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

L-type

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

Décembre 2020

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, de discuter avec d'autres utilisateurs 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

2 Besoins utilisateur

2.1 Besoins fonctionnels

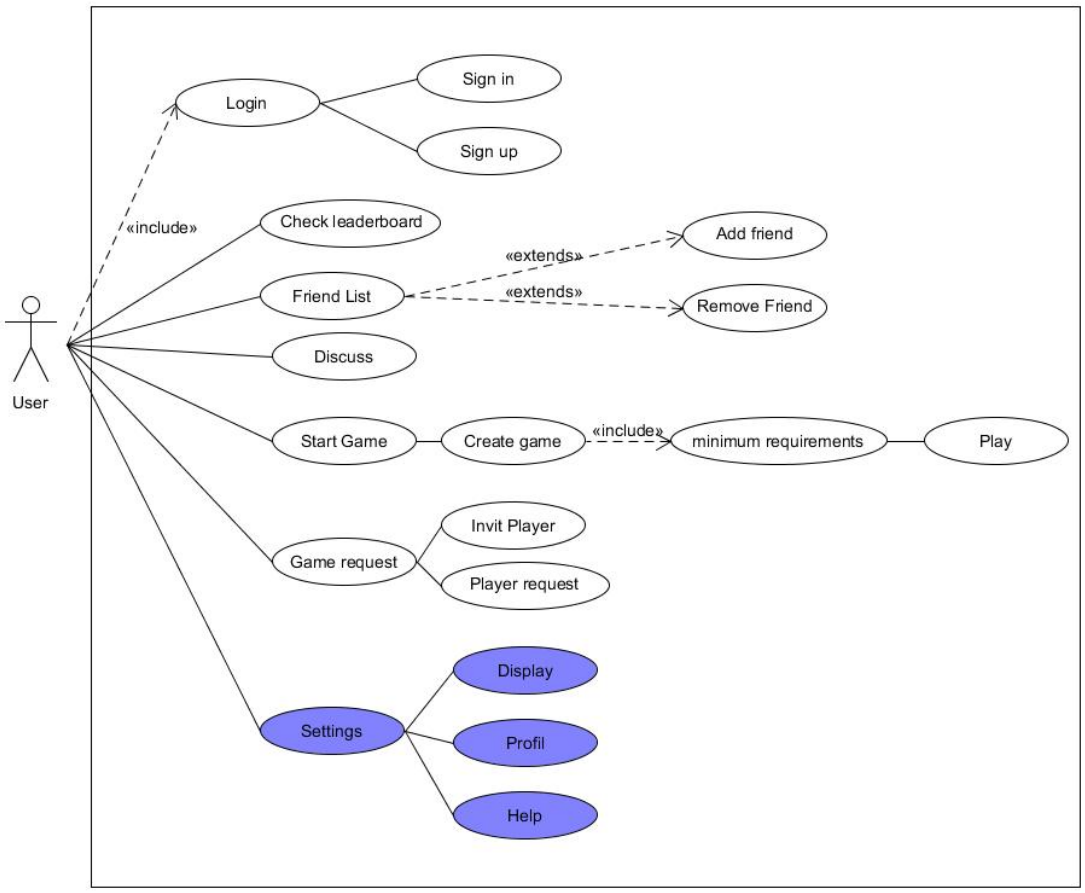

FIGURE 1 – Diagramme de Use Case côté utilisateur

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

- Discuter :

La liste des chats existants est présentée à l'utilisateur, il a le choix de discuter dans un chat existant ou d'en créer un nouveau avec un autre utilisateur, qu'ils soient amis ou non.

L'utilisateur peut aussi voir s'il a de nouveaux messages non-lus.

- Consulter amis :

La liste des amis de l'utilisateur est affichée lorsque cette option est sélectionnée. Il peut supprimer un ami, en ajouter un nouveau et aussi voir ses demandes d'amis.

La suppression d'un ami X par l'utilisateur Y implique également la suppression de Y dans la liste d'amis de X.

L'ajout d'un ami commence par une invitation. Le second utilisateur aura le choix d'accepter ou de refuser la demande.

- Consulter le classement :

Le classement affiche le score de tous les utilisateurs. S'il le souhaite, l'utilisateur peut afficher seulement le classement de ses amis.

- Envoyer une demande de jeu :

L'utilisateur a la possibilité d'inviter un autre utilisateur, avec un pseudonyme valide, à rejoindre une partie. Si ce dernier est connecté et qu'il n'est pas déjà en train de jouer, il a possibilité d'accepter ou de refuser la demande. S'il accepte, la connexion entre les deux joueurs sera établie, s'il refuse l'hôte ne sera pas averti.

- Consulter les demandes de jeu :

Lors d'une consultation de la liste d'invitations, la possibilité est laissée à l'utilisateur de les accepter ou décliner s'il en a reçu. Ainsi, accepter une invitation transfère l'utilisateur dans le lobby de l'hôte si celui-ci n'a pas atteint son nombre maximum de joueurs. L'invité ne peut pas modifier les paramètres que le premier

joueur aura choisi. À l'inverse, refuser une invitation n'avertit pas l'hôte.

- Lancer une partie :

Le lancement d'une partie se fait lorsque l'utilisateur crée une partie en gardant les paramètres de celle-ci par défaut ou en les redéfinissant.

- Paramètres :

Dans les paramètres, l'utilisateur peut consulter les règles du jeu à l'aide le bouton "Help". Configurer ses préférences audiovisuelles lui sont également permises. Il a aussi la possibilité d'accéder à son profil afin de consulter ses informations de compte s'il le souhaite.

2.1.3 Création de partie

La création d'une partie est une option qui envoie l'utilisateur vers une fenêtre de personnalisation permettant de modifier les paramètres du jeu. Cette fenêtre contient déjà des paramètres par défaut. S'il le souhaite, l'utilisateur peut définir :

- 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, il pourra, s'il le souhaite, inviter un autre joueur dans sa partie. Celle-ci ne se lancera que si le joueur invité accepte la demande de jeu.

- La difficulté de la partie :

L'utilisateur doit choisir une difficulté générale 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. C'est également la difficulté de la partie qui va déterminer la probabilité qu'aura un ennemi de lâcher un bonus après sa destruction.

- 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 inversément, que ses projectiles le soient pour l'invité.

- Le nombre de vies :

Le choix du nombre de vies est décidé par l'utilisateur.

2.1.4 Jeu

_Level :

Un jeu est composé de plusieurs niveaux. Les niveaux se différencient selon le nombre d'ennemis, leur résistance et la puissance de leurs projectiles. Le nombre d'obstacles varie également d'un niveau à l'autre.

_MapObjects :

Tous les objets apparaissant à l'écran et pouvant se déplacer sont des descendants de cette classe. Parmi elles, il y a les vaisseaux, les bonus, les projectiles et les obstacles.

Les projectiles sont créés par les vaisseaux lorsque ces derniers tirent. Ils peuvent sortir de l'écran, s'annuler en rencontrant d'autres projectiles ou causer des dégâts en atteignant leurs cibles.

Un vaisseau peut se déplacer dans toutes les directions, ainsi que subir et infliger des dégâts. Un vaisseau ennemi peut créer un bonus en se détruisant. La probabilité de lâcher ce bonus est déterminée par le niveau de difficulté choisi par l'utilisateur. Les bonus sont de plusieurs types et peuvent rapporter des améliorations d'armes au joueur qui réussit à les attraper. Le vaisseau d'un joueur ne peut pas lâcher de bonus mais peut en attraper.

_Player :

Un joueur commence sa partie avec un nombre de vies prédéterminé et un score nul. Il contrôle un vaisseau avec lequel il peut tirer des projectiles vers des ennemis, ce qui augmentera son score selon la quantité de dommages qu'il aura infligé. Son niveau de vie est réduit lorsqu'il subit des dégâts. Si ce niveau est nul, le vaisseau est détruit et le nombre de vies est décrémenté. Un nouveau vaisseau est attribué au joueur s'il possède encore au moins une vie. Sinon, la partie est terminée pour lui.

_MapHandler :

Le "MapHandler" gère tous les objets (MapObjects) se trouvant à l'écran. Il s'occupe de vérifier après chaque action si des objets n'entrent pas en collision. Cette classe gère la mise à jour des positions des MapObjects lorsqu'ils sont en déplacement.

_CurrentGame :

La classe CurrentGame représente un jeu 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 à quelle partie il doit envoyer l'action d'un utilisateur. Le jeu continue de s'exécuter tant qu'un joueur a au moins une vie et n'a pas terminé tous les niveaux. Elle peut également s'arrêter en cas de déconnexion d'un client.

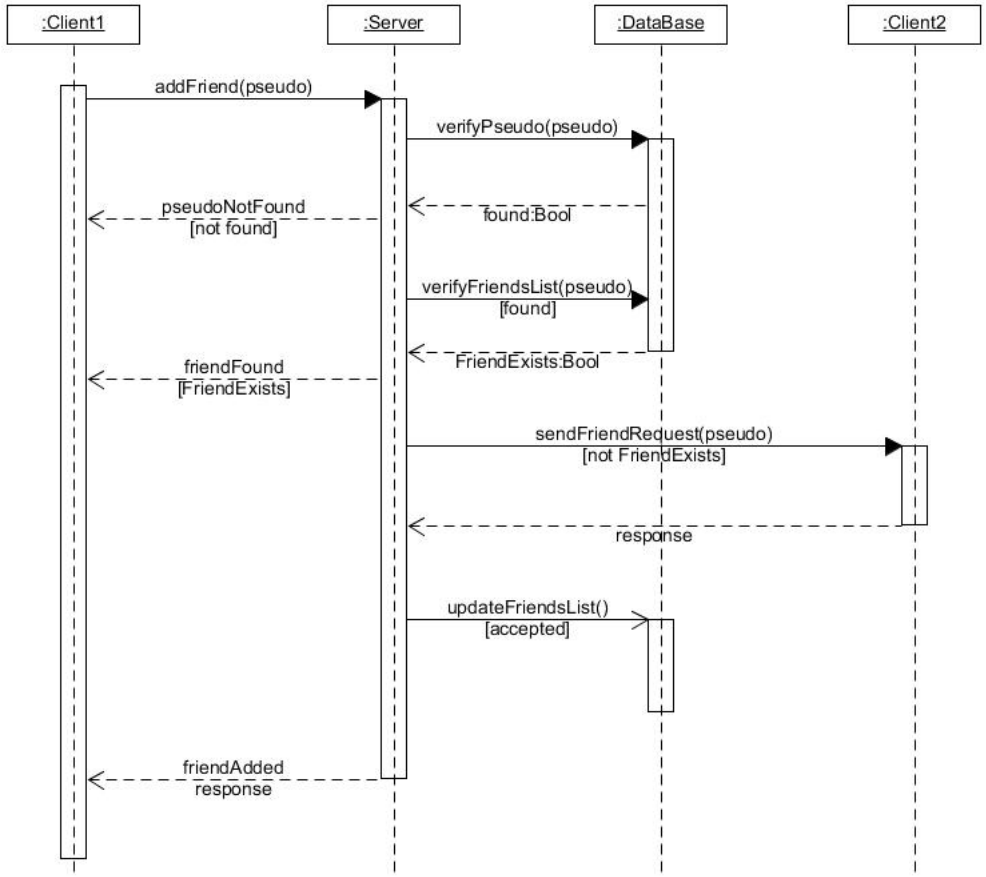

FIGURE 2 – Diagramme de Séquence pour l'ajout d'un ami

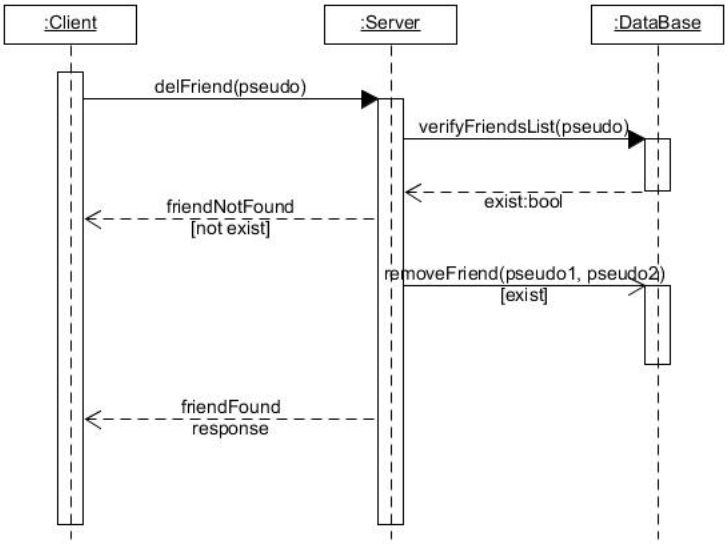

FIGURE 3 – Diagramme de Séquence pour la suppression d'un ami

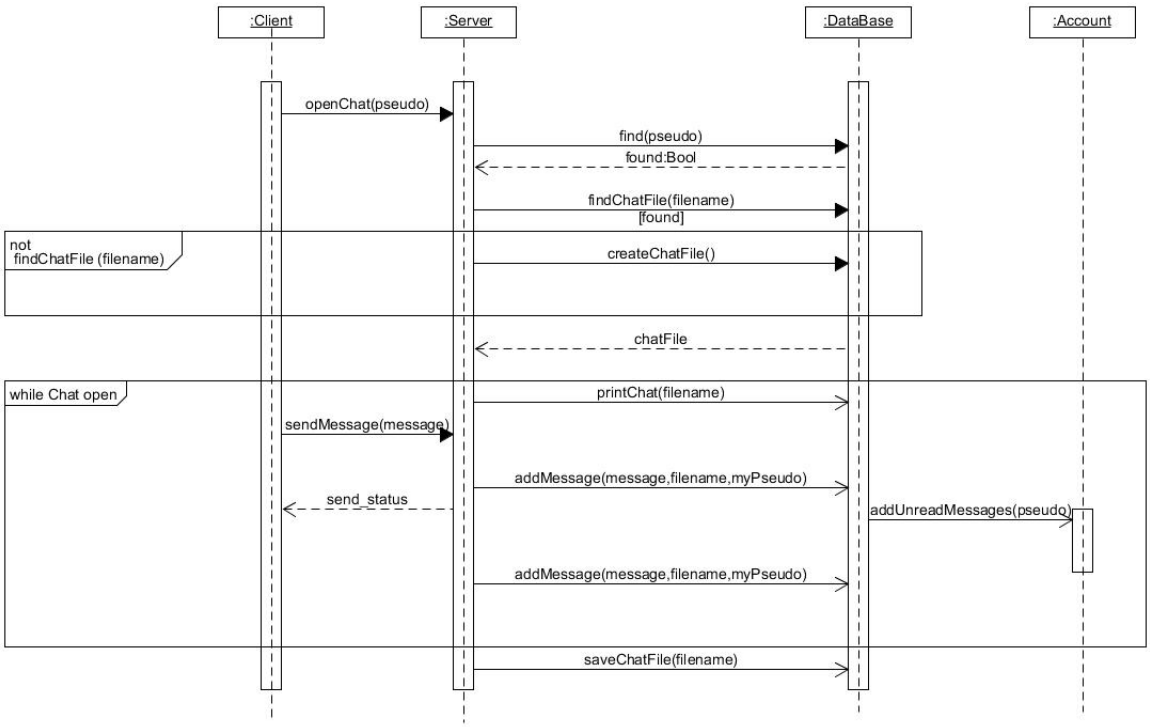

FIGURE 4 – Diagramme de Séquence pour le chat entre Clients

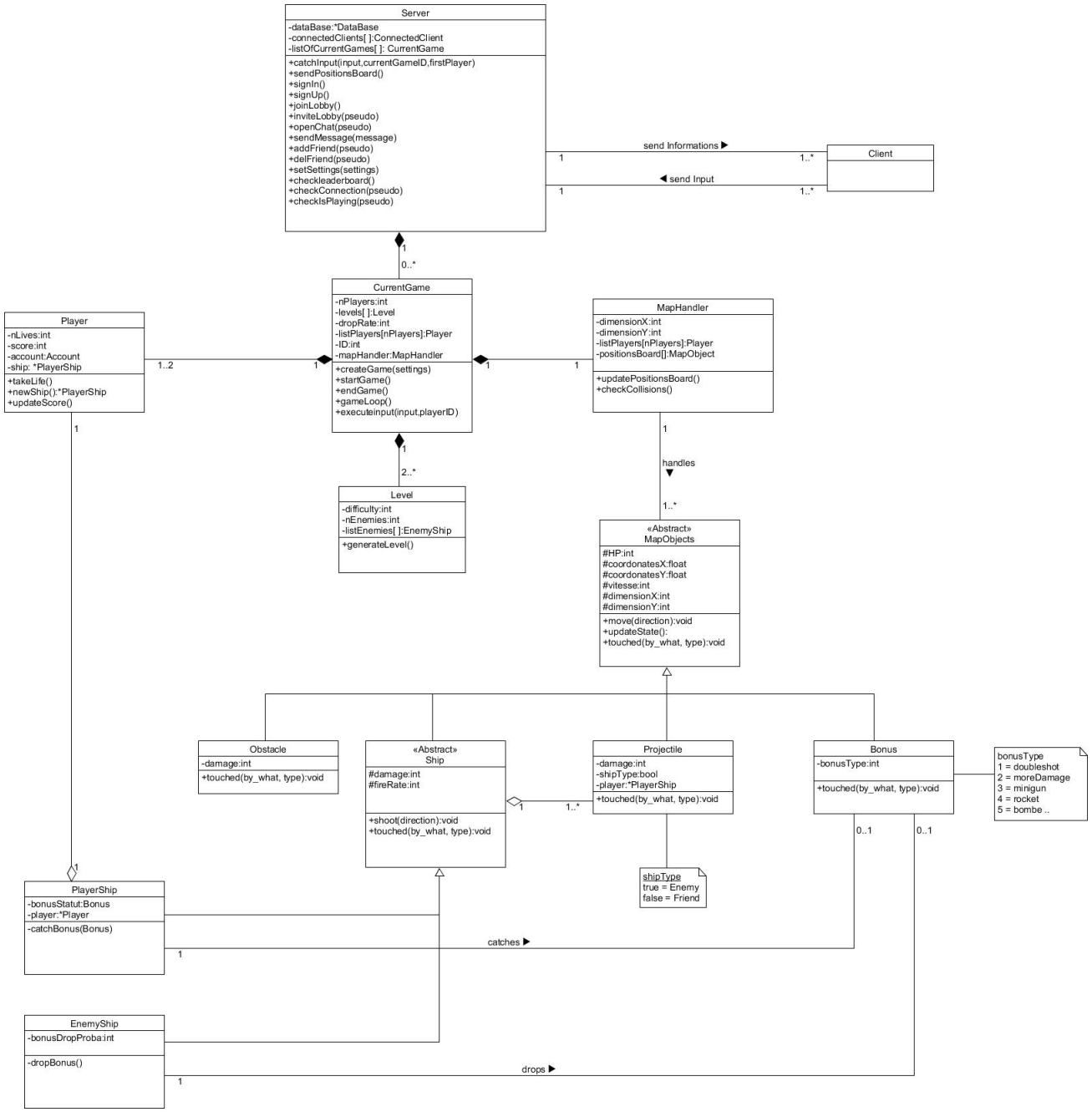

FIGURE 5 – Diagramme de Classe du jeu

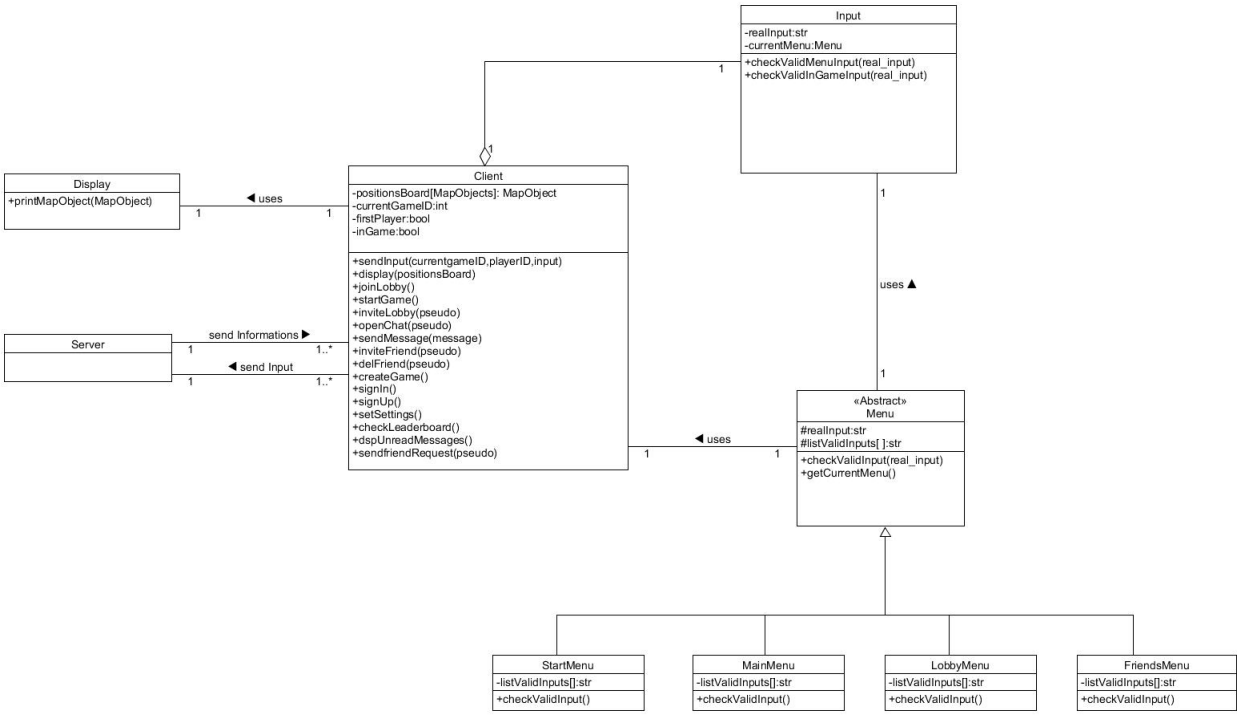

FIGURE 6 – Diagramme de Classe côté Client

3 Besoins système

3.1 Besoins fonctionnels

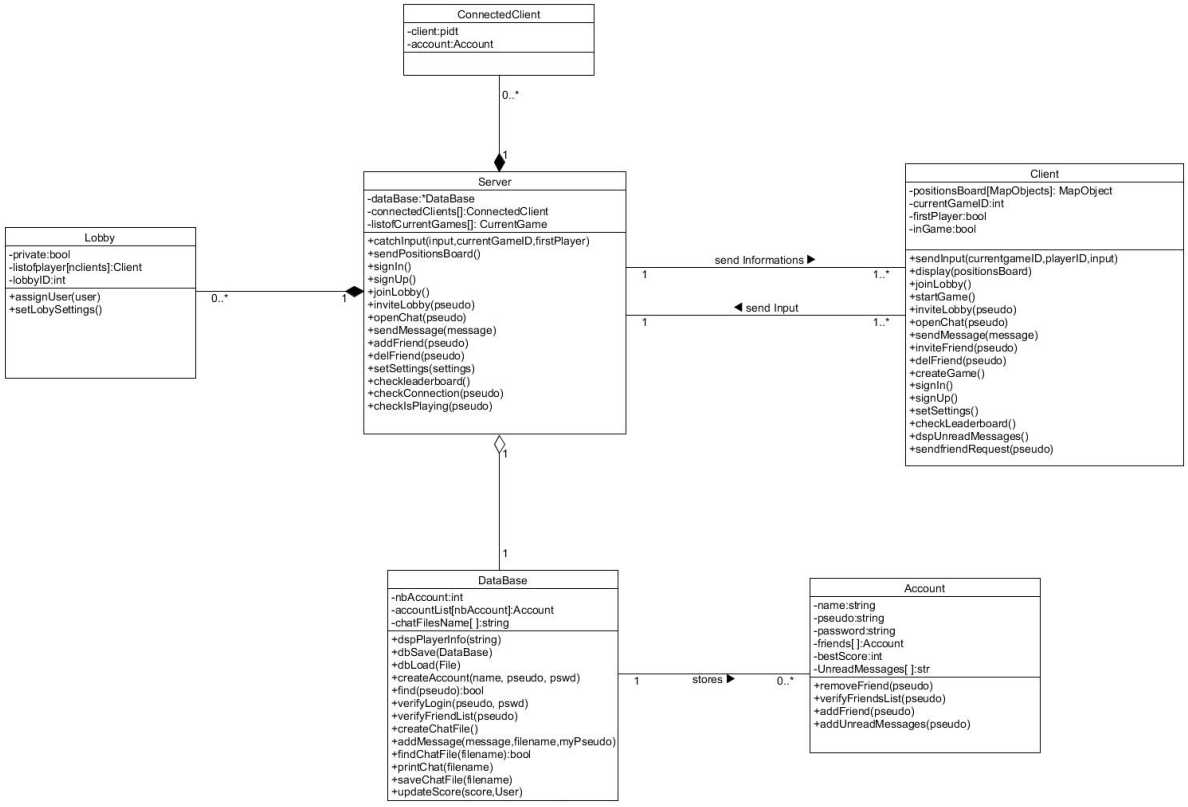

FIGURE 7 – Diagramme de Classe du Système

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 du jeu. Pour le bon fonctionnement du programme, cette classe devra toujours être active.
Parmi les fonctionnalités principales du serveur, on peut trouver :
- La connexion entre client et serveur
- L'interaction entre les utilisateurs
- La gestion des données personnelles
- La gestion de toutes les parties en cours

3.1.2 DataBase

Cette classe sauvegardera toutes les informations persistantes et nécessaires au bon fonctionnement du jeu.
Elle se chargera de :
- préserver et manipuler les informations personnelles relatives à chaque joueur
- conserver les historiques de discussions de tous les joueurs
- la vérification de l'existence des données

3.1.3 Lobby

Le Lobby va servir de salle d'attente pour un ou deux joueurs avant de lancer une partie commune. L'hôte d'un lobby va pouvoir décider de son accessibilité (privé/publique).

3.1.4 Client

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

— Se connecter
— s'inscrire
— Rejoindre un lobby
— Créer une partie
— Discuter avec d'autres utilisateurs
— Consulter le classement
— Ajouter ou supprimer un ami

Les actions citées ne sont possibles que si le client est connecté au serveur. Les entrées du client sont capturées par la classe Input.
La classe Input est connectée au menu affiché à l'écran, ce qui permet de transmettre les entrées au client qui les enverra ensuite au serveur.
La classe abstraite Menu a pour but de représenter une interface au niveau graphique ou terminal. Les classes héritières de la classe abstraite devront afficher le menu qui leur correspond et vérifier à la demande de la classe Input la validité d'une entrée.
La classe Display permet de gérer l'affichage des objets du jeu pour le client.

3.1.5 Gestion des comptes

Un objet Account contient toutes les informations relatives à un utilisateur (pseudo, mot de passe, etc). 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 le serveur. Si celui-ci n'est pas trouvé, un nouveau compte est bien créé et ajouté dans la base de données.

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) et une liste d'amis qu'il peut consulter à tout moment.

3.1.6 Gestion d'une partie

Tout client connecté sera placé dans un lobby privé par défaut. Lorsqu'il voudra lancer une partie, il aura la possibilité de choisir les paramètres de celle-ci comme expliqué plus haut (cf 2.1.3). Le joueur peut aussi changer le statut du lobby(public/privé). Si le nombre de joueurs choisi est 2, l'hôte doit obligatoirement inviter un autre utilisateur connecté dans le cas où le lobby est privé. Le serveur vérifie que la personne invitée existe dans la base de données, qu'elle est connectée et qu'elle n'est pas dans une partie en cours. S'il est public, tout utilisateur connecté pourra rejoindre le lobby qui deviendra donc privé. La partie ne peut pas commencer tant que le nombre de joueurs spécifiés ne correspond pas au nombre de personnes présentes dans le lobby.

3.1.7 Gestion des amis

- Ajout : Lorsque l'utilisateur veut ajouter un ami, le serveur 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 par le serveur.

- Suppression : Le serveur 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.

3.1.8 Chat

Les fichiers de chat entre deux utilisateurs sont stockés dans la base de données. Ces fichiers ont comme nom "pseudo1_pseudo2.txt" où pseudo1 est plus petit lexi-

cographiquement que pseudo2. Chaque ligne du fichier a comme format "pseudo : message".

Un utilisateur peut discuter avec n'importe quel autre utilisateur. Lorsqu'il veut ouvrir un chat, le serveur vérifie qu'il n'existe pas déjà un fichier chat partagé par ces deux personnes. Si oui, son contenu est affiché, sinon, un nouveau fichier est créé et stocké dans la base de données. Chaque nouveau message entraîne l'écriture dans le fichier et une incrémentation du nombre de messages non-lus du destinataire.

3.1.9 Classement

Après chaque partie, le serveur met à jour le score de chaque joueur si celui-ci est supérieur à son score actuel.

3.2 Besoins non fonctionnels

- Le lancement du programme nécessite un environnement Linux.
- Par souci de sécurité, toutes les requêtes d'un client doivent passer par le serveur.

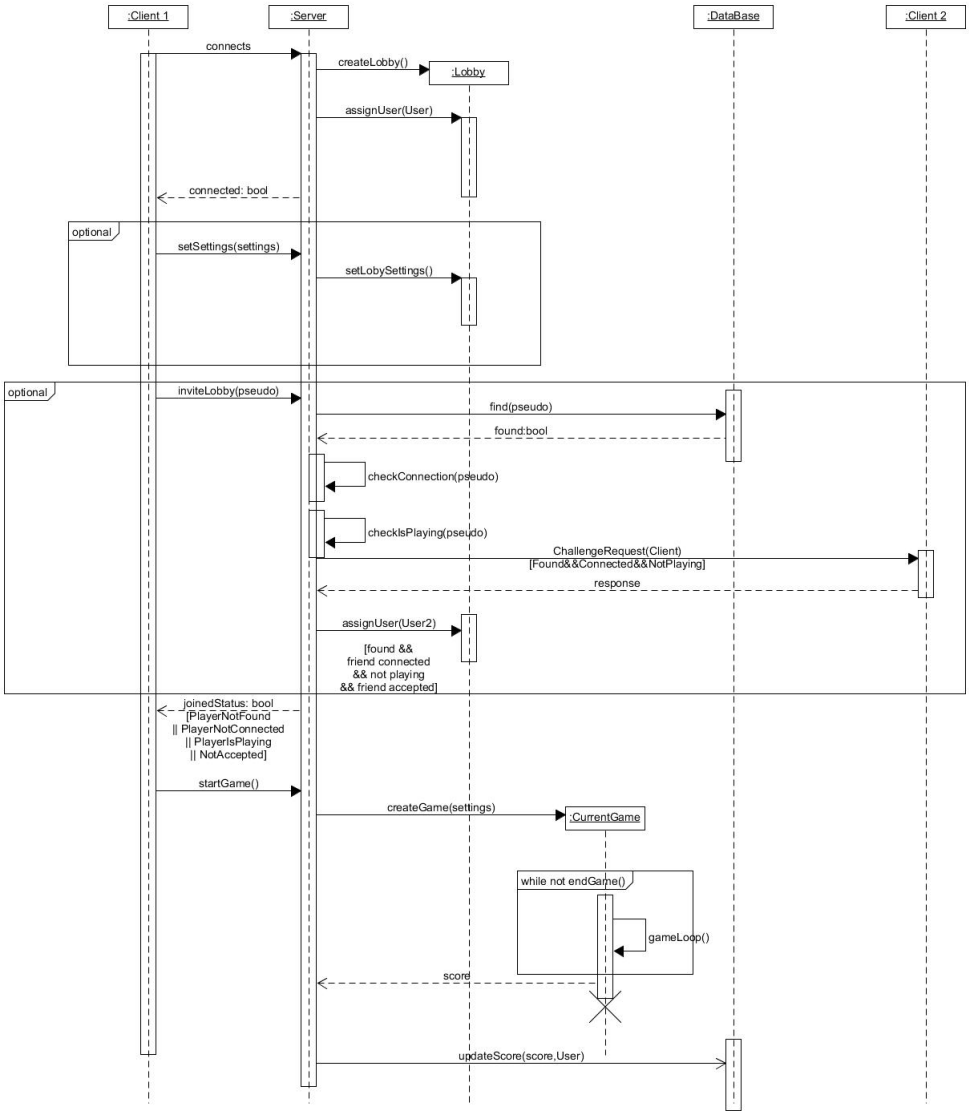

FIGURE 8 – Diagramme de Séquence de lancement du jeu

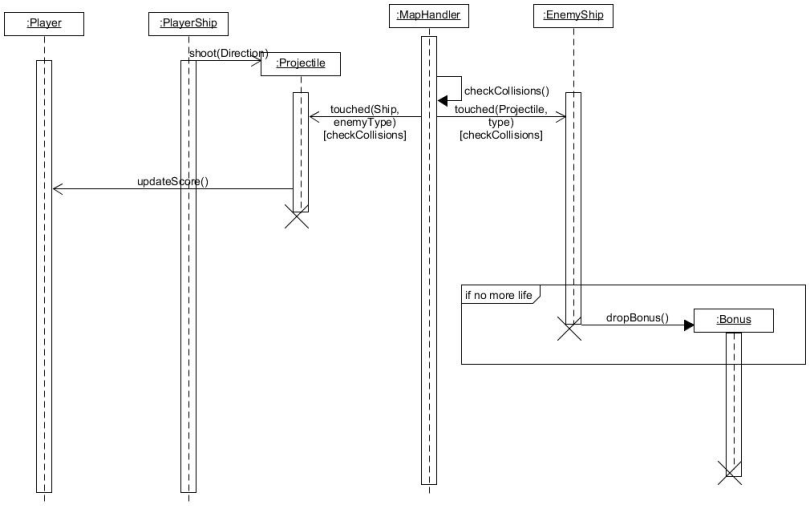

FIGURE 9 – Diagramme de Séquence d'un tir de joueur

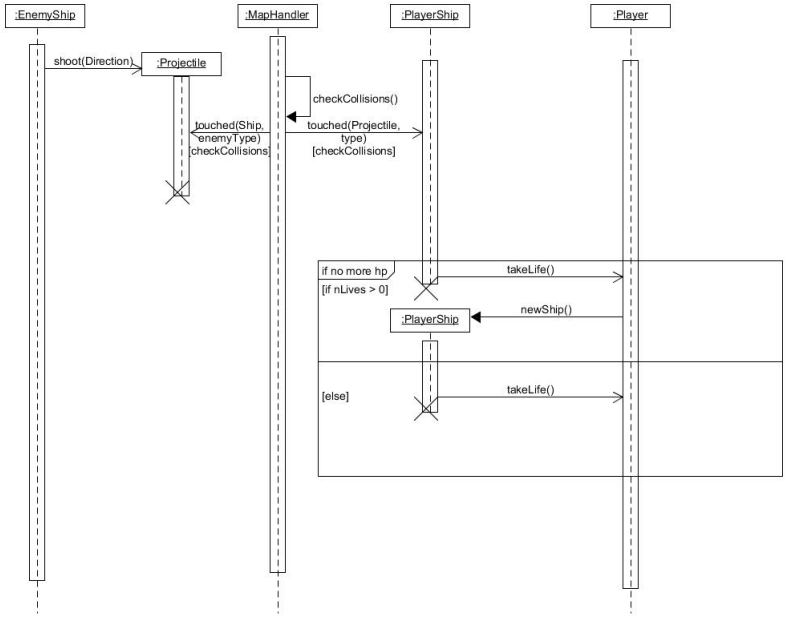

FIGURE 10 – Diagramme de Séquence d'un tir ennemi

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
Invite player	L'utilisateur doit être enregistré dans la base de données	Etablissement de la connexion via le serveur entre l'hôte et l'invité	Inviter un joueur avec son pseudo	Invitation d'un pseudo qui n'existe pas

Player request	L'utilisateur doit être enregistré dans la base de données et connecté pour recevoir une invitation via le serveur	Etablissement de la connexion via le serveur entre l'hôte et l'invité	Accepter/ Refuser une invitation de partie	La connexion échouera si : -Accepter une invitation dont l'hôte n'est plus connecté -Rejoindre un salon complet
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
Chat	L'utilisateur doit être enregistré dans la base de données	Le serveur effectue les liaisons entre les utilisateurs	Envoyer et recevoir des messages avec n'importe quel pseudo enregistré 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)

Settings	L'utilisateur doit être enregistré dans la base de données	Mise à jour des changements	L'utilisateur peut modifier ses paramètres de profil et/ou audiovisuels. Il peut également consulter les règles du jeu	Néant
Ship's controls	Créer une partie	Actualisation de l'état de jeu	Se déplacer, tirer recevoir des bonus	Néant
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, ...)