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

3 Mars 2019

Table des Matières

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 pourra jouer au 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. Néanmoins, le joueur pourra lancer une partie en tant que *visiteur* mais ne bénéficiera pas des avantages liés à la possession d'un compte.

1.2 Fonctionnement du Jeu d'Échec

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 remédier[1].

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.

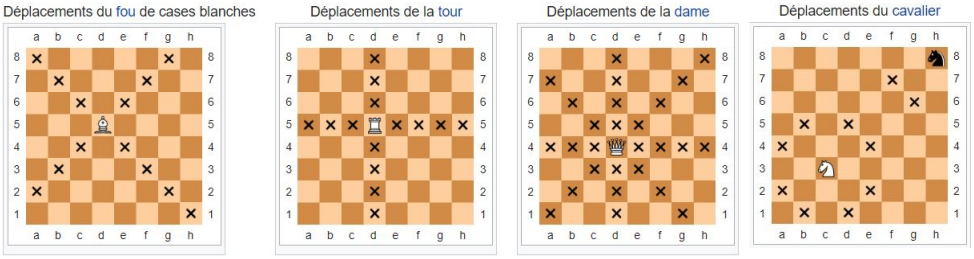

Figure 1: Image provenant de Wikipedia

[1] Wikipedia

1.3 Les Modes de Jeu

1.3.1 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.

1.3.2 Alice Chess

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

1.3.3 Dark Chess

Le jeu d'échecs sombre 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.

1.3.4 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. Le but du joueur noir est le but du jeu classique mais le but du joueur blanc est de prendre tous les pions du joueur noir. À part cela, le jeu se déroule selon les règles habituelles du jeu d'échecs.

1.4 Fonctionnalités pour Utilisateur Enregistré et Utilisateur Non-Enregistré

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 *Game*.

1.5 Fonctionnalités pour Utilisateur Enregistré

Les joueurs pourront communiquer via un chat global. Dans le système de matchmaking, ils auront le choix entre affronter un ami ou un joueur aléatoire parmi ceux qui sont connectés et libres[2]. À travers le menu principal, les utilisateurs accéderont à plusieurs sous-menus tels que *View Ranking, View Rules, View Statistics, Friends List*.

1.6 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 non-enregistré : Utilisateur ne s'étant pas identifié au système via un compte.

Un utilisateur enregistré : 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.

[2] ne se trouvant pas en partie.

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

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 illégal[3])

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.

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 mate 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 le joueur dont c'est le tour n'a pas de coup légal possible mais n'est pas en situation d'échec.

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, avec la même main, 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 adverse[4].

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 case[5].

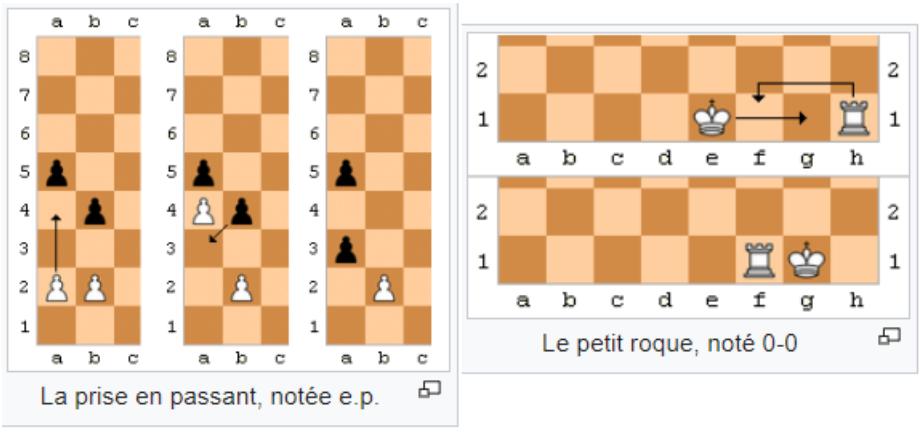

Figure 2: Image provenant de Wikipedia

[3]source Wikipedia
[4]source Wikipedia
[5]source Wikipedia

1.7 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

Historique des modifications

2 Besoins Utilisateurs : Fonctionnels

2.1 Connexion

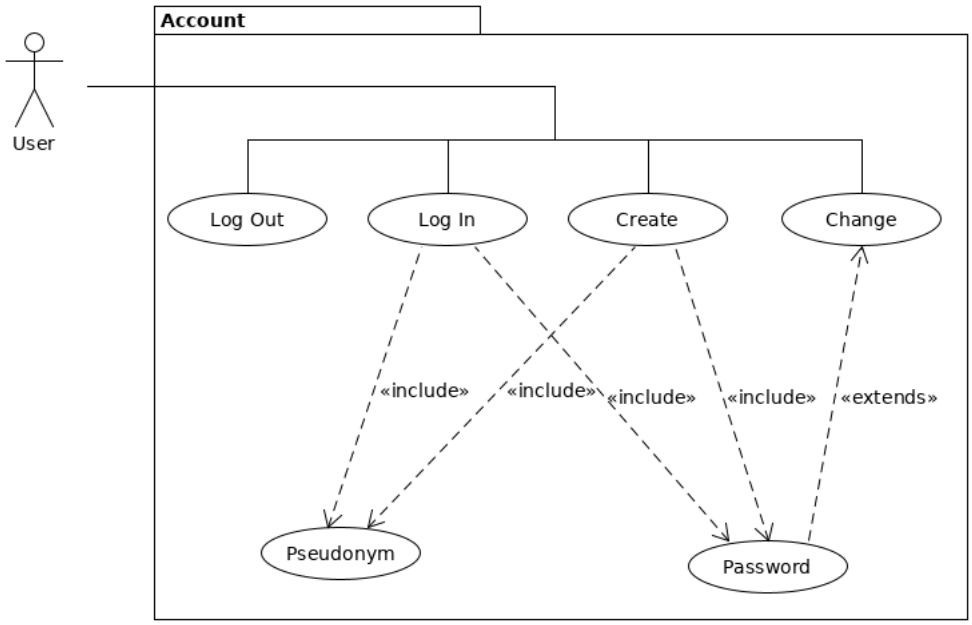

Figure 3: Diagramme de *use case* des actions possibles d'un utilisateur quant à son identification

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

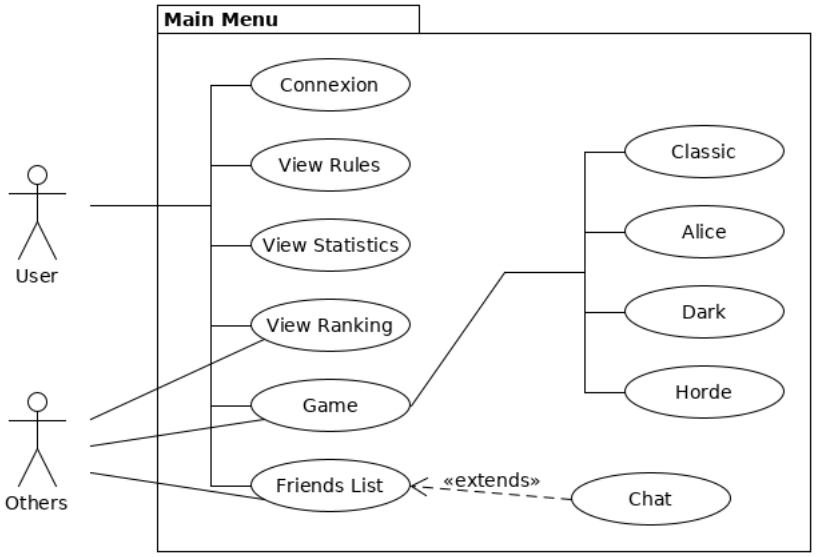

Figure 4: Diagramme de *use case* des actions possibles d'un utilisateur

2.2.1 Jouer

Acteur : *User*.
Relations avec d'autres cas d'utilisation : Modes de jeu proposés (*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 (*Classic*,*Alice*, *Dark* ou *Horde*).
Cas général : Le *user* peut sélectionner un mode de jeu parmi ceux proposés (*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.
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 (*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 et le serveur.
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 lorsque c'est à son tour de jouer. Il envoi 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 Discuter dans le Chat

Acteur : *User*.
Relations avec d'autres cas d'utilisation : *Friends List*.
Pré-conditions : Avoir un ami ou "adversaire" et qu'il soit connecté.
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.

2.3.3 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 pour pouvoir jouer leur tour.

4 Besoins Systèmes : Fonctionnels

4.1 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.

4.2 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.

4.3 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.

4.4 Voir les Statistiques Personnelles

Acteur : *Server*.
Relations avec d'autres cas d'utilisation : Néant.
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.

4.5 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.

4.6 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.

4.7 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 durant une partie mais également en dehors d'une partie s'ils sont amis.
Cas exceptionnels : Néant.

4.8 Gestion d'une Partie

4.8.1 Premier à Jouer

Le joueur blanc commence. Celui-ci est le premier joueur connecté et qualifiable de la file d'attente pour ce jeu.

4.8.2 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.

4.8.3 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.

4.8.4 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.

4.8.5 Déroulement de la Partie

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

4.8.6 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 l'on a affaire à un Pat. 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

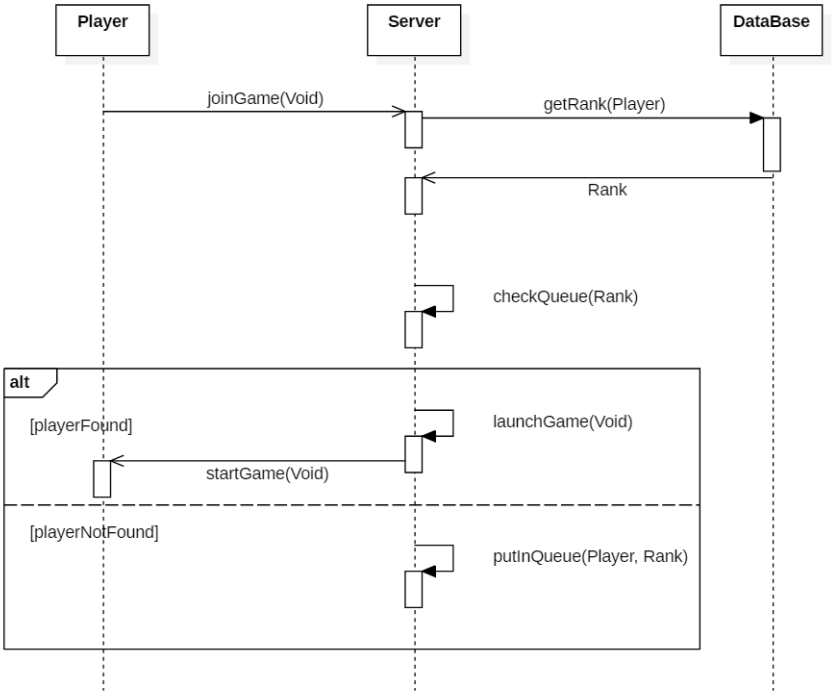

Figure 5: Diagramme de séquence pour le *Matchmaking*

6.1 Diagrammes de Classes

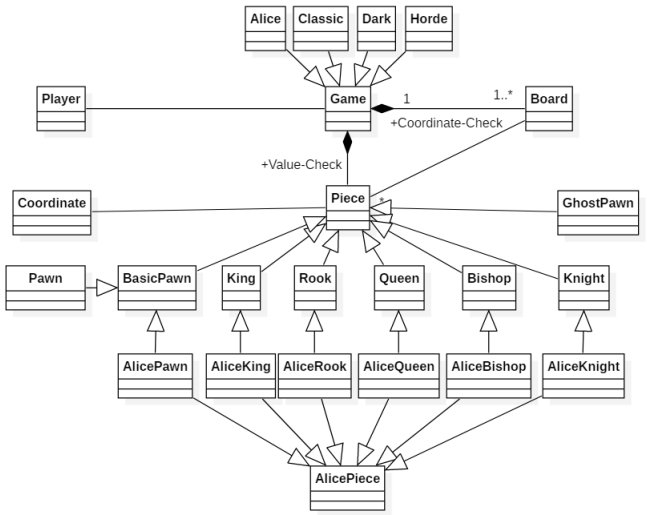

Figure 6: Diagramme de classes général

6.1.1 Les Différentes Pièces

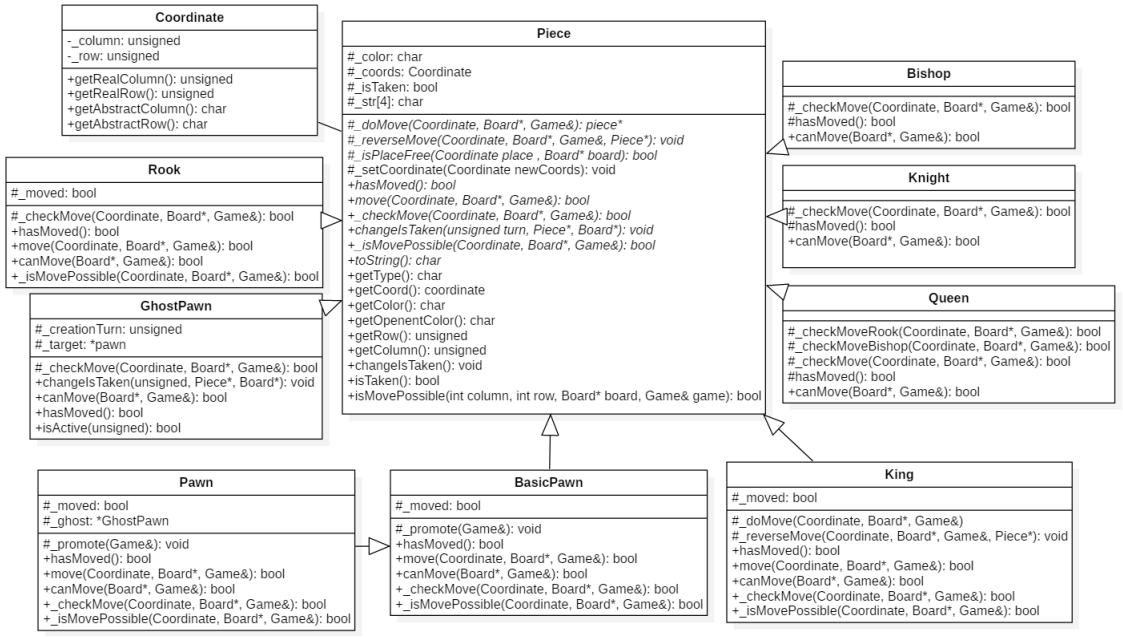

Figure 7: Diagramme des pièces et coordinate

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 si 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 ce dernier est actif (il est actif uniquement le tour juste après sa création). Lorsqu'un *GhostPawn* est pris par un *Pawn* et qu'il est actif, le *GhostPawn* se charge de faire disparaitre le *Pawn* qui l'a créer du plateau. *BasicPawn* est un *Pawn* qui ne peut effectuer de prise en passant. En effet, on voit qu'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 déplacement normalement valide.
   hasMoved : Retourne true si la pièce a déjà été déplacée ou non.
   isMovePossible : vérifie si un movement est possible en prenant en compte le fait que le *King* ne doit pas être en échec après ce mouvement.
   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

6.1.2 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.

6.1.3 Les Différents Modes De Jeu

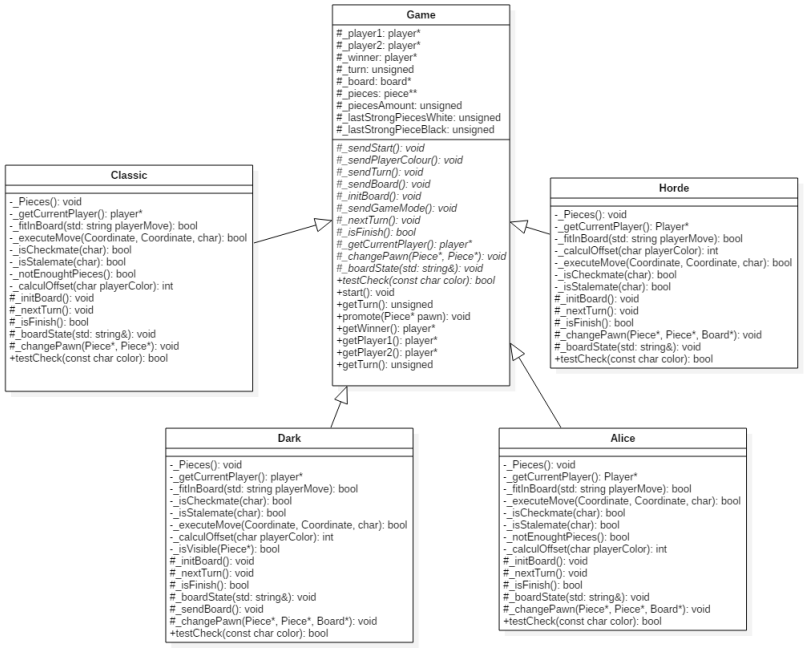

Figure 8: Diagramme des modes de jeu

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 possible et l'exécute si c'est le cas, sinon redemande un mouvement au joueur.
   isFinish : Retourne true si un joueur le joueur qui vient de jouer à gagner (auquel cas le l'attribut *winner* est mis à jour) ou si il y a égalité entre les 2 joueurs.
   testCheck : Retourne true si un *King* est en échec.
   promote : Exécute la promotion d'un *Pawn*.
   send*Info* : Envoie l'information *Info* au client.

2. *Méthodes Classic*
   getCurrentPlayer : Renvois le joueur courant.
   executeMove : Exécute le mouvement d'une pièce.
   isCheckmate : Retourne true si le joueur est en situation d'échec et mate.
   isStalemate : Retourne true si il y a un pat dans le *Board* courant.
   notEnoughtPieces : Retourne true si il n'y a plus assez de pièces que pour qu'un des deux joueurs puisse faire un échec et mate.
   sendGameMode : Envoie aux deux joueurs le type de jeux.
   boardState : Retourne l'état du *Board* sous forme d'un string et dont le format est celui qui est compréhensible par le client[6].

[6]Voir affichage de l'échiquier 6.5.6

3. *Méthodes Alice*

4. *Méthodes Dark*

5. *Méthodes Horde*

6.1.4 Player

6.1.5 Board

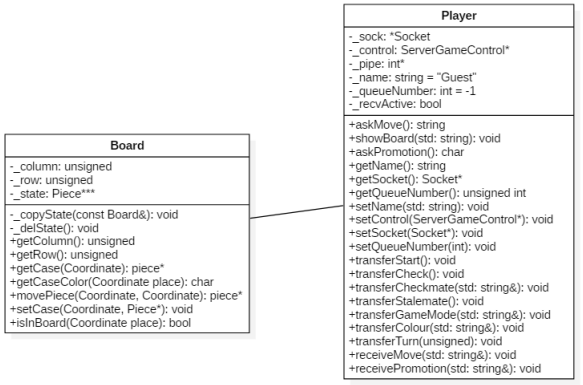

Figure 9: Diagram de Board et Player

6.2 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.

6.3 Les Actions du Joueur

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,lLe 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

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 chaine 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.

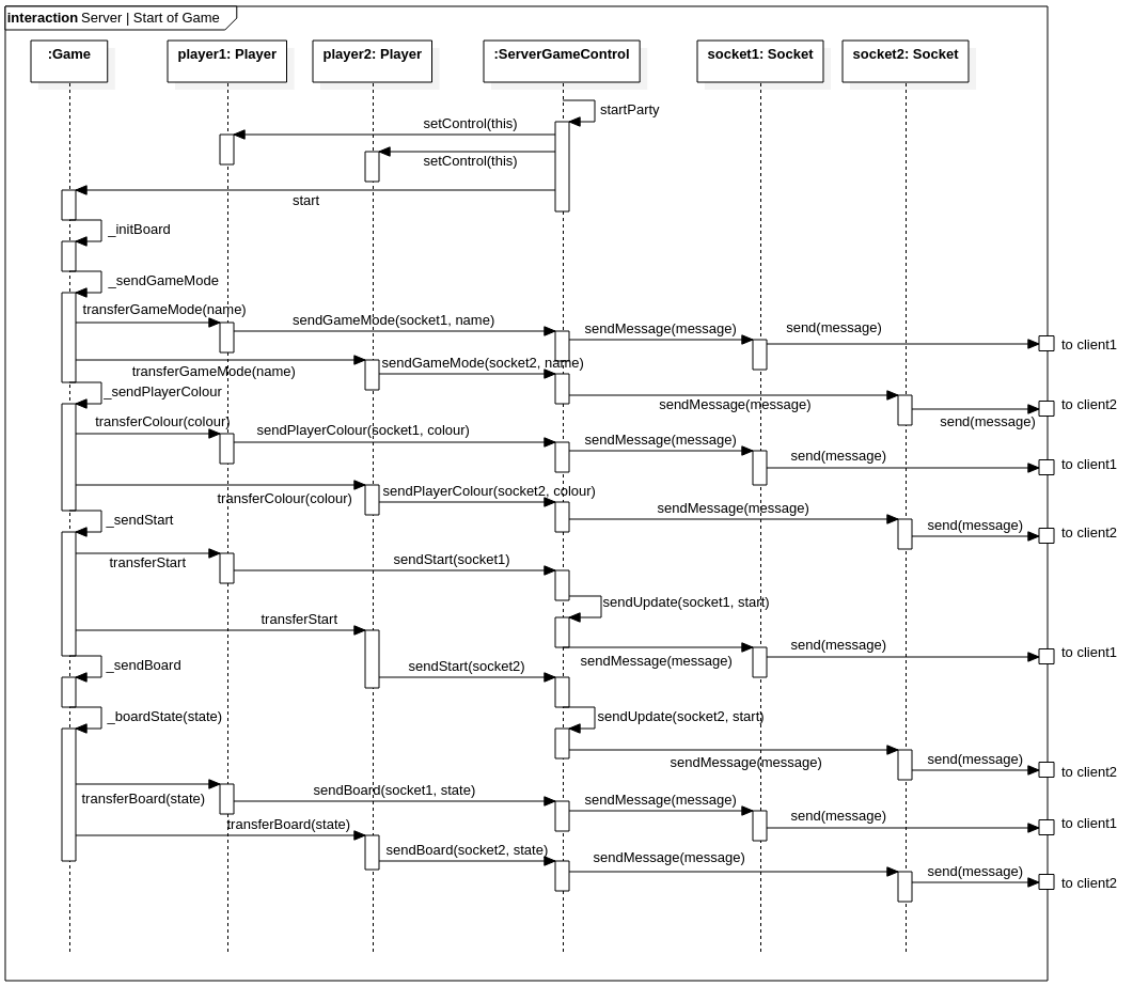

Figure 10: Diagramme de séquences d'initialisation du jeu côté serveur, dans le cas d'un jeu d'échecs classique.

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). Pour l'instant, le premier client connecté devient le premier joueur. Comme pour le point précèdent, ce message est placé dans une variable de l'instance d'affichage du jeu.

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.

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*48=144, plus les 2 symboles de séparation.

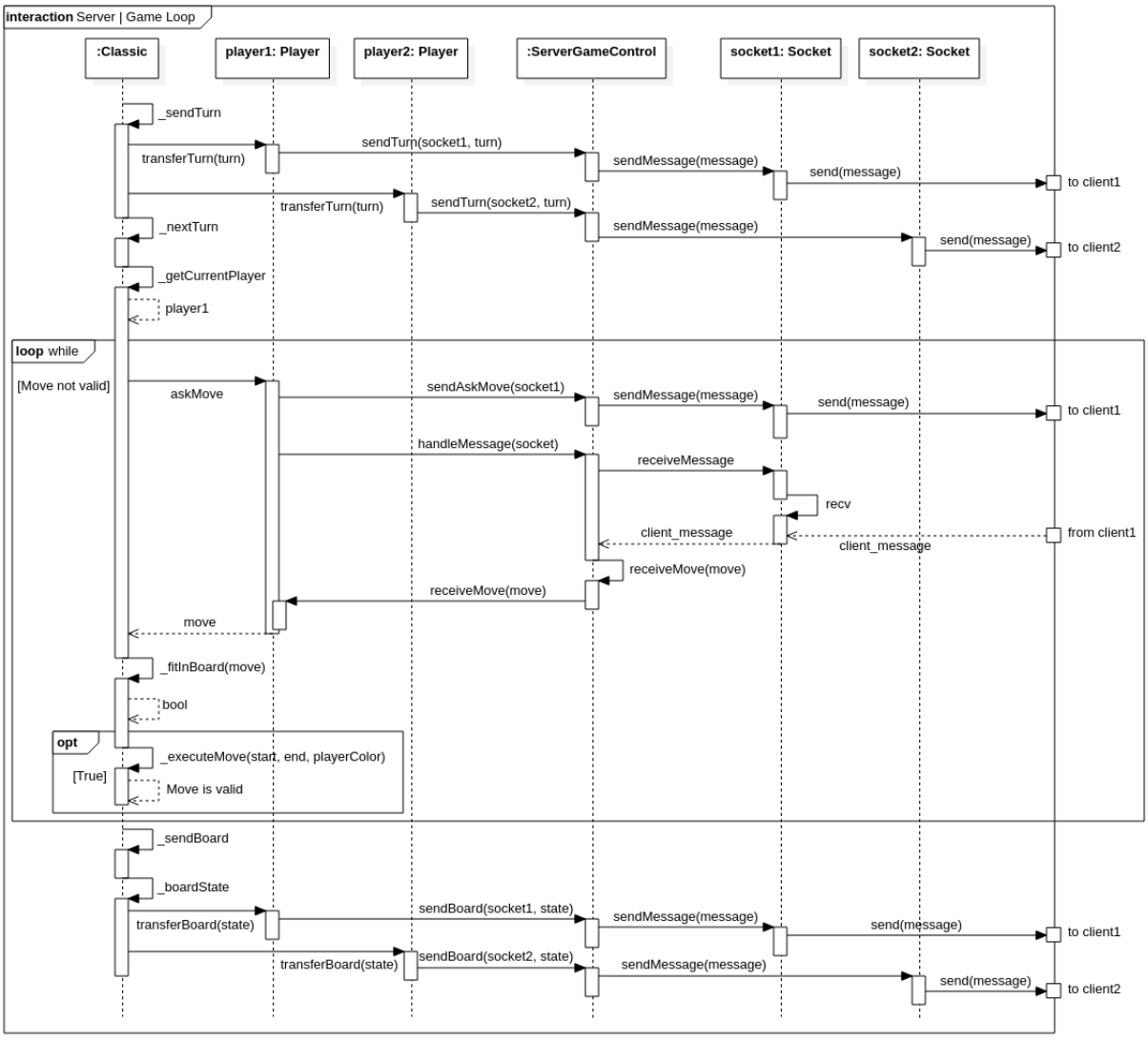

Figure 11: Diagramme de séquences des étapes à l'intérieur d'une boucle de jeu dans le cas d'un jeu d'échecs classique où le premier joueur est le joueur courant, représenté du côté serveur.

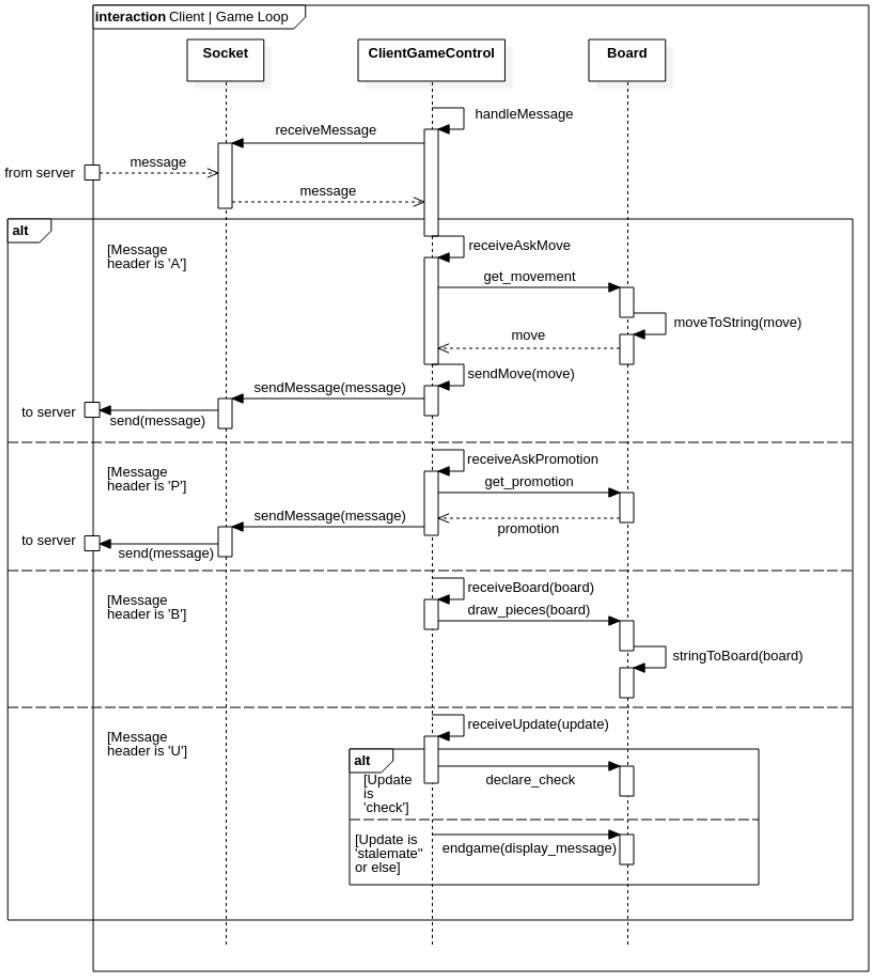

Figure 12: Diagramme de séquences de réception de messages durant une partie de jeu d'échecs classique, représenté du côté client.

6. *Envoi du nombre de tours de jeu*
   Mise en pratique : Le serveur envoie le numéro du tour actuel de jeu aux joueurs.
   Protocole : A partir d'un compteur dans la boucle de jeu, le serveur envoie le chiffre au client précédé de l'en-tête 'T' (turn), qui apparait 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, précédé 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 connaitre la promotion voulue. Une fois la demande recue, 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 deux cas (un des joueurs à 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 a 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

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 : La méthode *_get_choice*, toujours précédée par la méthode *_init_choicew*, est utilisée afin de pouvoir afficher les différentes options.

2. *Choix d'une option*
   Mise en pratique : L'utilisateur peut parcourir les différentes options en utilisant les touches directionnelle *Haut* et *Bas* et valider son choix en appuyant sur la touche *Entrée*
   Protocole : Après que la méthode *_get_choice* est appelée, le programme attend que l'utilisateur choisisse une option. Après que le choix a été fait, une fonction est appelée pour diriger l'utilisateur vers un autre menu.

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 4 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 et 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.

4. *Choix d'option du menu Ami*
   Mise en pratique : L'utilisateur a le choix entre 6 options : *Voir sa liste d'amis*, *Envoyer une demande d'ami*, *Retirer un ami*, *Accepter ou refuser une demande d'ami*, *Annuler sa demande d'ami* et *Retour*.
   Protocole : Après que la méthode *_get_choice* est appelée, le programme attend que l'utilisateur choisisse une option et après que le choix a été fait une fonction est appelée pour diriger l'utilisateur vers un autre menu.

5. *Voir sa liste d'amis*
   Mise en pratique : Après avoir choisi l'option, le nom de tout les 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 tout ses amis. Une fois que le nom des amis sont reçus du coté client, la méthode *init_friendsw* est utilisée pour afficher le nom de tout ses amis sur l'écran.

6. *Envoyer une demande d'ami*
   Mise en pratique : Après avoir choisi l'option, l'utilisateur est invité à rentrer un nom ou à rentrer une commande pour retourner au menu des amis.
   Protocole : Si l'utilisateur ne rentre pas la commande pour quitter alors un message est envoyé au serveur qui le traitera et renverra le résultat de l'opération. Plusieurs résultat sont possibles, soit la demande a été envoyée soit 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.

7. *Retirer un ami*
   Mise en pratique : Après avoir choisi l'option, l'utilisateur est invité à rentrer un nom ou à rentrer une commande pour retourner au menu des amis.
   Protocole : Si l'utilisateur ne rentre pas la commande pour quitter 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.

8. *Accepter ou refuser une demande d'ami*
   Mise en pratique : Après avoir choisi l'option, l'utilisateur est invité à rentrer une commande suivi d'un nom ou à rentrer une commande pour retourner au menu des amis.
   Protocole : Si l'utilisateur ne rentre pas la commande pour quitter 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 le nom de la personne entrée n'a envoyé aucune demande d'ami.

9. *Annuler sa demande d'ami*
   Mise en pratique : Après avoir choisi l'option, l'utilisateur est invité à rentrer un nom ou à rentrer une commande pour retourner au menu des amis.
   Protocole : Si l'utilisateur ne rentre pas la commande pour quitter 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 annulée soit le nom de la personne entré est incorrect.

10. *Choix d'option du menu Statistique*
    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.

11. *Choix d'option du menu Classement général*
    Mise en pratique : L'utilisateur a le choix entre 5 options : *Classic*, *Dark*, *Horde*, *Alice* et *Retour*.
    Protocole : Si l'utilisateur choisit une des 4 premières options alors un messages est envoyé au serveur qui renverra le nom et statistiques des 10 premiers joueurs du mode sélectionné. Ces informations seront ensuite affichées dans un rectangle. Sinon l'utilisateur revient au menu principal.

12. *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.
    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 envoyé correctement, soit la commande a été mal entrée et dans ce cas aucun message n'est envoyé.

7 Annexe