

Projet d'année - L-Type

Marco Luyckx, Tiago Marques Correia, Idiatou Diallo, Samed Tektas,
Victor Piryns, Alexandre De Groodt, Attilio Discepoli,
Edgardo Cuellar Sanchez, Diego Benitez Alberdi

2020-2021



Table des matières

1	Introduction	3
1.1	Présentation du projet	3
1.2	Glossaire	3
1.3	Historique de modification	4
2	Comment jouer ?	5
2.1	Lancement du jeu	5
2.2	Les touches pour la version terminal (Ncurses)	5
2.3	Les touches pour la version graphique (SFML)	5
3	Besoins utilisateur	6
3.1	Besoins fonctionnels	6
3.1.1	Connexion	6
3.1.2	Menu	6
3.1.3	Sociale	6
3.1.4	Jeu	6
3.1.5	Création de niveaux	6
3.2	Besoins non-fonctionnels	6
3.2.1	Expérience utilisateur (UX)	6
4	Besoins système	7
4.1	Besoins fonctionnels	7
4.1.1	Base de données	7
4.1.2	Serveur	7
4.1.3	Client	7
4.2	Communications	7
4.2.1	Matériel Informatique	7
4.3	Besoins non fonctionnels	7
4.3.1	Portabilité	7
4.3.2	Rétro compatibilité	7
4.4	Connexion	8
4.5	Petite explication pour le menu	8
5	Diagrammes Use case/Tableaux/Séquence/Classe	9
5.1	Connection Menu	9
5.2	Main Menu	10
5.3	Room Menu	11
5.4	Leaderboard	12
5.5	Social	12
5.6	Levels	13
5.7	Online Menu	18
5.8	Game	22
6	Explications pour le diagramme de la partie sur les commandes.	30
7	Screenshots	32

1 Introduction

1.1 Présentation du projet

Le projet a pour but de créer un jeu "shoot them up" multijoueur local nommée "L-type".

Tout utilisateur doit posséder un compte pour pouvoir avoir accès aux différentes fonctionnalités du jeu. Une fois le compte créé, l'utilisateur peut se connecter et par après créer ou rejoindre une partie, gérer une liste d'amis, consulter la liste des scores et gérer son compte. Une partie peut accueillir un à deux joueurs. Durant le déroulement d'une partie, les joueurs doivent parcourir plusieurs niveaux en détruisant les ennemis qui se présentent devant eux ainsi qu'en esquivant leurs tirs. Les vaisseaux dirigés par les joueurs peuvent récupérer des bonus d'armement lâchés par leurs nombreux adversaires.

Lorsqu'il y a deux joueurs dans une partie, la mort d'un des joueurs n'entraîne pas la fin de la partie, en d'autres mots la partie continue tant qu'un joueur possède encore plus d'un point de vie. Lorsqu'il n'y a qu'un seul joueur, la partie se termine lors de la mort du joueur. En résumé, le jeu L-Type est un Shoot'em up en scrolling vertical avec la possibilité de jouer en multijoueur local. Chaque joueur peut créer un compte.

Avant chaque partie, le joueur pourra modifier les paramètres de la partie qui va se dérouler (vitesse des ennemis, probabilité de power up, etc.) mais pas pendant la partie ! Trois niveaux de difficulté sont proposés au(x) joueur(s), chaque niveau étant plus dur que le précédent.

Un leaderboard permet de consulter les différents scores.

1.2 Glossaire

- Pseudo : Pseudonyme d'un joueur
- Client : Machine du joueur
- Serveur : Machine gérant la logistique du jeu (transparent au joueur)
- HUD : "Head-up display" Ensemble d'informations en haut de l'écran renseignant le joueur sur son état
- Room : Salle d'attente avant le début d'une partie
- Interpolation : L'application d'une interpolation linéaire ou sphérique sur un vecteur en visant à rapprocher le vecteur en question à un vecteur cible de X%, X étant le "poids" de l'interpolation. (Exemple : un vecteur (1,1) avec un vecteur cible (2,2), après interpolation avec un poids de 0.5 (=50%) sera égal au vecteur (1.5, 1.5))
- Pipe : Canal de communication entre le client et le serveur.
- Commande : Structure reprenant les actions à effectuer par l'autre processus.
- Spawner : Apparition d'entité
- Ship : Vaisseau
- Power up : Objet (bonus) lâché par un ennemi lors de sa mort
- Leaderboard : Tableau d'affichage des scores.
- Enumeration : Type de données consistant en un ensemble de valeurs constantes.
- Sérialisation : Codage d'une information sous la forme d'une suite d'informations plus petites.
- Structure : Regroupement de données.
- Friendly fire : Lorsque les tirs alliés et les collisions entre les joueurs sont activés.
- Like : Système de "j'aime" pour les niveaux du jeu.
- SFML : Librairie utilisée pour coder la version graphique du jeu.
- Ncurses : Librairie utilisée pour coder la version du jeu en terminal.
- VFX : effets visuels.
- Frame : Une étape d'animation du jeu, un instant du jeu.

1.3 Historique de modification

Date	Version	Personnes	Modification
21.11	0.1	Alexandre	Diagrammes Use Case et tables
29.11	0.2	Tout le groupe	Diagrammes de classes
04.12	0.3	Idiatou et Attilio	Ajout Glossaire, Introduction et Besoins Utilisateurs
06.12	0.4	Tout le groupe	Diagrammes de séquences
11.12	0.5	Tout le groupe	Ajout des Besoins Système
14.12	0.6	Diego, Edgardo et Attilio	Enrichissement des Besoins Utilisateurs
15.12	0.7	Alex et Samed	Mise en page, refs, diagrammes, correction ortho
15.12	0.8	Victor, Marco, Tiago	Enrichissement des Besoins Système
16.12	0.9	Tout le groupe	Correction collective et mise au point
18.12	0.10	Tout le groupe	Dernier check du SRD avant la remise
02.02	1.0	Tout le groupe	Modification du SRD avec les nouvelles règles
04.02	1.1	Tout le groupe	Vérifications des modifications
26.02	1.2	Samed et Diego	Mise à jour diagrammes de classes
28.02	1.3	Samed et Diego	Mise à jour diagrammes de classes + explications Use Case
01.03	1.4	Diego	Modification Besoins Utilisateurs et des Besoins système
01.03	1.5	Diego	Mise à jour diagrammes de classes
10.03	1.6	Diego	Mise à jour des diagrammes, du glossaire, explications commandes
17.04	1.7	Diego	Ajout de screenshots
18.04	1.8	Diego	Mise à jour des diagrammes + nouvelles sections
19.04	1.9	Diego	Mise à jour tableau Use Case + glossaire

2 Comment jouer ?

2.1 Lancement du jeu

Avant toute chose, il faut savoir comment lancer et jouer au jeu et pour ce faire, vous aurez besoin d'avoir les librairies de ncurses et sfml, qui sont respectivement libncurses-dev et libsfml-dev, ainsi que de sqlite pour la base de données : libsql3-dev. Pour lancer le jeu vous aurez besoin d'ouvrir un terminal depuis le dossier principal du jeu dans lequel vous allez exécuter les commandes suivantes :

- ./build.sh
- ./ltype_game_server

La dernière commande sert à lancer le serveur sur lequel le jeu va tourner. Veillez bien à le laisser tourner en fond. Ensuite afin d'exécuter le jeu vous aurez besoin d'un second terminal pour faire tourner le client à l'aide de la commande :

- ./ltype_game_client

Pour faire tourner le jeu dans sa version terminal,c-à-d avec la librairie "ncurses". Et :

- ./ltype_game_client -g

Pour le faire tourner dans sa version graphique, avec la librairie "SFML".

2.2 Les touches pour la version terminal (Ncurses)

- Z, UP_ARROW : se déplacer vers l'avant, naviguer dans les menus.
- Q, LEFT_ARROW : se déplacer vers la gauche, naviguer dans les menus.
- S, DOWN_ARROW : se déplacer vers le bas, naviguer dans les menus.
- D, RIGHT_ARROW : se déplacer vers la droite, naviguer dans les menus.
- ESPACE, ENTER : tirer en jeu.
- P : mettre pause en jeu.
- M : reprendre la partie en pause.
- O : quitter la partie depuis le menu pause.

2.3 Les touches pour la version graphique (SFML)

- MOUSE : pour naviguer dans les menus.
- Z, UP_ARROW : se déplacer vers l'avant J1, J2.
- Q, LEFT_ARROW : se déplacer vers la gauche J1, J2.
- S, DOWN_ARROW : se déplacer vers le bas J1, J2.
- D, RIGHT_ARROW : se déplacer vers la droite J1, J2.
- ESPACE, ENTER : tirer en jeu J1, J2.
- P : mettre pause en jeu.
- M : reprendre la partie en pause.
- O : quitter la partie depuis le menu pause.

3 Besoins utilisateur

3.1 Besoins fonctionnels

3.1.1 Connexion

Il est possible pour l'utilisateur de créer un compte avec mot de passe, un pseudonyme associé s'il n'en a pas encore et une question secrète. Par après il pourra se connecter au jeu à l'aide de ces données. (*cf* Connection menu 1)

3.1.2 Menu

Une fois connecté, l'utilisateur se retrouve dans le menu principal et a accès à différents choix d'options : créer une partie, avoir accès à son menu "Social", accéder au "leaderboard" qui regroupera les meilleurs scores de certains joueurs ou bien même "levels" si il joue sous la version graphique du jeu. (*cf* Main menu 2)

3.1.3 Sociale

L'utilisateur a accès à un menu "Social" regroupant son profil avec sa liste d'amis, et où il peut de ce fait envoyer et recevoir des demandes d'amis. Il peut aussi éditer son profil et voir ses niveaux créés en version graphique. (*cf* Social menu 7)

3.1.4 Jeu

Un joueur peut créer une partie, modifier différents paramètres (résistance des ennemis, nombre de vies des joueurs, ...) et il peut aussi jouer en mode 2 joueurs dans ce cas le deuxième joueur se connecte à son compte ou joue en tant qu'invité. (*cf* Game 10)

3.1.5 Crédit de niveaux

Un utilisateur pourra, sous la version graphique(SFML) du jeu, créer ses propres niveaux. Il pourra par la suite jouer ses niveaux et même les niveaux des autres joueurs. Un système de "Like" est aussi disponible, ce qui donne la possibilité de like ses niveaux préférés! (*cf* Main menu 2)

3.2 Besoins non-fonctionnels

3.2.1 Expérience utilisateur (UX)

Une interface agréable d'utilisation, un bel agencement des menus et des boutons, pour une utilisation optimale de l'interface.

4 Besoins système

4.1 Besoins fonctionnels

4.1.1 Base de données

Une base de données existe pour permettre le stockage des données. Elle existe du côté du serveur, se chargeant de lire et d'écrire les données nécessaires pour le lancement/déroulement d'une partie, les options possibles, la gestion des menus et le leaderboard.

4.1.2 Serveur

Le serveur est une instance séparée du jeu, qui contient toutes les fonctionnalités essentielles pour son déroulement correct. Il lit les inputs venant du client, mettant à jour la représentation interne du jeu, résolvant toutes les collisions et renvoyant les nouvelles positions au client. Il peut aussi être utilisé pour interpréter les événements des menus, enregistrer des données dans la base de données et/ou lire à partir de celle-ci. Le serveur est un processus qui vit indépendamment du client, sur la même machine.

4.1.3 Client

Le client lit les commandes venant du pipe et modifie les positions des entités dans le jeu pour être une copie parfaite de la version du jeu qui existe sur le serveur. Il est aussi responsable de lire les inputs venant de la méthode d'affichage (stdin ou window-events) et de les envoyer au serveur.

Le client a deux modes de visualisation du jeu, ASCII et 2D graphique. Vu que les deux versions doivent être compatibles entre elles, et que les positions sont des flottants, elles seront arrondies pour la version ASCII. La librairie utilisée pour la réalisation des interfaces est Ncurses pour l'affichage sur le terminal. Le nombre de client pouvant tourner en même temps a été limité pour une question de rapidité et d'optimisation, même si le serveur a la capacité d'accepter autant de clients qu'il faut.

4.2 Communications

Les clients et le serveur communiquent à travers des pipes nommés, qui sont une interface de communication entre différents processus existants sur la même machine. Chaque processus peut écrire des "commandes" sur le pipe pour indiquer à l'autre processus les actions qu'il doit effectuer.

4.2.1 Matériel Informatique

L'utilisateur communique par le biais de son clavier pour la version du jeu dans le terminal et avec son clavier et sa souris pour la version GUI.

4.3 Besoins non fonctionnels

4.3.1 Portabilité

Le code du projet doit être réalisé en C++ et pouvoir s'exécuter sur les machines Linux du bâtiment N.O de l'ULB.

4.3.2 Rétro compatibilité

Deux joueurs possédants des versions graphiques différentes du jeu (terminal/GUI) doivent pouvoir jouer ensemble.

Ceci peut être réalisé grâce à un grand découpage entre la partie logique du jeu et la partie qui va prendre en compte l'affichage. Le client est un front end pour le jeu, sa seule responsabilité est de lire les inputs, les envoyer au serveur pour effectuer un traitement et après mettre à jour les nouvelles positions/données.

4.4 Connexion

Un joueur qui oublie son mot de passe peut le changer grâce au bouton "reset password". Lors de la création de son compte, le joueur choisit une question secrète de sécurité et crée une réponse à cette question. Pour pouvoir changer son mot de passe, le joueur doit répondre correctement à la question sans limite d'essais. Une fois que la bonne réponse est trouvée le client proposera alors au joueur de mettre son nouveau mot de passe et de le confirmer.

4.5 Petite explication pour le menu

Pour l'affichage du menu, il y a une classe principale qui s'occupe de gérer tous les menus en recevant celui à afficher grâce à un MNAME qui est une énumération contenant les noms des différents menus. Elle s'occupe aussi de gérer les commandes avec le serveur pour les menus afin de faire ce qu'il faut au bon moment.

5 Diagrammes Use case/Tableaux/Séquence/Classe

5.1 Connection Menu

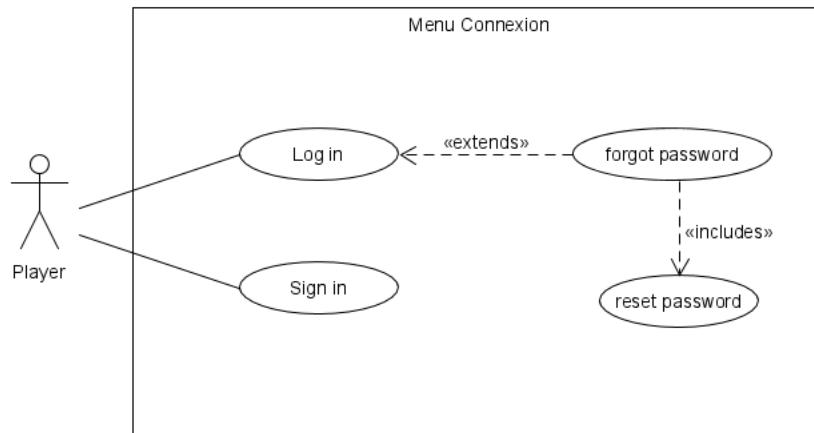


FIGURE 1 – Diagramme Use Case du menu de connexion

Use Case	Pré-conditions	Post-conditions	Cas général	Cas exceptionnels
Sign In	le joueur n'est pas inscrit	le joueur est inscrit et connecté	définir un pseudo, mot de passe et une réponse à la question secrète	le pseudo est déjà utilisé
Log In	le joueur est inscrit	le joueur est connecté	il indique le pseudo et le mot de passe qui sont comparés à ceux de la base de donnée	le pseudo/mot de passe ne correspondent pas à ceux de la base de donnée
Forgot Password	le joueur est inscrit	entre dans le reset password	il répond à sa question secrète	néant
Reset Password	le joueur a bien répondu à la question secrète	le mot de passe du joueur est changé	défini un nouveau mot de passe dans la base de données	néant

Explication Use Case de Connection Menu : Processus du fonctionnement de la connexion d'un joueur au jeu et de la réinitialisation du mot de passe.

5.2 Main Menu

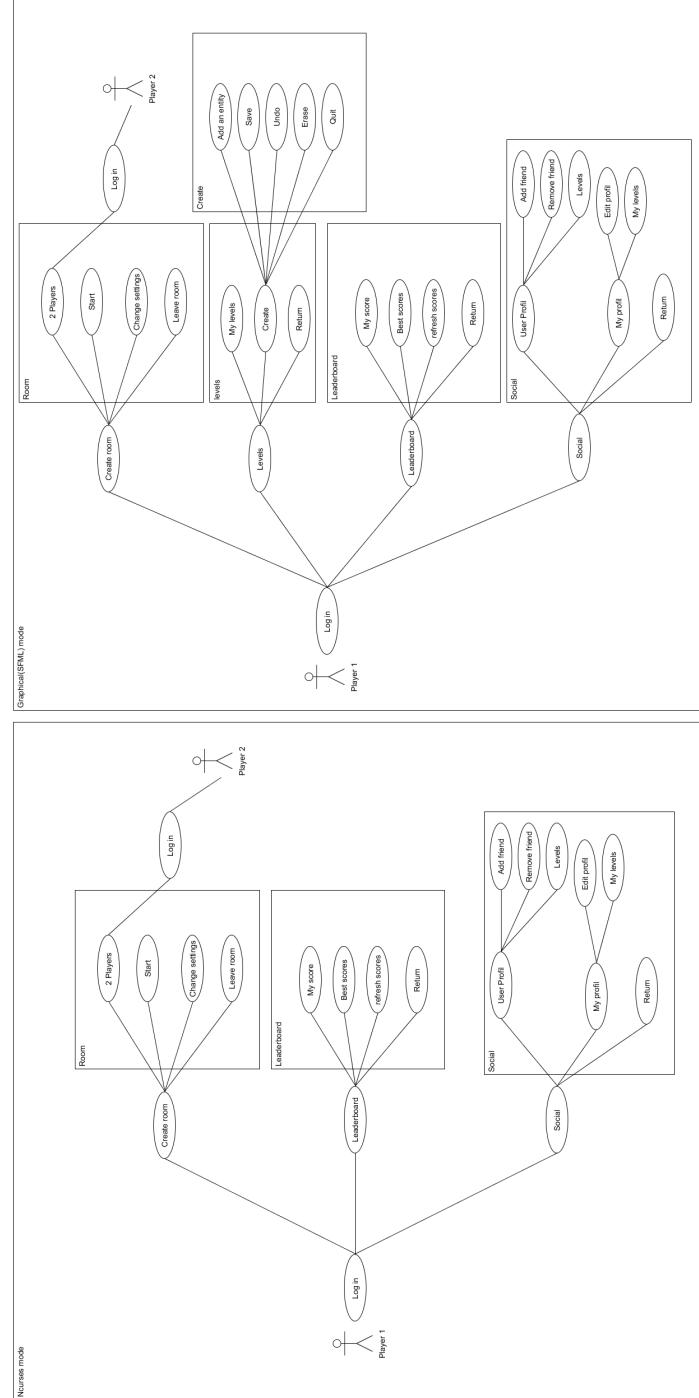


FIGURE 2 – Diagramme Use Case du menu

Use Case	Pré-conditions	Post-conditions	Cas général
Create Room	néant	est dans une room	changer les settings
Levels	être en partie graphique	est dans le sous menu levels	affiche un second choix de menu
Leaderboard	néant	est dans le menu du leaderboard	affiche les scores
Social	néant	est dans le menu "sociale"	affiche son profil et ses amis

Explication du Menu : accès aux différents menu ; la création d'une partie, le leaderboard et le menu pour le côté sociale.

5.3 Room Menu

Use Case	Pré-conditions	Post-conditions	Cas général
Start	néant	est dans une partie qui correspond aux options	jouer seul/à deux
Change Settings	player=host	l'option désirée a changé	change un des paramètres : 2 joueurs, nombrevies, la difficulté, la chance d'avoir un bonus et le friendly fire
Leave	néant	la room n'existe plus et est dans le menu principal	fermer la room et revenir dans le menu principal
2 Players	avoir quelqu'un avec qui jouer	est connecté dans la partie	se connecte à son compte ou en créer un

Explication de Room Menu : Commencer une partie avec un(e) ami(e), changer les paramètres de la partie et de pouvoir quitter le salon.

5.4 Leaderboard

Use Case	Pré-conditions	Post-conditions	Cas général
Leaderboard	néant	est dans le leaderboard	affiche le score du joueur et les meilleurs scores
my score	a déjà joué	le leaderboard s'affiche	montre constamment son score
refresh scores	quelqu'un a joué depuis le dernier refresh	le leaderboard s'est actualisé	met à jour les scores en temps réel.
best scores	quelqu'un a déjà joué	le leaderboard s'actualise	montre les meilleurs scores
return	néant	est dans le menu principal	retour dans le menu principal

Explication de Leaderboard : Menu dans lequel nous avons la possibilité de consulter les meilleurs scores réalisés de tout le monde, de ses amis et de soi-même.

5.5 Social

Use Case	Pré-conditions	Post-conditions	Cas général
My profil	néant	est dans son profil	peut éditer son profil, voir ses amis et ses niveaux
User profil	a cherché un compte qui existe	le profil s'est affiché	ajouter ou supprimer l'utilisateur, voir ses niveaux
return	néant	est dans le menu principal	retour dans le menu principal

Explication de Social : Menu dans lequel nous avons la possibilité de consulter son profil, le profil d'un autre joueur, de gérer sa liste d'amis et voir ses niveaux.

5.6 Levels

Use Case	Pré-conditions	Post-conditions	Cas général
My levels	a déjà créé un niveau	voit ses niveaux créés	affiche les niveaux qu'il a créé
Create	néant	a créé ou non un niveau	rentre dans un mode d'édition de niveau
return	néant	est dans le menu principal	retour dans le menu principal

Explication de Levels : Menu où se trouve 2 sous menus, "my levels" qui contient les niveaux créés par l'utilisateur et "create" qui permet de rentrer dans un mode d'édition de niveau.

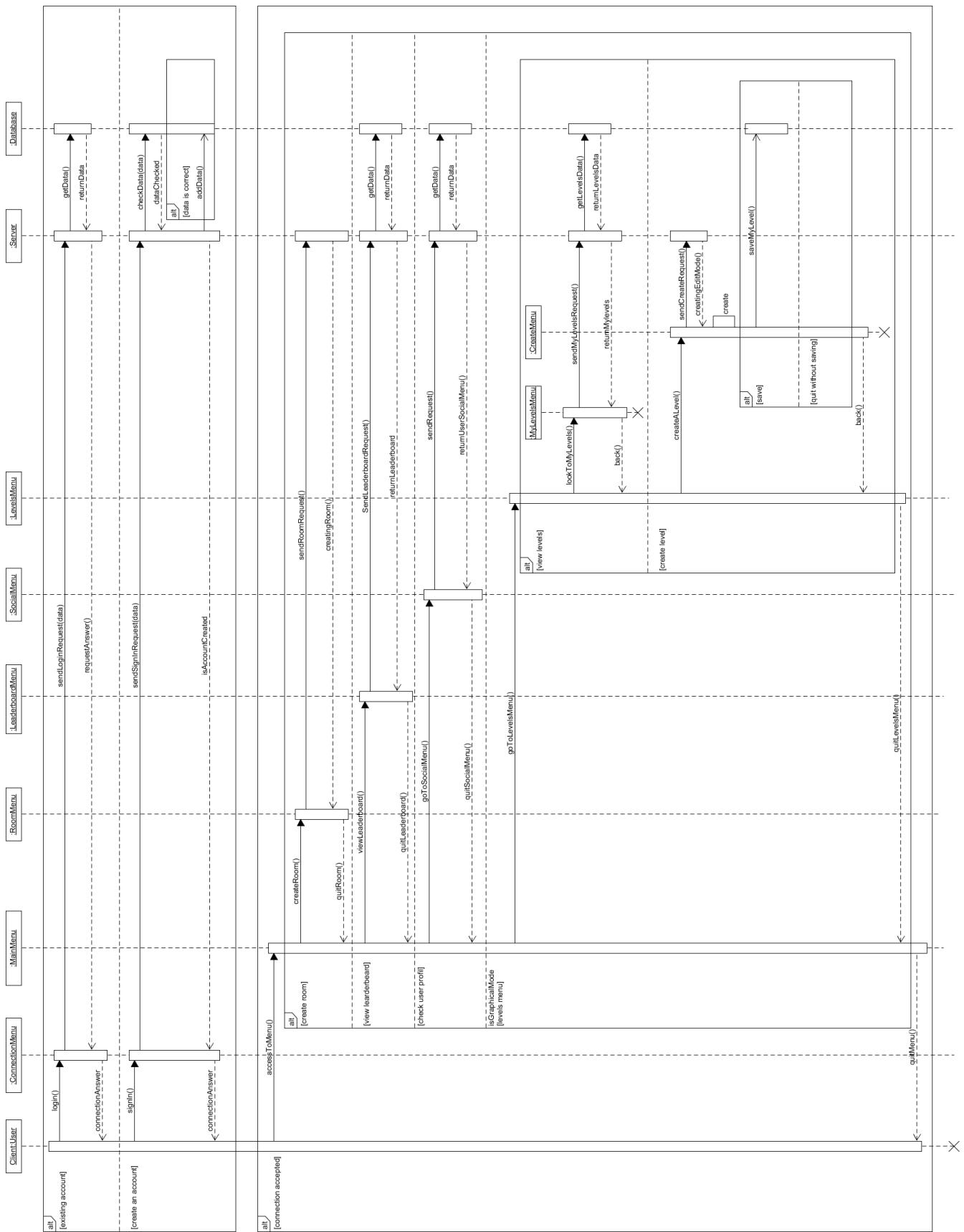


FIGURE 3 – Diagramme de sequence du menu

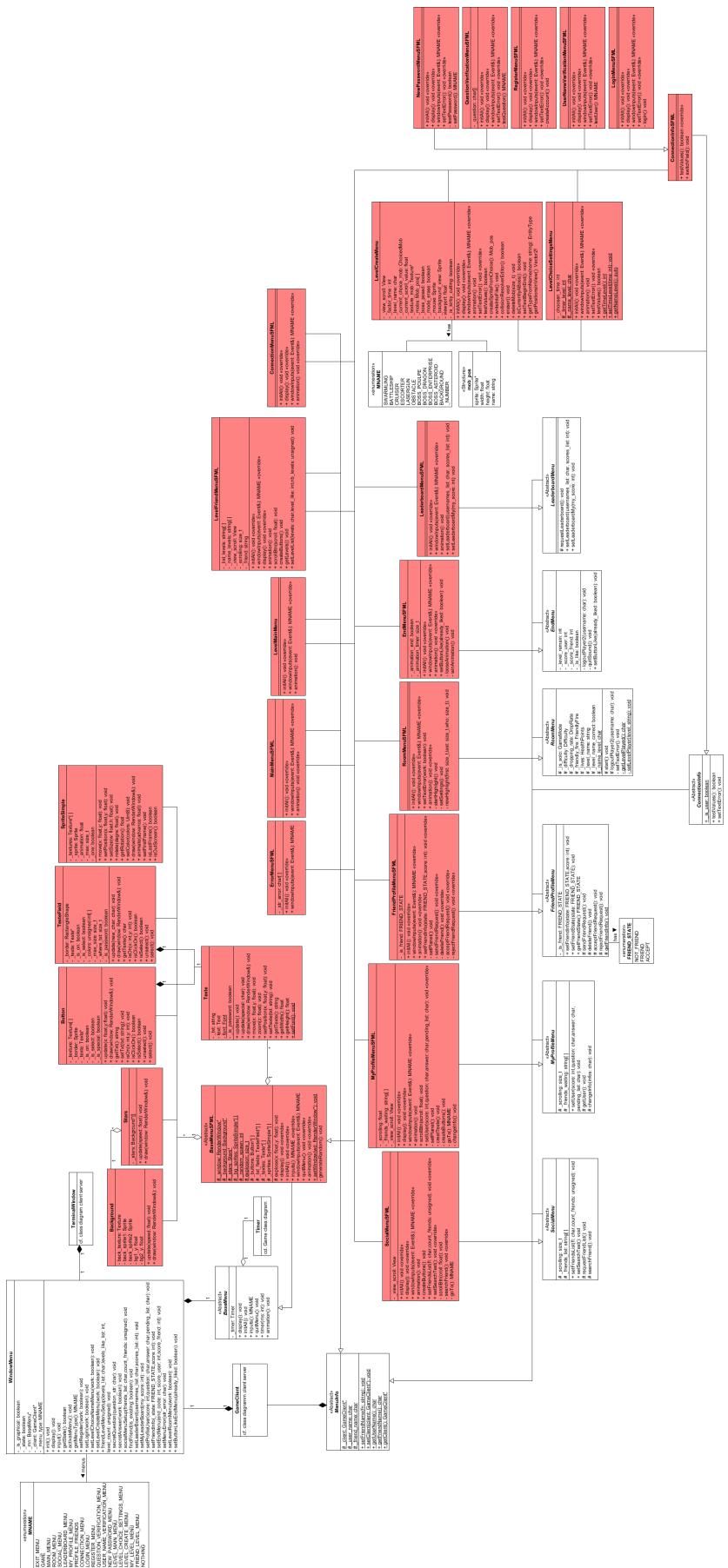


FIGURE 4 – Diagramme de classe du menu - partie graphique(SFML)

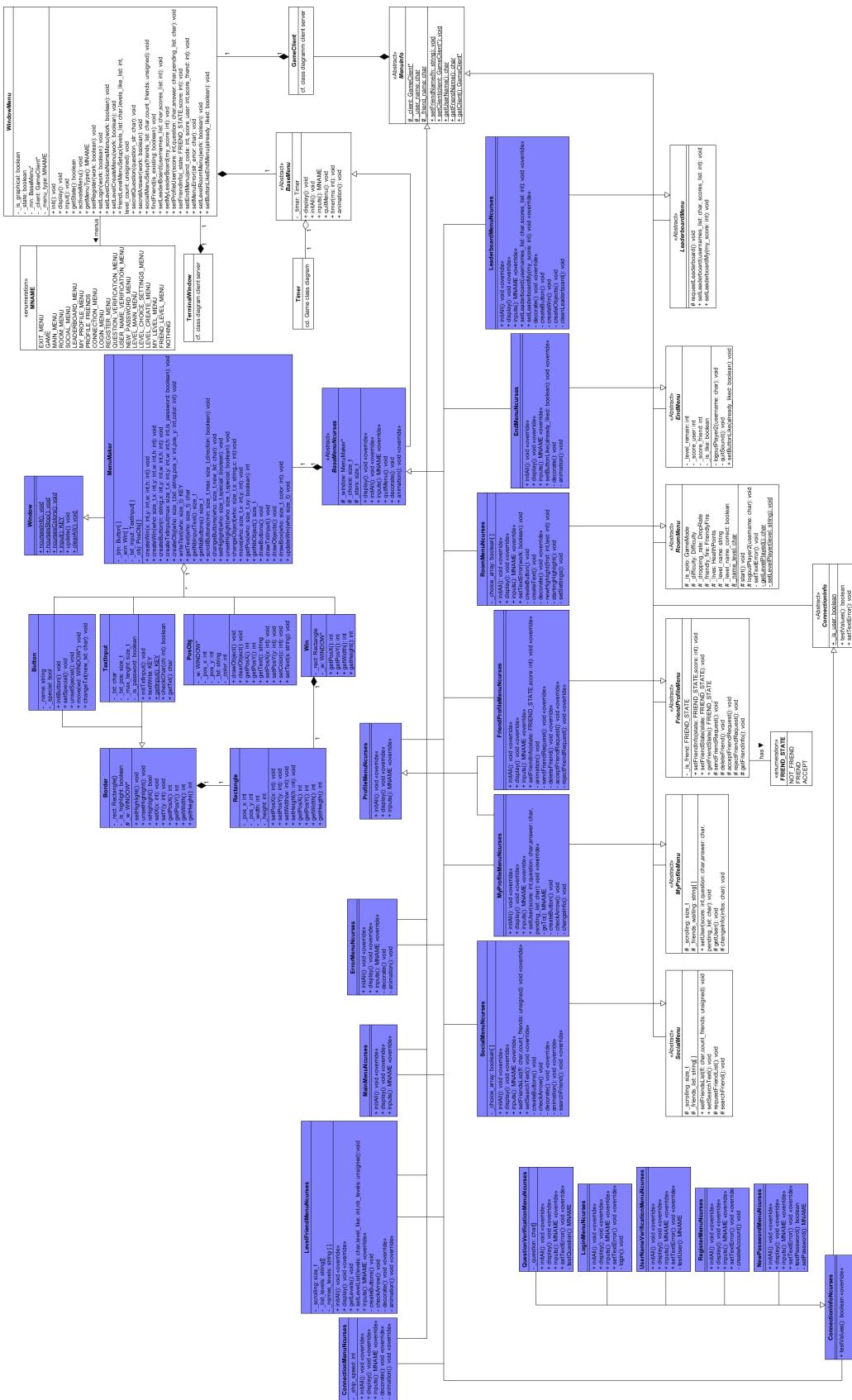


FIGURE 5 – Diagramme de classe du menu - partie terminal(Ncurses)

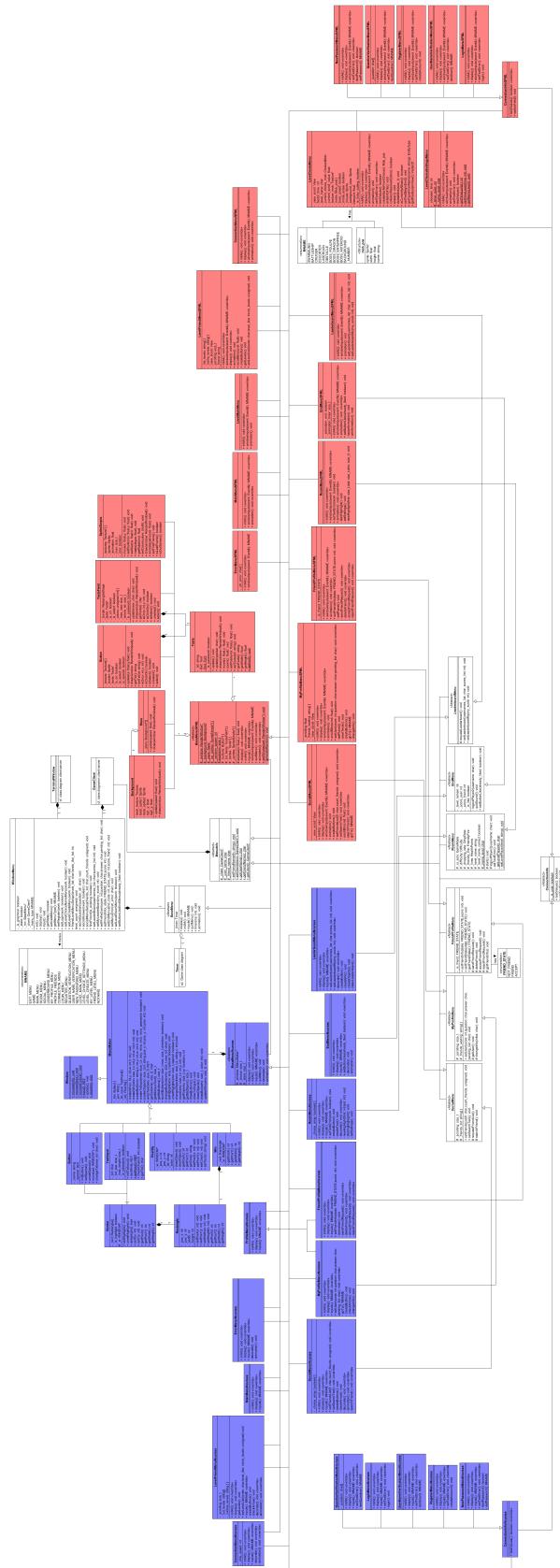


FIGURE 6 – Diagramme de classe du menu- vue d'ensemble

5.7 Online Menu

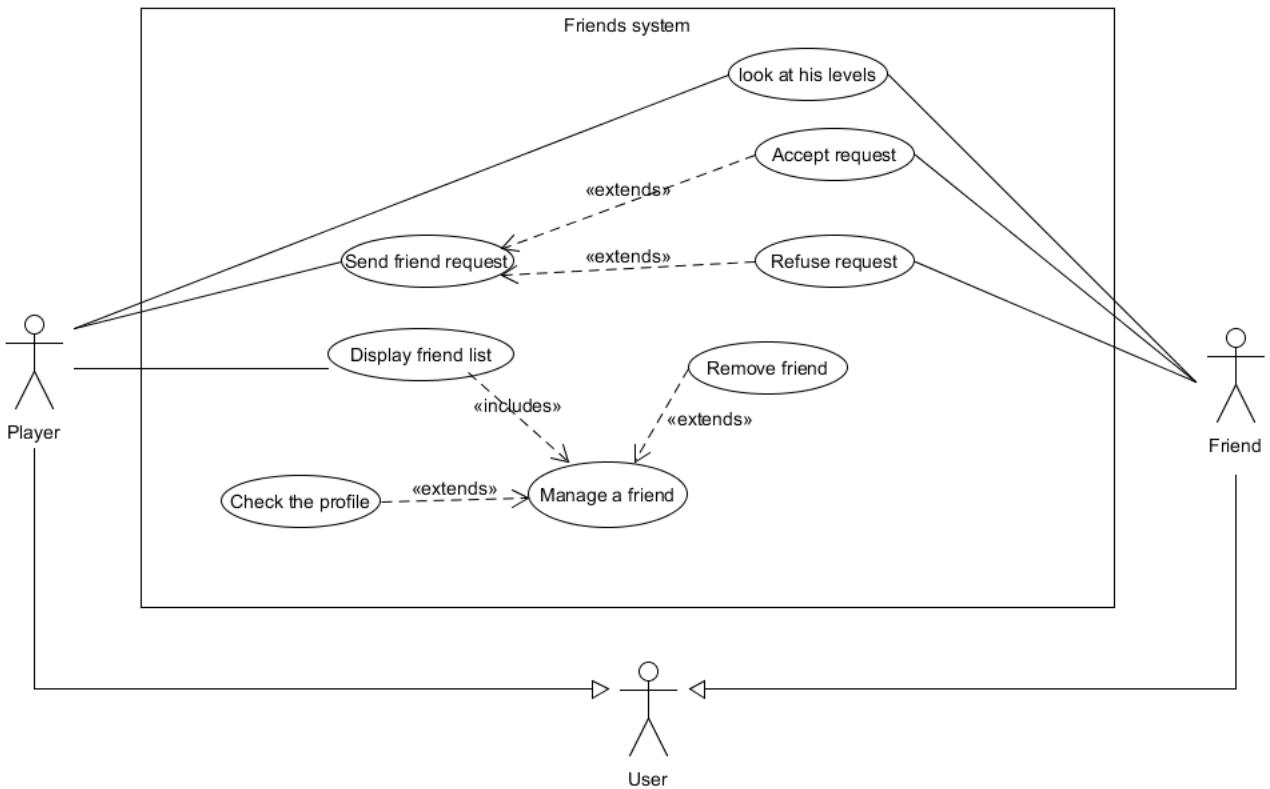


FIGURE 7 – Diagramme Use Case du côté social

Use Case	Pré-conditions	Post-conditions	Cas général	Cas exceptionnels
Display friendlist	Avoir au moins un ami	Néant	Affiche les amis du joueur	Néant
Manage a friend	Choisir un amicissement	Néant	Remove friend	Néant
Remove Friend	Être ami avec le joueur	Ne plus avoir le joueur en ami	Retirer l'ami de sa liste d'ami	Néant
Send friend request	Avoir le pseudo exact du joueur	Néant	La demande d'ami est envoyée	Le pseudo n'est pas correct
Look at his levels	Avoir le pseudo exact du joueur	voit les niveaux de l'utilisateur	affiche les niveaux d'un utilisateur	aucun niveau est présent
Accept request	Avoir reçu une demande d'ami	Avoir un nouvel ami	Les deux joueurs s'ajoutent mutuellement dans leur liste d'amis	Néant
Refuse request	Avoir reçu une demande d'amis	Néant	La demande d'ami est supprimée	Néant
Check profile	Avoir le pseudo exact	Néant	Affiche le profil du joueur au pseudo correspondant	Le pseudo n'est pas correct

Explication Use Case de Online Menu : Menu où nous regardons notre liste d'ami(e)s et pouvons envoyer une invitation, supprimer un(e) ami(e), annuler une demande, etc.

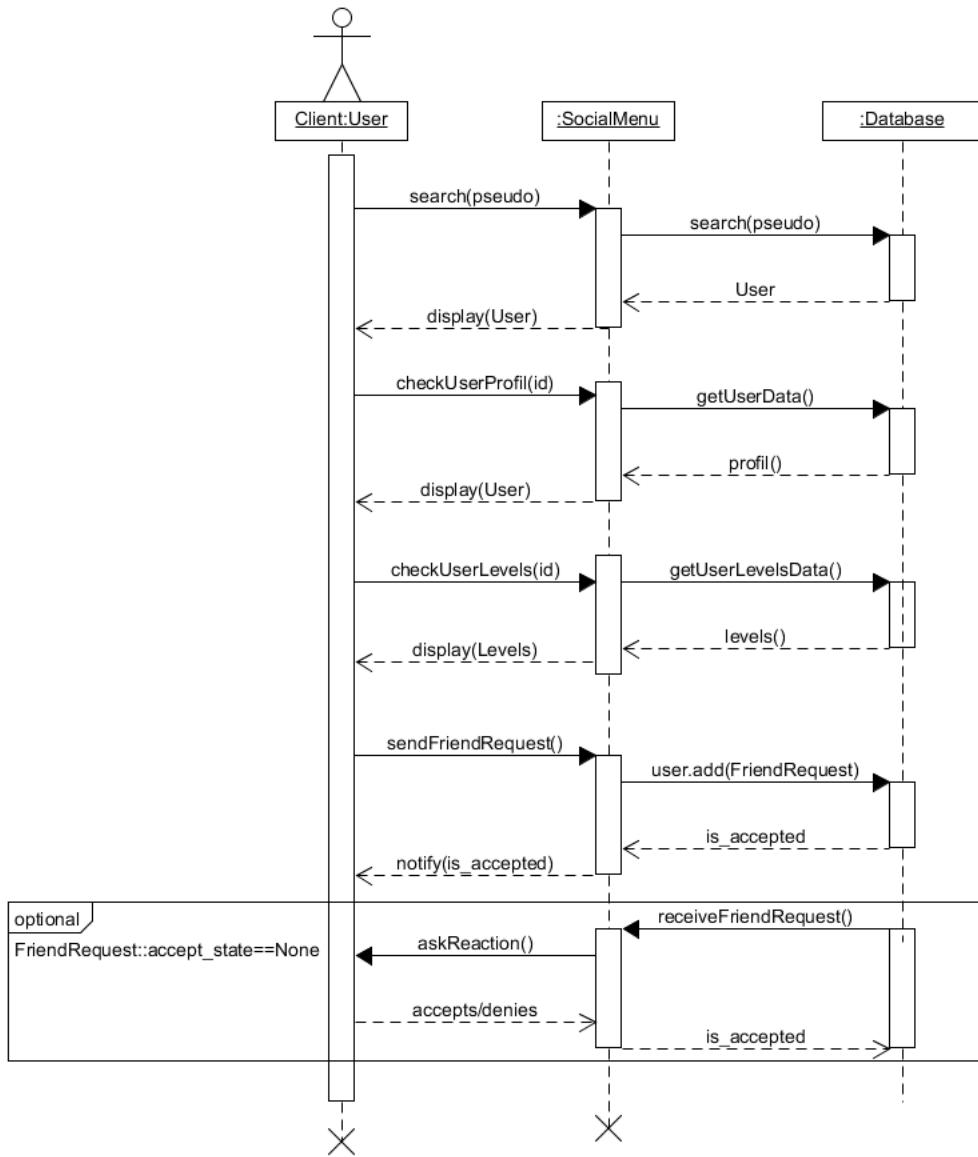


FIGURE 8 – Diagramme de séquence des amis

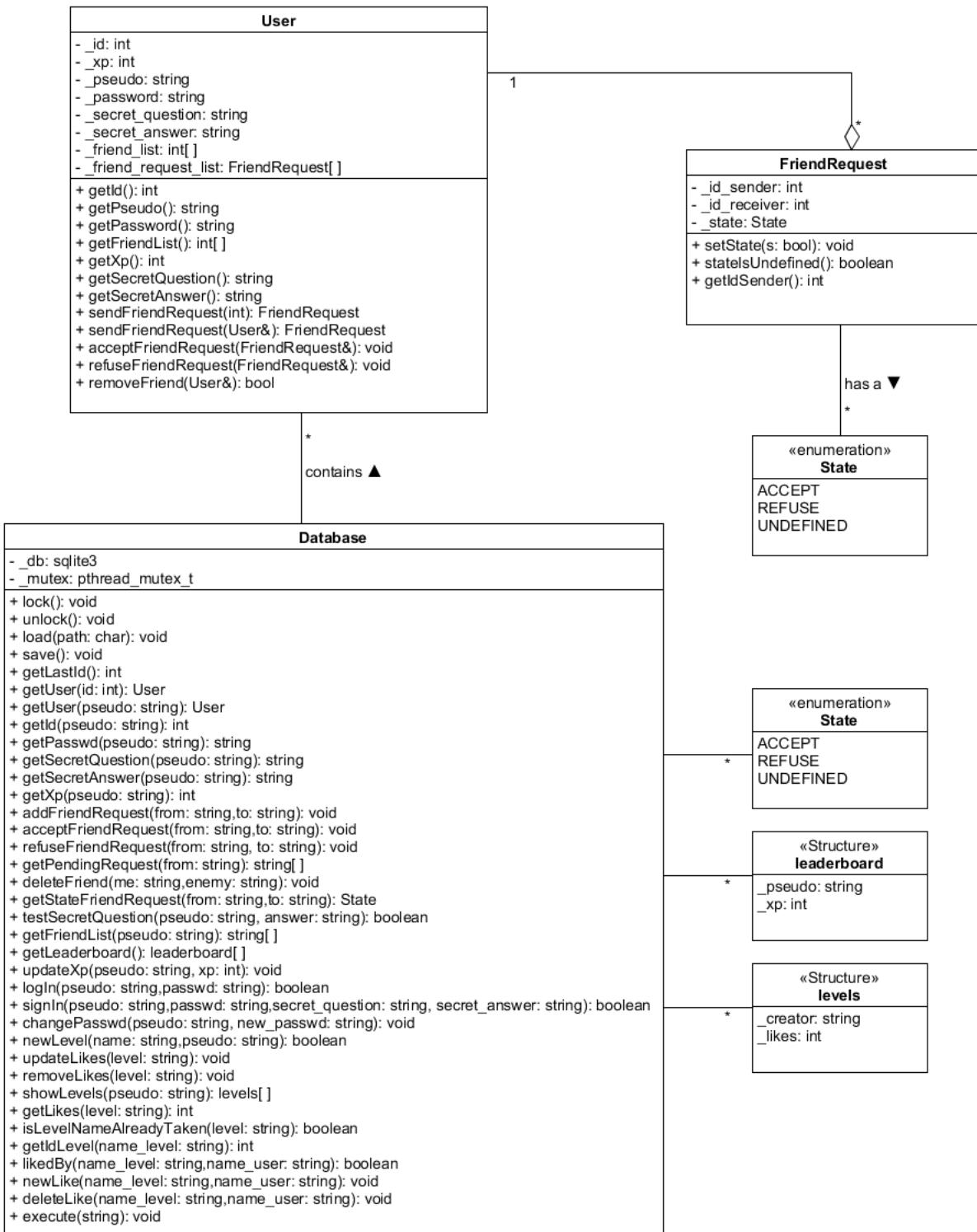


FIGURE 9 – Diagramme de classe de la database

5.8 Game

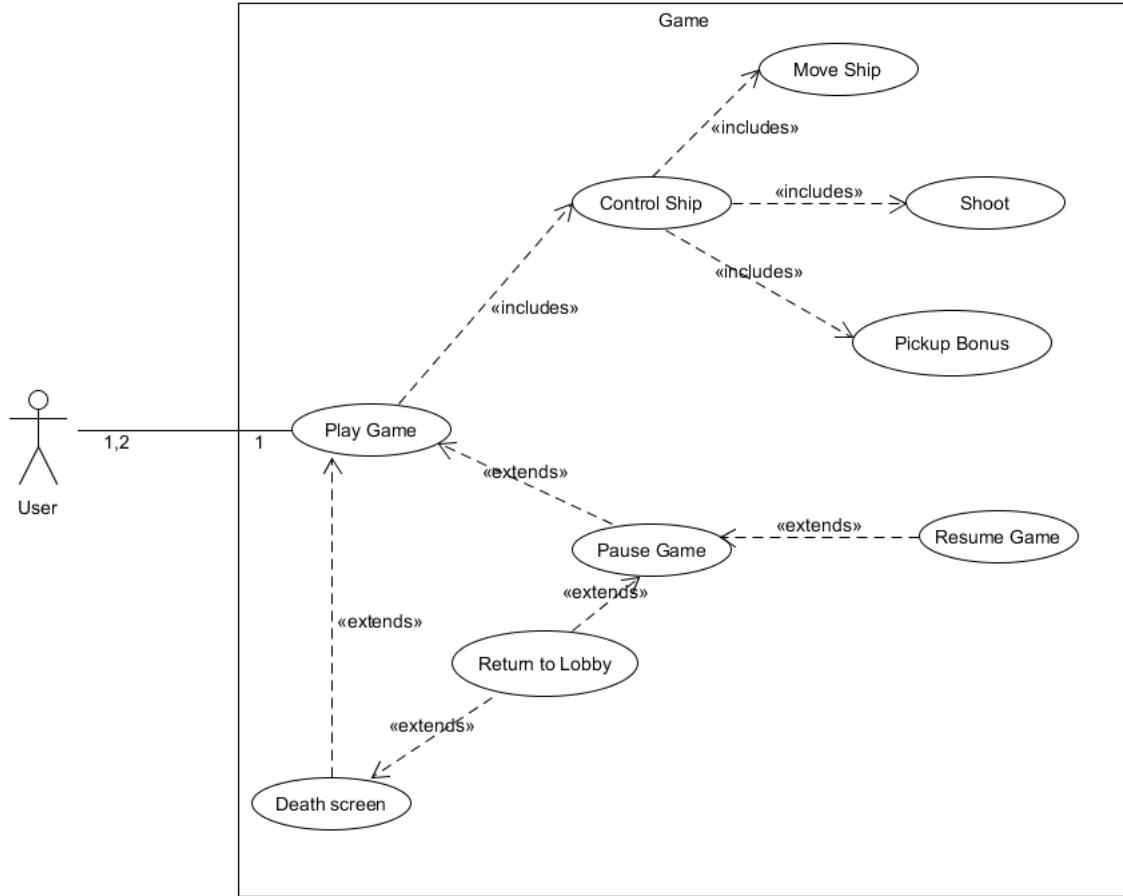


FIGURE 10 – Diagramme Use Case de la partie

Use Case	Pré-conditions	Post-conditions	Cas général
Play Game	est connecté au jeu	néant	c'est l'écran principal pour les 1/2 joueur(s)
Pause Game	le jeu tourne	le jeu est arrêté	le jeu est mis en pause
Resume Game	le jeu est arrêté	le jeu tourne	le jeu est remis en route
Return to Lobby	le jeu est arrêté	le joueur est dans le lobby	termine la partie et retourne dans le lobby
Control Ship	le jeu tourne	néant	tout ce qui peut être fait avec le vaisseau
Move Ship	le jeu tourne	le vaisseau bouge	le vaisseau peut bouger dans les directions sans sortir de l'écran
Shoot	le jeu tourne	néant	tire des projectiles qui endommagent à la collision
Pickup Power Ups	le jeu tourne	le vaisseau a un bonus	bouger le vaisseau vers le power up pour utiliser l'arme
Death Screen	le joueur a perdu	quitter vers le lobby	le menu qui apparaît quand le joueur perd
End Credits	gagner	retour au menu principal	les crédits sont montrés à la fin du jeu
Like	joue un niveau créé	le niveau a un like en plus	ajoute un like au niveau

Explication Use Case de Game : Le déroulement du jeu ainsi que ses possibilités.

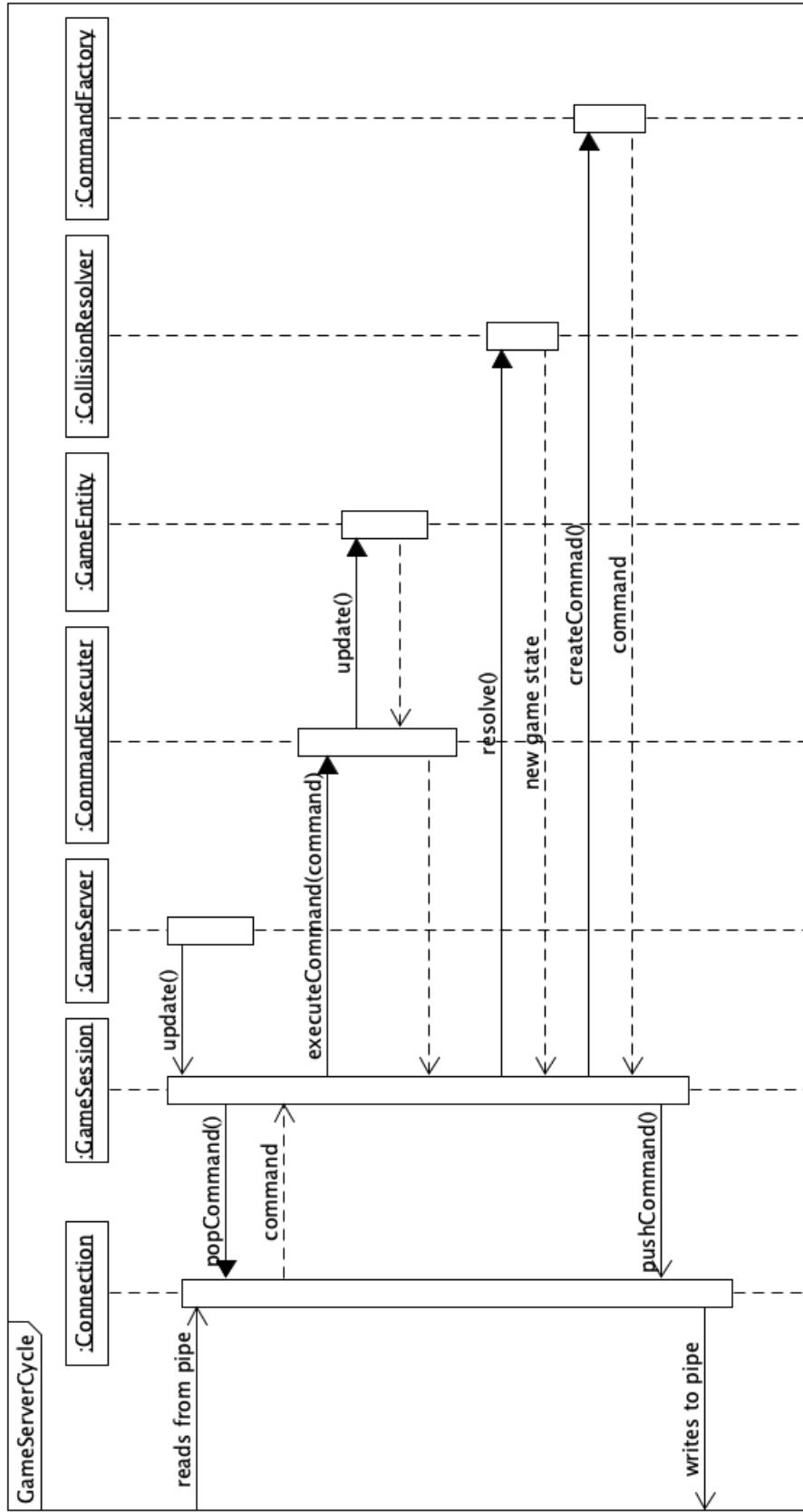


FIGURE 11 – diagramme de séquence serveur de jeu

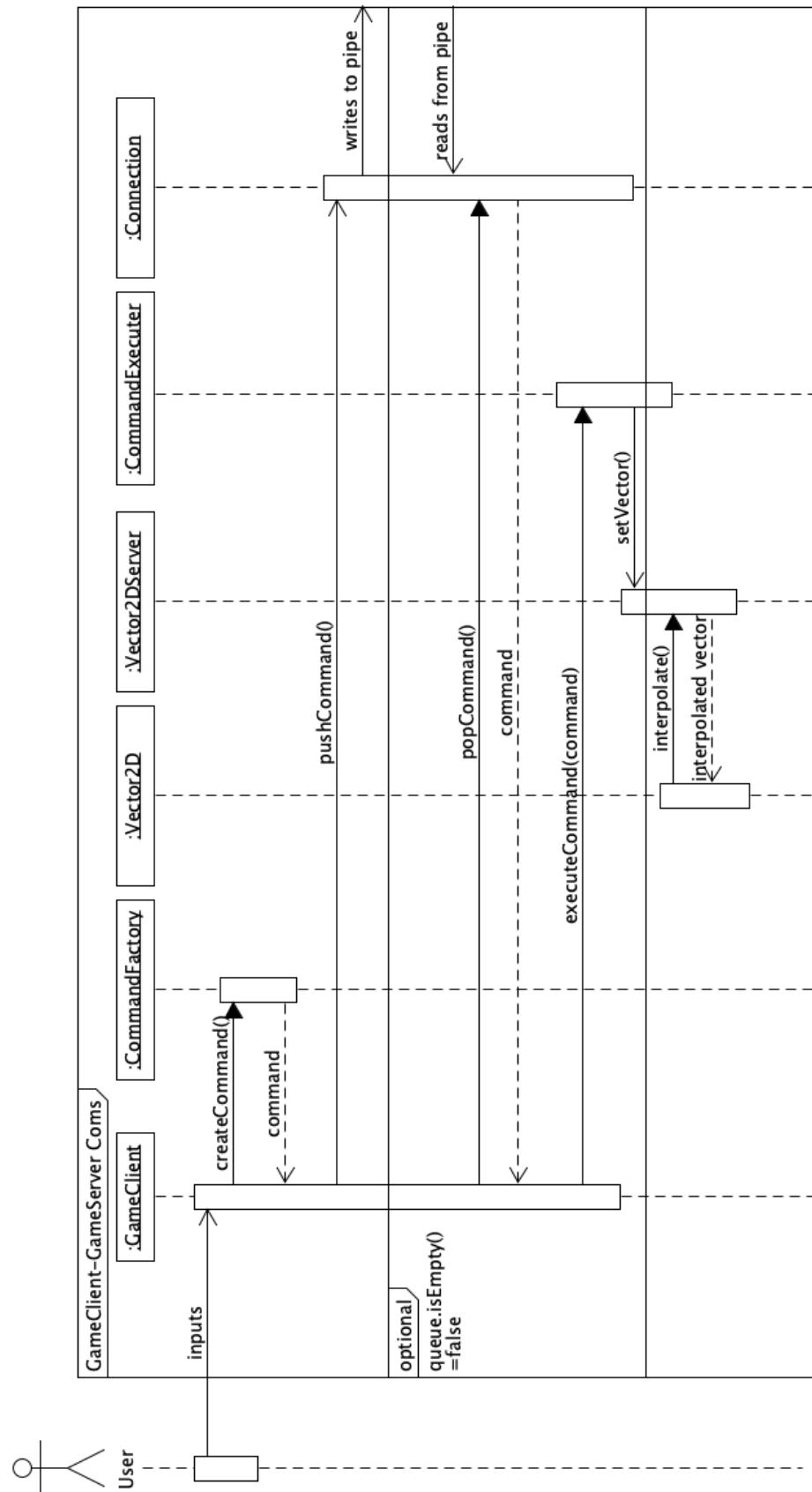


FIGURE 12 – diagramme de séquence client serveur de jeu

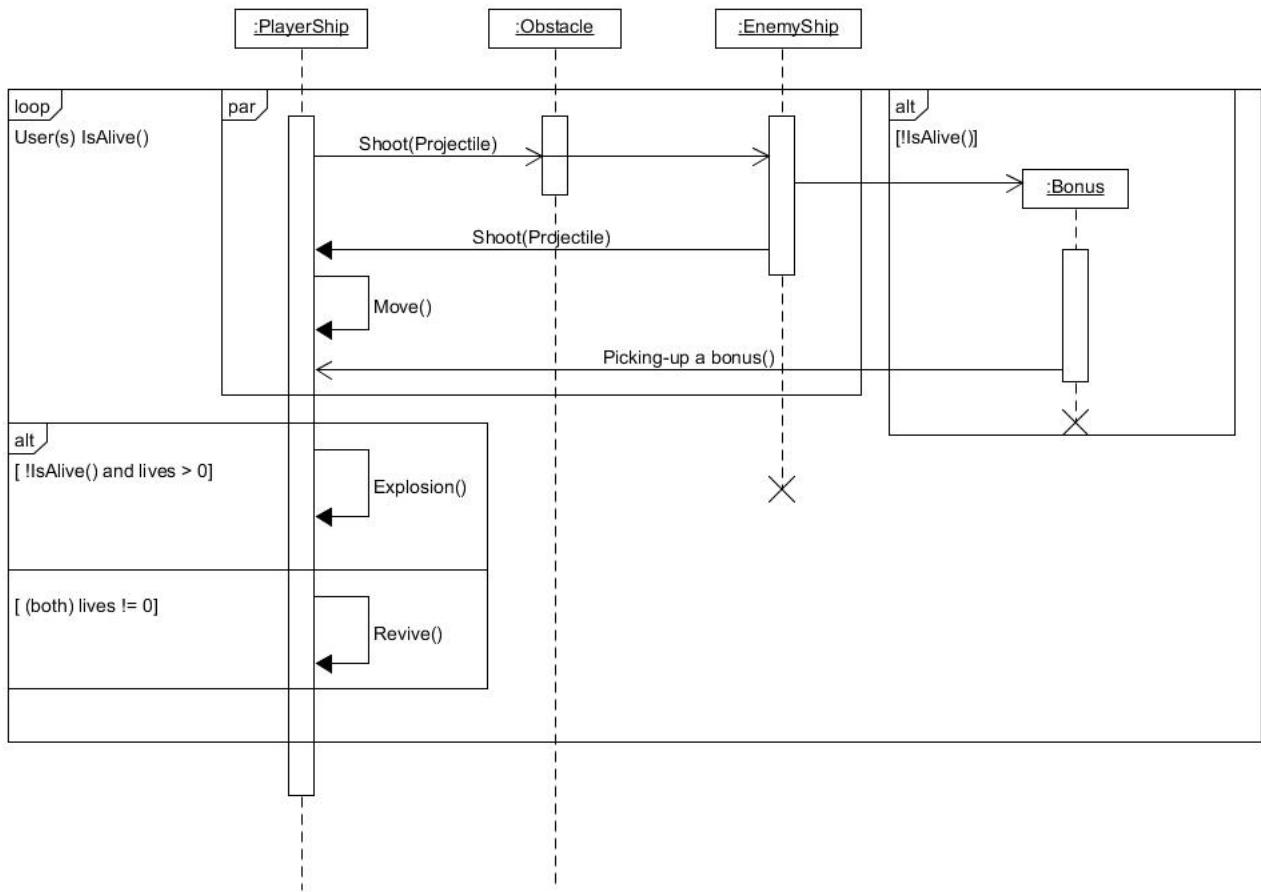


FIGURE 13 – diagramme de séquence de la partie

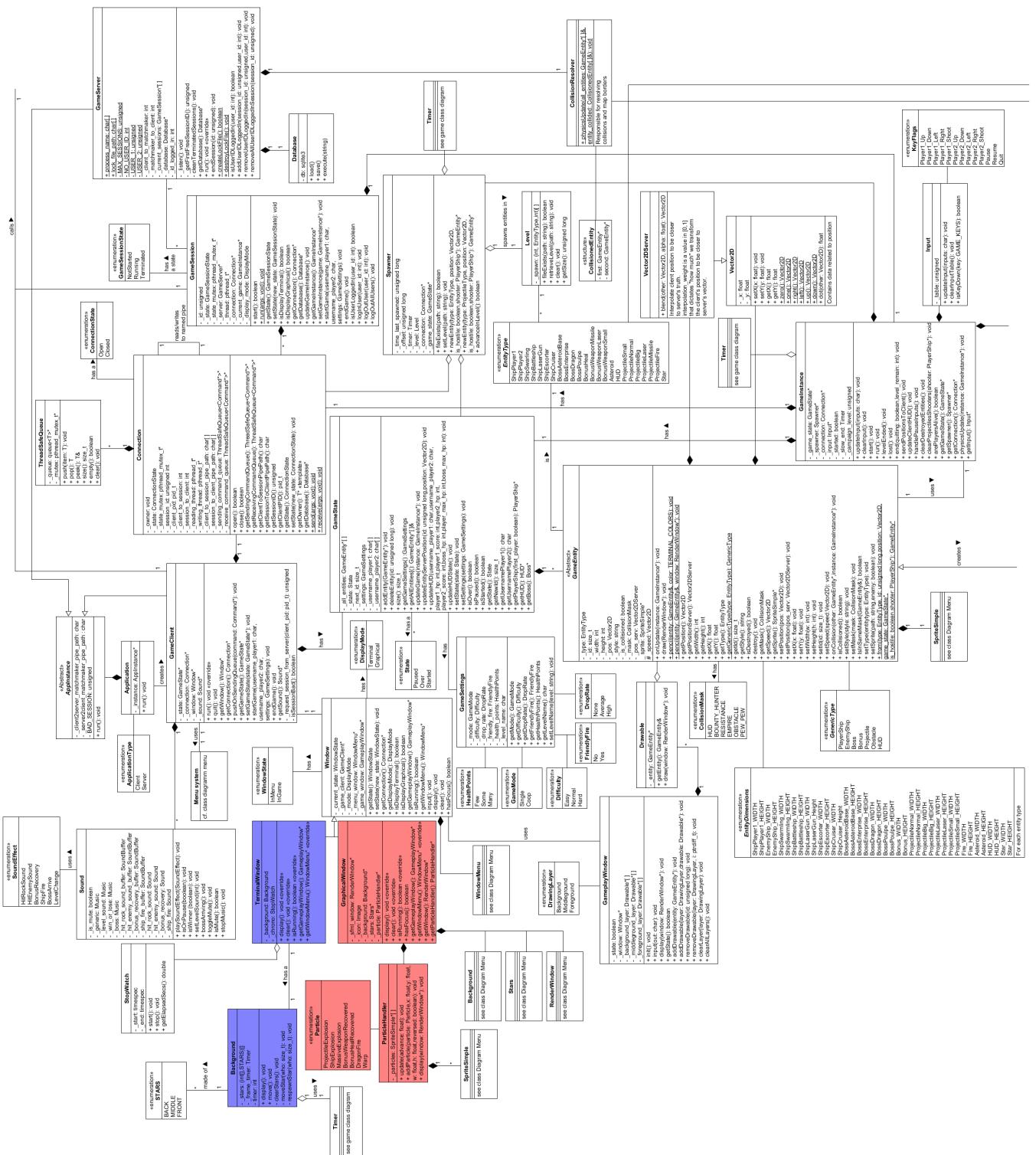


FIGURE 14 – Diagramme de classe du jeu(1)- partie sur le serveur et une partie du client

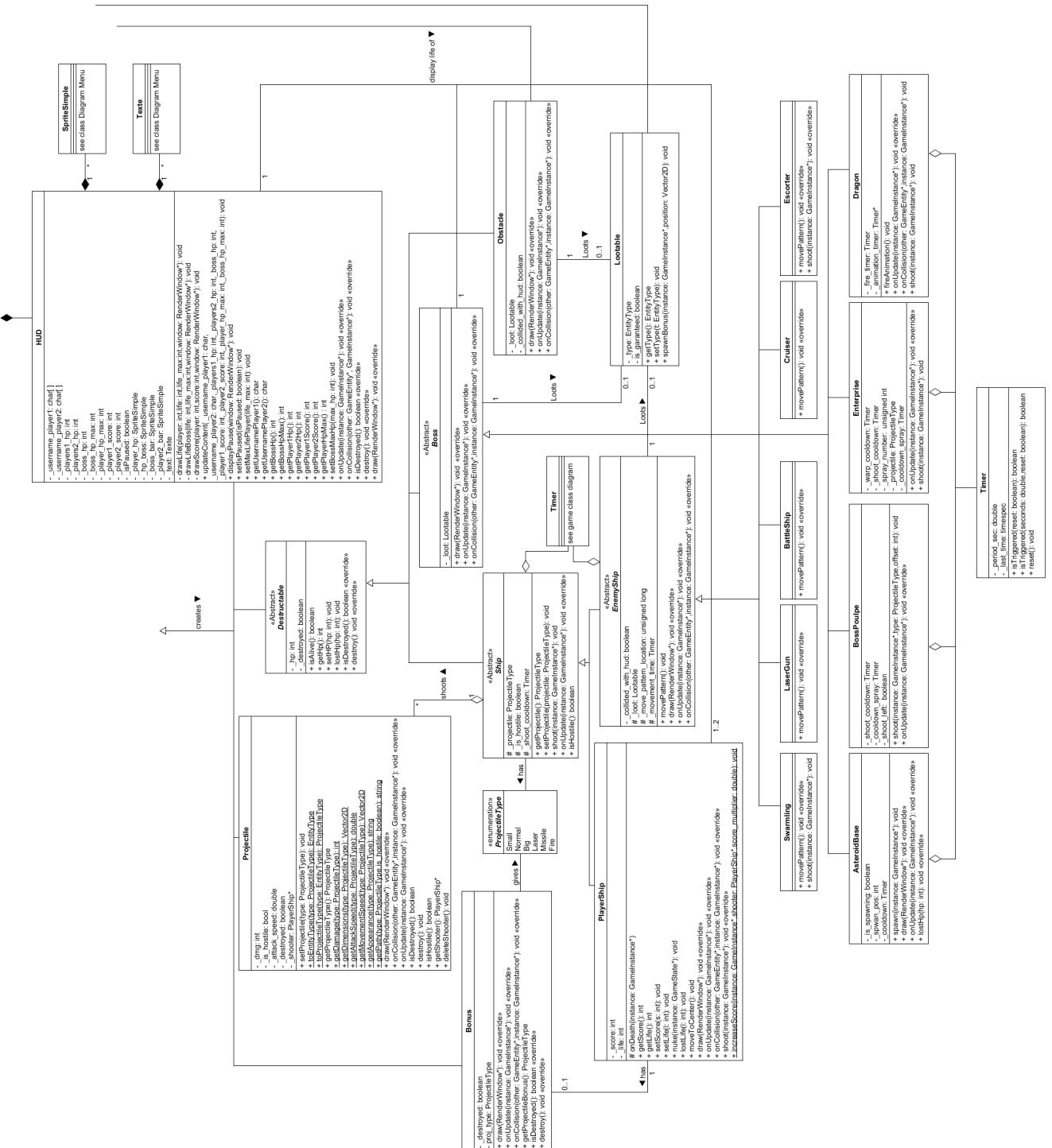


FIGURE 15 – Diagramme de classe du jeu(2)- partie sur la logique du jeu

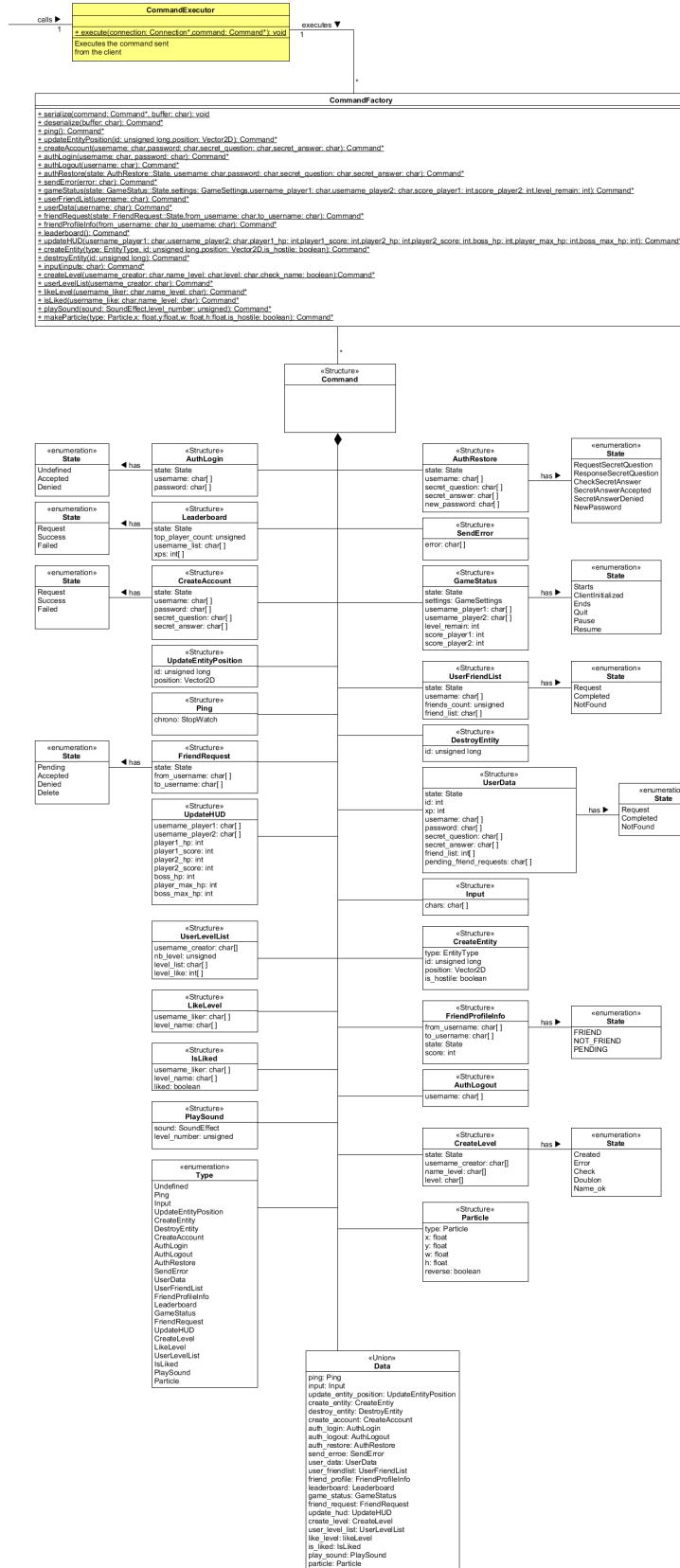


FIGURE 16 – Diagramme de classe du jeu(3) - partie sur les commandes

6 Explications pour le diagramme de la partie sur les commandes.

La structure Command représente une commande, elle peut être lu et écrite depuis un pipe. L'union quant à elle est la liste de toutes les commandes possibles (une structure par type de commande). L'énumération "Type" représente tous les types de commandes afin de faciliter la sérialisation. Les différentes commandes sont :

- Ping : Fais un aller-retour jusqu'au serveur et affiche le temps que cela a mis. Cela sert surtout à debugger et à montrer que le client se connecte au serveur.
- Input : Envoie une liste des touches appuyées dans une frame par le client. La session les récupère et réalise les actions demandées.
- UpdateEntityPosition : Le serveur envoie cette commande au client une fois que les collisions sont résolues afin de mettre à jour le vecteur de position présent sur le serveur.
- CreateEntity : Le serveur envoie cette commande au client quand il crée une nouvelle entité.
- DestroyEntity : Le serveur envoie cette commande au client quand il détruit une entité.
- CreateAccount : Commande envoyée par le client au serveur ("Request"). Le serveur crée potentiellement le compte dans la base de données ("Success", "Failed") et renvoie l'information au client.
- AuthLogin : Le client envoie cette commande au serveur. Le serveur renvoie une réponse qui peut avoir plusieurs états "undefined", "Accepted", "Denied" qui sont respectivement l'état de base que le client envoie au serveur, l'état signifiant que le login est une réussite et l'état de l'échec du login.
- AuthLogout : Commande envoyée par le client au serveur pour déconnecter un utilisateur, principalement utilisé pour déconnecter le second joueur étant donné que le premier utilisateur ne peut pas se déconnecter du jeu sans le quitter.
- AuthRestore : Commande utilisée lorsqu'un utilisateur a oublié son mot de passe. Il fera plusieurs allers-retours entre le client et le serveur et aura plusieurs états pendant l'exécution comme AuthLogin.
- SendError : Envoie une erreur à l'utilisateur.
- UserData : Le client l'envoie afin de recevoir les informations de l'utilisateur. Elle a aussi plusieurs états en cours de route.
- UserFriendList : Une commande envoyée par le client afin d'avoir les informations concernant la liste d'amis de l'utilisateur.
- FriendProfileInfo : Commande afin de récupérer les informations sur le profil d'un ami.
- Leaderboard : Commande qui demande les informations du leaderboards du serveur.
- GameStatus : Préviens la session/le client qu'une partie a commencé ou s'est terminée.
- FriendRequest : Le client envoie cette commande au serveur pour le notifier qu'un utilisateur a envoyé une demande d'ami à un autre utilisateur.
- UpdateHUD : Commande envoyée par le serveur au client afin de mettre à jour l'HUD.
- CreateLevel : Une commande que le client peut utiliser pour faire 2 choses : savoir si un nom est déjà utilisé pour un niveau et pour créer un niveau.
- UserLevelList : Renvoie au client tous les niveaux créés par l'utilisateur (username_creator).
- LikeLevel : Like un niveau.
- PlaySound : Commande que le serveur envoie au client afin de jouer un son.
- Particle : Commande qui joue les VFX.

La classe "CommandFactory" est responsable de créer ces commandes en fonction des événements et de les mettre dans la file d'attente de Connection.

La classe "CommandExecutor" est responsable d'exécuter les commandes en fonction du type de celle-ci et de ses données prises depuis la file d'attente de Connection.

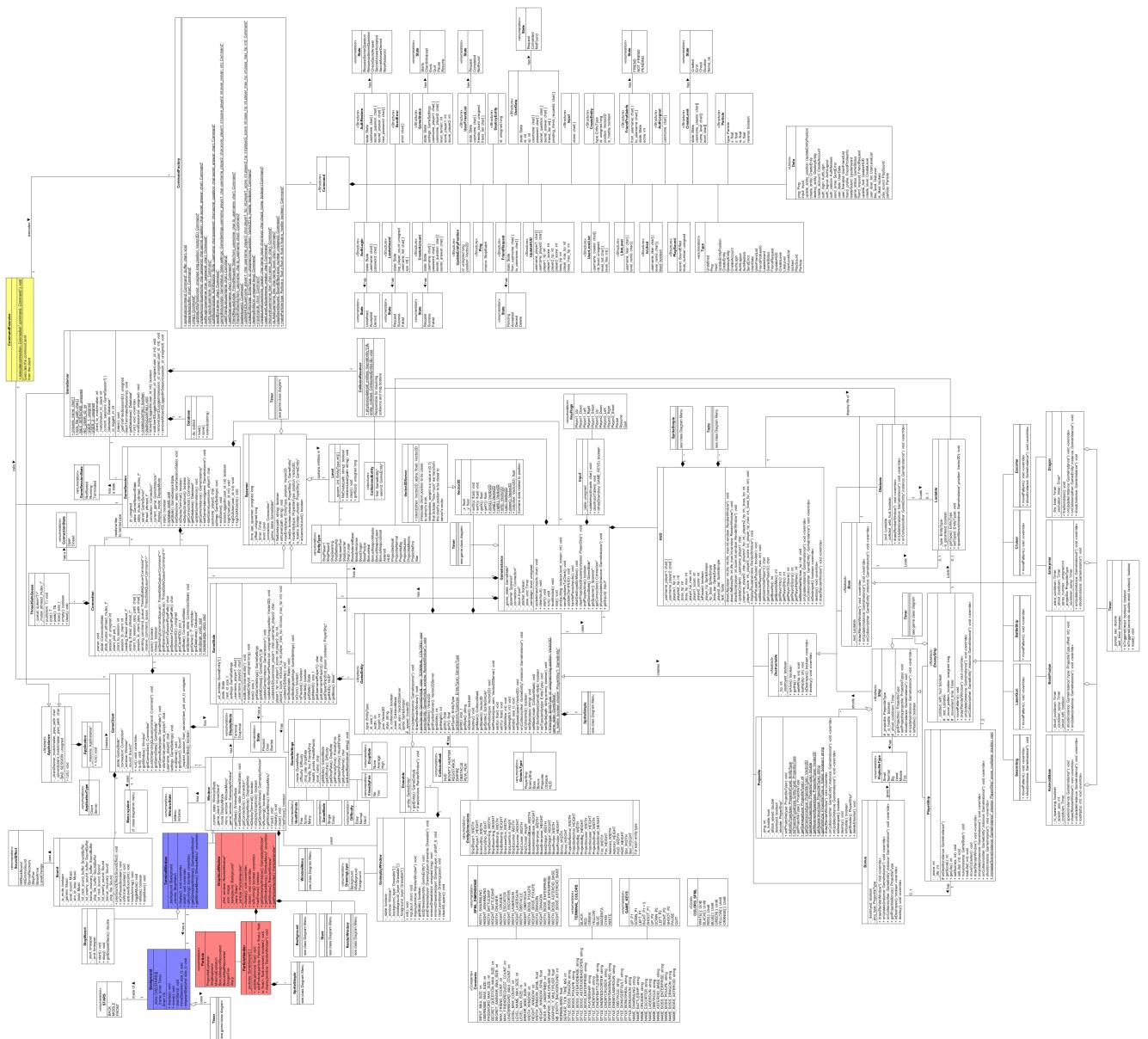
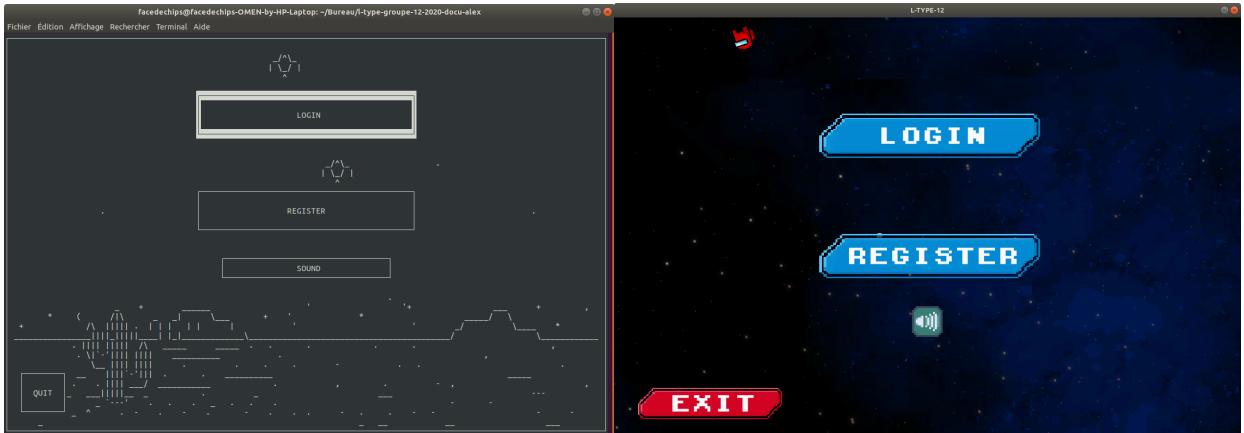


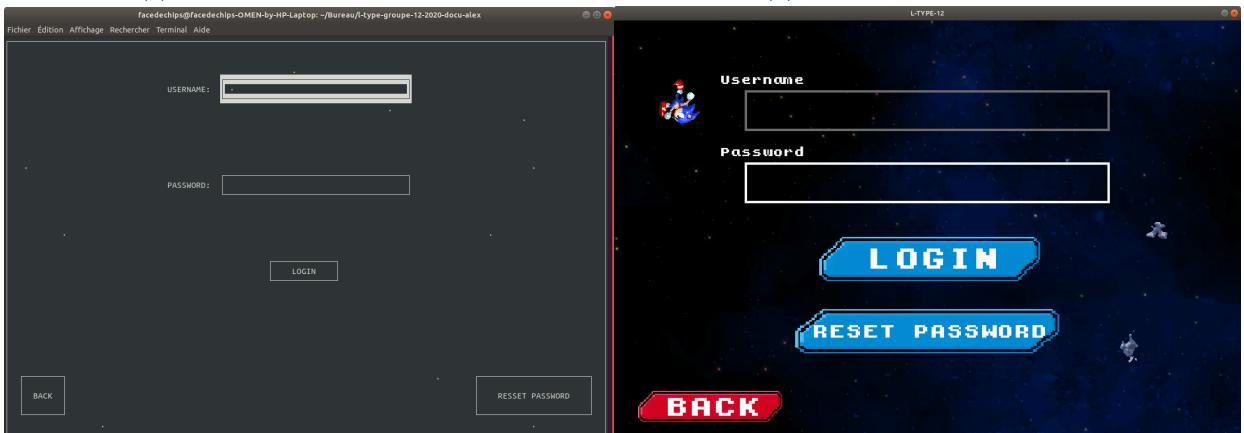
FIGURE 17 – Diagramme de classe du jeu- vue d'ensemble

7 Screenshots



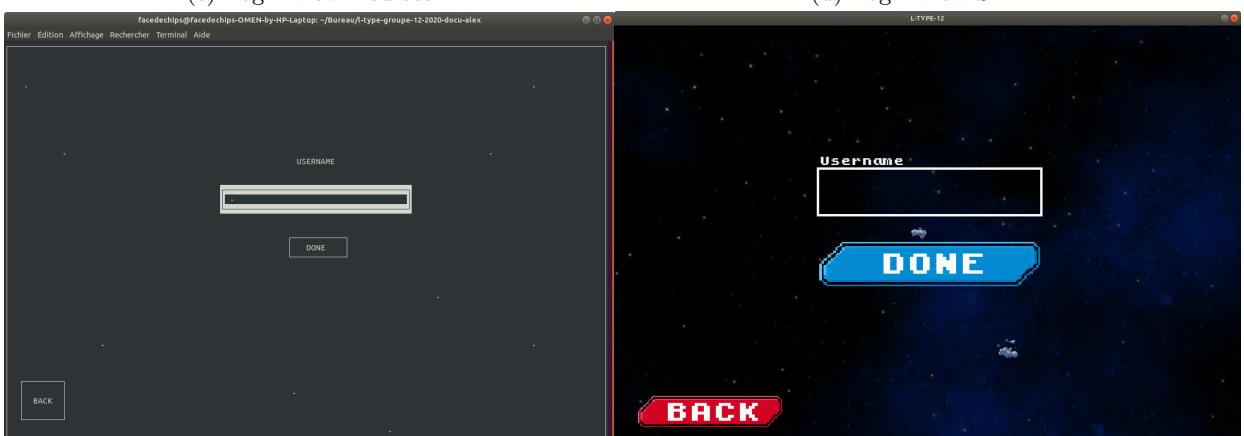
(a) Connection window Ncurses

(b) Connection window SFML



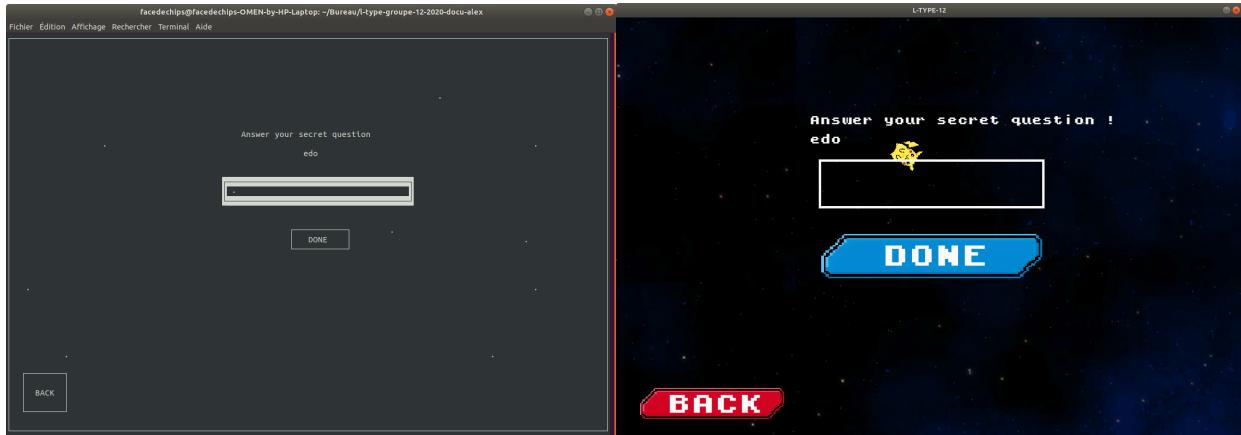
(c) Login view Ncurses

(d) Login view SFML



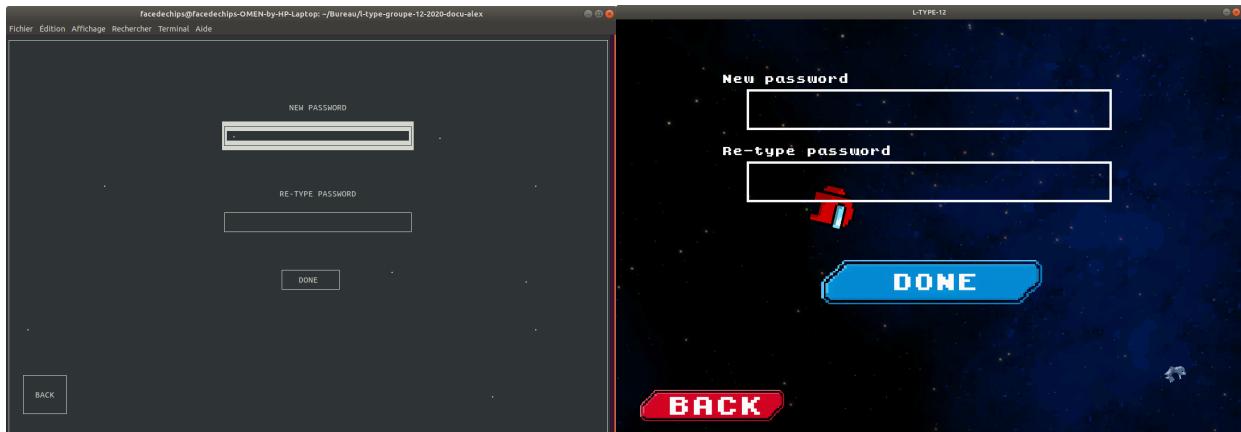
(e) Reset password Ncurses(1)

(f) Reset password SFML(1)



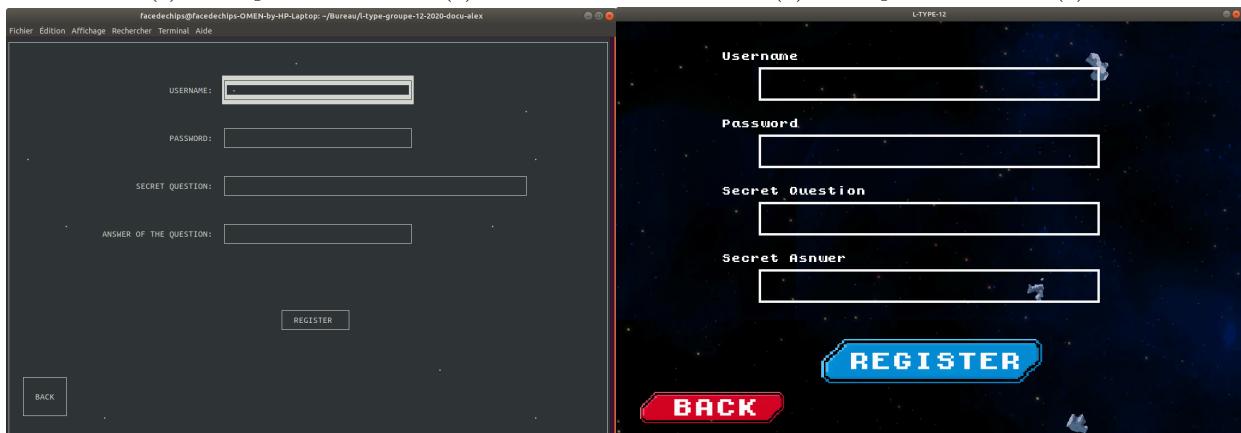
(a) Reset password Ncurses(2)

(b) Reset password SFML(2)



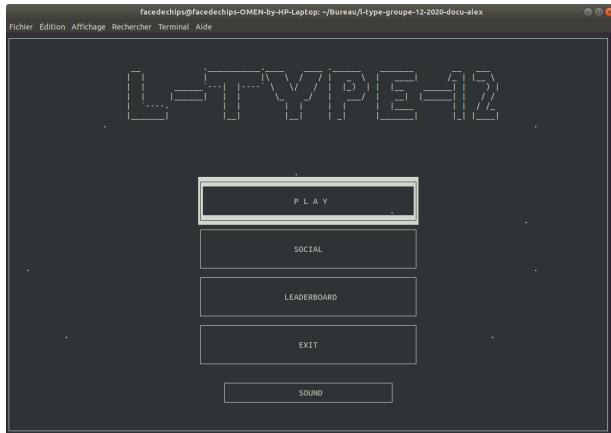
(c) Reset password Ncurses(3)

(d) Reset password SFML(3)



(e) Register view Ncurses

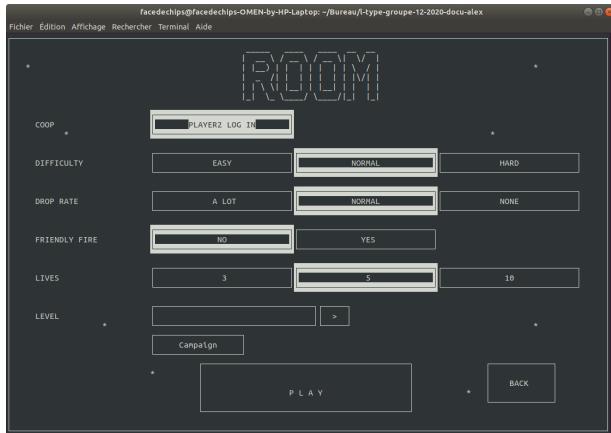
(f) Register View SFML



(a) Main menu Ncurses



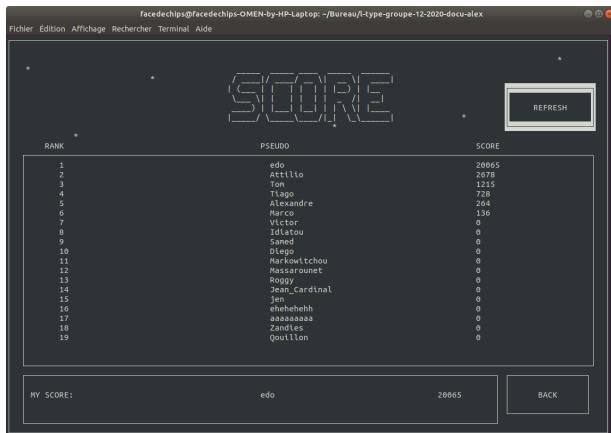
(b) Main menu SFML



(c) Room menu Ncurses



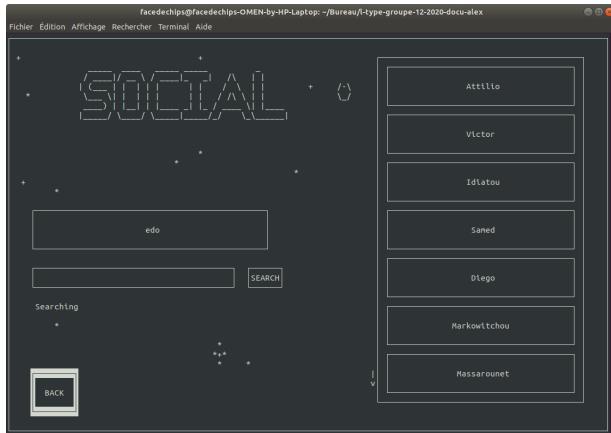
(d) Room menu SFML



(e) Leaderboard menu Ncurses



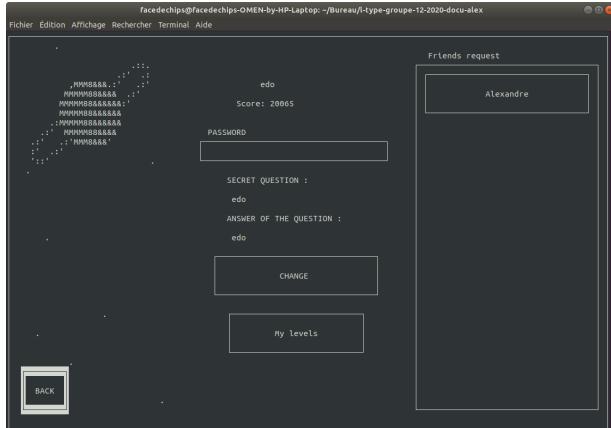
(f) Leaderboard menu SFML



(a) Social menu Ncurses



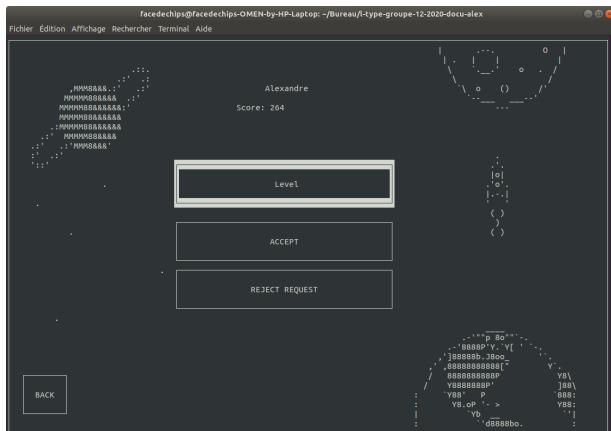
(b) Social menu SFML



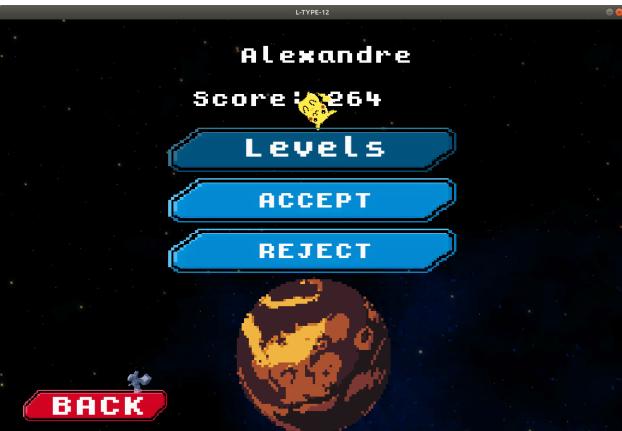
(c) My profil view Ncurses



(d) My profil view SFML



(e) User profil view Ncurses



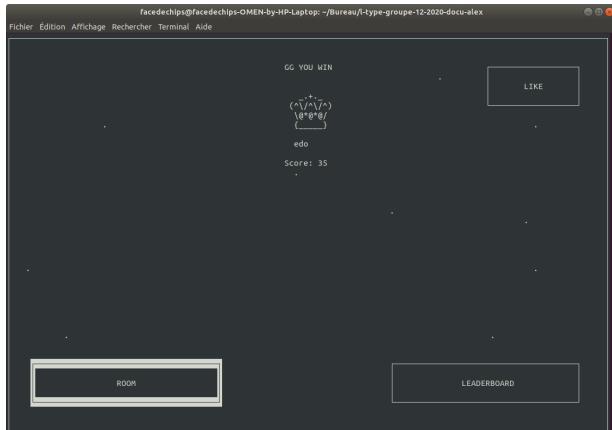
(f) User profil view SFML



(a) In game view Ncurses



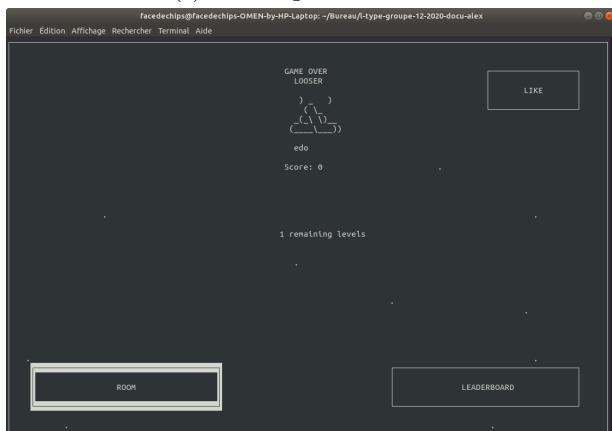
(b) In game view SFML



(c) Winning view Ncurses



(d) Winning view SFML



(e) Losing view Ncurses

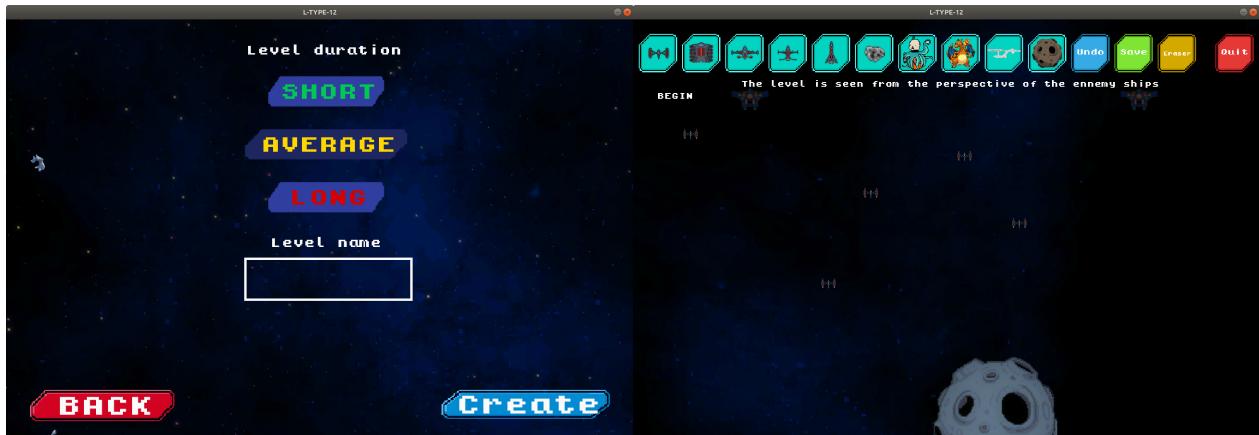


(f) losing view SFML



(a) Levels menu SFML

(b) My levels view SFML



(c) Create a level menu SFML

(d) Editing mode view SFML

L-TYPE-12