisées entre les package model, manager et GUI pour correspondre à la disposition de ce modèle.

Voici une description générale de cette organisation.

## **MODEL:**

La partie modèle comporte les classes qui gèrent les données du jeu et la définition des objets. Notre classe principale est la classe Game, qui comporte la fonction principale run et s'occupe de déclencher l'initialisation et la mise à jour de tous les autres éléments du jeu. Nous avons également une classe GameManager qui gère les fonctions plus complexes en lien avec l'objet Game.

Le plateau est l'objet qui représente la base du jeu en lui-même, composé de plusieurs instances de Case. Nos personnages sont divisés en ceux qui sont jouables (Player) et ceux qui ne le sont pas (Ennemi). Les ennemis sont spécialisés et ont des comportements différents.

Pour faciliter l'initialisation de certains de ces éléments, nous utilisons des classes "factory". De plus, nous avons des classes secondaires qui ne font pas partie directement du jeu mais offrent à l'utilisateur des fonctionnalités additionnelles telles qu'un mode d'édition ou une sauvegarde. (Plus de détails dans la partie sur les fonctionnalités du jeu.)

## **VIEW:**

Pour la partie graphique de notre jeu, on a différentes pages qui comportent le plateau de jeu, le mode d'édition, les paramètres, etc. Pour organiser ces pages, on utilise un CardLayout et on change la page affichée en fonction des actions de l'utilisateur et des conditions définies. Par exemple, le bouton [Play] est configuré pour afficher la page [PlayPage] lorsqu'il est actionné, en mettant cette page en avant dans le CardLayout.

Pour dessiner nos éléments graphiques, on utilise la bibliothèque Graphics, qui nous permet de créer des visuels personnalisés et interactifs. Cette approche facilite la gestion de l'interface utilisateur et permet une transition fluide entre les différentes sections du jeu, rendant l'expérience utilisateur plus cohérente et intuitive.

## **CONTROLS:**

Pour gérer le lien entre le modèle et la vue, on utilise des classes "manager" qui jouent un rôle central en faisant le pont entre la logique du modèle et l'interface utilisateur (UI).

Ces managers incluent des fonctions pratiques comme les calculs de trajectoire, les mises à jour des états des éléments du jeu, et d'autres opérations essentielles. Ils accèdent et modifient les données du modèle, tout en communiquant avec l'UI pour mettre à jour l'affichage graphique. En séparant clairement les responsabilités entre le modèle, la vue et les managers, on obtient un code plus modulaire et maintenable, facilitant la gestion, le débogage et l'extension du jeu.

En Annexe 1 et 2: Les schémas MVC et la relation entre les classes.

## II. Initialisation du jeu

Dans cette partie on va décrire un peu plus en détail comment sont composées et sont initialisés des éléments essentiels au jeu: le Plateau et les Factory d'Ennemis. On explique ces éléments en amont puisque c'est essentiel afin de comprendre la logique de nos autres classes et mécanismes de jeu.

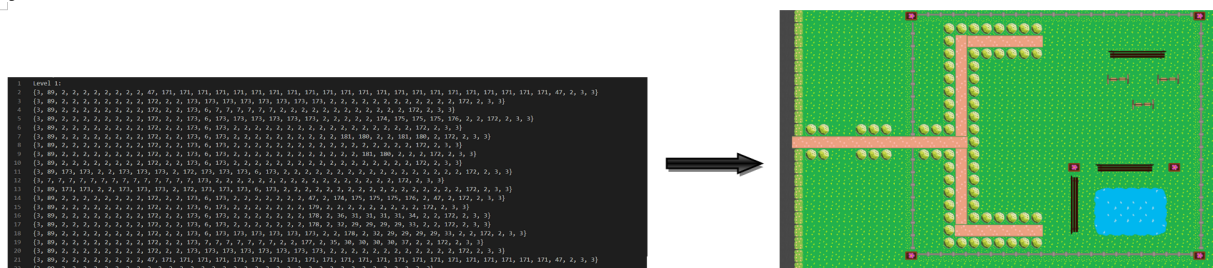
## PLATEAU

Le plateau est une grille rectangulaire de tuiles représentant l'arène de jeu, composée de 20x36 cases. Chaque case est un objet distinct avec une image, un mode de fonctionnement, un identifiant unique (ID) et un type spécifique.

Pour conserver les différentes dispositions du plateau à travers les niveaux, on utilise l'ID des cases. Un fichier texte (PlateauLevels.txt) contient des tableaux imbriqués d'entiers représentant les plateaux des différents niveaux.

Lors de l'initialisation et tout au long du jeu, le fichier PlateauLevels.txt est lu pour charger le bon niveau dans le plateau de jeu actuel à l'aide de `PlateauLevelLoader`. L'ID de chaque case est également utilisé pour afficher la bonne image sur le plateau de jeu. Chaque chiffre fait référence à une image dans `imagecombinee.png`, qui est un tableau d'images des cases du jeu. Avec `BufferedImage`, on découpe les images et les associe aux ID correspondants. Ainsi, lors de la lecture du fichier texte, chaque chiffre peut être associé à l'ID de l'image correspondante pour dessiner les niveaux. Nos images représentent des objets de la maison pour s'accorder avec notre thème.

Le type des cases introduit un élément interactif au jeu. En associant un type à chaque case, on peut influencer le comportement du joueur ou des ennemis, enrichissant ainsi la jouabilité.



## FACTORY

Pour faciliter la gestion des différents ennemis on comprend une "chaîne d'assemblage" qui permet de créer facilement de nouvelles variations d'un même élément : ce sont les "factory". C'est la mise en œuvre d'un patron de conception qui pourra être utilisé pour créer des instances. On utilise des factories pour créer nos différents ennemis. Cette approche nous permet de séparer la logique de création d'objets de leur utilisation effective dans le code. En utilisant une interface générale, `PersonnageFactory`, nous pouvons définir un ensemble de méthodes standard pour la création de personnages, telles que `createPersonnage()` et `getCoordinates()`. Ensuite, nous implémentons cette interface dans les factories spécifiques à chaque type d'ennemi.

Par exemple, dans la classe `Enemy1Factory`, nous implémentons la méthode `createPersonnage()` pour créer et retourner des instances de `EnemyBasique`, un type spécifique d'ennemi. Cette approche facilite l'ajout de nouveaux types d'ennemis dans notre jeu, car il suffit de créer une nouvelle classe de factory pour ce type d'ennemi en implémentant l'interface `PersonnageFactory`. Ainsi les factory rendent notre code plus modulaire, flexible et extensible.

On utilise le modèle de conception Factory ainsi qu'un mécanisme de chargement de niveau pour créer dynamiquement des ennemis en fonction des données chargées à partir d'un fichier. La classe `EnemyLevelLoader` est responsable du chargement des données d'ennemis pour un niveau spécifique à partir d'un fichier et de la création des instances d'ennemis en conséquence. On maintient une liste de Personnages dans `Game` pour stocker les factories pour les ennemis de chaque niveau. La méthode `loadLevelEnemies()` lit les données d'ennemis à partir d'un fichier, crée des factories pour les ennemis spécifiés pour le niveau en cours à l'aide de `createFactoriesForEnemyTypes()`, et les stocke dans la carte. La méthode `createEnemiesForLevel()` récupère les factories pour le niveau actuel à partir de la carte et crée des instances d'ennemis à l'aide de ces factories. En utilisant ce modèle, on peut définir des configurations d'ennemis pour chaque niveau dans un fichier et créer dynamiquement des instances d'ennemis pendant le jeu en fonction de ces configurations.

Schéma de fonctionnement des Factory Pattern en Annexe 3.

### III. Fonctionnalités Principales


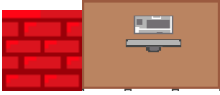

Cette section détaille les mécaniques clés du jeu et présente les différents éléments interactifs.

On va décrire les compositions des Cases du jeu, le fonctionnement du Joueur, le fonctionnement des Ennemis, l'implémentation des Armes et l'enchaînement des niveaux.

#### CASES






Nous avons implémenté des cases ayant des effets interactifs. Certaines cases bloquent le passage du joueur ou limitent la zone de déplacement, et d'autres cases influencent de manière significative le déplacement du joueur sur le plateau. Ce choix a été réalisé pour correspondre à une charge minimale au système tout en augmentant la complexité et l'intérêt du jeu.

Les cases qui ne changent pas le déplacement du joueur:

Nom	Type	Exemple
Simple	Permettent le déplacement sans effet particulier	le sol, l'herbe 
Indestructible	Ni le joueur ni les ennemis ne peuvent se déplacer dessus. Les munitions peuvent les traverser ou non, par exemple les murs et les éléments bas comme les lits.	Les murs porteurs (rouges), le bureau 
Destructible	Ni le joueur ni les ennemis ne peuvent se déplacer dessus, mais les munitions peuvent les détruire. Un effet de cassure apparaît, et la case devient un sol. Par exemple, les murs cassables ou les commodes.	Les cloisons (orange), les portes 



Les cases qui sont interactives avec le joueur:

Nom	Type	Exemple
Ralentissante	Ralentissent la vitesse de déplacement du joueur, ce qui peut permettre aux ennemis de le rattraper plus facilement.	Tapis 
Accélératrice	Accélèrent la vitesse de déplacement du joueur, lui permettant de s'échapper plus rapidement des situations dangereuses.	Les flaques de savons 
Blocage temporaire	Il va être bloqué pendant 3 secondes et il ne pourra pas tirer.	Pot de fleur 
Tournie	Les déplacements du joueur sont perturbés comme s'il glissait	Flaque d'eau 
Perte de point de vie	Lors de son passage sur la case, la vie va diminuer de 2 points	Bouts de verre 

Ces cases sont judicieusement réparties dans les niveaux pour créer des zones de danger et des passages stratégiques.

Par exemple, un couloir étroit avec des cases ralentissantes peut obliger le joueur à trouver une route alternative ou à utiliser une capacité spéciale pour traverser rapidement. De même, une zone entourée de cases qui diminuent la vie peut contenir le dernier ennemi à abattre pour gagner, ajoutant un risque calculé pour le joueur.

L'objectif de ces mécanismes est de forcer le joueur à planifier ses mouvements et à adapter sa stratégie en temps réel. En connaissant les effets des différentes cases, le joueur doit décider quand prendre des risques pour atteindre des objectifs ou éviter les pièges. Ces choix stratégiques enrichissent l'expérience de jeu et ajoutent de la profondeur à chaque niveau.

## JOUEUR

Le joueur à des attributs tels que la santé, une vitesse, une position ...

Au cours de la partie, le joueur pourra effectuer les actions suivantes pour jouer:

- Se déplacer à l'aide des touches directionnelles. Les déplacements seront entravés par des murs, des obstacles et des éléments de décor, ajoutant ainsi une dimension stratégique à la navigation dans les niveaux. Au cours de son déplacement, un mouvement va se dessiner par l'enchaînement continue de 3 images comme les praxinoscopes. Cela va donner l'illusion plus réelle d'un déplacement. De plus, pour accentuer le réalisme, des traces de pas ont été initiées. Son orientation se fait au travers du placement de la souris, qui prend l'image d'une cible. Il est possible de faire une rotation de 360°.
- Se défendre contre ses ennemis en utilisant une variété d'armes. Au travers de la cible qui illustre la souris, un système de tir de projectiles avec une gestion précise des trajectoires et des collisions, offrant ainsi aux joueurs un contrôle précis sur leurs attaques et interactions avec l'environnement. Il aura accès à un arsenal comprenant différentes armes, chacune ayant ses propres caractéristiques telles qu'une portée, une puissance et des capacités spéciales comme la capacité à détruire des murs ou à faire rebondir les projectiles ( cf Armes). En outre, le joueur pourra placer des armes au sol telles que des mines et des grenades, créant ainsi des stratégies défensives complexes avec des effets retardés qui influencent également le décor.
- Choisir son arme en appuyant sur la touche espace. Au fur et à mesure de sa progression dans le jeu, le joueur débloque une gamme d'armes de plus en plus variée, offrant ainsi une expérience de jeu toujours renouvelée. Une indication permanente sur l'arme équipée et le nombre de munitions restantes sera affichée à l'écran.



## ENNEMI

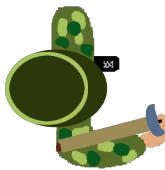


Les ennemis sont des Personnages avec des comportements différents et des statistiques de jeu spécifiques. Les objets ennemis eux-mêmes (classe Enemy) n'ont pas de comportement intrinsèque ; on définit leur comportement avec des fonctions basées sur un algorithme de flood-fill.

Le flood-fill est un algorithme de calcul du "plus court chemin". On initialise un tableau imbriqué d'entiers qui est un miroir de notre plateau. On place dans ce tableau la case correspondant à la position actuelle du joueur dans le plateau à 0. Les objets infranchissables comme les murs sont marqués avec une valeur élevée, par exemple 1000, dans le tableau de flood-fill. Ensuite, à partir de la position du joueur marquée par 0, on "remplit" le tableau avec les distances relatives au joueur. L'algorithme "inonde" le tableau et "l'eau" arrive à 0 en suivant le chemin le plus court défini par les distances que nous avons établies.

Ainsi, un ennemi compare sa position dans le plateau avec les possibilités de mouvement offertes par le flood-fill et se dirige vers la case avec la valeur la plus petite (la distance la plus courte). Ce processus est mis à jour dynamiquement : le tableau de flood-fill, la position du joueur et la position des ennemis changent au cours du jeu. Cela permet aux ennemis de réagir de manière dynamique et adaptative aux mouvements du joueur et aux changements dans l'environnement.

Les différents ennemis vont utiliser les informations du flood fill différemment pour avoir leur comportements distincts.

Ennemi Basique	<p>Il est statique mais a une grande portée de tir.</p> 
Ennemi Medium	<p>Il a un rayon de détection du joueur, qui lui permet de suivre le joueur quand il le détecte et de lui tirer dessus.</p> 

Ennemi IA	<p>Il poursuit le joueur dès le début de la partie et ne fait des dégâts que en corps à corps.</p> 
Ennemi Sniper	<p>Il se déplace vers le joueur, dès qu'il est à portée de tir sans obstacle (mur..) il s'arrête et tire.</p> 
Ennemi qui évite les balles	<p>Lorsque le joueur lui tire dessus, il va essayer d'éviter les balles sinon il s'approche du joueur et tire.</p> 





Les ennemis ont aussi un effet de mouvement de marche, de par l'enchaînement de trois images continue qui varie selon les éléments interactifs. Chaque ennemi aura l'accès à une unique arme pour se défendre.

## ARMES

Comme évoqué précédemment, les armes seront caractérisées par un nombre de munitions, une portée et parfois des capacités spéciales. Nos armes ne sont pas définies par une réelle construction d'un objet arme qui pourrait tirer des munitions. Elles se définissent plutôt par les munitions elle-même, qui apparaissent à partir du bras du joueur ou de l'ennemi. Ce choix a été fait, d'une part, pour faciliter l'implémentation. D'autre part, en vue de dessus, il est difficile de donner assez de détails pour bien différencier le style de l'arme sans nécessiter une gestion assez complexe d'un changement d'image.

Dans le jeu, on retrouve deux types d'armes : les projectiles et les pièges.

- *Projectiles* : Les joueurs peuvent tirer différents types de projectiles, chacun ayant ses propres caractéristiques et effets.

Pistolet	Arme de base qui permet de tirer 30 munitions à une petite portée infligeant 10 dégâts à l'ennemi, et permet de détruire les objets cassants.	
Bazooka	Arme à longue portée, plus développée, qui permet de tirer 10 munitions infligeant 35 dégâts à l'ennemi, et permet de détruire les objets cassants.	
Fusil d'assaut	Arme développée permettant de tirer 25 balles à très longue portée, infligeant 15 dégâts à l'ennemi. Pratique pour éliminer les ennemis avant qu'ils n'atteignent le joueur.	
Gun	Arme spéciale à longue portée contenant 20 munitions et infligeant 40 dégâts aux ennemis. Les balles de cette arme permettent de détruire les objets cassants, mais aussi de rebondir sur les murs afin de bénéficier de plusieurs points de tir.	

- *Pièges* : Les joueurs peuvent placer des pièges sur le plateau pour piéger les ennemis. On distingue les mines et les grenades, qui peuvent toutes deux être déposées sur le plateau, à l'exception des murs et des obstacles. Elles ne pourront pas être placées de l'autre côté d'un mur.
  - **Mines** : Infligent des dégâts aux ennemis comme au joueur s'ils passent dessus. Stratégiquement, elles peuvent être placées sur les chemins fréquemment empruntés par les ennemis.
  - **Grenades** : Infligent des dégâts aux ennemis et au joueur s'ils se trouvent à proximité après cinq secondes, lors de leur explosion. Idéales pour les attaques retardées et les zones de forte densité ennemie.

Au cours de la partie, l'arme que le joueur possède sera affichée dans un cadran d'information en haut à droite. Ce cadran contient le nom de l'arme et une image, permettant au joueur de se faire une idée de la puissance de l'arme. En outre, le joueur a accès au nombre de munitions restantes pour gérer sa consommation pendant la partie. Si le joueur descend à 10 munitions restantes, une alerte clignotante rouge signale le niveau critique, l'incitant ainsi à changer de stratégie ou d'arme.

L'idée de recharger des munitions au cours de la partie avait été envisagée. Cependant, après avoir construit les niveaux, nous nous sommes rendu compte que cela n'était pas nécessaire. Le nombre d'ennemis présents et d'éléments destructibles ont été pris en compte dans le réglage des armes. Il est donc tout à fait possible de gagner avec le nombre d'armes et de munitions données.

## LES NIVEAUX

Chaque niveau du jeu décrit l'environnement d'une maison, commençant par l'extérieur. Dans le mode normal, les joueurs traversent différents espaces de la maison dans l'ordre suivant ;

- le parc
- le jardin et l'entrée de la maison
- salle à manger
- chambre parentale
- chambre d'enfants
- chambre d'invités

Les niveaux sont envahis d'ennemis variés qui augmentent en difficulté à mesure que le joueur progresse dans les niveaux. Par exemple, dans le parc, les ennemis sont être plus faibles pour saisir les commandes du jeu, tandis que dans les niveaux intérieurs comme la cuisine ou le salon, les ennemis deviennent plus puissants et stratégiquement placés.

Certains niveaux peuvent également introduire des défis environnementaux, comme des pièges ou des obstacles dynamiques, qui ajoutent une couche supplémentaire de difficulté. Ou bien des structures interactives qui complexifient le jeu comme des portes à casser qui dévoilent des pièces “secrètes” ou sortir de la pièce où on est confiné. Ces fonctionnalités sont faites grâce aux cases décrites précédemment.

La condition de fin de niveau est remplie lorsque tous les ennemis présents sur le plateau ont été vaincus par le joueur. Une fois cette condition atteinte, le niveau est considéré comme terminé et le joueur peut passer au niveau suivant. Cette structure garantit une progression logique et continue, permettant au joueur de découvrir progressivement les différentes zones de la maison et de relever des défis de plus en plus complexes. La progression à travers les niveaux est conçue pour maintenir l'intérêt du joueur, avec une courbe de difficulté bien équilibrée.

Chaque niveau implique des mécaniques de jeu uniques et des défis spécifiques pour les joueurs à surmonter, offrant ainsi une variété d'expériences de jeu pour les joueurs. Cette section offre un aperçu détaillé des mécaniques de jeu, des éléments interactifs du jeu de shooter en vue de dessus, fournissant ainsi une base solide pour la compréhension de l'expérience de jeu proposée.

## IV. Fonctionnalités secondaires

Pour faciliter et rendre le jeu plus attractif, nous avons ajouté différentes fonctionnalités de notre jeu : un mode édition pour créer des niveaux, une sauvegarde pour que le joueur puisse revenir sur sa partie et différentes autres features. Nous allons ainsi nous intéresser à ces derniers.

### EDITING & MODE CRÉATIF

Le mode Editing est une fonctionnalité qui permet au joueur de créer son propre niveau, ajoutant ainsi une dimension supérieure au jeu et une continuité au jeu après avoir complété le mode normal. Il offre la possibilité au joueur de créer des plateaux à son goût, modifiant la difficulté pour leurs propres défis, créer sa propre histoire.

Ce mode a ainsi pour but de prolonger l'expérience du jeu pour les joueurs en quête de défi plus dure et ceux qui apprécient les personnalisations.

Les utilisateurs peuvent créer de nouveaux niveaux ou modifier des niveaux existants.

Pour ce mode, le plateau de jeu est au centre de la fenêtre, entouré de panneaux pour sélectionner les éléments du niveau.. Pour le modifier, le joueur a accès aux différentes cases, représentées sous forme de différentes listes : les murs, le sol, les décors et les obstacles. Cela permet au joueur de se retrouver parmi les multitudes cases créés. Le joueur peut également placer différents ennemis sur le plateau pour rendre son niveau intéressant.

La sauvegarde de niveau est possible seulement si au moins un ennemi a été placé. De plus, le joueur a accès au niveau précédemment créé ce qui lui donne la possibilité de modifier des niveaux personnalisés déjà existants ou de créer de nouveau niveau en utilisant les anciens niveaux en tant que Template.

Nous avons ainsi pu utiliser ce mode pour créer les différents niveaux pour le mode campagne/normal du jeu à l'aide d'un booléen utilisé pour déterminer les fichiers textes cibles pour les sauvegardes et modifications : les fichiers creativeTmp.txt et enemyTmp.txt pour le mode créatif et les fichiers PlateauLevels.txt et EnnemiesForLevels.txt pour le mode normal.

Pour afficher les anciens niveaux, on utilise les mêmes fonctions que pour le plateau du jeu : le fichier txt du Plateau est lu pour sélectionner le bon niveau et retranscrit en tableau à l'aide de PlateauLevelLoader et lorsque l'on sauvegarde, on ajoute à la suite du fichier txt le nouveau niveau.

On a ainsi un mode créatif dans lequel l'utilisateur joue les niveaux personnalisés qu'il l'a créé l'un à la suite. Dans ce mode, contrairement au mode normal, le fichier creativeTmp.txt est lu pour initialiser le plateau et le fichier enemyTmp.txt est lu pour la création des ennemis avec les factory. Cependant contrairement au mode normal, si le



joueur qui en pleine partie et qu'il revient dans le mode créatif, il doit rejouer à partir du niveau 1 : il s'agit d'un marathon de niveau avec l'accès à toutes les armes qu'il a débloquées.

Cela rend le mode plus attrayant en empêchant le joueur de finir tous ses niveaux en une seule fois.

Les modes Créatif et Édition enrichissent ainsi l'expérience de jeu en permettant aux joueurs de créer et de personnaliser leurs propres niveaux, augmentant ainsi la rejouabilité et offrant une plateforme pour la créativité. En plus de prolonger la durée de vie du jeu, ils permettent également d'explorer de nouvelles façons de jouer.

## **SAUVEGARDE**

La mise en place de la sauvegarde permet d'offrir une expérience de jeu fluide et immersive aux joueurs. Une sauvegarde a été créée pour permettre au joueur de reprendre là où il s'est arrêté et de retrouver les niveaux qu'il a créés. Elle leur permet de retrouver facilement leur progression, ce qui permet de maintenir leur intérêt et les encourager à continuer à jouer. De plus, la possibilité de sauvegarder leurs propres niveaux créés permet de valoriser leur créativité et leur investissement dans le jeu.

En intégrant cette fonctionnalité dans les paramètres du jeu, les joueurs peuvent aisément enregistrer leur partie à tout moment, offrant ainsi une flexibilité appréciable. Cette sauvegarde génère trois fichiers texte au nom du joueur. Ces fichiers contiennent des données pour la progression dans le jeu, notamment les informations relatives aux niveaux créatifs, et la liste des ennemis et d'autres éléments nécessaires à la continuité de l'expérience de jeu. La création de ces trois fichiers texte garantit une sauvegarde complète et détaillée de leur progression, y compris les niveaux qu'ils ont personnalisés.

Le fichier .ser (abréviation de "serialized") est utilisé pour stocker l'état d'un objet en Java afin qu'il puisse être sauvegardé et restauré ultérieurement.

Lorsque le joueur enregistre sa progression, le programme crée un fichier .ser qui contient une version sérialisée de l'objet Player. La sérialisation convertit l'état de l'objet en un format binaire qui peut être écrit sur le disque. Ce fichier contient toutes les données nécessaires pour recréer l'objet Player plus tard, comme le nom du joueur, son niveau actuel, ses scores, et autres informations pertinentes. Lorsque le joueur souhaite reprendre sa partie, le programme lit le fichier .ser et déséréalise les données pour recréer l'objet Player. La déséréalisation est le processus inverse de la sérialisation. Elle reconstruit l'objet Player en mémoire avec toutes ses données et son état tel qu'il était au moment de la sauvegarde.

Lorsque le joueur se « reconnecte » à la page d'accueil, il a la possibilité de charger son ancienne partie en utilisant ces fichiers, lui permettant ainsi de reprendre là où il s'était arrêté. La possibilité de charger facilement leur ancienne partie renforce le sentiment de continuité et de familiarité pour les joueurs. Cela leur évite de devoir recommencer depuis le début, ce qui peut être frustrant et décourageant, et leur permet de se plonger

directement dans l'action là où ils l'avaient laissée. En fin de compte, cette fonction de sauvegarde améliore considérablement la satisfaction globale du joueur et contribue à rendre l'expérience de jeu plus engageante et gratifiante.

## **AUTRES :**

Pour améliorer l'expérience du jeu, des musiques de fond ont été ajoutées au jeu : une pour l'introduction du lore du jeu et une deuxième correspondant à celle du jeu. Elle est peut être activés/désactivés dans les settings. Cela permet de rendre le jeu plus immersif.

De plus, le jeu, comme expliqué dans l'introduction, possède du lore.

L'introduction de l'histoire de notre personnage plonge les joueurs dans le contexte et l'univers du jeu. Elle permet de créer une immersion dès les premières minutes, captant l'intérêt et la curiosité des joueurs. En retraçant l'histoire du protagoniste et les circonstances actuelles du jeu, l'histoire établit une connexion émotionnelle entre le joueur et le personnage principal.

Le scénario d'introduction du jeu a été conçu pour être à la fois engageant et informatif. L'objectif est de fournir une transition entre la réalité du joueur et le monde virtuel du jeu. L'introduction narrative raconte l'histoire de Sam, le protagoniste, qui revient dans son village natal après plusieurs années d'absence, seulement pour découvrir que celui-ci est envahi. Cette trame de fond pose les bases de la motivation du personnage à combattre les envahisseurs, et par extension, motive le joueur à s'impliquer dans l'aventure.

Le texte de l'histoire est affiché progressivement, caractère par caractère, créant une lecture immersive. Des images de fond illustrent chaque étape de l'histoire, renforçant l'immersion visuelle. Un bouton "Skip intro" apparaît après quelques secondes, permettant aux joueurs impatients de passer directement au jeu.

L'affichage du texte se fait dans un panneau semi-transparent, assurant une bonne lisibilité tout en permettant de voir les images de fond. La progression de l'histoire est gérée par des indices pour les phrases et les images, ainsi que par des minuteurs pour synchroniser le texte et les transitions visuelles. Une fois l'histoire terminée, le jeu passe automatiquement à la scène suivante, sauf si le joueur a choisi de sauter l'introduction.

## V. Déroulement (gameplay écrit)

Une fois on a exécuté le fichier bash, voici en ordre ce qu'il se passe:

1. Page de présentation
2. Narration de l'histoire d'origine
3. Page de Login
  - a. création de compte
  - b. connexion à compte
4. Menu
5. Play (Normal)
  - a. Niveau est affiché dans PlayPage, les armes sont chargées, le joueur et les ennemis sont placés
  - b. Le joueur peut se déplacer et tirer sur les ennemis
  - c. Le joueur peut gagner le niveau
    - enchaînement au prochain niveau
    - le joueur peut débloquent des armes au fur et à mesure
  - d. Le joueur peut perdre le niveau
    - affichage de Game Over et retour au Menu
6. Mode édition: le joueur peut créer son propre niveau ou modifier les niveaux qu'il a déjà créé
  - Ces niveaux peuvent être joué en appuyant sur [ Créatif ] dans Menu
7. Créative
  - a. Niveau personnalisé est affiché, les armes sont chargées, le joueur et les ennemis sont placés
  - b. Le joueur peut se déplacer et tirer sur les ennemis
  - c. Le joueur peut gagner le niveau
    - enchaînement au prochain niveau
    - le joueur peut débloquent des armes au fur et à mesure
  - d. Le joueur peut perdre le niveau
    - affichage de Game Over et retour au Menu
8. Menu
  - le joueur peut relire les règles du jeu
  - regarder les stat de jeu en appuyant sur [Game Info]
  - sauvegarder la session en appuyant sur [Sauvegarde]

## **VI. Pour finir: améliorations, organisation et retours**

### **ORGANISATION**

Le processus de développement a suivi les étapes suivantes :

1. Conception : Cette phase a débuté par une analyse approfondie des besoins, suivie de la définition des spécifications et de la création de maquettes pour visualiser le jeu. Ces maquettes ont permis de clarifier et de valider les concepts avant de passer à l'étape suivante.
2. Implémentation : Ensuite, nous avons développé les fonctionnalités selon les spécifications établies, en adoptant des méthodes agiles pour favoriser la flexibilité et l'adaptabilité. Cette approche a permis des itérations rapides et des ajustements en cours de route.
3. Tests : À chaque étape du développement, nous avons validé les fonctionnalités implémentées, en mettant particulièrement l'accent sur les tests. Cela a garanti la stabilité et la fiabilité du jeu tout au long du processus.

### **Répartition du Temps sur les 12 Semaines du Projet**

Semaines 1-6 : Pendant cette période, nous nous sommes concentrés sur la conception initiale du jeu. Cela inclut l'élaboration du concept, la définition des fonctionnalités principales, et la planification du développement. Ensuite, nous avons entamé la phase d'implémentation en travaillant sur les fonctionnalités de base telles que la génération de l'arène, les mécaniques de déplacement et de tir, ainsi que la mise en place des ennemis et des obstacles. À ce stade, le jeu était une version de base sans effets graphiques avancés, avec des joueurs et des ennemis représentés par des ronds et des couleurs différenciant les différentes cases de base. Pour cette première partie, l'ensemble de l'équipe travaillait sur le code.

Semaines 7-9 : Durant ces semaines, nous avons implémenté l'interface graphique et ajouté des fonctionnalités avancées telles que le comportements intelligents des Ennemis (flood fill). Cette phase a permis au jeu de prendre sa forme finale avec une interface utilisateur complète et fonctionnelle. Le travail était alors plus interne. Nous avons décidé de diviser l'équipe en deux parties : une partie se concentrant sur l'interface graphique avec Majoori, Ines, et Eléacine, et une autre sur les mécanismes de jeu avec Leonor et Marylou.

Semaines 10-12 : Enfin, nous avons consacré les dernières semaines à la finalisation du projet. Cela comprenait des tests exhaustifs, le débogage, l'optimisation des performances, et la préparation de la documentation finale. Ces étapes ont assuré que le jeu soit prêt pour une utilisation stable et sans bugs, tout en étant bien documenté pour les utilisateurs.

En suivant cette répartition du temps, le projet a pu progresser de manière structurée et efficace, garantissant un produit final de haute qualité.

## **DÉFIS ET DIFFICULTÉES**

La réalisation de notre projet de jeu vidéo a été ponctuée de nombreuses difficultés et défis qui ont mis à l'épreuve notre gestion du temps, notre compétence technique et notre capacité à collaborer efficacement. À travers cette section, nous examinerons en détail les obstacles les plus significatifs auxquels nous avons dû faire face, ainsi que les solutions que nous avons mises en place pour les surmonter. De la gestion des délais aux problèmes de compatibilité, en passant par les complexités techniques et la création de contenu, chaque étape du développement présente ses propres défis uniques.

### **Contraintes de Temps**

La principale difficulté rencontrée a été le respect des délais impartis pour chaque phase du projet, exacerbée par le rythme de nos deux licences. Le défi de gérer efficacement notre temps entre les obligations académiques et le développement du jeu a été constant. Pour y remédier, nous avons mis en place une série de réunions régulières pour surveiller les progrès, réévaluer les priorités et ajuster le planning si nécessaire. Cette approche proactive a permis de maintenir le cap malgré les contraintes de temps strictes.

### **Problèmes de Compatibilité**

Adapter le jeu pour qu'il fonctionne sur différentes plateformes, notamment entre Mac et Windows, a posé des défis uniques. Les erreurs présentes sur certains PC et absentes sur d'autres ont nécessité une grande patience et une approche méthodique. Nous avons dû aborder chaque problème individuellement pour s'assurer qu'ils étaient résolus sur chaque type d'ordinateur, sans faire de concessions sur la qualité. Cette démarche a impliqué des tests rigoureux et des ajustements spécifiques pour garantir une expérience utilisateur homogène sur toutes les plateformes.

### **Défis Techniques**

La réalisation du flood fill a été d'une complexité notable. Ce n'était pas tant la compréhension de l'algorithme qui posait problème, mais plutôt les réglages nécessaires pour l'intégrer correctement avec les graphismes. Pendant une période prolongée, les ennemis ne détectaient pas les obstacles et passaient au-dessus, malgré des

HANON Marylou, HUSSON Eléacine, ZENATI Ines, SOL Léonor, VIJAYAKUMAR Majoori 21

coordonnées théoriquement correctes. Grâce à des études approfondies et de nombreux tests, nous avons finalement résolu ce problème. Un problème similaire s'est présenté avec le joueur, mais il a également été résolu avec succès grâce à une approche collaborative et persistante.

## Création de Contenu

Créer des graphismes visuels (personnages, environnements, animations) cohérents et esthétiquement plaisants a demandé du temps et des compétences artistiques considérables. Par exemple, la conception des personnages a nécessité plusieurs itérations avant d'atteindre un style visuel satisfaisant. Développer une histoire captivante et des dialogues intéressants a également représenté un défi. Cette tâche a nécessité une bonne compréhension de la narration interactive et des retours constants des testeurs pour affiner le contenu. La collaboration étroite entre les artistes et les développeurs a été cruciale pour créer un univers de jeu immersif et engageant.

Les défis rencontrés lors de la réalisation de notre jeu vidéo ont été nombreux et variés. De la gestion du temps aux problèmes de compatibilité, en passant par les défis techniques et la création de contenu, chaque obstacle a été surmonté grâce à une combinaison de planification rigoureuse, de tests approfondis et de collaboration efficace. Ces expériences ont non seulement renforcé nos compétences techniques et artistiques, mais ont également solidifié notre capacité à travailler en équipe sous pression. En fin de compte, ces défis ont contribué à la création d'un jeu vidéo plus robuste, plus engageant et mieux conçu.

## AMÉLIORATIONS

Pour maximiser son potentiel et garantir une expérience de jeu captivante et immersive, nous devons envisager plusieurs améliorations. Bien que nous n'ayons pas encore reçu de retours d'utilisateurs, nous pouvons nous appuyer sur notre analyse interne et les tendances actuelles de l'industrie pour identifier les domaines clés où des améliorations pourraient enrichir l'expérience de jeu. Ces suggestions visent à augmenter la complexité et le défi du jeu, améliorer la jouabilité et les graphismes. Voici un aperçu des améliorations potentielles que nous pourrions mettre en œuvre.

- **Ajout de Nouveaux Niveaux** : Nous pourrions enrichir le jeu en ajoutant de nouveaux niveaux avec des arènes plus complexes et des défis plus difficiles. Cela offrirait aux joueurs une expérience plus variée et stimulante, augmentant ainsi la longévité et l'intérêt du jeu.
- **Introduction de Nouveaux Personnages Jouables** : L'ajout de nouveaux personnages de type ennemi, chacun doté de compétences uniques,

enrichirait considérablement l'expérience de jeu. En créant de nouveaux algorithmes de flot fill, nous pourrions développer des ennemis plus performants et complexes, augmentant ainsi la difficulté et l'attrait du jeu. Par exemple, nous pourrions introduire des ennemis armés de munitions rebondissantes, ajoutant une dimension stratégique à l'évitement et à la défense. De plus, un ennemi capable de suivre les traces de pas du joueur offrirait un défi supplémentaire, nécessitant une réflexion tactique accrue et une adaptation constante de la part des joueurs. Ces innovations rendraient le gameplay plus dynamique et engageant, tout en augmentant la rejouabilité et l'intérêt à long terme.

- **Amélioration des Mécaniques de Jeu** : Optimiser les mécaniques de jeu existantes pour une jouabilité plus fluide et plus immersive améliorerait significativement l'expérience utilisateur. Une attention particulière pourrait être portée à la réactivité des contrôles, à la fluidité des animations et à l'équilibrage des niveaux de difficulté
- **Ajout d'une Boutique et de la Monétisation** : Intégrer un système de monétisation en parallèle avec une boutique en jeu permettrait aux joueurs d'acheter des bonus de vie, des nouvelles munitions ou des améliorations de capacités. De plus, permettre le changement de personnage principal ajouterait une dimension de personnalisation et d'optimisation stratégique au jeu
- **Effets Graphiques Avancés** : L'intégration d'effets visuels avancés tels que des effets de particules, des ombres dynamiques et des animations fluides renforcerait considérablement l'immersion des joueurs. En augmentant le nombre de cases interactives dotées de ces effets, comme des cases de téléportation pour le joueur, nous pourrions ajouter une dimension de surprise et de stratégie supplémentaire. Ces améliorations visuelles rendraient le jeu non seulement plus attrayant, mais aussi plus engageant, contribuant ainsi à une expérience de jeu plus riche et captivante.
- **Multijoueur Local** : L'ajout de la fonctionnalité multijoueur local permettrait à plusieurs joueurs de s'affronter sur la même console ou sur des appareils connectés en réseau. Bien que les premiers tests n'aient pas été concluants, retravailler cette fonctionnalité pourrait offrir une dimension sociale et compétitive au jeu, augmentant ainsi sa rejouabilité et son attrait pour un public plus large

- **Flood fill bis:** On a essayé d'implémenter des flood fill alternatives pour encore plus dynamiser le comportement des ennemis. Mais on a pas pu avancer à cause de certains problèmes de liaison entre l'algorithme de model et ce qu'il se passait dans le jeu. Plus précisément on n'as pas réussi à calculer les trajectoires de tir possibles par rapport à l'orientation de l'ennemi lors de l'essai d'un flood fill "sniper".

## RETOUR

Notre expérience dans l'élaboration du jeu vidéo a été à la fois enrichissante et formatrice, nous offrant de précieuses leçons sur la planification, la communication, et l'adaptabilité. La rigueur de notre planification a été un pilier central de notre progression, nous rappelant l'importance d'estimer correctement les ressources et de définir des jalons clairs pour suivre les progrès. De plus, une communication ouverte et transparente entre les membres de l'équipe s'est avérée essentielle pour résoudre rapidement les problèmes et maintenir la motivation. Cette collaboration étroite a favorisé l'échange d'idées et la résolution de problèmes complexes, renforçant ainsi notre cohésion d'équipe. Notre capacité à être agile et adaptable a également été mise à l'épreuve, nous apprenant l'importance de rester flexibles face aux défis imprévus et de trouver des solutions créatives pour les surmonter.

En ce qui concerne nos points forts, la conception créative du jeu avec son thème et ses mécaniques de jeu engageantes a été un atout majeur. De plus, l'utilisation efficace de techniques telles que le floodfill et les factory a permis d'optimiser le développement et d'améliorer l'efficacité de notre travail. L'engagement de l'équipe dans la réalisation du projet a également été un point fort, avec une collaboration étroite et une communication efficace tout au long du processus.

Cependant, nous avons également identifié certains points faibles qui méritent d'être adressés. Certaines fonctionnalités n'ont pas été implémentées selon nos attentes initiales en raison de contraintes de temps, ce qui a souligné la nécessité d'une meilleure gestion des délais. En effet, une perte de motivation face aux différents échecs, ajoutée à la pression, a engendré un retard de trois semaines. Une fois les solutions trouvées à différents problèmes nous avons réussi à reprendre un rythme convenable. De plus, la qualité du code pourrait être améliorée dans certaines parties du projet pour une maintenance future plus facile, soulignant l'importance de l'attention aux détails et de la recherche constante de l'excellence dans notre travail. En fin de compte, notre expérience dans l'élaboration du jeu vidéo nous a permis de grandir en tant qu'individus et en tant qu'équipe, nous préparant à relever de nouveaux défis avec confiance et détermination.



# Conclusion

Dans ce rapport, nous avons exploré en détail le processus de développement du jeu de shooter, en mettant en avant les différentes phases du projet, les éléments clés de conception et les défis rencontrés en cours de route. Nous avons également discuté des améliorations apportées au projet initial, de l'organisation du travail et des leçons apprises tout au long du processus de développement.

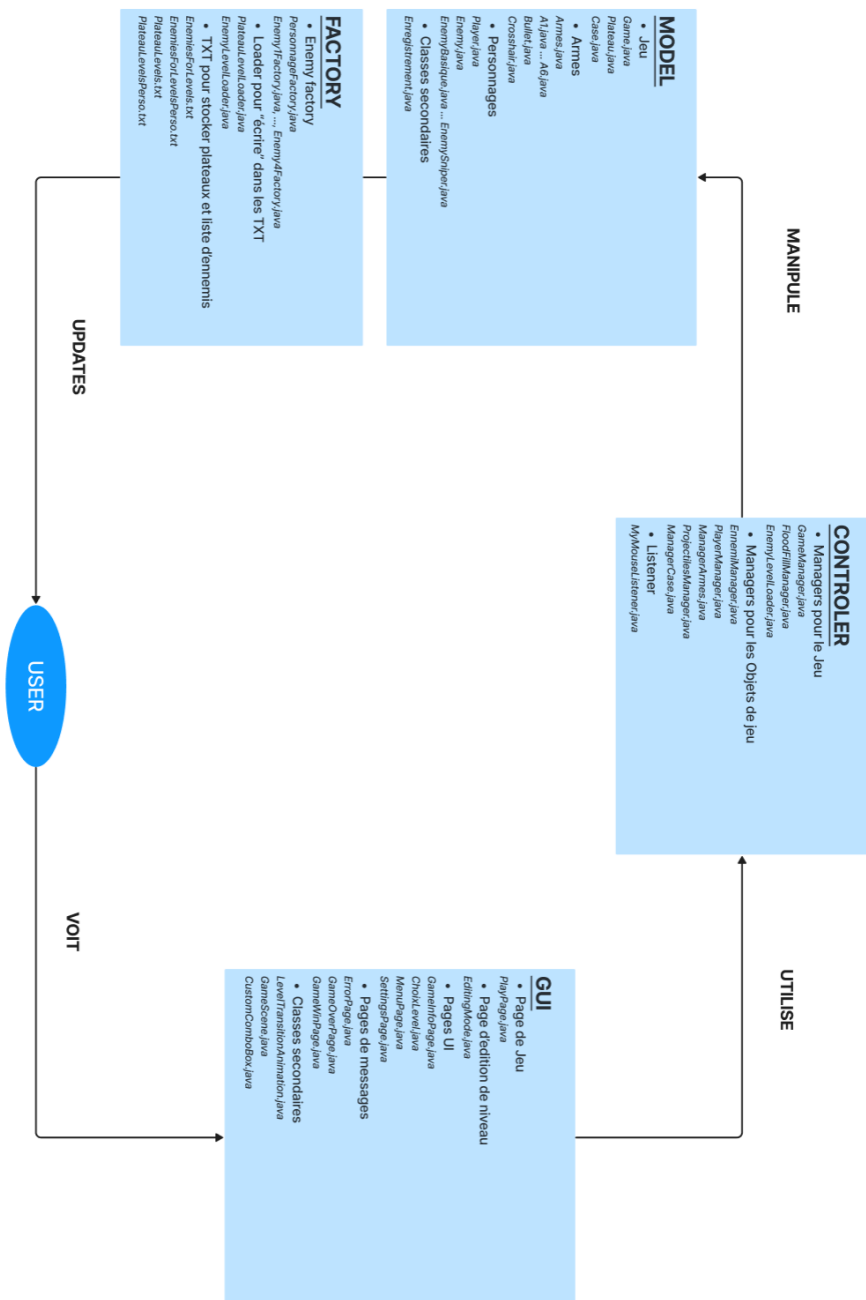
L'expérience de développement du jeu a été à la fois stimulante et enrichissante. Nous avons été confrontés à des défis techniques et organisationnels, mais grâce à notre détermination, notre collaboration et notre persévérance, nous avons réussi à surmonter ces obstacles et à livrer un projet dont nous sommes fiers. Cette expérience nous a permis de développer nos compétences techniques, notre esprit d'équipe et notre capacité à gérer des projets complexes.

Nous tenons à exprimer nos sincères remerciements à tous les membres de l'équipe pour leur dévouement, leur travail acharné et leur contribution précieuse à ce projet. Nous remercions également nos encadrants pour leur soutien, leurs conseils et leur expertise tout au long du processus de développement. Sans leur contribution, ce projet n'aurait pas été possible.

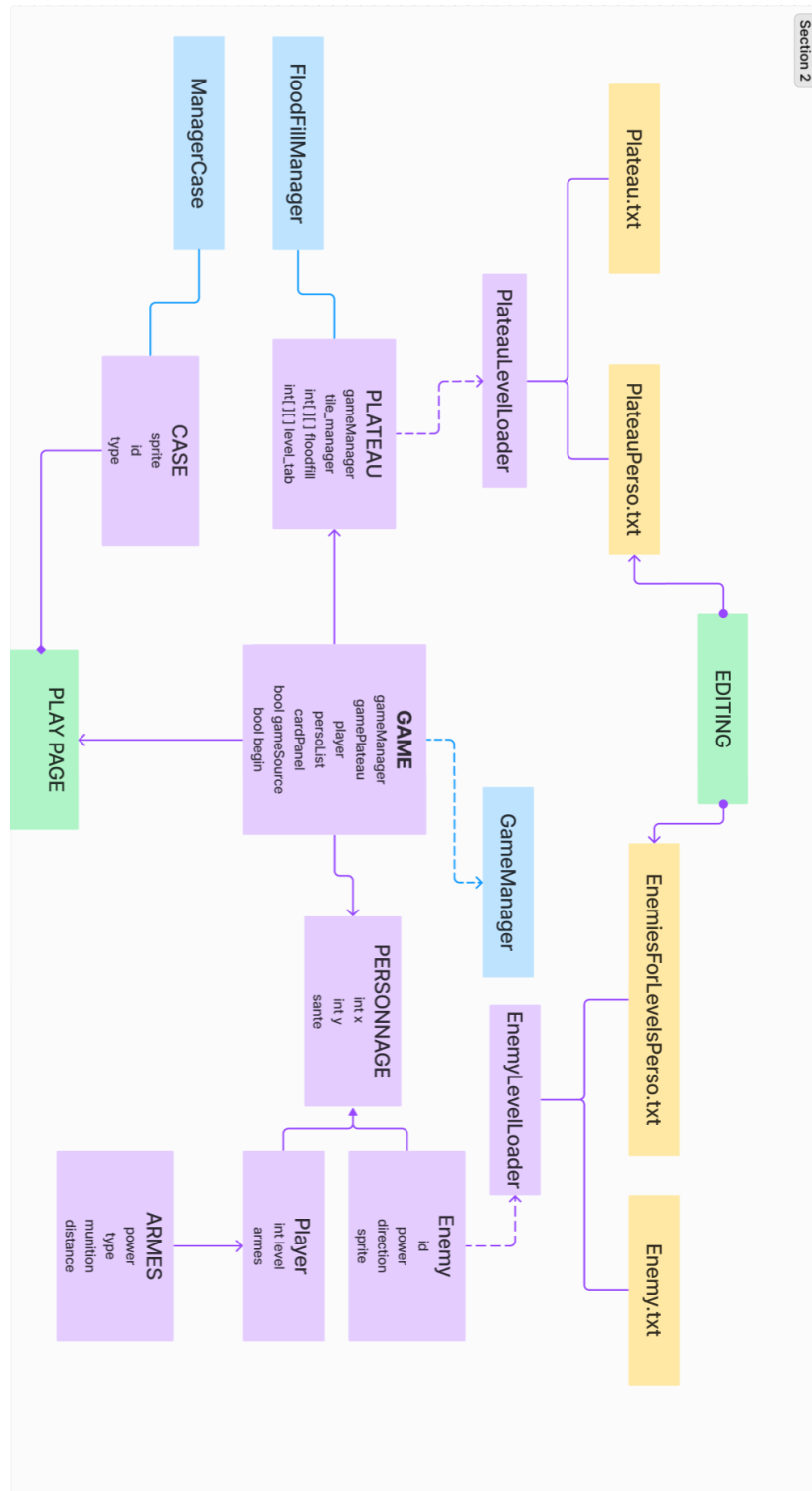
En conclusion, le développement du jeu de shooter en vue de dessus a été une expérience inoubliable qui nous a permis de repousser nos limites, d'apprendre de nouvelles compétences et de créer quelque chose dont nous sommes fiers. Nous espérons que ce jeu apportera du plaisir aux joueurs, tout comme il nous a apporté du plaisir et de la satisfaction pendant son développement.

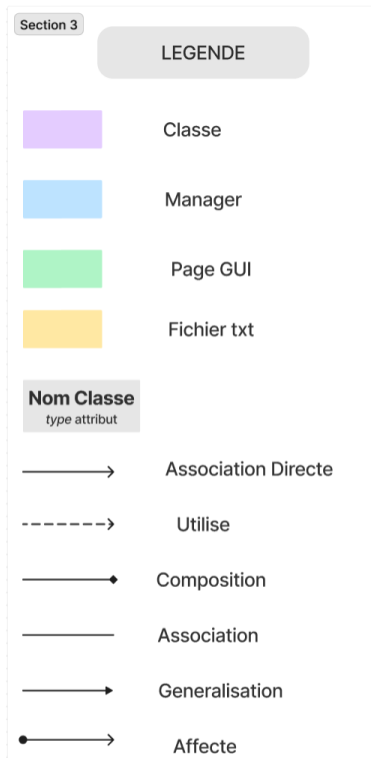
# **ANNEXE**

## **Annexe 1: MVC (non exhaustif)**



## Annexe 2: Relation entre classes (non exhaustif)





## Annexe 3 : Utilisation de factory pattern

