Université Libre de Bruxelles

Projets d'informatique II

Tetristitude

SRD 3V-CUBE

Auteurs:
Derras Eiman
Drion Elisha
Bouali Abderrahmane

Rouaux Thomas
Libert Alexandre
Jouniaux Alexane

March 5, 2018

Contents

1 Introduction

Le but du projet est de développer un jeu en réseau à l'aide d'outils acquis depuis le début de notre cursus. Celui-ci doit être une reproduction du célèbre "Tetris", disposant d'un mode multijoueur ainsi que d'un système de gestion des challengers. Se nommant 3V-CUBE, il est intégralement écrit en C++.

Il est demandé d'accorder une attention particulière aux concepts vus aux cours d'INFO-F-201 et d'INFO-F-204. Le client-serveur gérant le jeu et le SRD seront donc les points phares de ce projet.

Comme tout jeu en réseau, le joueur a la possibilité de créer un compte pour suivre ses statistiques, trouver des parties adaptées à son niveau mais également de discuter avec ses amis ainsi que d'autres fonctionnalités s'offrant à lui.

Il y a 4 modes de jeu disponibles : classique, marathon, sprint et VS (en ligne).

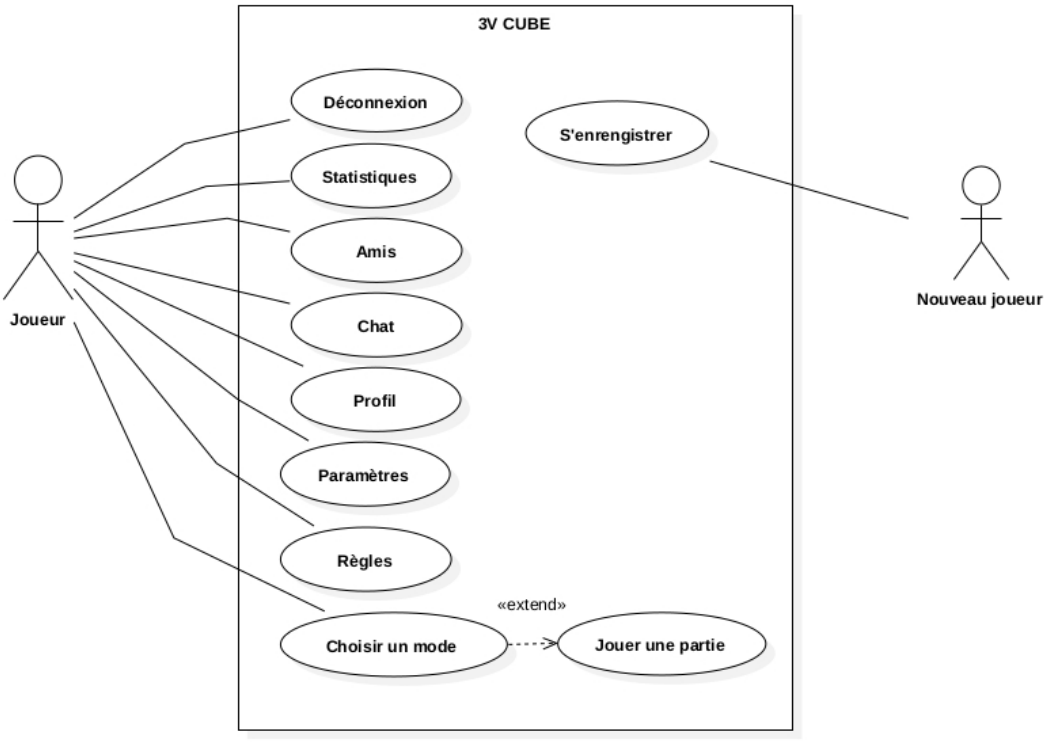
2 Glossaire

Gravité : Constante qui détermine la vitesse de chute des tetriminos.
Salon : Endroit virtuel où des joueurs attendent de jouer/jouent ensemble.
Tetrimino : Forme géométrique de 4 carrés que le joueur manipule.
Matchmaking : Processus de mise en relation de deux joueurs afin de créer une partie.

3 Historique du document

Version	Date de modification	Auteur	Résumé
0.1	26/11/2017	Eiman	Ajout de l'introduction, de la grille d'historique et des USE CASE
0.2	27/11/2017	Elisha	Ajout glossaire, remaniement structure besoins fonctionnels, màj introduction
0.3	28/11/2017	Abderr, Thomas, Alexandre, Alexane	Use case matchmaking
0.4	01/12/2017	Eiman	Ajout de besoins fonctionnels, remaniement structure matchmaking
0.5	13/02/2018	Abderr, Alexane	Ajout des diagrammes de classes, de séquence et d'activité et remaniement structure globale
0.6	22/02/2018	Alexane	Mise à jour diagramme de séquence et d'activité
0.7	24/02/2018	Alexane	Remaniement texte et structure
0.8	28/02/2018	Alexane	Ajout des diagrammes de classe,des use-case (et remaniement de certains) et texte complété

4 Diagramme général

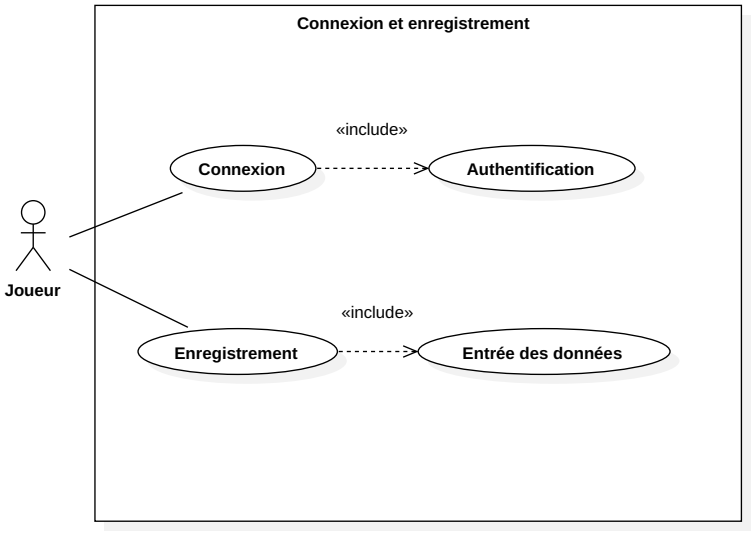


L'utilisateur aura une multitude de choix devant lui. Il pourra entre autre consulter ses statistiques personnelles et son classement mondial, il pourra aussi chatter avec ses différents amis et évidemment regarder les statistiques de ceux-ci. Mais également, avoir le choix de changer les paramètres comme il l'apprécie. Enfin, il pourra choisir un mode de jeu et jouer en solo ou faire une recherche de salon favorable à celui-ci.

5 Besoins fonctionnels de l'utilisateur

5.1 Connexion et enregistrement

Condition générale : Serveur doit être en ligne.



5.1.1 Connexion

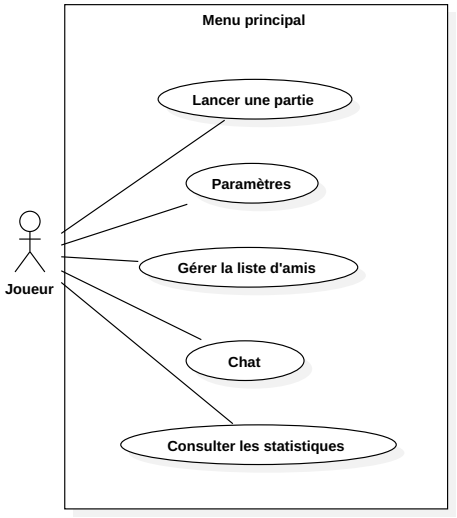
- **Acteur(s) :** Joueur.
- **Pré-conditions :** Le joueur doit avoir un compte.
- **Post-conditions :** Le joueur accède au menu principal.
- **Cas particulier :** Le joueur entre de mauvais identifiants. Les champs sont vidés et un message est affiché, l'invitant à entrer des identifiants valides.

5.1.2 Enregistrement

- **Acteur(s) :** Joueur.
- **Pré-conditions :** Aucune.
- **Post-conditions :** Le joueur accède à une fenêtre où il pourra s'enregistrer.
- **Cas particulier :** Le joueur n'entre pas d'identifiant ou de mot de passe ou ceux-ci ne correspondent pas au format demandé.

5.2 Menu principal

Condition générale première : Serveur doit être en ligne. Ceci concerne tous les choix du menu sauf les paramètres.
Condition générale seconde : Le joueur doit être connecté.



5.2.1 Lancer une partie

Le joueur décide à quel mode il jouera.
Pré-conditions : Le joueur doit être enregistré.

5.2.2 Paramètres

Le joueur peut modifier les touches permettant de déplacer les Tetriminos ainsi que le volume de la musique dans le jeu.

5.2.3 Gérer la liste d'amis

Le joueur peut consulter sa liste d'amis et ajouter ou supprimer des amis.

5.2.4 Chat

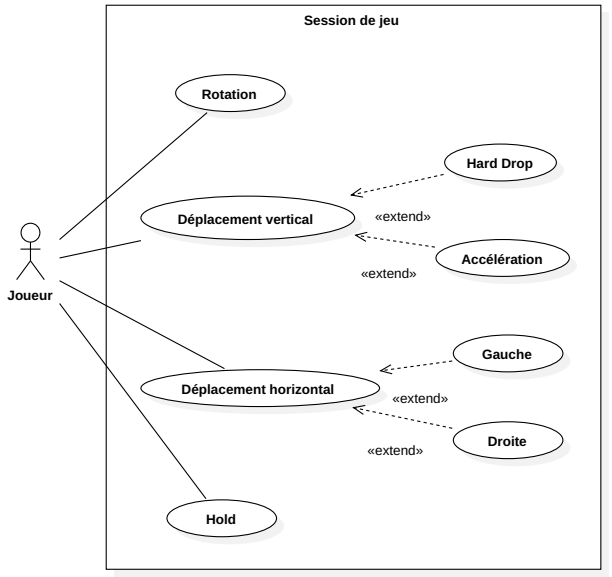
Depuis le menu principal, on peut accéder à un salon de chat. Là, on peut discuter avec tous les utilisateurs qui sont également en ligne.

5.2.5 Consulter les statistiques

Le joueur peut consulter ses statistiques personnelles ainsi que celles de ses amis.

5.3 Session de jeu

Dans une session de jeu classique, l'utilisateur peut déplacer horizontalement les tétriminos ou les tourner dans le sens souhaité. Tous ces déplacements se mesurent en unité. Les déplacements simples se mesurent par une unité quel que soit le sens ou le type de mouvement. Ainsi, indépendamment du mode de jeu, le tétriminos peut chuter directement vers le bas sans que le joueur ne puisse le redéplacer.



Enfin, l'utilisateur a la possibilité de mettre pause à toute session de jeu si celle-ci est une partie dont le mode n'est pas multijoueurs.

5.3.1 Rotation

Le joueur a la possibilité de faire tourner les pièces.

- **Rotation à gauche :** Le tetriminos effectue une rotation de 90° vers la gauche.
- **Rotation à droite :** Le tetriminos effectue une rotation de 90° vers la droite.
- **Acteur(s) :** Joueur.
- **Pré-condition(s) :** Le tetriminos doit être en train de chuter.
- **Post-condition(s) :** Le tetriminos a tourné de 90° vers la droite ou la gauche.
- **Cas particulier(s) :** Néant.

5.3.2 Déplacement pièce

- **Acteur(s) :** Joueur.
- **Pré-condition(s) :** Le tetriminos doit être en train de chuter.

Plusieurs déplacements sont possibles :

- **Déplacement horizontal :** Le joueur peut déplacer le tetriminos horizontalement durant sa chute :
  - **Déplacement à gauche :** Le tetriminos se déplace d'une unité vers la gauche.
  - **Déplacement à droite :** Le tetriminos se déplace d'une unité vers la droite.
- **Post-condition(s) :** Le tetriminos se trouve une position plus à droite/à gauche.
- **Cas particulier(s) :** Néant.
- **Déplacement vertical :** Les tetriminos "chutent" verticalement durant toute la partie. Cependant, le joueur peut influencer leur chute de deux manières différentes :
  - **Fall :** Le tetriminos descend d'une unité.
  - **Drop :** Le tetriminos tombe jusqu'en bas de la grille.
- **Post-condition(s) :** Le tetriminos se trouve une unité plus bas/tout en bas de la grille.
- **Cas particulier(s) :** Néant.

5.3.3 Hold

Le joueur décide de sauvegarder la pièce pour l'utiliser plus tard.

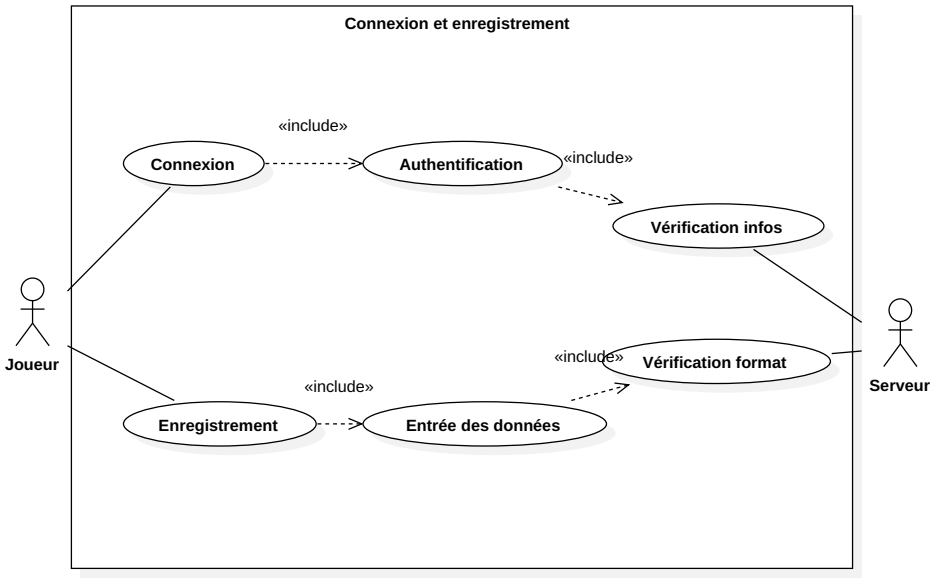
- Acteur(s) : Joueur
- Pré-condition(s) : La pièce doit être en train de chuter.
- Post-condition(s) : La pièce est sauvegardée.
- Condition(s) particulière(s) : Néant.

5.4 Salon de jeu

L'utilisateur voulant jouer en réseau peut inviter un ami dans un salon de jeu privé ou bien faire une recherche de salon. Pour l'un comme l'autre, ceux-ci doivent attendre dans une file d'attente que le serveur fasse une recherche de salon appropriée aux paramètres du joueur. Une fois qu'il a trouvé un salon approprié, les joueurs peuvent commencer à jouer.
Si le serveur n'a pas trouvé de salon approprié, le joueur est redirigé vers le menu principal.

6 Besoins fonctionnels système

6.1 Connexion et enregistrement



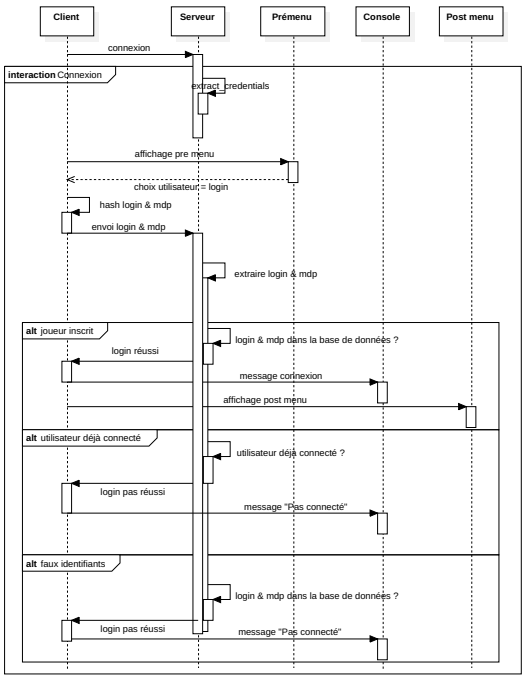
6.1.1 Vérification infos

- Acteur(s) : Serveur
- Pré-condition(s) : Le serveur doit être en ligne.
- Post-condition(s) : Si l'authentification est correcte, le menu principal s'affiche.
- Condition(s) particulière(s) : Si l'authentification est incorrecte, un message d'erreur est affiché et l'utilisateur peut réessayer de se connecter.

6.1.2 Vérification format

Ce diagramme montre l'ensemble d'actions effectuées lorsqu'un joueur se connecte.

- Acteur(s) : Serveur
- Pré-condition(s) : Le serveur doit être en ligne.
- Post-condition(s) : Si la vérification est concluante, le joueur est inscrit dans le fichier répertoriant les login et mots de passes et le menu principal s'affiche.
- Condition(s) particulière(s) : Si la vérification n'est pas concluante, un message d'erreur est affiché et l'utilisateur peut réessayer de s'enregistrer. Voici l'ensemble des actions effectuées lorsque un utilisateur se connecte.



6.2 Gestion des comptes

6.2.1 Création d'un compte

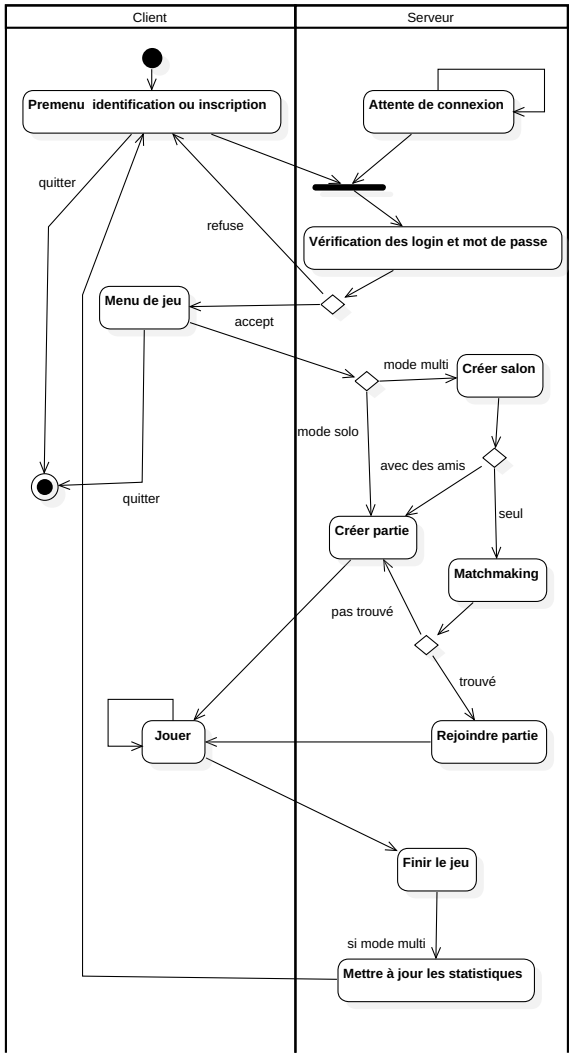
Lorsqu'il lance le jeu pour la première fois, un joueur a la possibilité de se créer un compte. Pour cela, une fenêtre est affichée et il doit rentrer un identifiant et un mot de passe. Si ceux-ci sont valides, le joueur pourra dorénavant se connecter grâce à eux.

6.3 Gestion du classement

La gestion du classement se fait grâce à une base de données dans laquelle on enregistre le nombre de victoires et de défaites de chaque joueur.

6.4 Création d'une partie

Ce diagramme montre l'ensemble d'actions effectuées de la création d'une partie jusqu'à sa fin.

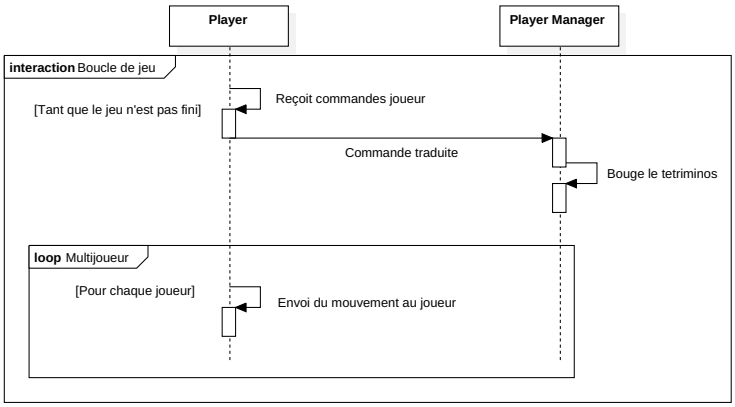


6.5 Session de jeu

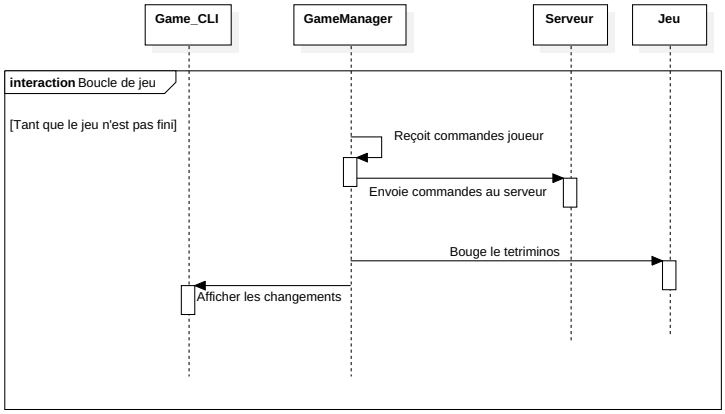
Cette section décrit l'ensemble d'actions possibles que le serveur peut devoir gérer pendant la partie.

6.5.1 Boucle de jeu

Voici l'ensemble d'actions qu'exécute le serveur pendant une partie.



Voici l'ensemble d'actions qu'exécute le client pendant une partie.



6.5.2 Apparition des tetriminos

Les tetriminos sont générés aléatoirement. Le tetriminos courant est généré en même temps que le suivant.

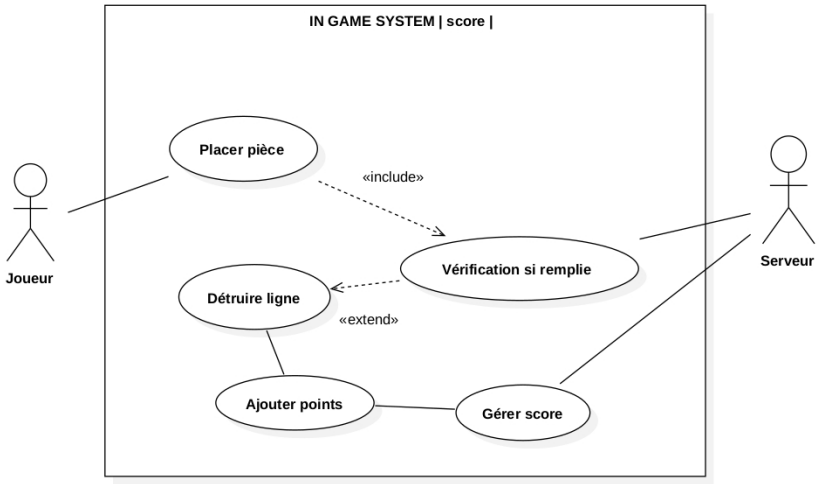
6.5.3 Disparition des tetriminos

Quand une ligne est complète, elle doit être supprimée. Pour cela, on "vide" les blocs de la ligne puis on fait descendre celle qui se trouvait au-dessus.μ

6.5.4 Difficulté du jeu

La difficulté du jeu est la vitesse à laquelle les tetriminos apparaissent et tombent. Elle reste constante dans tous les modes de jeu sauf le mode marathon. Dans celui-ci, elle augmente chaque fois que le joueur complète 10 lignes.

6.5.5 Score du joueur



- **Placer pièce**
  - **Acteur(s) :** Joueur
  - **Pré-condition(s) :** Il reste une place de la taille et de la forme de la pièce dans la grille.
  - **Post-condition(s) :** La pièce est placée dans la grille.
  - **Cas particulier(s) :** Néant.
- **Vérification ligne remplie**
  - **Acteur(s) :** Serveur.
  - **Pré-condition(s) :** Plusieurs pièces doivent avoir été placées dans la grille.
  - **Post-condition(s) :** Néant.
  - **Cas particulier(s) :** Néant.
- **Ajouter points**
  - **Acteur(s) :** Serveur.
  - **Pré-condition(s) :** Une ligne doit être complète.

– **Post-condition(s) :** Le score du joueur a augmenté.

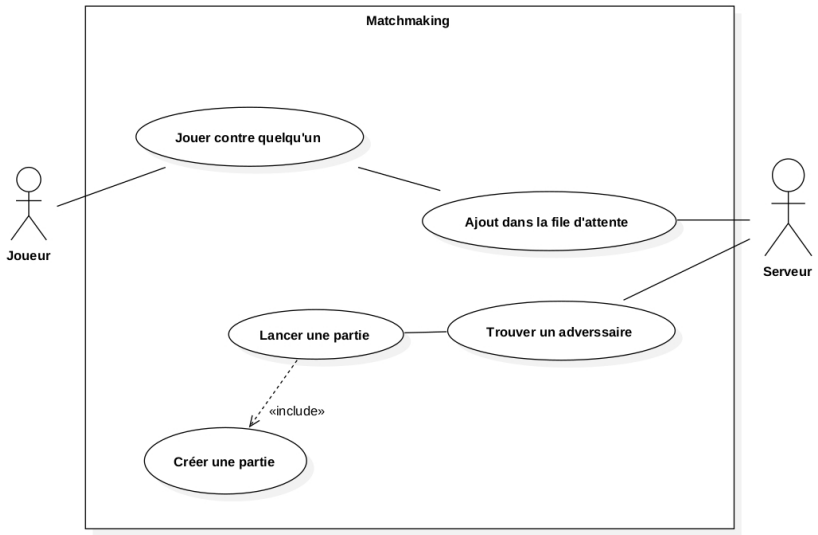
– **Cas particulier(s) :** Néant.

La formule pour calculer les points ajoutés à un joueur est : $score = score + 50 \times 2^{nbre\ lignes\ complètes} \times (level\ du\ joueur + 1)$.

6.6 Salon de jeu

Le joueur a également la possibilité de jouer en mode multijoueurs. Le serveur va donc remplir les mêmes fonctions que dans une partie classique. Quand un joueur veut faire une partie en multijoueur, le serveur cherche un salon avec une place de libre. Dès qu'il en trouve une, il ajoute le joueur dedans et la partie commence. S'il n'y en a pas de libre, le serveur crée un salon et le joueur doit alors attendre qu'un autre joueur le rejoigne.

6.6.1 Création d'une partie multijoueur



- **Ajout dans la file d'attente**

  – **Acteur(s) :** Serveur.

  – **Pré-condition(s) :** Serveur doit être connecté. La liste d'attente ne doit pas être pleine.

  – **Post-condition(s) :** Le joueur est ajouté à la liste d'attente.

- **Condition(s) particulière(s) :** Néant.

- **Trouver un adversaire**
  - **Acteur(s) :** Serveur.
  - **Pré-condition(s) :** On doit avoir deux joueurs de même niveau dans la file d'attente.
  - **Post-condition(s) :** On a bien deux joueurs pour la partie multijoueur.
  - **Condition(s) particulière(s) :** Si on ne trouve pas d'adversaire, on retourne au menu principal.

6.6.2 Choix du vainqueur

Le vainqueur est le joueur ayant remporté le plus de points et donc rempli le plus de lignes de la grille.

7 Besoins non fonctionnels

7.1 Utilisateur

Il a des besoins essentiels qui permettent une expérience utilisateur des plus agréables tout au long de sa période de jeu.

- L'utilisateur est déconnecté après une inactivité trop étendue.
- Le jeu en ligne se veut réactif en évitant le moins de latence possible.
- L'interface se veut épurée rendant l'utilisation très interactive.

7.2 Système

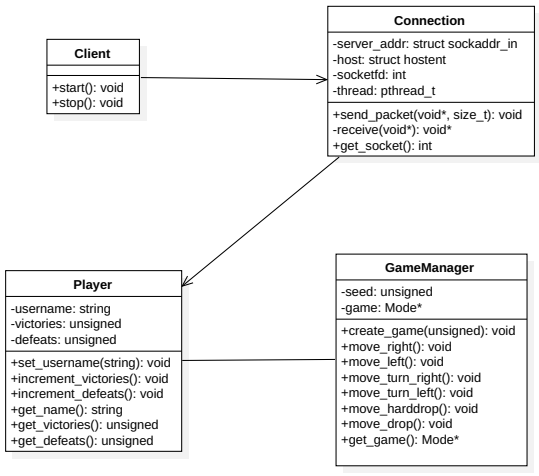
- Le code se veut facilement maintenable.
- Le projet sera codé dans le langage C++, un langage orienté objet qui a été exigé par les assistants.
- Pour le stockage de données, nous avons privilégié le langage mysql.
- La gestion des données sensibles des utilisateurs doivent être pour la plupart, cryptées afin d'assurer une sécurité permanente.
- Le jeu peut s'exécuter sur plusieurs types de machines de façon à garder une expérience minimale acceptable.

8 Design du système

Le système comprend un client et un serveur.

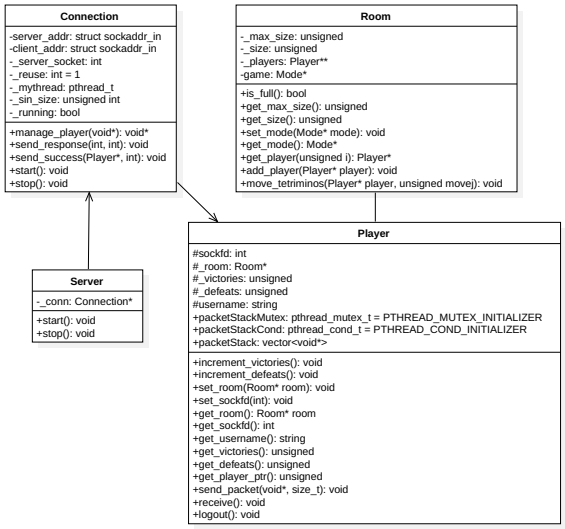
8.1 Client

Voici le diagramme de classes du client.



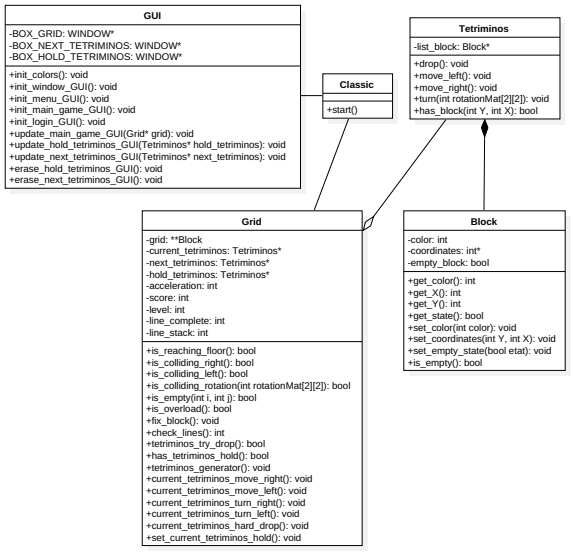
8.2 Serveur

Voici le diagramme de classes du serveur



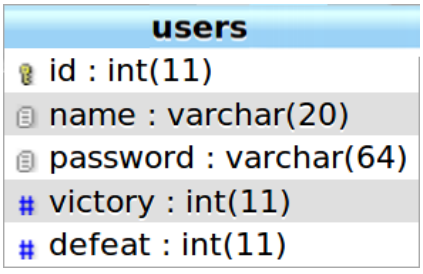
8.3 Jeu

Voici un diagramme de classes représentant le jeu.

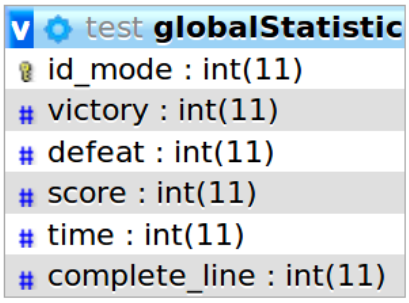


8.4 Base de données

Les données sont stockées dans une base de données SQLite qui contient différentes tables dont la description va suivre. Le choix de SQLite a été fait car il permet d'utiliser le langage SQL. Celui-ci permet de manipuler simplement les données en C++.
Tout d'abord, une table principale users qui stocke tous les noms d'utilisateurs et les mots de passe associés. Les mots de passes sont hachés. Elle contient également les nombres de victoires et de défaites des joueurs.

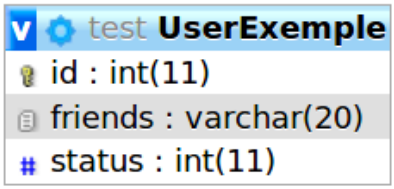


Ensuite, une table globalStatistic qui contient le nombre de victoires et de défaites pour les 4 modes de jeu différents.



id_mode	victory	defeat
1	10	6
2	5	3
3	4	6
4	11	7

Enfin, pour chaque utilisateur, on crée une autre table où seront stockés les amis des utilisateurs ainsi que les demandes d'amis en attente. Un utilisateur peut envoyer une demande d'amis à n'importe quel joueur. Dans la base de données, l'utilisateur à qui on envoie une demande verra dans sa table une nouvelle ligne nommée "friends" où on trouverale nom de la personne qui l'a demandé en ami et la ligne "status" sera égale à 0 (demande d'amis). Une fois la demande d'amis accepetée, le "status" de la personne qui a fait la demande d'amis passe à 1.



8.5 Interface

8.5.1 Interface console

Le jeu peut se jouer en mode console. La librairie utilisée est ncurses. C'est la classe GUI qui gère l'affichage des différents menus et du jeu.