

INFO-F-109 : Projet d'Informatique 2
Software Requirement Document
Pawn Hub

Huwart Maxence, Boonen Jacques
Pham Hong Phuc, Nguyen Duc, Forest Caroline
Antunes Andre, Mardulyn Romain

1 Avril 2019

Table des Matières

1	Introduction	4
1.1	Description du Projet	4
1.2	Fonctionnement du Jeu d'Échec Classique	4
1.2.1	Règles du Jeu	4
1.2.2	Les Pièces	4
1.3	Mécanismes Du Jeu	5
1.3.1	Tour par Tour	5
1.3.2	En Temps Réel	5
1.4	Les Modes de Jeu	5
1.4.1	Classique	5
1.4.2	Alice Chess	5
1.4.3	Dark Chess	5
1.4.4	Horde Chess	5
1.5	Fonctionnalités pour Utilisateur	6
1.6	Fonctionnalités Supplémentaires	6
1.7	Glossaire	6
1.8	Historique des Modifications	8
2	Besoins Utilisateurs : Fonctionnels	9
2.1	Connexion	9
2.1.1	S'identifier	9
2.1.2	Créer un Compte	9
2.2	Menu Principal	10
2.2.1	Jouer	10
2.2.2	Voir les Statistiques Personnelles	10
2.2.3	Voir le Classement	11
2.2.4	Voir sa Liste d'Amis	11
2.2.5	Envoyer une Demande d'Ami	11
2.2.6	Annuler une Demande d'Ami	11
2.2.7	Accepter/Refuser une Demande d'Ami	11
2.2.8	Retirer un Ami de sa Liste d'Amis	12
2.2.9	Discuter dans le Chat	12
2.3	Durant une Partie	12
2.3.1	Avancer une Pièce	12
2.3.2	Déclarer Forfait	12
3	Besoins Utilisateurs : Non-Fonctionnels	12
4	Besoins Systèmes : Fonctionnels	13
4.1	Connexion	13
4.1.1	Connexion au Serveur	13
4.1.2	Enregistrement d'un Nouveau Compte	13
4.2	Menu Principal	13
4.2.1	Création d'une Partie	13
4.2.2	Voir les Statistiques Personnelles	13
4.2.3	Voir le Classement	14
4.2.4	Voir sa Liste d'Amis	14
4.2.5	Envoyer une Demande d'Ami	14

4.2.6	Annuler une Demande d'Ami	14
4.2.7	Accepter/Refuser une Demande d'Ami	14
4.2.8	Retirer un Ami de sa Liste d'Amis	15
4.2.9	Envoi de Messages	15
4.3	Gestion d'une Partie	15
4.3.1	Premier à Jouer	15
4.3.2	Déplacer une Pièce	15
4.3.3	Promotion	15
4.3.4	Roque	16
4.3.5	Déroulement de la Partie	16
4.3.6	Fin de la Partie	16
5	Besoins Systèmes : Non-Fonctionnels	16
5.1	Système d'Exploitation	16
5.2	Réseau	16
5.3	Mise à Jour des Différentes Fonctionnalités	16
6	Design du Système	17
6.1	Matchmaking	17
6.2	Diagrammes de Classes	18
6.2.1	Les Différentes Pièces	18
6.2.2	Coordinate	20
6.2.3	Les Différents Modes De Jeu	20
6.2.4	Player	22
6.2.5	Board	22
6.2.6	RealTimeBoard	22
6.2.7	ChainedMove	22
6.3	Connexion du Joueur	23
6.4	Client-Serveur	23
6.5	Déroulement d'une partie	24
6.6	Menu	29
6.6.1	Affichage du menu	29
6.6.2	Choix d'une option	29
6.6.3	Choix d'option du menu Jouer	30
6.6.4	Choix d'option du menu Ami	30
6.6.5	Voir sa liste d'amis	30
6.6.6	Envoyer une demande d'ami	30
6.6.7	Retirer un ami	30
6.6.8	Accepter ou refuser une demande d'ami	31
6.6.9	Choix d'option du menu Statistiques	31
6.6.10	Choix d'option du menu Classement général	31
6.6.11	Utilisation du chat	31
7	Annexe	31

1 Introduction

1.1 Description du Projet

Ce projet aura pour but de recréer un grand classique des jeux de plateau: les échecs. Le jeu sera jouable en réseau via un mode multijoueurs. Le joueur aura la possibilité de jouer en mode tour par tour ou en temps réel, c'est à dire que les joueurs ne devront pas attendre leur tour pour jouer. Le joueur le plus rapide verra son coup s'effectuer en premier. Il devra ensuite choisir entre le mode classique ou à une de ses variantes telles que *AliceChess*, *DarkChess* et *HordeChess*, décrites plus loin. De plus, il aura également la possibilité de créer un compte à partir duquel il pourra effectuer plusieurs actions qui seront détaillées ci-dessous.

1.2 Fonctionnement du Jeu d'Échec Classique

1.2.1 Règles du Jeu

Chaque joueur commence le jeu avec 16 pions: un roi, une dame, deux tours, deux fous, deux cavaliers et huit pions. Le but du jeu est d'infliger à son adversaire un *échec et mat*, une situation dans laquelle le roi d'un joueur est en prise sans qu'il soit possible d'y remédier¹.

1.2.2 Les Pièces

Le Pion (Pawn) : Depuis sa position initiale, le pion peut avancer d'une ou deux cases en avant, ensuite d'une case en avant uniquement. Cependant, le pion prend une autre pièce en diagonale. Également, lorsque le pion arrive à la dernière rangée, il effectue une promotion et est remplacé par n'importe quelle autre pièce, à part le roi, au choix du joueur . De plus, il peut également effectuer la prise en passant qui est expliquée dans le glossaire.

La Tour (Rook) : La tour peut se déplacer d'un nombre quelconque de cases sur les rangées et les colonnes.

Le Fou (Bishop) : Le fou peut se déplacer uniquement en diagonale d'un nombre quelconque de cases.

La Reine (Queen) : La reine peut se déplacer dans toutes les directions d'un nombre quelconque de cases.

Le Roi (King) : Le roi peut se déplacer dans toutes les directions d'une case uniquement. Il peut également effectuer un roque qui est expliqué dans le glossaire.

Le Cavalier (Knight) : Le cavalier se déplace en "L", voir image ci-dessous.

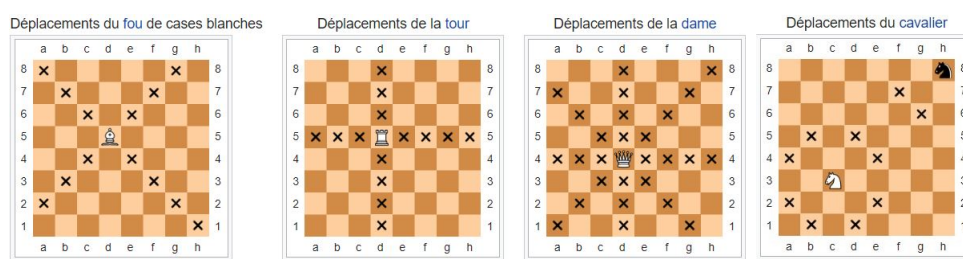


Figure 1: Image provenant de Wikipedia

¹Wikipedia

1.3 Mécanismes Du Jeu

1.3.1 Tour par Tour

Comme son nom l'indique, avec cette option, le déroulement du jeu est divisé en parties bien définies et visibles, appelées tours. Les joueurs jouent donc l'un après l'autre. Chaque joueur dispose d'une limite de temps de 45 minutes pour jouer une partie. Le chronomètre se décrémente au tour du joueur et se met en pause lorsque c'est au tour de l'adversaire

1.3.2 En Temps Réel

Avec cette option, les joueurs peuvent jouer sans la contrainte de tour. Chaque joueur peut donc déplacer ses pièces quand bon lui semble à condition que la pièce déplacée ait terminé son dernier déplacement depuis plus de 10 secondes et que le mouvement soit légal du point de vue du mode de jeu jouer en temps réel sans prendre en compte le fait que le roi soit en échec. Le but du jeu est de prendre le roi adverse. La prise en passant² est possible tant que le pion n'a pas dépasser les 10 secondes d'immobilité après avoir bougé.

Le déplacement des pièces n'est plus instantané. Les pièces se déplacent avec une vitesse d'une case par seconde et commence et s'arrête de se déplacer au milieu d'une case. Lorsque 2 pièce de couleurs différente se trouve sur la même case il y a *collision*. Si l'une des pièces ne bouge pas, elle se fait prendre, sinon c'est la dernière pièce à avoir bougé qui se fait prendre.³

1.4 Les Modes de Jeu

1.4.1 Classique

Dans une partie de jeu d'échecs classique, le but est de mettre le roi adverse en échec sans qu'il ait la possibilité au tour suivant de ne plus l'être.

1.4.2 Alice Chess

Le jeu d'échecs d'Alice est une variante du jeu d'échecs jouée en utilisant 2 plateaux. Après chaque coup joué, la pièce déplacée est téléportée vers l'autre plateau. Un coup n'est valide que si la case où doit se téléporter la pièce ne comporte aucune pièce de sa couleur. Mis à part ce changement d'affichage, le jeu se déroule selon les règles habituelles du jeu d'échecs.

1.4.3 Dark Chess

Le jeu d'échecs Dark est une variante du jeu d'échecs où le joueur ne voit que ses propres pièces et les cases où il peut légalement les déplacer. Le but des joueurs est de prendre le roi adverse. Contrairement aux règles classiques, le roi d'un joueur peut être en échec à la fin de son tour. De plus, il n'y a pas de prises en passant possibles. À part cela, le jeu se déroule selon les règles habituelles du jeu d'échecs.

1.4.4 Horde Chess

Le jeu d'échecs de horde est une variante du jeu d'échecs où le joueur noir possède 32 pions et le joueur blanc la collection standard du jeu d'échecs classique. Pour gagner la partie, le joueur blanc doit prendre tous les pions du joueur noir tandis que celui-ci doit mettre le roi blanc en échec et mat. À part cela, le jeu se déroule selon les règles habituelles du jeu d'échecs.

²voir description des différent modes de jeux plus bas

³Pour une description plus complète des règles voir <http://kfc.kunuk.dk/rules.htm>

1.5 Fonctionnalités pour Utilisateur

Depuis le menu principal, les joueurs pourront accéder à l'option *View Rules* à travers lequel ils pourront lire en détail les règles du jeu d'échecs et de ses variantes. Les joueurs lanceront une partie via l'option *Play*.

Les joueurs pourront communiquer via un chat global. À travers le menu principal, les utilisateurs accéderont à plusieurs sous-menus tels que *View Rules*, *Statistics*, *Friends List*.

1.6 Fonctionnalités Supplémentaires

Le joueur aura la possibilité de jouer en mode console ou via une interface graphique. S'il choisit l'interface graphique, des publicités défileront pendant ses parties.

1.7 Glossaire

Par souci de clarté et de lisibilité, toutes les informations relatives aux règles et au fonctionnement officiels des échecs et de ses variantes ne seront pas parcourues ci-dessous.

Un utilisateur : Utilisateur s'étant identifié au système via un compte.

Joueur : Utilisateur en partie.

Matchmaking : Système qui met en relation deux joueurs avant le lancement d'une partie.

Chat : Système de communication par messages instantanés entre deux joueurs, ou plus.

Echiquier (Board) : Plateau sur lequel se joue le jeu d'échec.

Coup/Mouvement : Déplacement d'une pièce d'une case à une autre de l'échiquier.

Coup Légal : est un coup qu'un joueur a le droit de jouer en vertu des règles du jeu (par opposition à un coup illégal⁴)

Prise : Déplacement d'une pièce sur une case où se trouve une pièce adverse, ce qui implique un retrait du jeu de la pièce adverse.

Collision : Pour les modes de jeux en temps réel, c'est le cas où plusieurs pièces de couleurs différentes se trouvent sur une même case.

Echec : Un joueur est en échec lorsque son roi est dans une position telle qu'il peut être pris par son adversaire.

Echec et mat : Un joueur est échec et mat il est en situation d'échec et qu'il n'a aucun coup légal lui permettant de se sortir de cette situation.

Pat : Il y a pat lorsque au tour du joueur, son roi n'est pas en échec mais ne peut effectuer de coups légaux sans se mettre en échec. Le pat met immédiatement fin à la partie et la partie est considérée comme une égalité.

Le Roque : Le roque consiste à déplacer en un seul coup le roi et l'une des tours. On déplace d'abord le roi de deux cases vers la tour puis, on fait passer la tour de l'autre côté, juste à côté du roi (voir le diagramme ci-dessous). Les conditions suivantes sont nécessaires pour pouvoir roquer : aucune pièce ne se trouve entre le roi et la tour concernée ; le roi et la tour concernée n'ont encore jamais joué et le roi n'est pas en échec ; la case traversée par le roi n'est contrôlée par aucune pièce adverse⁵.

⁴source Wikipedia

⁵source Wikipedia

La Prise en Passant: Au jeu d'échecs, la prise en passant est une possibilité particulière de capturer un pion. Lorsqu'un pion avance de deux cases et que le joueur adverse possède un pion qui aurait pu prendre ce pion si ce dernier n'avait avancé que d'une case, alors l'adversaire peut, au tour suivant uniquement, prendre le pion venant d'avancer de 2 cases comme si ce dernier n'avait avancé que d'une seule case⁶.

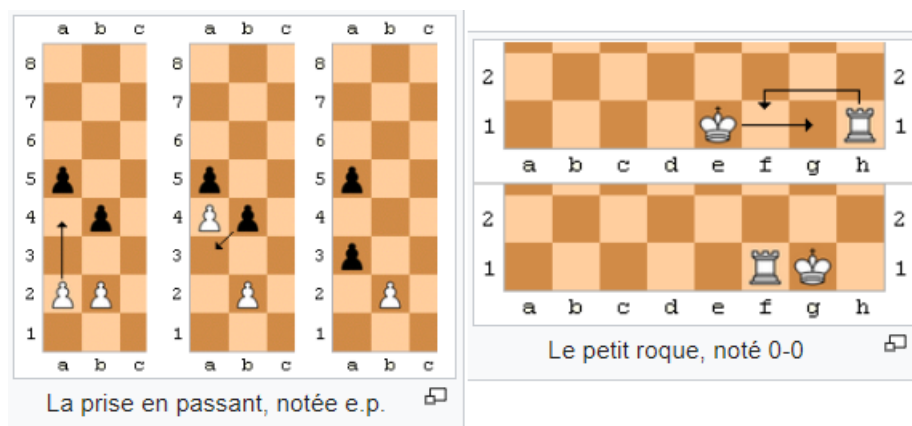


Figure 2: Image provenant de Wikipedia

⁶source Wikipedia

1.8 Historique des Modifications

version	date	auteur	description
1	4/12	Pham Hong Phuc	Création du SRD + ajout structure
1.1	6/12	Forest Caroline	Ajout de diagrammes UML
1.2	8/12	Boonen Jacques	1.Introduction
1.3	11/12	Huwart Maxence	Besoins systèmes
1.4	12/12	Mardulyn Romain	Explication des différent modes de jeu
1.5	12/12	Antunes André	Explication des options durant une partie
1.6	12/12	Nguyen Duc	Modification des diagrammes de séquence
1.7	13/12	Pham Hong Phuc	Correction et réajustement
2	12/02	Boonen Jacques	Fonctionnement du jeu + ajout images
2.1	13/02	Boonen Jacques	Déroulement d'une partie
2.2	28/02	Forest Caroline	Mise à jour du déroulement d'une partie
2.2.1	02/03	Forest Caroline	Diagrammes de séquences pour le déroulement d'une partie
2.3	3/03	Nguyen Duc	Protocole de communication du menu
2.4	3/03	Antunes André	Rajout des méthodes pour Alice, Dark, Horde, Classique
2.5	3/03	Boonen Jacques	Nouveaux Diagrammes de classes
2.6	28/03	Boonen Jacques	Realtime + diagrammes système + relecture finale
2.7	28/03	Antunes André et Forest Caroline	Relecture

Historique des modifications

2 Besoins Utilisateurs : Fonctionnels

2.1 Connexion

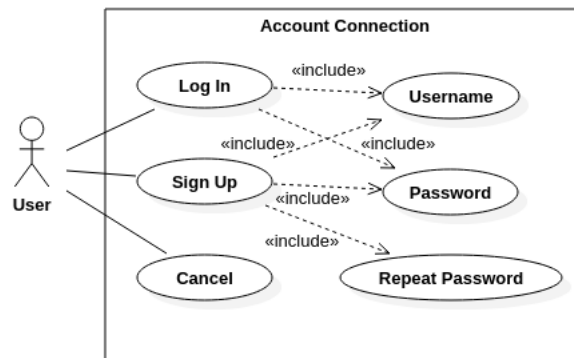


Figure 3: Diagramme de *use case* des actions possibles d'un utilisateur quant à son identification

2.1.1 S'identifier

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant.

Pré-conditions : Le *user* doit avoir sélectionné le sous-menu *Connexion* puis l'option *Log In* dans le menu principal du jeu.

Post-conditions : Le *user* doit encoder un nom d'utilisateur et un mot de passe d'un compte déjà existant dans le système.

Cas général : Après s'être identifié, le *user* peut profiter de tous les sous-menus du menu principal.

Cas exceptionnels : Si un utilisateur est déjà connecté sur un compte, un autre joueur ne peut se connecter sur ce compte. Également, si l'utilisateur encode mal ses identifiants, le système lui renvoie un message d'erreur afin qu'il les réécrive correctement.

2.1.2 Créer un Compte

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant.

Pré-conditions : Le *user* doit avoir sélectionné le sous-menu *Connexion* puis l'option *Create* dans le menu principal du jeu.

Post-conditions : Le *user* doit encoder un nom d'utilisateur et un mot de passe afin de créer son compte.

Cas général : Un nom d'utilisateur et un mot de passe correspondant sont encodés dans la base de données du système.

Cas exceptionnels : Si le nom d'utilisateur encodé est déjà dans la base de données, l'utilisateur doit en ré-encoder un nouveau.

2.2 Menu Principal

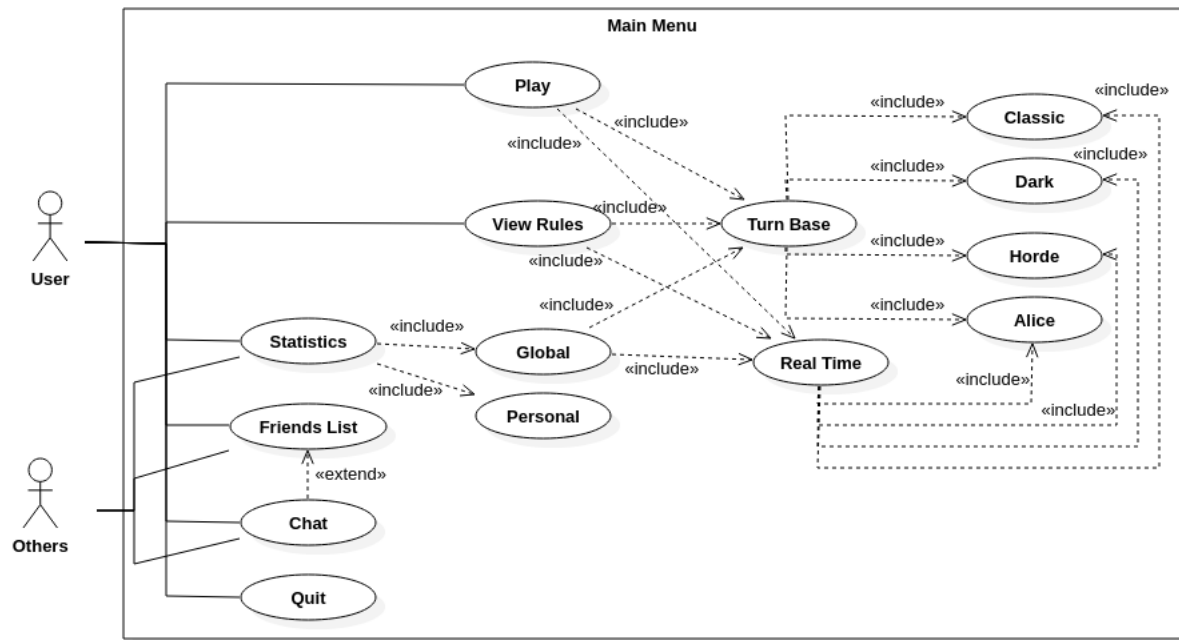


Figure 4: Diagramme de *use case* des actions possibles d'un utilisateur

2.2.1 Jouer

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Modes de jeu proposés (*TurnBasedGame*, *RealTimeGame*, *Classic*, *Alice*, *Dark* ou *Horde*).

Pré-conditions : Le *user* doit avoir sélectionné l'option *Game* dans le menu principal du jeu.

Post-conditions : Le *user* sélectionne un mode de jeu parmi ceux proposés (*TurnBasedGame*, *RealTimeGame*, *Classic*, *Alice*, *Dark* ou *Horde*).

Cas général : Le *user* peut sélectionner un mode de jeu parmi ceux proposés (*TurnBasedGame*, *RealTimeGame*, *Classic*, *Alice*, *Dark* ou *Horde*), ce qui lance une requête de *matchmaking* (voir Figure 5) au serveur.

Cas exceptionnels : Néant.

2.2.2 Voir les Statistiques Personnelles

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant.

Pré-conditions : Le *user* doit avoir sélectionné l'option *View Statistics* dans le menu principal du jeu.

Post-conditions : Le *user* accède à ses statistiques personnelles.

Cas général : Le *user* peut consulter ses statistiques en envoyant une requête au serveur. Il pourra consulter son nombre de victoires, défaites, matchs nuls et son elo.

Cas exceptionnels : Néant.

2.2.3 Voir le Classement

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant.

Pré-conditions : Le *user* doit avoir sélectionné l'option *View Ranking* dans le menu principal du jeu.

Post-conditions : Le *user* accède au classement global des joueurs (*user* et *others*).

Cas général : Le *user* peut consulter le classement global des joueurs et leur elo associé. (*user* et *others*) en envoyant une requête au serveur.

Cas exceptionnels : Néant.

2.2.4 Voir sa Liste d'Amis

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Chat*.

Pré-conditions : Le *user* doit avoir sélectionné l'option *Friends List* dans le menu principal du jeu.

Post-conditions : Le *user* accède à sa liste d'utilisateurs "amis", avec lesquels il peut choisir de discuter via l'option *chat*.

Cas général : Le *user* peut consulter sa liste d'utilisateurs "amis" en envoyant une requête au serveur. Il peut aussi discuter avec eux via messages instantanés.

Cas exceptionnels : Néant.

2.2.5 Envoyer une Demande d'Ami

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Friends List*

Pré-conditions : Le *user* doit avoir sélectionné cette option dans le menu principal du jeu, ensuite il doit encoder le nom du joueur avec lequel il souhaite être ami.

Post-conditions : Une demande d'ami a été envoyée.

Cas général : Une demande d'ami a été envoyée.

Cas exceptionnels : Il n'est pas possible d'envoyer une demande d'ami à "Guest" ou "all". Le *user* est déjà ami avec la personne à qui il veut envoyer une demande. Une demande a déjà été envoyée à la personne à qui il veut envoyer une demande. Le *user* ne peut pas s'envoyer une demande à lui-même. Le nom de l'ami ne peut excéder 10 caractères, ni être composé de '|' et ' '.

2.2.6 Annuler une Demande d'Ami

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Friends List*

Pré-conditions : Le *user* doit avoir sélectionné cette option dans le menu principal du jeu.

Post-conditions : Annule la demandé d'ami.

Cas général : La demande d'ami a été annulée et le joueur concerné ne peut plus l'accepter ou la refuser.

Cas exceptionnels : Aucune demande n'a été envoyée à la personne à qui on veut annuler la demande.

2.2.7 Accepter/Refuser une Demande d'Ami

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Friends List* et *Envoyer une demande d'ami*.

Pré-conditions : Le *user* doit avoir sélectionné cette option dans le menu principal du jeu, ensuite il doit accepter ou refuser la demande d'ami qui lui a été envoyée.

Post-conditions : Si le *User* accepte la demande d'ami, le nouvel ami est rajouté à sa liste d'amis. Sinon, la demande cesse d'apparaître et ils ne sont pas amis.

Cas général :

Cas exceptionnels : Aucune demande d'ami n'a été envoyée.

2.2.8 Retirer un Ami de sa Liste d'Amis

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Friends List*

Pré-conditions : Le *user* doit avoir sélectionné cette option dans le menu principal du jeu et être ami avec la personne qu'il souhaite supprimer de sa liste d'amis.

Post-conditions : L'ami est retiré de la liste d'amis.

Cas général :

Cas exceptionnels : La personne que le *User* veut retirer de sa liste d'amis, n'est pas ami avec lui.

2.2.9 Discuter dans le Chat

Acteur : *User*.

Relations avec d'autres cas d'utilisation : *Friends List*.

Pré-conditions : Si le joueur veut parler avec une personne en particulier, il doit l'avoir en ami, sinon aucune.

Post-conditions : Néant.

Cas général : Le *user* ouvre avec un bouton (à définir) le chat et peut écrire sur un textfield pour communiquer avec les autres personnes.

Cas exceptionnels : Néant.

2.3 Durant une Partie

2.3.1 Avancer une Pièce

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant

Pré-conditions : C'est au tour du *user* de jouer. Il doit préalablement choisir une pièce et indiquer un déplacement possible et valide.

Post-conditions : La pièce se retrouve à la position désignée par le *user* après déplacement.

Cas général : Le *user* doit déplacer un pion lors de son tour de jeu. Il envoie ainsi une requête de déplacement au serveur qui vérifiera à son tour la validité ou non du coup.

Cas exceptionnels : Le coup enregistré par le *user* n'est pas valide. Il doit donc rejouer son tour.

2.3.2 Déclarer Forfait

Acteur : *User*.

Relations avec d'autres cas d'utilisation : Néant.

Pré-conditions : Il faut que le *user* soit dans un jeu.

Post-conditions : Informer l'adversaire sur la déclaration de forfait et terminer le jeu.

Cas général : Lorsque le joueur veut terminer la partie.

Cas exceptionnels : Lorsque le joueur perd la connexion le joueur déclare automatiquement forfait.

3 Besoins Utilisateurs : Non-Fonctionnels

Afin d'avoir des parties de jeux dynamiques, les joueurs se verront imposer une limite de temps de 45 minutes pour pouvoir jouer une partie.

Afin d'avoir des parties équilibrées, les joueurs se verront attribués des adversaires de même niveau via le matchmaking. Chaque joueur dispose d'un elo, celui-ci est calculé d'après la formule de Arpad Elo. L'elo est donc un nombre.⁷

4 Besoins Systèmes : Fonctionnels

4.1 Connexion

4.1.1 Connexion au Serveur

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : *Log In*.

Pré-conditions : *Server* en ligne et réception du signal correspondant.

Post-conditions : Utilisateur connecté, redirection vers le menu principal, sous-menus réservés aux utilisateurs débloqués, remplacement du sous-menu *Log In/Create an account* par *Log Out*.

Cas général : Utilisateur connecté.

Cas exceptionnels : Si un utilisateur est déjà connecté sur un compte, un autre joueur ne peut se connecter sur ce compte. De même, si l'utilisateur encode mal ses identifiants, le système lui renvoie un message d'erreur afin qu'il les réécrive correctement.

4.1.2 Enregistrement d'un Nouveau Compte

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : Créer un compte.

Pré-conditions : *Server* en ligne et réception du signal correspondant.

Post-conditions : Compte enregistré, message de succès de création de compte envoyé à l'utilisateur.

Cas général : Compte enregistré.

Cas exceptionnels : Si le compte est déjà existant, un message d'erreur est envoyé à l'utilisateur.

4.2 Menu Principal

4.2.1 Création d'une Partie

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : Choisir un mode de jeu.

Pré-conditions : *Server* en ligne et réception du signal correspondant.

Post-conditions : Vérifie dans la file d'attente si un joueur dans la même fourchette de classement et désirant jouer au même mode est disponible. Si oui, le *Server* crée la partie. Sinon, l'utilisateur est mis en file d'attente.

Cas général : Utilisateur mis en file d'attente si aucun adversaire n'est trouvé, sinon une partie est créée.

Cas exceptionnels : Néant.

4.2.2 Voir les Statistiques Personnelles

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : Néant.

⁷Arpad Elo est un scientifique, mathématicien et joueur d'échecs américain d'origine hongroise. Il est surtout connu pour son système d'estimation de la force des joueurs d'échecs, le classement Elo. Wikipedia

Pré-conditions : *Server* en ligne et réception du signal correspondant.
Post-conditions : Envoi des statistiques personnelles à l'utilisateur correspondant.
Cas général : Envoi des statistiques personnelles de l'utilisateur.
Cas exceptionnels : Néant.

4.2.3 Voir le Classement

Acteur : *Server*.
Relations avec d'autres cas d'utilisation : Néant.
Pré-conditions : *Server* en ligne et réception du signal correspondant.
Post-conditions : Envoi du classement.
Cas général : Envoi du classement.
Cas exceptionnels : Néant.

4.2.4 Voir sa Liste d'Amis

Acteur : *Server*.
Relations avec d'autres cas d'utilisation : *Envoi de messages*.
Pré-conditions : *Server* en ligne et réception du signal correspondant.
Post-conditions : Envoi de la liste d'amis de l'utilisateur en question.
Cas général : L'utilisateur reçoit sa liste d'amis et peut interagir avec elle via des requêtes serveur (ajout d'un ami, suppression, etc).
Cas exceptionnels : Néant.

4.2.5 Envoyer une Demande d'Ami

Acteur : *Server*.
Relations avec d'autres cas d'utilisation : Néant.
Pré-conditions : *Server* en ligne et réception du signal correspondant.
Post-conditions :
Cas général :
Cas exceptionnels : Si le nom encodé n'existe pas, le *User* retourne au menu principal. Si 2 *User* se sont mutuellement envoyés une demande d'ami, ils deviennent directement ami sans devoir accepter l'invitation de l'autre.

4.2.6 Annuler une Demande d'Ami

Acteur :
Relations avec d'autres cas d'utilisation :
Pré-conditions : *Server* en ligne et réception du signal correspondant.
Post-conditions :
Cas général :
Cas exceptionnels :

4.2.7 Accepter/Refuser une Demande d'Ami

Acteur :
Relations avec d'autres cas d'utilisation :

Pré-conditions : *Server* en ligne et réception du signal correspondant.

Post-conditions :

Cas général :

Cas exceptionnels :

4.2.8 Retirer un Ami de sa Liste d'Amis

Acteur :

Relations avec d'autres cas d'utilisation : *Friends List*

Pré-conditions : *Server* en ligne et réception du signal correspondant. L'ami en question doit être présent dans la liste d'ami concernée.

Post-conditions : L'ami en question est effacé de la liste d'ami concernée.

Cas général :

Cas exceptionnels :

4.2.9 Envoi de Messages

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : *Friends List*, jouer.

Pré-conditions : *Server* en ligne et réception du message à envoyer.

Post-conditions : Envoi du message.

Cas général : Les utilisateurs peuvent s'envoyer des messages privés en dehors d'une partie s'ils sont amis, sinon ils peuvent communiquer sur un chat commun avec tous les autres *User*.

Cas exceptionnels : Néant.

4.3 Gestion d'une Partie

4.3.1 Premier à Jouer

Le premier joueur connecté à la partie se voit attribuer la couleur blanche et commence par conséquent en premier.

4.3.2 Déplacer une Pièce

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : *Avancer une pièce*.

Pré-conditions : Réception du signal correspondant, coup valide (pas de pièces qui bloquent, etc).

Post-conditions : Pièce déplacée.

Cas général : Pièce déplacée.

Cas exceptionnels : Si le coup n'est pas valide, on redemande au joueur de faire déplacer une de ses pièces.

4.3.3 Promotion

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : *Avancer une pièce*.

Pré-conditions : Un pion est arrivé à l'autre bout du plateau.

Post-conditions : Pion promu en dame, fou, cavalier ou tour au choix du joueur qui possède le pion en question. Le jeu est ré-analysé pour identifier une potentielle mise en échec due au changement de statut du

pion.

Cas général : Pion promu.

Cas exceptionnels : Néant.

4.3.4 Roque

Acteur : *Server*.

Relations avec d'autres cas d'utilisation : *Avancer une pièce*.

Pré-conditions : Un joueur déplace le roi sur une case provoquant le roque, le roi et la tour en question n'ont pas bougé depuis le début de la partie. Aucune pièce ne bloque.

Post-conditions : Tour déplacée, position du roi mise à jour, le roi et la tour marqués comme "ayant déjà bougé".

Cas général : Roque effectué.

Cas exceptionnels : Si l'une des pré-conditions n'est pas respectée, on redemande au joueur de faire déplacer une de ses pièces.

4.3.5 Déroulement de la Partie

Voici le déroulement de la partie représentée à l'aide d'un diagramme de séquence :

4.3.6 Fin de la Partie

À la fin de chaque tour de jeu, la logique du jeu se trouvant côté serveur vérifie s'il y a un gagnant ou si un Pat s'est produit. Si oui, le résultat est envoyé à chacun des joueurs et la partie se termine.

5 Besoins Systèmes : Non-Fonctionnels

5.1 Système d'Exploitation

Le programme doit être capable de tourner sous Unix.

5.2 Réseau

Le jeu se joue en réseau local, une connexion internet est donc requise. Le serveur affiche son nom d'hôte afin que les clients puissent savoir où se connecter.

5.3 Mise à Jour des Différentes Fonctionnalités

Le système se doit de mettre assez rapidement à jour les statistiques personnelles et le classement global après chaque partie.

6 Design du Système

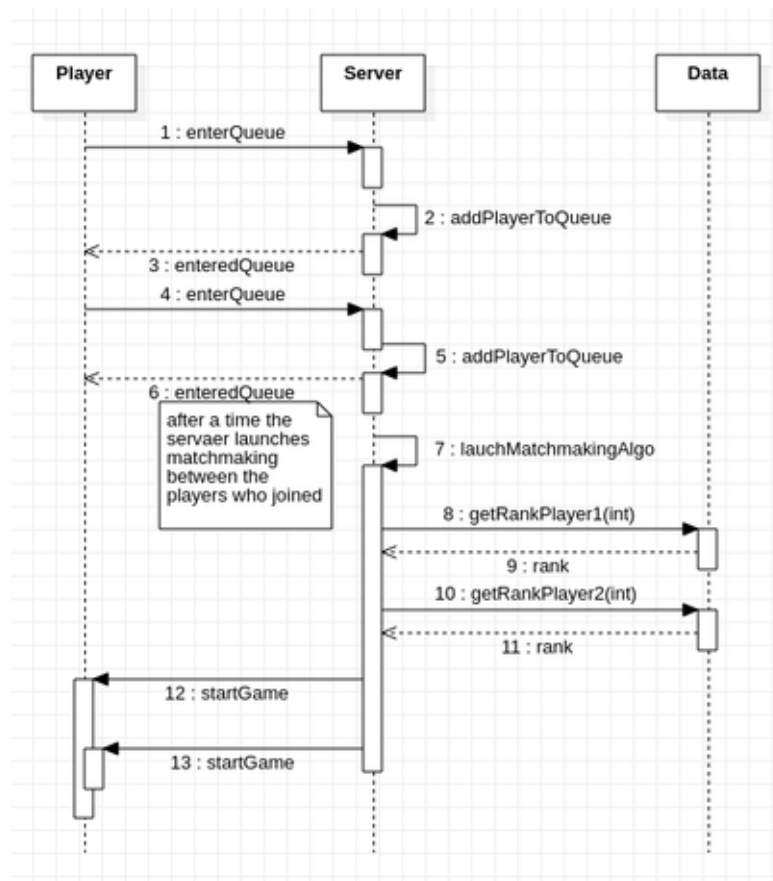


Figure 5: Diagramme de séquence pour le *Matchmaking*

6.1 Matchmaking

Le système de Matchmaking range les différents joueurs selon leur *Elo* dans différentes files d'attente. Par exemple, la file d'attente 1 contiendrait tous les joueurs entre 1000 et 1200 d'elo. Si il n'y a pas assez de joueurs dans une file d'attente, on regarde dans la file d'attente supérieur, par exemple, la file d'attente 2 qui contiendrait les joueurs entre 1200 et 1400 d'elo. S'il y a un nombre impair de joueurs dans les différentes files d'attente, le dernier joueur sera enlevé du système de matchmaking actuel et sera mis en première place dans le prochain matchmaking. En effet, l'ordre de priorité dans le matchmaking est fait selon l'ordre d'arrivée des joueurs.

6.2 Diagrammes de Classes

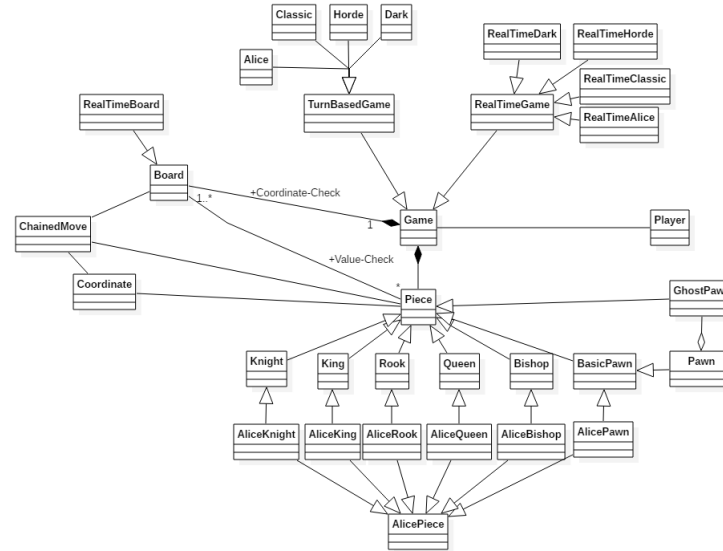


Figure 6: Diagramme de classes général

6.2.1 Les Différentes Pièces

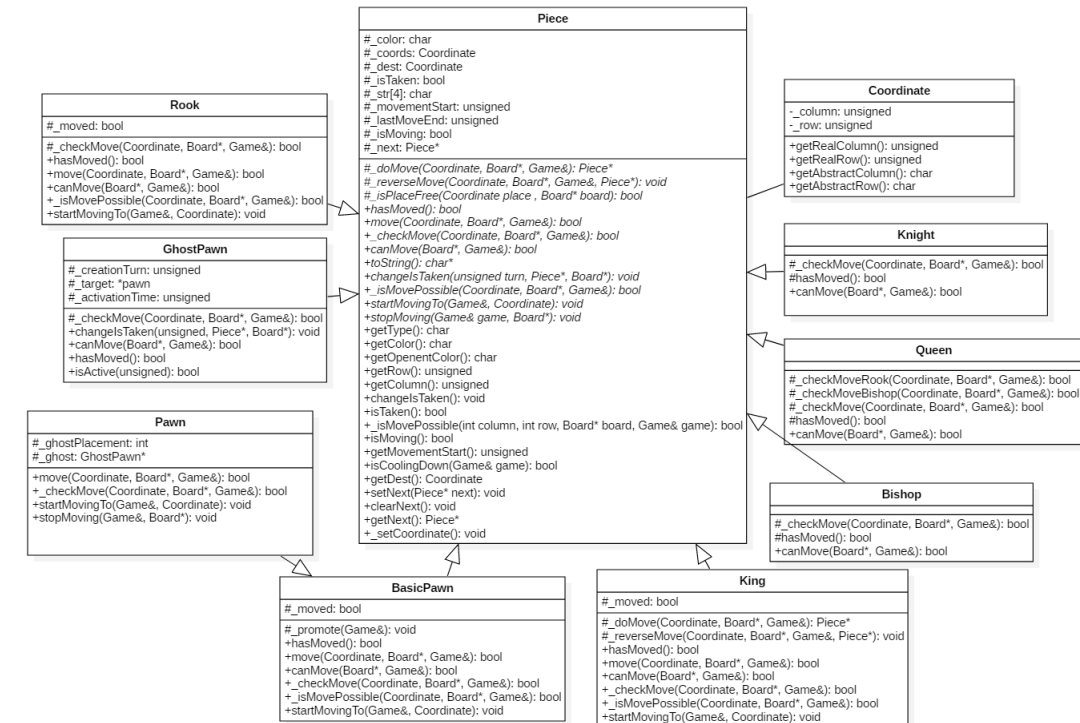


Figure 7: Diagramme des pièces et coordinate

Mode classique Chaque type de pièces, c'est à dire, *Queen, King, Bishop, Pawn, GhostPawn, Knight, Rook*, héritent de la classe mère *Piece*. *GhostPawn* est utilisé pour la prise en passant. Lorsqu'un *Pawn* avance de deux case, il laisse sur la case où il serait s'il n'avait avancer que d'une case, un *GhostPawn* qui est invisible pour le joueur. Un *Pawn* considère un *GhostPawn* comme une pièce prenable si cette dernière est active(elle est active uniquement le tour juste après sa création). Lorsqu'un *GhostPawn* est pris par un *Pawn* et qu'il est actif. Ensuite, le *GhostPawn* se charge de faire disparaître le *Pawn* qui l'a créé du plateau. *BasicPawn* est un *Pawn* qui ne peut effectuer de prise en passant. En effet, il ne possède pas d'attribut *ghost*. Cette pièce est indispensable dans le mode *Alice* car après chaque *Move*, les pièces changent de *Board* et ne peuvent donc pas effectuer de prise en passant.

1. **canMove** : Retourne true si la pièce peut se déplacer.
checkMove : Retourne true si le mouvement demandé d'une pièce par le joueur est un coup légal.
hasMoved : Retourne true si la pièce a déjà été déplacée ou non.
isMovePossible : vérifie si le coup d'un joueur ne met pas son propre *King* en échec. Si non, le coup n'est pas légal.
Move : est une méthode qui selon *checkMove* effectue ou non, le mouvement demandé par le joueur sur la pièce. Cette méthode effectue d'abord un *checkMove* à l'endroit du coup désiré par le joueur. Si, le coup est possible, elle effectue un *doMove* ce qui permet d'avancer la pièce à l'endroit du coup. Dans le cas où le coup met le *King* du joueur en échec, on effectue un *ReverseMove* qui remet le *Board* dans son état initial. Si le *Board* à changé, la fonction retourne true.

Mode Alice Tout comme dans le mode classique, les différentes pièces héritent de la classe mère *AlicePiece*. Néanmoins dans le mode Alice, il existe 2 échiquiers, c'est pourquoi, les pièces ont un attribut booléen *2Dimension* qui signale au système dans quel échiquier se trouve la pièce concernée.

1. **swapdimension**: Change l'attribut *2Dimension* par son contraire.
isPlaceFree : Retourne true si la case sur laquelle une pièce devrait arriver est libre ou non.
getdimension : Retourne la variable *2Dimension*.

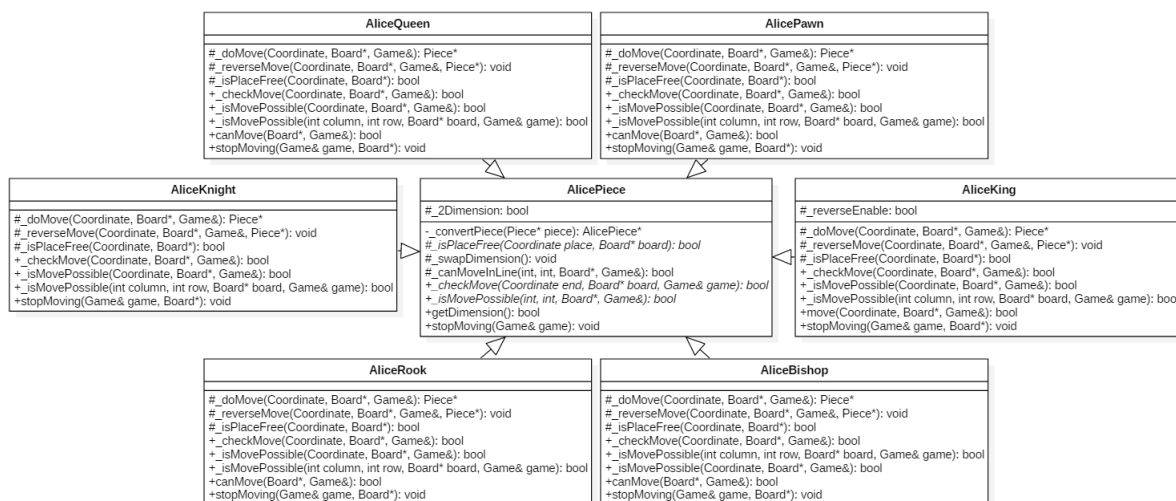


Figure 8: Diagramme des pièces du mode de jeu Alice

6.2.2 Coordinate

Par soucis de facilité et d'homogénéité dans le code, *Coordinate* est une classe qui retraduit chaque adresse réelle sur l'échiquier en une adresse virtuelle compréhensible pour les board, et inversement . En effet, le board coté Serveur est une matrice composée de lignes et rangées numérotées de 0 à 7 alors que l'échiquier coté Client est numérotées de A à H et 1 à 8.

6.2.3 Les Différents Modes De Jeu

Que cela soit en tour par tour ou en temps réel, *TurnBasedGame* et *RealTimeGame* héritent de la classe mère *Game*.

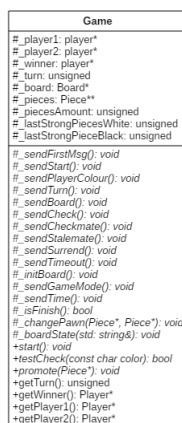


Figure 9: Diagramme de Game

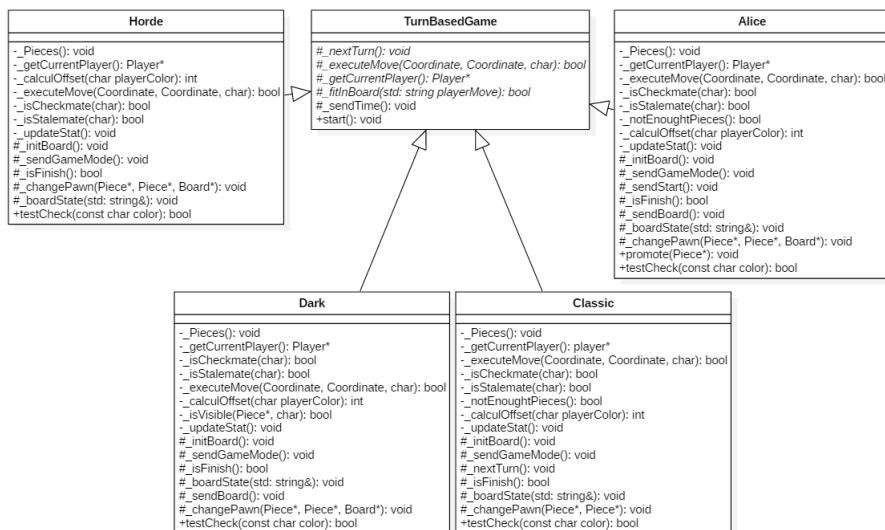


Figure 10: Diagramme des modes de jeu en tour par tour

Tour Par Tour

1. Méthodes Communes

start : Lance la partie et se termine en cas de victoire d'un des 2 joueurs ou en cas d'égalité.

initBoard : Place les pièces sur leur case de départ dans le *Board*.

nextTurn : Demande au joueur dont c'est le tour un mouvement. Ensuite, teste si le mouvement est légal et l'exécute si c'est le cas, sinon redemande un mouvement au joueur.

isFinish : Retourne true si un joueur est en échec et mat ou s'il y a égalité entre les 2 joueurs. Le joueur qui vient de jouer a gagné (auquel cas, l'attribut *winner* est mis à jour).

testCheck : Retourne true si un *King* est en échec.

promote : Exécute la promotion d'un *Pawn*.

sendInfo : Envoi de l'information *Info* au client.

2. Méthodes Classic

getCurrentPlayer : Renvoie le joueur courant.

executeMove : Exécute le mouvement d'une pièce.

isCheckmate : Retourne true si le joueur est en échec et mat.

isStalemate : Retourne true si il y a un pat dans le *Board* courant.

notEnoughPieces : Retourne true si il n'y a plus assez de pièces que pour qu'un des deux joueurs puisse faire un échec et mat.

sendGameMode : Envoie aux deux joueurs le mode de jeu.

boardState : Retourne l'état du *Board* sous forme d'un string et dont le format est celui qui est compréhensible par le client⁸.

Le rôle des méthodes de *Horde*, *Alice* et *Dark* sont les mêmes que pour le mode *Classic*. Néanmoins, le code de ces méthodes divergent selon le mode afin de s'adapter à la spécificité de chacun.

3. Méthodes Alice

Alice possède en plus:

boardState : Le *boardState* prend en considération la dimension du plateau, paramètre propre à ce mode de jeu. Le reste du code se comporte selon la méthode classique.⁹

4. Méthodes Dark

Dark possède en plus:

boardState : Le *boardState* prend en considération la visibilité des différentes pièces, paramètre propre à ce mode de jeu. Le reste du code se comporte selon la méthode classique.¹⁰

isVisible : Cette méthode indique si la pièce est visible par l'adversaire, c'est à dire si celui-ci est dans la possibilité de prendre la pièce. Si tel est le cas, renvoi d'un true.

Temps Réel Les modes en temps réel ne prenant pas en compte les tours des joueurs, il attend dans sa boucle principale de recevoir le mouvement d'un joueur, de devoir déplacer une pièce ou de devoir envoyer au joueur l'état actuel de l'échiquier.

Quand un mouvement est reçu, si il est légal, on construit un *ChainedMove* et le met dans une file d'attente où il sera exécuter au moment voulu.

Si un mouvement doit s'exécuter, le jeu gère les collisions et ensuite, fait bouger la pièce. La gestion des collisions se fait dans l'ordre suivant: on commence par gérer le cas où il y a une pièce immobile sur la case, ensuite, la pièce va se "battre" un à un avec chacune des pièces d'une autre couleur qu'elle présentent sur la case dans l'ordre où elle se sont arrivée sur la case, en commençant par la première arrivée. Si la pièce

⁸Voir affichage de l'échiquier 6.5.6

⁹Voir méthodes Classique de l'échiquier 6.1.3

¹⁰Voir méthodes Classique de l'échiquier 6.1.3

en mouvement perd, elle ne se "battr" pas avec les suivantes. Chaque pièce "battue" devenant une pièce prise. Une pièce gagne une "bataille" si elle à commencer à se déplacer avant l'autre.

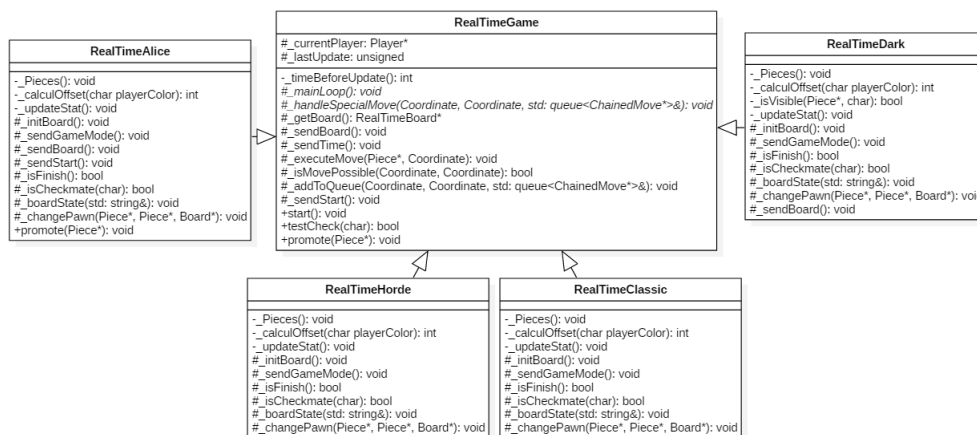


Figure 11: Diagramme des modes de jeu en temps réel

6.2.4 Player

Player se charge de transférer à la partie serveur bon nombre d'informations sur l'état de la partie via ses différentes fonctions *transfer*.

6.2.5 Board

Board garde en mémoire les différentes positions des pièces. Ainsi, il est plus rapide de savoir si une case est occupée ou non.

6.2.6 RealTimeBoard

RealTimeBoard et un *Board* mais qui permet en plus de lock¹¹/unlock une case et qui permet à plusieurs pièces d'être sur la même case à condition qu'il n'y ai au maximum qu'une seule pièce qui n'est pas en train de se déplacer par case.

6.2.7 ChainedMove

ChainedMove décompose le mouvement d'une pièces en une série de déplacements de 1 case et permet aussi de garder en mémoire le moment où le prochain déplacement sera effectué ainsi que la pièce à déplacée.

¹¹Une case est lock si le mouvement d'une pièce se terminera sur cette case.

6.3 Connexion du Joueur

Le joueur se connecte à l'aide du *hostname* du serveur, qui s'affiche à l'exécution de celui-ci. Il accède ensuite à un menu lui permettant de se connecter à son compte ou d'en créer un nouveau.

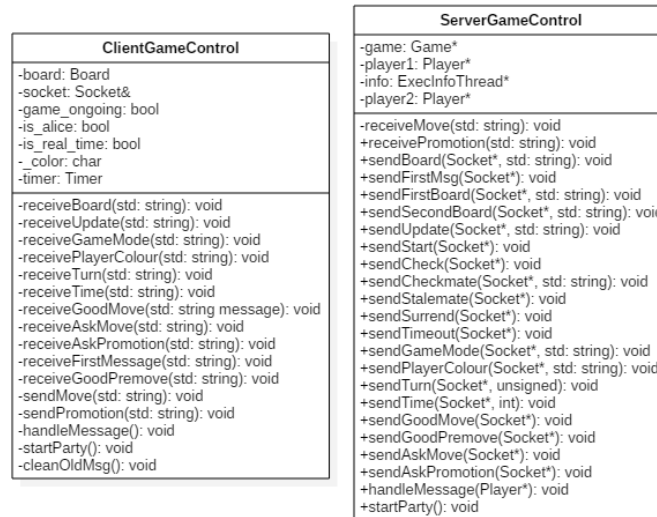


Figure 12: Diagramme de classe des *GameControl*

6.4 Client-Serveur

Dans le cadre de ce projet, il a été décidé que la partie serveur s'occupera de toute la logique du jeu tandis que la partie client s'assurera d'afficher correctement l'interface du jeu en temps réel et pour ce faire, recevra des requêtes du serveur pour connaître les mouvements de pièces voulus par les joueurs.

Il convient également de spécifier que l'échange d'information entre le serveur et le client se fera sous le format de chaînes de caractères. Par exemple, le serveur enregistrera les noms de compte et mots de passe des joueurs dans un fichier texte et non une base de données. Et cela pour au moins 3 raisons décrites ci-dessous :

1. Comme la plupart des tâches effectuées par nos programmes se font à l'aide de chaînes de caractères et par souci d'homogénéité, il est préférable et plus simple que l'échange d'information entre le serveur et le client se fasse également sous cette forme.
2. Une chaîne de caractère n'a pas besoin d'être traduite en binaire, ce qui diminue la complexité et le temps d'exécution de nos programmes.
3. La taille du message envoyée n'est pas un problème, car une taille de 150 octets suffit pour envoyer toutes les informations d'un message. Le plus grand message actuel est le plateau, qui ne consiste que de 147 octets lorsqu'il est complet dans un jeu de *Horde Chess* (voir 6.5, point 5).

Si l'on souhaite tout de même envoyer des messages plus longs que la taille standard définie, le système de messagerie utilise du padding à l'aide du symbole ']' afin d'identifier la fin du message. La classe *Socket* effectue ce padding ainsi que l'envoi et la réception de messages à l'aide des méthodes *sendMessage* et *receiveMessage*.

Le serveur tâchera de vérifier les différentes actions des joueurs et veillera également à envoyer régulièrement la disposition du plateau aux clients afin qu'il s'affiche correctement pour les joueurs.

Afin d'identifier les messages échangés et le protocole à suivre suite à la réception d'un message, ceux-ci sont précédés d'un en-tête.

6.5 Déroulement d'une partie

Les différentes étapes de mise en place et de déroulement d'une partie sont décrites ici, telles qu'effectuées lors d'une connexion avec un client utilisant la GUI ncurses.

1. *Initialisation du plateau côté serveur*

Mise en pratique : Le serveur initialise son objet de plateau en entrant toutes les pièces adaptées au jeu demandé, à leurs coordonnées respectives.

Protocole : L'instance de jeu fait appel à une méthode `_initBoard`, définie selon le mode de jeu.

2. *Envoi du mode de jeu aux clients*

Mise en pratique : Le serveur envoie le nom du mode de jeu choisi aux clients afin qu'ils l'affichent.

Protocole : L'instance de jeu fait appel à une méthode `_sendGameMode`, qui déclenche l'envoi d'une chaîne de caractères décrivant le mode de jeu, précédé de l'en-tête 'G' (gamemode). Une fois reçu, le message est identifié à l'aide de l'en-tête et placé dans une variable de l'instance d'affichage du jeu.

3. *Assignation de la couleur des pièces aux joueurs*

Mise en pratique : Le serveur envoie la couleur du joueur (noir ou blanc) à son client.

Protocole : Le serveur envoie 'white' au premier joueur, le joueur blanc, et 'black' pour le second joueur, précédé de l'en-tête 'X' (colour).

4. *Envoi de la confirmation de début de partie par le serveur*

Mise en pratique : Le serveur envoie un message pré-défini aux clients.

Protocole : Le serveur envoie 'start' aux clients, précédé de l'en-tête 'U' (update), ce qui déclenche le passage du mode d'affichage du menu au mode d'affichage du jeu.

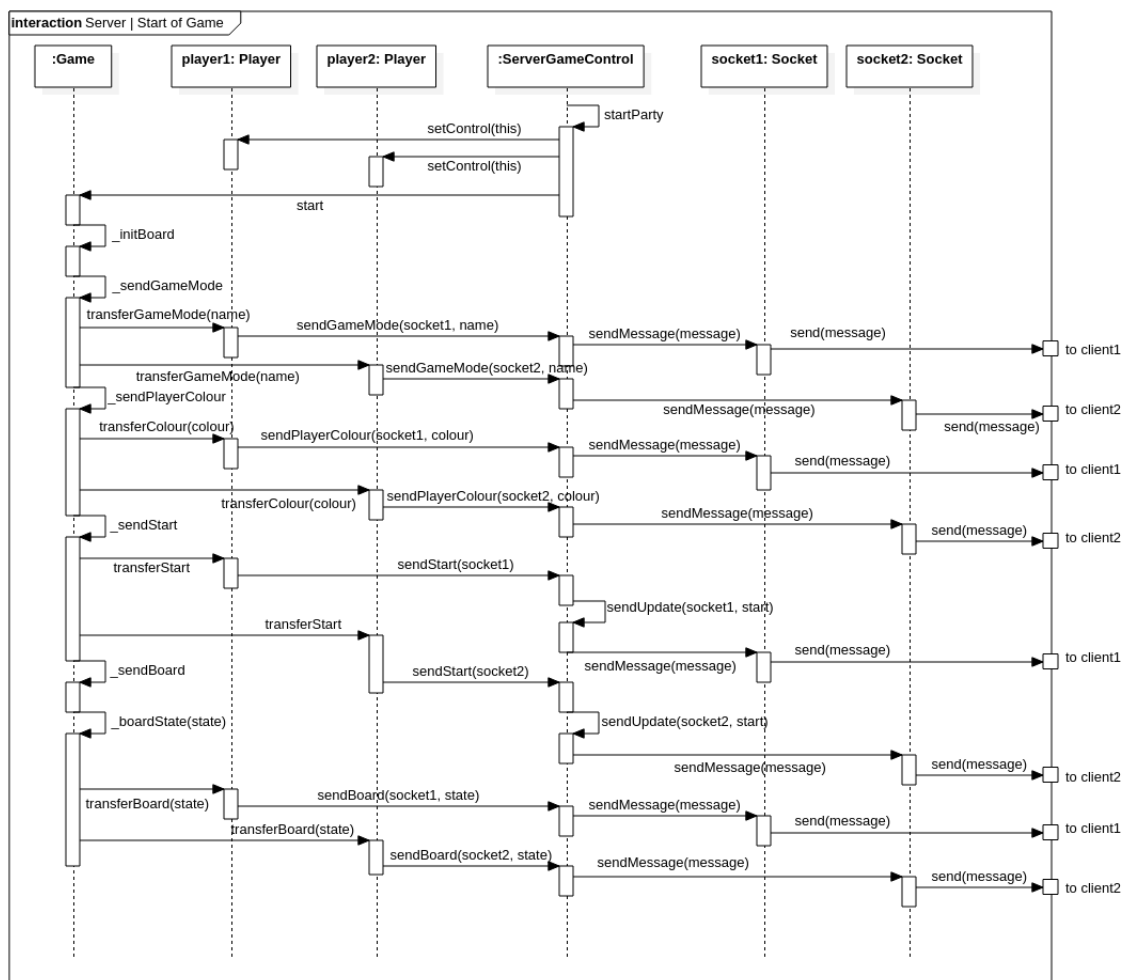


Figure 13: Diagramme de séquences d'initialisation du jeu côté serveur, dans le cas d'un jeu d'échecs classique.

5. Affichage de l'échiquier

Mise en pratique : Le serveur envoie le plateau aux joueurs.

Protocole : Le message est une suite de toutes les pièces présentes sur l'échiquier, chacune représentée par 3 caractères, précédée de l'en-tête 'B' (board). Le premier caractère est une lettre désignant le type de la pièce considérée. Les 2 autres, sa position, en commençant par la colonne, selon la représentation classique d'un échiquier, c'est à dire, les colonnes sont représentées de "A" à "H", et les lignes numérotées de 1 à 9. Les pièces blanches sont d'abord toutes envoyées, séparées des pièces noires par un symbole '!' puis les pièces noires sont envoyées, suivies du symbole '#'. Exemple : pA2hA4rD5qC7kE1bF8!bA8qG4kD7#. L'envoi d'un plateau contenant toutes les pièces du jeu d'échecs de horde est donc de 146 octets (sans compter le padding): 1 octet par caractère, 3 caractères par pièce, 48 pièces sur un plateau, soit $3 \times 48 = 144$, plus les 2 symboles de séparation.

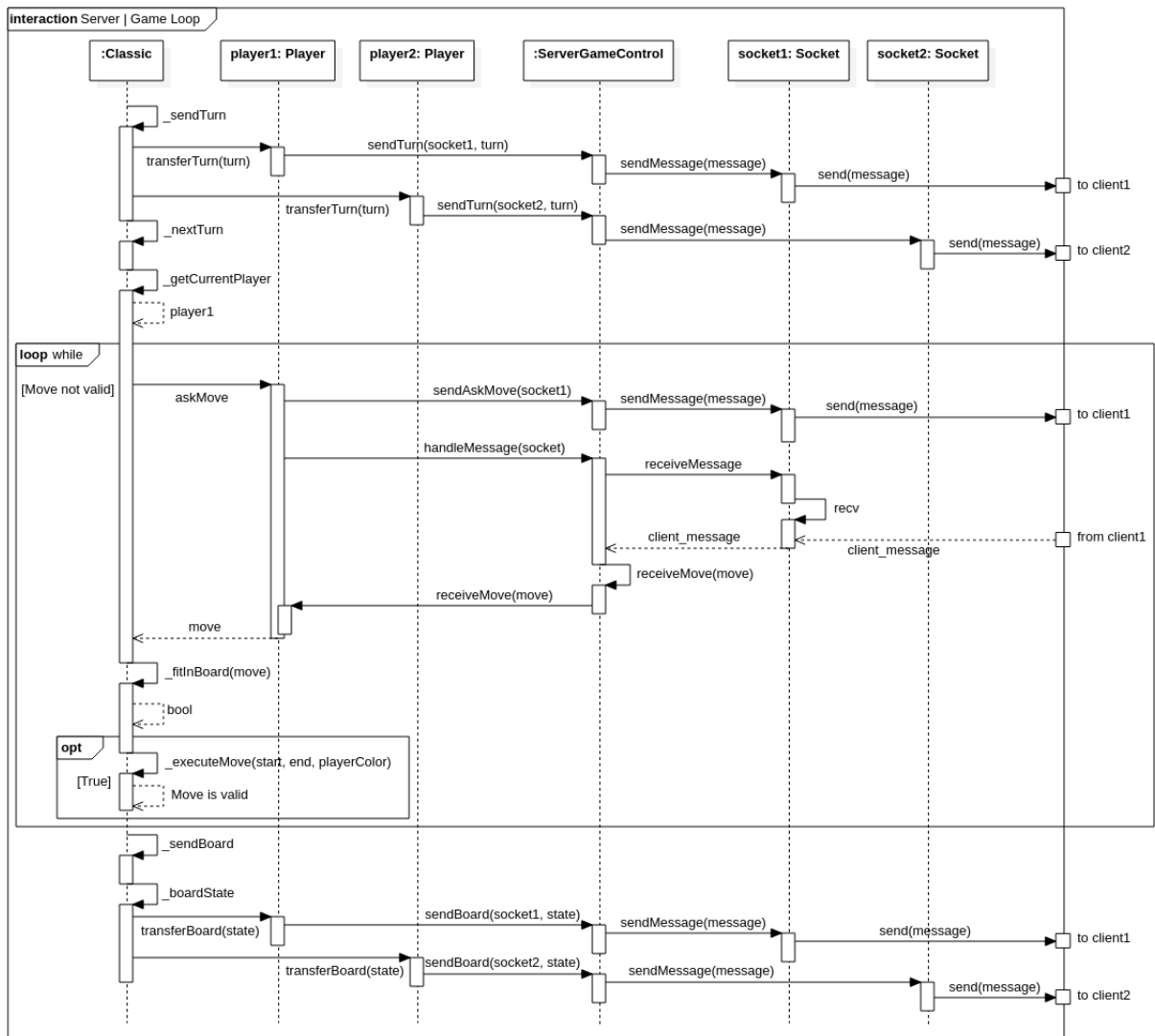


Figure 14: Diagramme de séquences des étapes à l'intérieur d'une boucle de jeu dans le cas d'un jeu d'échecs classique où le premier joueur est le joueur courant, représenté du côté serveur.

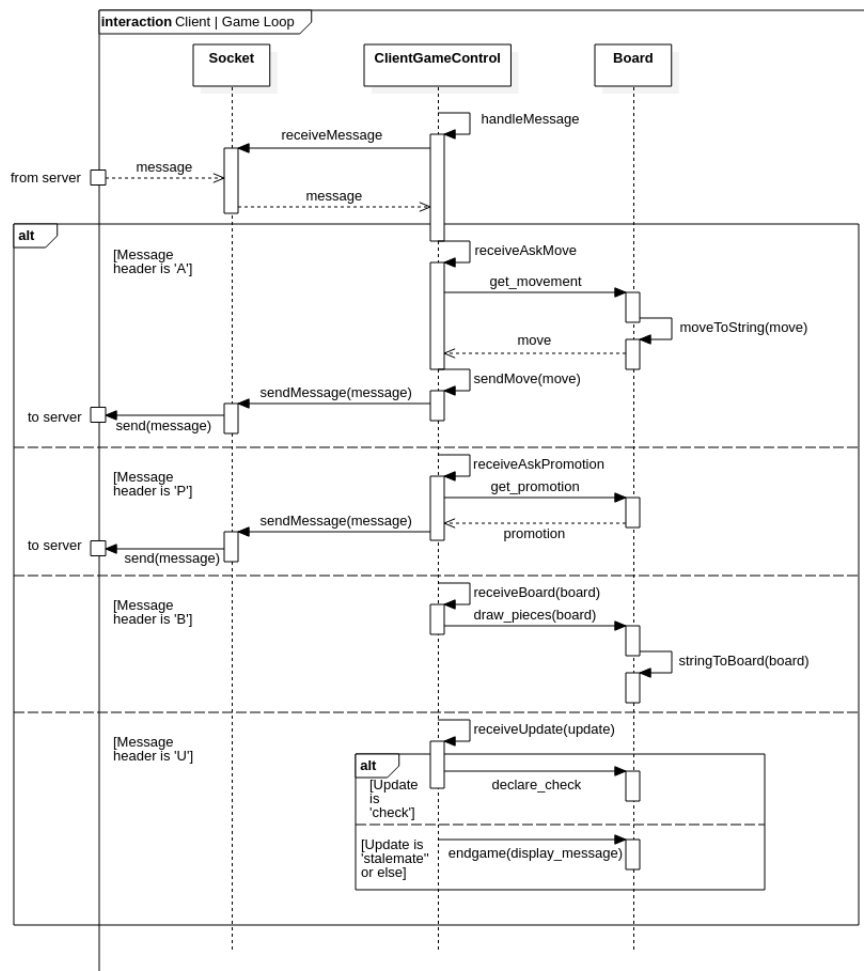


Figure 15: Diagramme de séquences de réception de messages durant une partie de jeu d'échecs classique, représenté du côté client.

6. Envoi du nombre de tours de jeu

Mise en pratique : Le serveur envoie le numéro du tour actuel de jeu aux joueurs.

Protocole : À partir d'un compteur dans la boucle de jeu, le serveur envoie le chiffre au client précédé de l'en-tête 'T' (turn), qui apparaît dans la fenêtre d'information de l'affichage du jeu.

7. Requête de mouvement du joueur

Mise en pratique : Le serveur identifie le joueur qui doit effectuer un mouvement et envoie une requête au client approprié. Celui-ci affiche la requête au joueur, qui entre le mouvement de pièce voulu. Le mouvement est renvoyé au serveur, qui vérifie la légalité de la demande - tant que le mouvement demandé par le joueur n'est pas autorisé par les règles du jeu, celui-ci doit rentrer une nouvelle demande de mouvement.

Protocole : La boucle de jeu présente dans la logique du serveur identifie l'instance de joueur concernée par la demande de mouvement et déclenche un envoi de message à son client ayant comme en-tête 'A' (askmove). Ce message est interprété par le client, qui demande au joueur de proposer un mouvement pour une de ses pièces. Le mouvement est alors renvoyé au serveur pour être vérifié. Ce message est

composé de 4 caractères, les 2 premiers représentant l'emplacement du début de tour de la pièce et les 2 derniers, la position future de la pièce, précédée de l'en-tête 'M' (move). Le serveur vérifie si le mouvement demandé est possible, et renvoie une demande de mouvement tant qu'il ne l'est pas.

8. *Mouvement de pièce*

Mise en pratique : Une fois que le mouvement demandé par le joueur est identifié comme possible, le serveur met à jour son instance de plateau et renvoie le plateau aux deux clients afin qu'ils l'affichent.

Protocole : voir 5.

9. *Promotion de pièce*

Mise en pratique : Si le mouvement du joueur qui vient d'être effectué permet une promotion de pion (voir 1.2.2), le serveur envoie une requête au client pour connaître la promotion voulue. Une fois la demande reçue, le serveur met à jour le plateau et le renvoie.

Protocole : Le serveur envoie un message ayant pour en-tête 'P' (promote) au client concerné. Celui-ci affiche la demande au joueur, qui choisit la pièce qu'il souhaite obtenir. Le client envoie la demande au serveur, qui consiste en un message de un caractère, représentant la pièce choisie ('q': reine/queen, 'b': fou/bishop, 'h': cavalier/horse/knight, 'r': tour/rook), précédé de l'en-tête 'P' (promote).

10. *Fin du tour de jeu*

Mise en pratique : Le serveur vérifie si le jeu est terminé, ce qui consiste en deux cas (un des joueurs a placé l'autre en échec et mat, il a gagné; aucun des joueurs n'est en mesure de placer l'autre en échec et mat, il y a pat). Si le jeu n'est pas fini, le serveur passe au tour suivant.

Protocole : L'instance de jeu du serveur fait un appel régulier à une méthode `_isFinish` pour identifier une fin de partie. Si c'est le cas, un message est envoyé au client, précédé de l'en-tête 'U' (update). Dans le cas d'un échec et mat, la couleur du gagnant est envoyée; dans le cas d'un pat, le message 'stalemate' est envoyé. Si le jeu n'est pas fini, le compteur de tours est incrémenté et la boucle de jeu reprend du début (6).

6.6 Menu

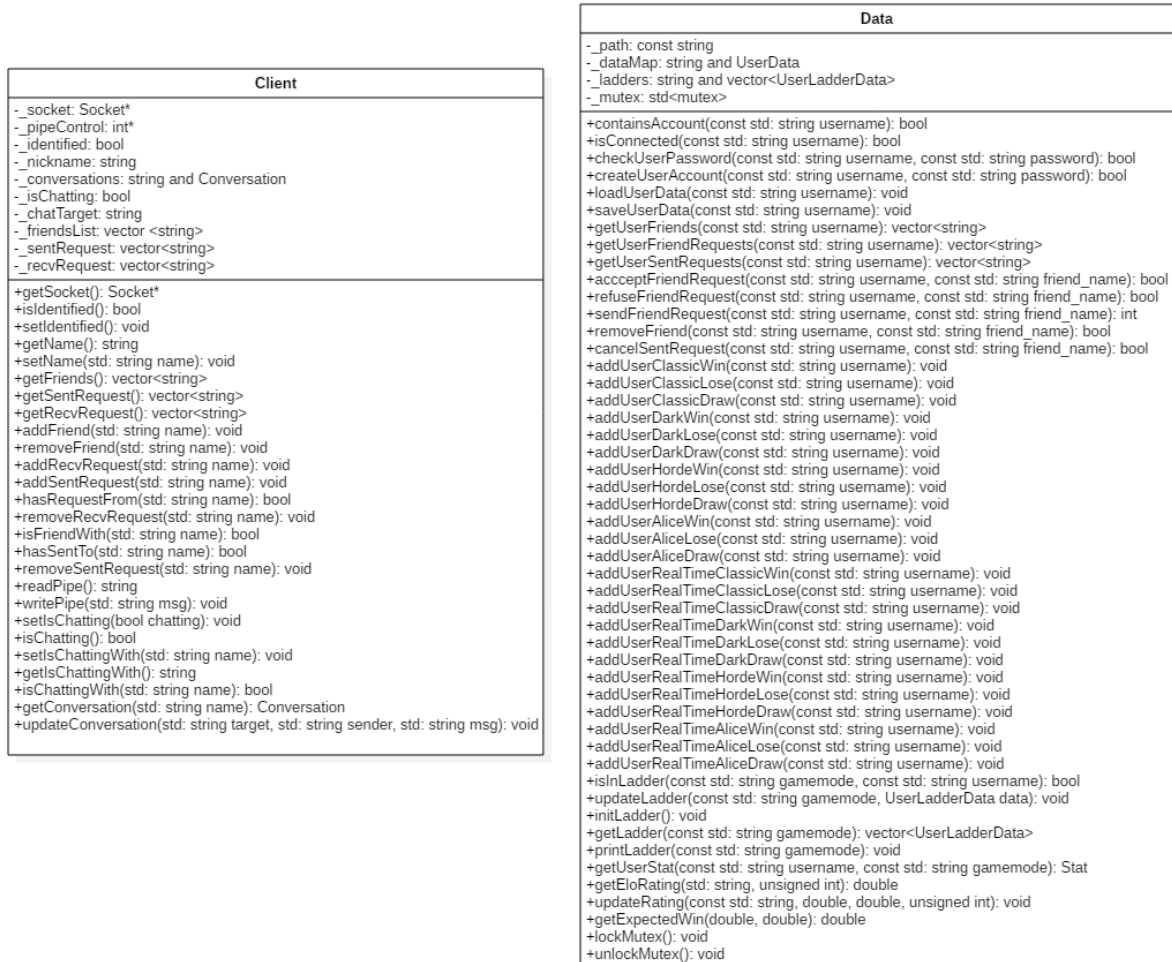


Figure 16: Diagramme de classe du *Client* et *Data*

6.6.1 Affichage du menu

Mise en pratique : Les méthodes de l'objet permettant l'affichage du menu sont utilisées et les options suivantes sont proposées à l'utilisateur : *Jouer*, *Amis*, *Statistique* et *Quitter*.

Protocole : Si l'utilisateur joue en mode console : La méthode `_get_choice`, toujours précédée par la méthode `_init_choicew`, est utilisée afin de pouvoir afficher les différentes options. Avec l'interface graphique

6.6.2 Choix d'une option

Mise en pratique : L'utilisateur peut parcourir les différentes options en utilisant les touches directionnelles *Haut* et *Bas* et valider son choix en appuyant sur la touche *Entrée*

Protocole : Si l'utilisateur joue en mode console : Après avoir appelé la méthode `_get_choice`, le programme attend que l'utilisateur choisisse une option. Si le choix a été fait, une fonction est appelée pour

diriger l'utilisateur vers un autre menu.

6.6.3 Choix d'option du menu Jouer

Mise en pratique : Les 4 modes de jeu sont affichés et l'utilisateur peut rejoindre une file d'attente vers le mode de jeu approprié en sélectionnant une de ces 6 options ou revenir au menu principal en choisissant l'option *Retour*.

Protocole : Si l'utilisateur choisit un mode de jeu, la fonction *playGame* est appelée et elle envoie un message au serveur qui placera le joueur dans une file d'attente. Ensuite l'utilisateur sera bloqué jusqu'à ce qu'un autre joueur rejoigne la même file d'attente. Sinon l'utilisateur sera dirigé vers le menu principal.

6.6.4 Choix d'option du menu Ami

Mise en pratique : L'utilisateur a le choix entre 5 options : *Voir sa liste d'amis*, *Envoyer une demande d'ami*, *Retirer un ami*, *Accepter ou refuser une demande d'ami* et *Retour*.

Protocole : Après que l'utilisateur a cliqué sur le bouton correspondant, le programme attend que l'utilisateur choisisse une option. Si un choix a été fait, une fonction est appelée pour diriger l'utilisateur vers un autre menu.

6.6.5 Voir sa liste d'amis

Mise en pratique : Après avoir choisi cette option, les noms des amis de l'utilisateur sont affichés dans un rectangle.

Protocole : Un message est envoyé au serveur qui traitera la demande et qui renverra le nom de tous ses amis. Lorsque l'entièreté des noms des amis sont reçus du côté client, leur nom est affiché sur l'écran de l'utilisateur.

6.6.6 Envoyer une demande d'ami

Mise en pratique : Après avoir choisi cette option, l'utilisateur est invité à rentrer un nom ou une autre commande pour retourner au menu des amis.

Protocole : Si l'utilisateur ne rentre pas la commande pour quitter ce menu, alors un message est envoyé au serveur qui le traitera et renverra le résultat de l'opération. Plusieurs résultats sont possibles, soit la demande a été envoyée soit l'une de ces erreurs s'est produite : l'utilisateur s'est envoyé une demande d'ami à lui-même, la personne se trouve déjà dans sa liste d'amis, le nom n'est lié à aucun compte ou alors une demande a déjà été envoyée.

6.6.7 Retirer un ami

Mise en pratique : Après avoir choisi cette option, l'utilisateur est invité à rentrer un nom ou une autre commande pour retourner au menu des amis.

Protocole : Si l'utilisateur ne rentre pas la commande pour quitter ce menu, alors un message est envoyé au serveur qui le traitera et renverra le résultat de l'opération. Plusieurs résultats sont possibles, soit la personne a été retirée de la liste d'amis soit l'utilisateur a entré le nom d'une personne qui ne fait pas partie de ses amis.

6.6.8 Accepter ou refuser une demande d'ami

Mise en pratique : Après avoir choisi cette option, l'utilisateur est invité à rentrer une commande suivi d'un nom ou autre une commande pour retourner au menu des amis.

Protocole : Si l'utilisateur ne rentre pas la commande pour quitter ce menu, alors un message est envoyé au serveur qui le traitera et renverra le résultat de l'opération. Plusieurs résultats sont possibles, soit la demande a été acceptée ou refusée, soit aucune demande d'ami n'a été envoyée par le nom de la personne encodé.

6.6.9 Choix d'option du menu Statistiques

Mise en pratique : L'utilisateur a le choix entre 3 options : *Voir ses statistiques*, *Voir le classement général* et *Quitter*.

Protocole : Si l'utilisateur décide de voir ses statistiques, la fonction *checkMyStat* est appelée et enverra un message au serveur qui retournera les statistiques de l'utilisateur qui seront affichées dans un rectangle. Si l'utilisateur veut voir le classement général, il sera redirigé vers un autre menu. Sinon, l'utilisateur retourne au menu principal.

6.6.10 Choix d'option du menu Classement général

Mise en pratique : L'utilisateur a le choix entre 9 options : *Classic*, *Dark*, *Horde*, *Alice* et ses variantes en *RealTime* et *Retour*.

Protocole : Si l'utilisateur choisit une des 4 premières options, alors un messages est envoyé au serveur qui renverra le nom et les statistiques des 10 premiers joueurs du mode sélectionné. Ces informations seront ensuite affichées dans un rectangle. Sinon l'utilisateur revient au menu principal.

6.6.11 Utilisation du chat

Mise en pratique : Une fois que l'utilisateur a sélectionné cette option, il peut discuter soit avec ses amis soit discuter avec n'importe quelle autre personne dans le canal global. L'utilisateur navigue entre les différents chats via la commande */change [nom ami]* ou */change [all]* pour le chat global.

Protocole : Une fois que l'utilisateur rentre la commande pour discuter, un message sera envoyé au serveur qui le traitera et renverra le résultat de l'opération. Plusieurs résultats sont possibles, soit le message de l'utilisateur s'est correctement envoyé, soit la commande a été mal entrée et dans ce cas, aucun message n'est envoyé.

7 Annexe