INFO-F209 - Projets d'informatique 2
Software Requirements Document

L-type

ABDOUL-AZIZ Aïssa
ADEGNON Kokou
BARBER Jeremy
DEMIREL Helin
ELKENZE Camelia
KINSOEN Alexandre
LATOUNDJI Salim
MASSIMETTI Mario
VANNESTE Martin

Mars 2021

Table des matières

1 Introduction

L'objectif de ce projet consiste en la réalisation d'un jeu d'action de style shoot 'em up en multijoueur. Dans ce jeu, un ou deux joueurs doivent parcourir plusieurs niveaux en détruisant les ennemis qui se présentent devant eux, tout en esquivant les tirs provoqués par ces derniers. Les vaisseaux dirigés par les joueurs peuvent récupérer des bonus d'armement lâchés par leurs nombreux adversaires, pour mieux les éliminer. L'objectif des joueurs étant de terminer tous les niveaux sans que leur compteur de vies ne se retrouve à 0. En effet, un joueur en possède un nombre déterminé. Si un projectile d'un joueur touche un ennemi, son score est augmenté. À la fin d'une partie, le score de chaque utilisateur est mis à jour si celui-ci est meilleur que son score actuel.

En dehors du jeu, un utilisateur a la capacité de gérer sa liste d'amis et de consulter le classement général des joueurs.

Le jeu ne sera exécutable que sous le système d'exploitation Linux.

1.1 Historique

Dates	Sujets	Noms
13/11/20	UseCase Utilisateur	Camelia, Jeremy, Salim
22/11/20	Annexe	Jeremy, Camelia
10/12/20	Besoins utilisateur	Kokou, Camelia, Helin, Aissa
11/12/20	Version finale diagramme de classe	Jeremy, Martin, Salim, Alexandre
14/12/20	Version final des diagrammes de sequence	Helin, Aissa, Martin, Kokou, Camelia, Mario, Alexandre, Jeremy, Salim
15/12/20	Introduction du SRD	Helin, Aissa, Mario
15/12/20	Besoins système partie serveur	Alexandre, Jeremy, Martin, Camelia, Aissa, Helin
15/12/20	Besoins système partie client	Kokou, Mario, Salim
16/12/20	Description du diagramme de classe	Helin, Mario, Salim, Aissa, Alexandre, Jeremy, Martin

03/02/21	Modifications concernant le SRD	Helin, Mario, Salim, Aissa, Alexandre, Jeremy, Martin, Kokou, Camelia
15/02/21	Modification diagrammes de classe jeu	Alexandre, Mario, Salim
26/02/21	Modification diagrammes de classe jeu, client et serveur	Aissa, Jeremy, Helin, Camelia, Martin
10/03/21	Dernières vérifications	Kokou

2 Besoins utilisateur

2.1 Besoins fonctionnels

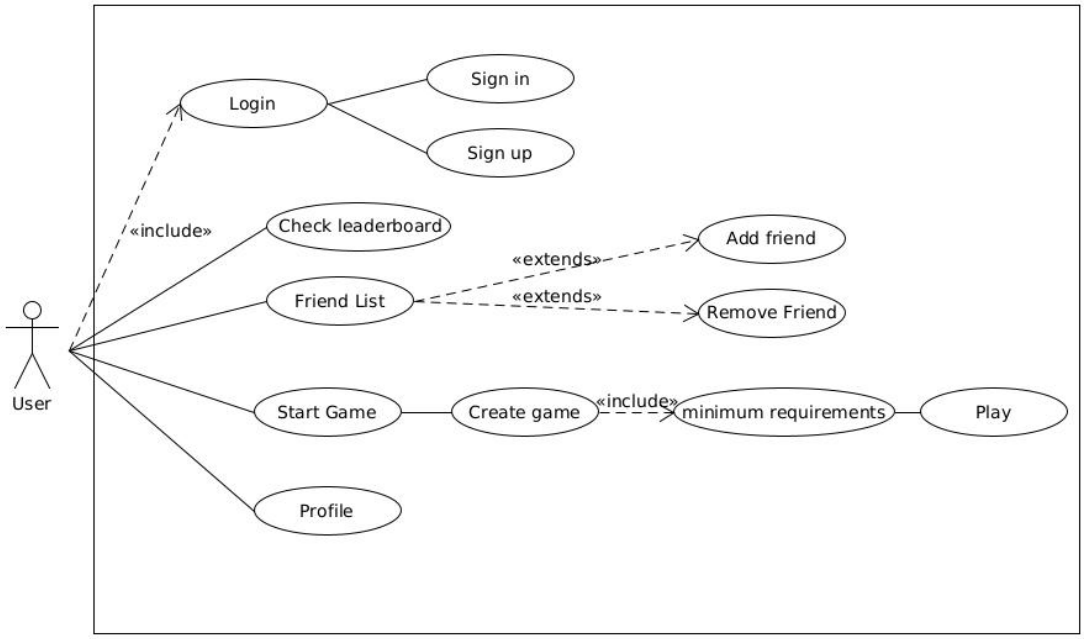

FIGURE 1 – Diagramme de use case côté utilisateur

2.1.1 Connexion

En lançant le programme, l'utilisateur est invité à s'inscrire ou se connecter.

A l'inscription, un pseudonyme unique et un mot de passe lui sont demandés tant que le pseudo entré est déjà pris par quelqu'un d'autre.

Dans le cas de la connexion, on invite l'utilisateur à saisir son pseudo et son mot de passe tant que le pseudo est inexistant ou que le mot de passe ne correspond pas. Après s'être connecté, l'utilisateur accède au menu principal.

Dans les deux cas, une option de retour à la page d'accueil sera disponible.

2.1.2 Menu principal

- Gestion des amis :

Dans cette option, l'utilisateur peut consulter sa liste d'amis et voir le score de ces derniers. Il à la possibilité d'envoyer des demandes d'amitié, en accepter ou en

refuser. L'utilisateur peut également supprimer des personnes de sa liste d'amis.

- Consulter le classement :

Le classement affiche tout les utilisateurs de l'application ainsi que leurs scores. La liste est trié du joueur ayant le plus haut score à celui avec le plus bas. Cette option permet au client de prendre connaissance des pseudonymes des autres pour envoyer des demandes d'amitiés.

- Lancer une partie :

Lorsque l'utilisateur souhaite créer une partie, il se trouve dans un lobby. Il peut y modifier les paramètres par défaut du jeu et inviter un second joueur. La partie est lancée quand l'hôte appuie sur play.

- Profile :

Dans le menu profile, l'utilisateur peut consulter ses informations de compte s'il le souhaite. A savoir, son pseudo et son score.

2.1.3 Création de partie

La création d'une partie est une option qui envoie l'utilisateur dans un lobby permettant de modifier les paramètres du jeu. Cette fenêtre contient déjà des paramètres par défaut.

S'il le souhaite, l'utilisateur peut modifier les options suivantes :

- Le nombre de joueurs :

L'utilisateur a la possibilité de choisir entre un ou deux participants. Dans le cas où le nombre de participants équivaut à deux, la deuxième personne est invitée à se connecter.

- La difficulté de la partie :

L'utilisateur doit choisir le niveau de difficulté de sa partie. La difficulté de chaque niveau sera adaptée en fonction de ce choix, mais elle augmente progressivement au fur et à mesure, après chaque victoire de niveau.

- Le tir allié :

La possibilité d'activer le tir allié ne peut être accordée que dans le cas où le nombre de joueurs est supérieur à un. Le joueur aura donc le droit de choisir s'il souhaite que les projectiles de l'invité soient inoffensifs ou non et inversement. Cette option active aussi la collision entre les joueurs.

- Le nombre de vies :

Le choix du nombre de vies est limité à 3 pour l'utilisateur. Il peut donc seulement la diminuer.

- Bonus :

Le joueur peut choisir la probabilité d'apparition des bonus. Ainsi il peut rendre le jeu plus ou moins difficile.

2.1.4 Jeu

- Partie en cours :

Elle reçoit les actions effectuées par les joueurs à travers le serveur et les applique au jeu. Lors de sa création, elle possède les paramètres choisis par le joueur ainsi qu'un identifiant. Ce dernier permet au serveur de déterminer a quelle partie il doit envoyer l'action d'un utilisateur.

Le joueur commence chaque partie avec une à trois vies constituée(s) de 100 points de vie et contrôle un vaisseau avec lequel il peut tirer des projectiles vers des ennemis, ce qui augmentera son score à chaque tir atteignant sa cible. Sa quantité de points de vie est réduite lorsqu'il subit des dégâts. S'il n'a plus de points de vie, le joueur perd une vie. Lorsque le joueur n'a plus de vie, son vaisseau est détruit et si plus aucun joueur n'a de vie, la partie se termine. Elle peut également s'arrêter en cas de déconnexion d'un utilisateur.

- Niveau :

La partie est divisée en 5 niveaux de difficulté croissante. Après les 5 niveaux, le(s) joueur(s) devra/ont faire face à un Boss pour terminer la partie. En passant au niveau suivant, la santé, les dégâts et le nombre d'ennemis accroissent.

- Entités à l'écran :

Parmi les entités apparaissant a l'écran et pouvant se déplacer, il y a les vaisseaux, les bonus, les projectiles et les obstacles.

• Ship : Dans ce jeu, il existe 3 types de ships : les PlayerShip, les EnemyShip et les Boss. Tous ces vaisseaux peuvent se déplacer dans toutes les directions, ainsi que subir et infliger des dégâts. Les PlayerShip sont les vaisseaux alliés, ce sont ceux que les joueurs contrôlent. Les EnemyShip peuvent lâcher un bonus en se détruisant, la probabilité de lâcher ce bonus est déterminée par l'utilisateur avant de lancer la partie. Le Boss est l'ennemi final a affronter pour finir les niveaux.

• Projectile : Les projectiles sont créés lorsque les vaisseaux tirent. Ils peuvent sortir de l'écran, s'annuler en rencontrant d'autres projectiles ou causer des dégâts en atteignant leur cible.

• Bonus : Les bonus sont de plusieurs types et peuvent rapporter des améliorations d'armes et de santé au joueur qui réussit

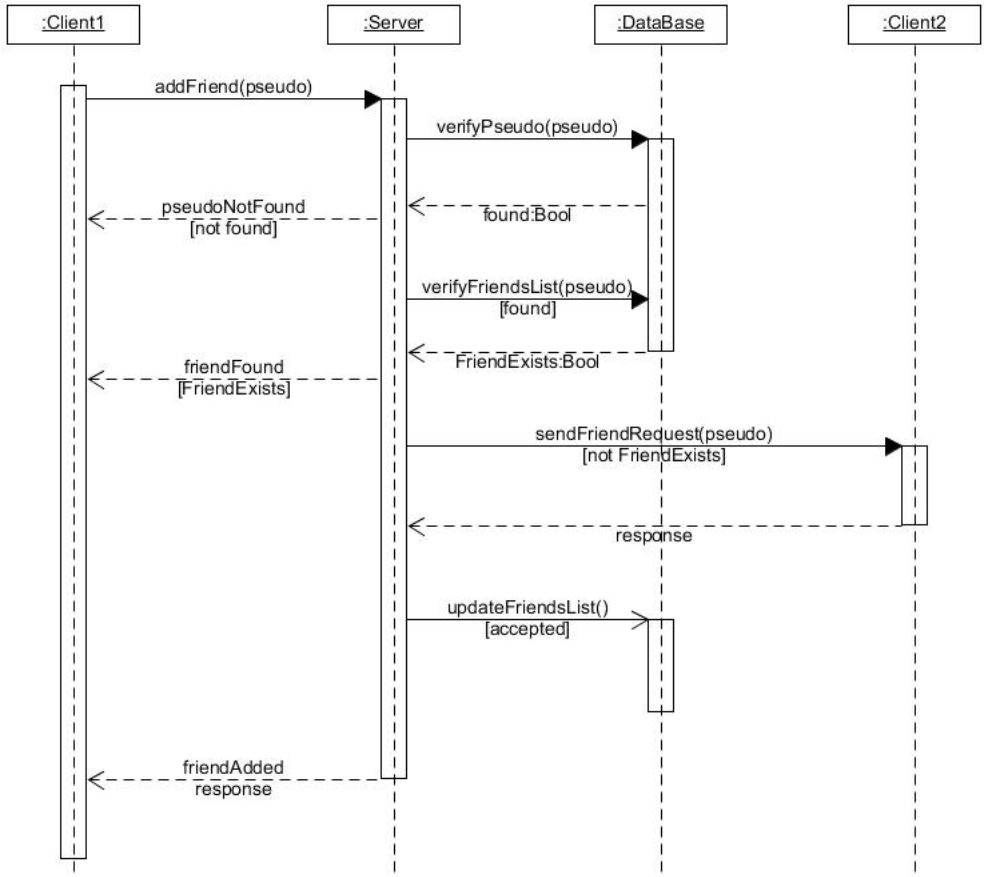

FIGURE 2 – Diagramme de séquence pour l'ajout d'un ami

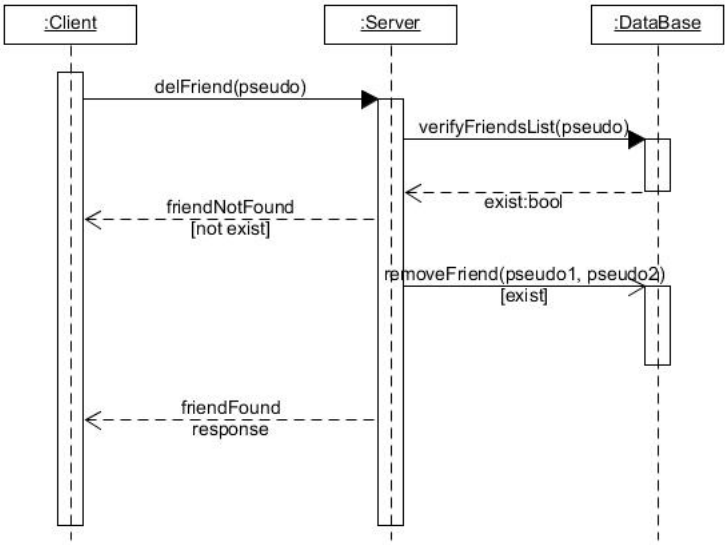

FIGURE 3 – Diagramme de séquence pour la suppression d'un ami

3 Besoins système

3.1 Besoins fonctionnels

3.1.1 Serveur

Le serveur est la classe principale du système. Il va gérer la majorité des interactions du programme et va servir d'intermédiaire entre le client et toutes les fonctionnalités de l'application. Pour le bon fonctionnement du programme, cette classe devra toujours être active.

- Communication client-serveur :
Le programme client et le serveur communiquent entre eux à travers des tubes de communications privés et publics. Les messages envoyés entre eux sont codifiés et limités. Lorsqu'une session utilisateur est lancée, il notifie le serveur à travers un tube public en envoyant son identifiant (pid du processus). Dés lors, le serveur lui crée des tubes privées pour conserver l'intégrité des données.

- Connexion :
Lorsqu'un client souhaite se connecter ou s'inscrire, il envoie son pseudo et mot de passe au serveur. Ce dernier se charge de vérifier l'existence ou la disponibilité de

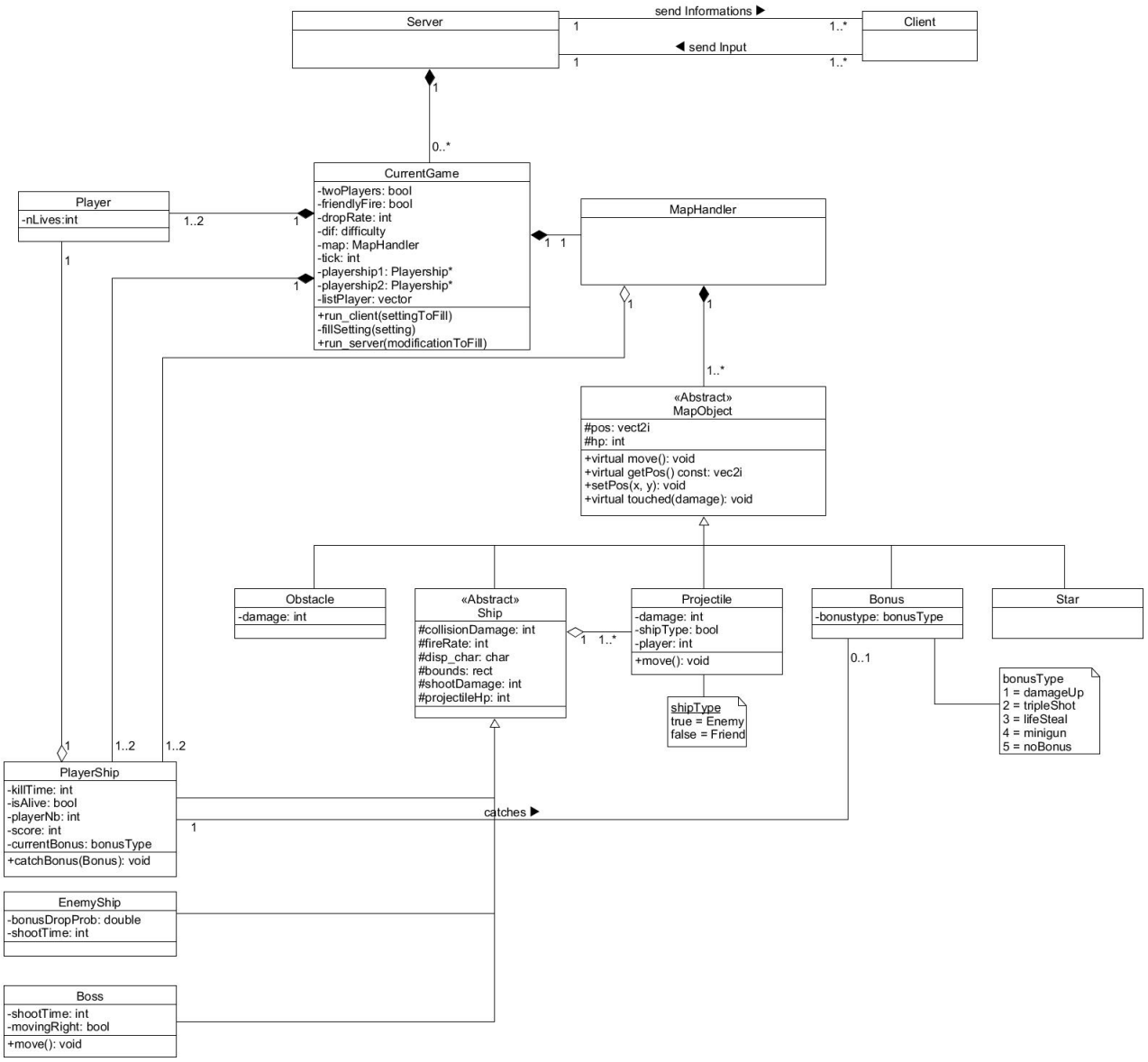

FIGURE 4 – Diagramme de classe du jeu

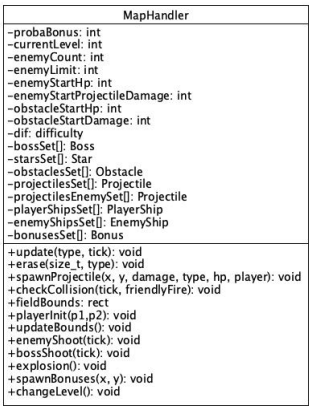

FIGURE 5 – classe MapHandler

ces informations dans la base de données et ainsi ouvrir une session utilisateur ou non.

- Interactions entre utilisateurs :
Un client peut ajouter et supprimer des amis ainsi que voir le classement de tous les utilisateurs. Ces actions sont rendues possibles grâce au serveur qui sert de lien entre tous les programmes clients. En effet, ces derniers envoient les messages correspondants à leurs requêtes et le serveur se charge d'interroger la base de donnée et d'envoyer les réponses adéquates.

- Jeu :
Toutes les parties lancées par les utilisateurs sont gérées par le serveur. En effet, il a pour rôle de décider de toutes les actions indépendantes du client et d'appliquer les mouvements de celui-ci. Ce qui signifie que c'est lui qui invoque les ennemis, les obstacles et les bonus. C'est aussi au serveur de recevoir les coups des joueurs, de les appliquer et de vérifier les collisions. Dés lors, le seul rôle du programme utilisateur est d'écouter les instructions du serveur et de les appliquer.

Une fois qu'une partie est terminée, le serveur se charge de mettre à jour le score des joueurs. Le score final est commun et est la somme des scores de chaque joueur.

- Terminaison :
Lorsqu'une session cliente est terminée, le serveur est mis au courant et se charge donc de supprimer les tubes de communications. S'il y a une partie en cours, elle est arrêtée. Dans le cas où le serveur est arrêté, tous les programmes utilisateurs ainsi que les parties lancées sont stoppées et la base de données est sauvegardée.

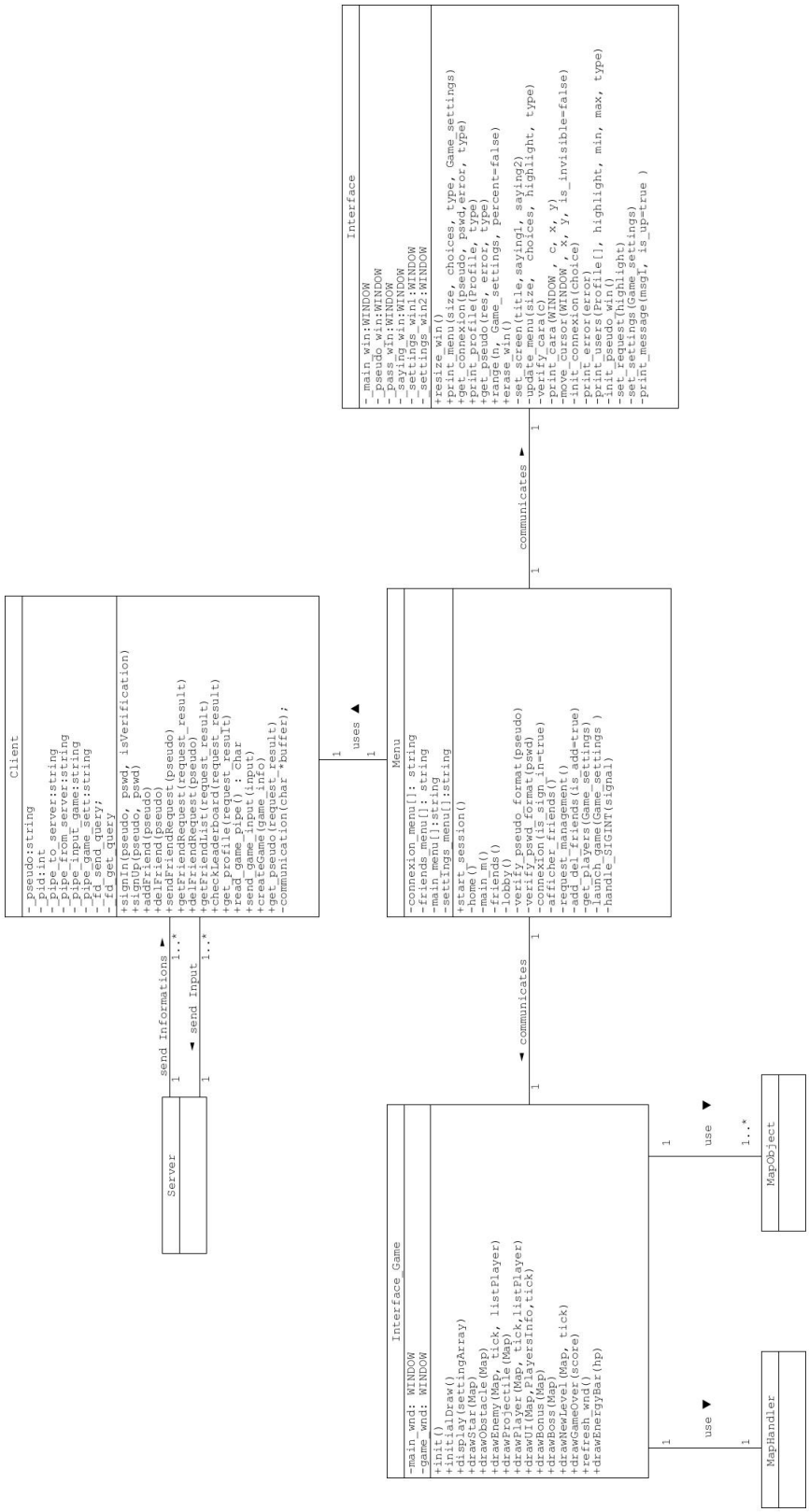

FIGURE 6 – Diagramme de classe côté client

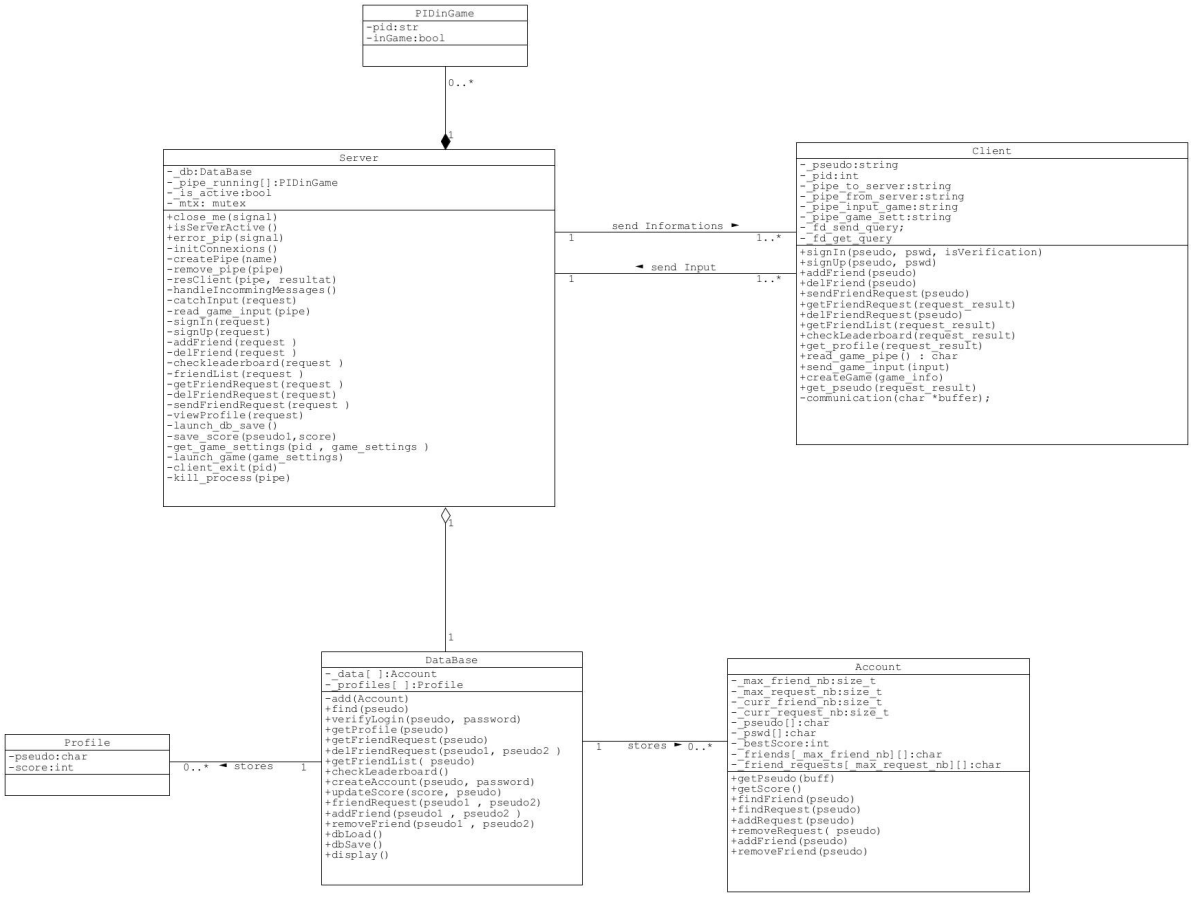

FIGURE 7 – Diagramme de classe du système

3.1.2 DataBase

La base de données a pour rôle de sauvegarder les informations relatives aux utilisateurs.
Un certains nombre d'opérations lui sont applicable à travers le serveur. En effet, ce dernier permet le transfert d'informations entre les clients et la base de données.
Les informations contenues dans la base de données sont les pseudonymes, mots de passe et le meilleur score de chaque compte.

3.1.3 Client

La classe Client permet à l utilisateur de communiquer avec le serveur à travers diverses actions possibles affichées sur le menu. Toutes les fonctionnalités décrites dans la section 2.1 sont possibles. A savoir :

— Se connecter
— S'inscrire
— Créer une partie
— Consulter le classement
— Gérer ses amis
— Consulter son profile

3.1.4 Gestion des comptes

Un objet Account contient toutes les informations relatives à un utilisateur (pseudo, mot de passe et score, sa liste d'amis et ses requêtes). Ces informations sont stockées dans la base de données et les différentes demandes d'accès à celle-ci sont traitées par le serveur.

- Accès :

Lors de la création d'un compte, la disponibilité du pseudo est vérifiée par la base de données. Si celui-ci n'est pas trouvé, un nouveau compte est bien créé et ajouté dans la base.

Lors de la connexion, c'est encore le serveur, à travers la base de données qui vérifie que le pseudo et le mot de passe saisis correspondent à un compte existant.

- Contenu d'un compte :

Chaque compte possède des informations sur l'utilisateur auquel il appartient, notamment ses identifiants (pseudo, mot de passe), son score (pour qu'il apparaisse dans le classement général des joueurs), une liste d'amis et une liste d'invitations qu'il peut consulter à tout moment.

3.1.5 Gestion d'une partie

Lorsque le joueur voudra lancer une partie, il aura la possibilité de choisir les paramètres de celle-ci comme expliqué plus haut (cf 2.1.3). Si le nombre de joueurs choisi est 2, l'hôte doit obligatoirement inviter un autre utilisateur à se connecter.

La partie ne peut pas commencer tant que le second joueur ne réussit pas à se connecter.

3.1.6 Gestion des amis

- Ajout : Lorsque l'utilisateur veut ajouter un ami, la requête est envoyée à la base de données via le serveur. Elle effectue des vérifications et envoie la demande, si elle est valide, à la personne concernée. Si la demande est acceptée, la liste d'amis des deux utilisateurs est mise à jour.

- Suppression : La base de données fait une recherche de l'ami à supprimer et l'efface de la liste d'amis de l'utilisateur. L'utilisateur sera également supprimé de la liste de son ancien ami.

3.1.7 Classement

Après chaque partie, le serveur met à jour le score de chaque joueur si celui-ci est supérieur à son score actuel. Ce qui permet de garder le classement tous le temps à jour.

3.2 Besoins non fonctionnels

- Le lancement du programme nécessite un environnement Linux possédant un dossier nommé "tmp" à la racine, afin de pouvoir stocker et supprimer les tubes créés par le serveur.
- Par souci de sécurité, toutes les requêtes d'un client doivent passer par le serveur.
- Pour des raisons esthétiques, la taille du terminal doit être de 80x24. - La librairie open source "ncurses" est indispensable.

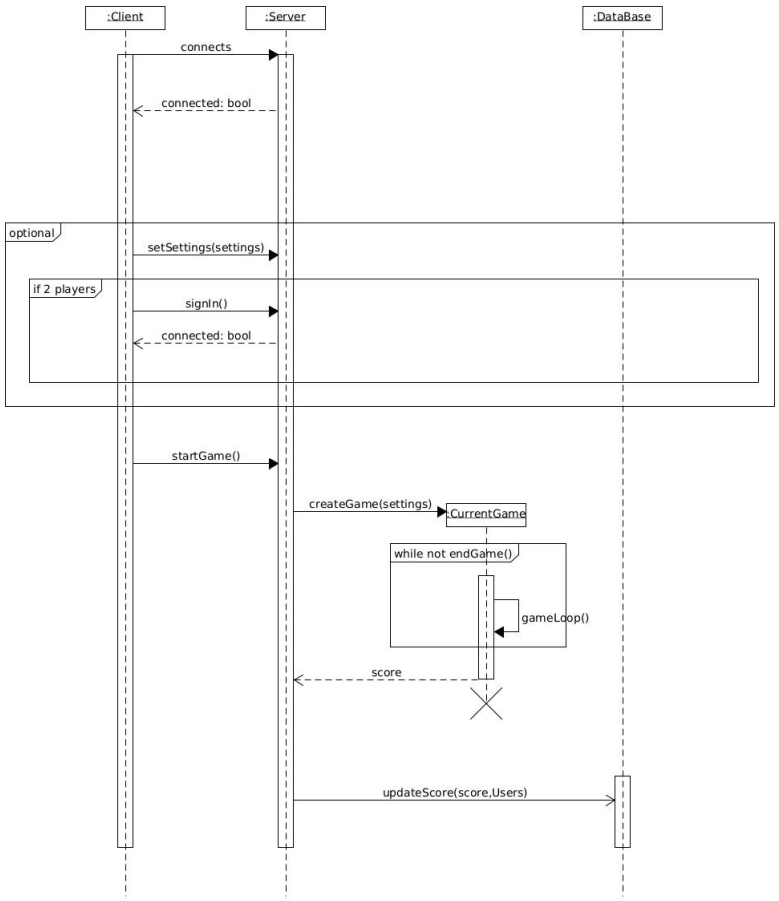

FIGURE 8 – Diagramme de séquence de lancement du jeu

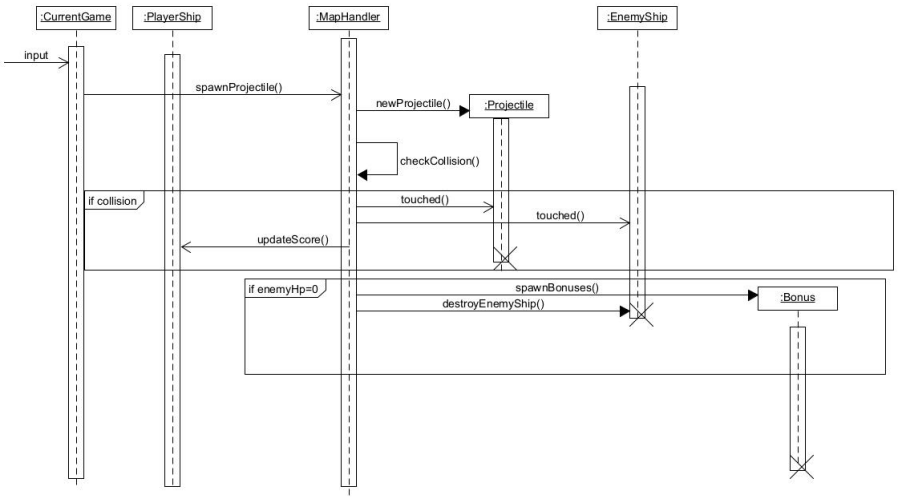

FIGURE 9 – Diagramme de séquence d'un tir de joueur

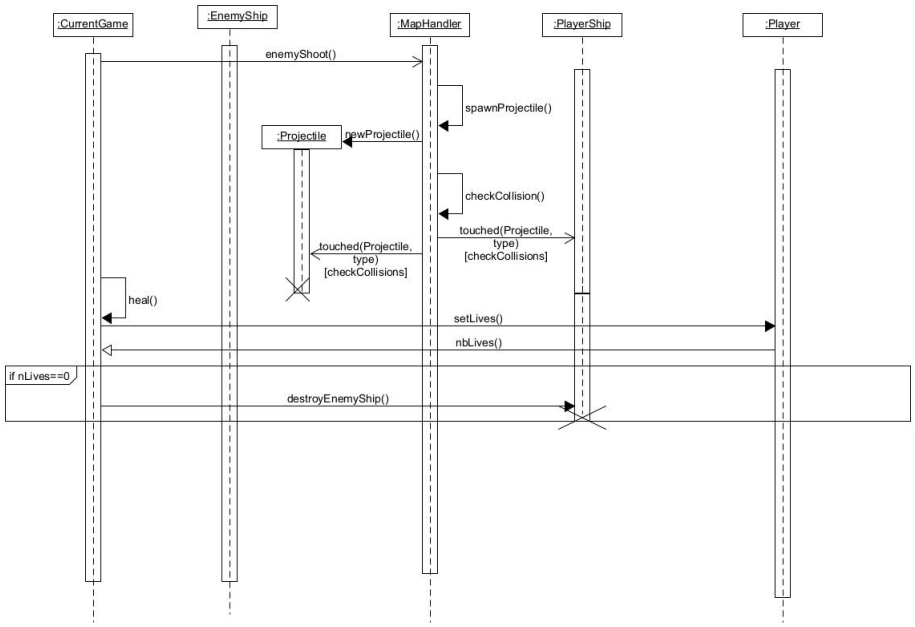

FIGURE 10 – Diagramme de séquence d'un tir ennemi

4 Annexes

4.1 Description du diagramme use case utilisateur

USE CASE	Pré-conditions	Post-conditions	Cas Général	Cas exceptionnels
Sign in	L'utilisateur doit être enregistré dans la base de données	L'utilisateur est connecté à sa base de données et le menu principal est affiché	L'utilisateur déjà enregistré se connecte à son compte en entrant son pseudonyme et mot de passe. Le serveur vérifie que les données soient correctes et donne accès au compte du client	Si l'identifiant ou le mot de passe sont incorrects, affiche un message d'erreur est affiché à l'utilisateur
Sign up	L'utilisateur n'est pas présent dans la base de données	Ajout d'un compte dans la base de données et affichage du menu principal	L'utilisateur crée un compte en introduisant un pseudo et un mot de passe	Si les données entrées ne respectent pas le format requis ou que le pseudonyme est déjà utilisé, un message d'erreur est affiché et l'utilisateur peut recommencer l'action jusqu'à ce qu'elle soit valide
Create game	L'utilisateur doit être enregistré dans la base de données	Possibilité de sauvegarder les paramètres par défaut d'une partie	L'utilisateur peut lancer une partie après avoir rempli les conditions minimales	Néant
Check Leaderboard	L'utilisateur doit être enregistré dans la base de données	Néant	Le joueur peut consulter le classement des scores en envoyant une requête au serveur qui va lui renvoyer les informations	Néant

View friend list	L'utilisateur doit être enregistré dans la base de données	Néant	Consultation de liste d'ami dans la base de données	Néant
Add friend	L'utilisateur doit être enregistré dans la base de données	Si invitation acceptée, ajout d'amis dans la base de donnée (bidirectionnel)	Entrer le pseudo d'un utilisateur. Le système va rechercher dans la base de données si le pseudo existe et lui envoyer une invitation.	Ajouter un pseudo qui n'existe pas (affiche une erreur)
Delete friend	L'utilisateur doit être enregistré dans la base de données et avoir au moins un ami.	Suppression dans la liste d'amis par le serveur(bidirectionnel)	Entrer le pseudo d'un ami. Le serveur va rechercher dans la base de données si le pseudo existe et le supprimer.	Supprimer un ami qui n'existe pas (affiche une erreur)
Profile	L'utilisateur doit être enregistré dans la base de données	Mise à jour des changements	L'utilisateur peut consulter ses informations de profile. Neant	Néant
Ship's controls	Créer une partie	Actualisation de l'état de jeu	Se déplacer, tirer recevoir des bonus	Néant
://fr.overleaf.com/project/6048c7f9f8c384f807254e17				
Leave party	Être en train de jouer	Retour au menu principal	Le joueur arrête la partie en cours.	Néant
Leave game	Etre connecté	Fermeture du jeu	L'utilisateur est connecté et veut quitter le jeu	L'utilisateur est connecté et force sa sortie du jeu (CTRL+C, ...)