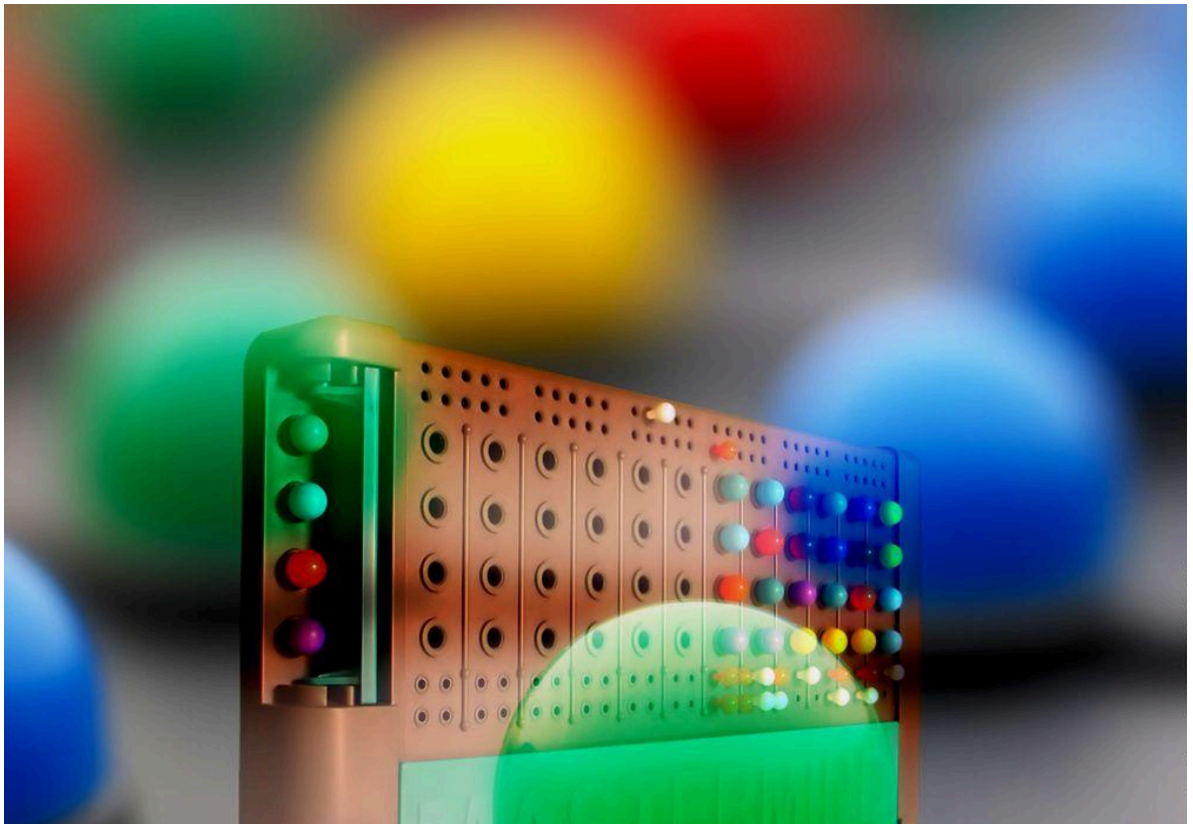
centralelille
IG2I

PROJET PRS

Création d'un mastermind multijoueur



Martin Ducoulombier
Delphine Chichery

Encadré par Mme Le Glaz

centralelille
IG2I

INTRODUCTION

Dans le cadre de notre parcours académique en PRS, nous entreprenons un projet de développement visant à appliquer concrètement les connaissances et compétences acquises en cours. Notre objectif est de concevoir une version repensée du jeu Mastermind, adaptée au jeu en réseau et à la collaboration entre plusieurs joueurs.

Ce projet constitue une occasion idéale de mettre en pratique les concepts théoriques abordés en cours, notamment en matière de programmation système et de communication inter processus. En développant cette version multijoueur du Mastermind, nous cherchons à approfondir notre compréhension des processus de communication, de la gestion des ressources partagées et de la synchronisation des actions entre les différents participants.

Notre jeu permettra à un groupe de 1 à 4 joueurs de se connecter à un serveur centralisé pour participer à des parties simultanées. La collaboration entre les joueurs et la compétition pour résoudre la combinaison secrète de couleurs apporteront une dimension sociale et ludique à l'expérience de jeu.

Chaque joueur sera confronté au défi de décoder la combinaison secrète de couleurs générée aléatoirement, en utilisant les indices fournis par le système. Avec une limite de 12 essais, l'objectif sera de trouver la combinaison secrète avec le moins de tentatives possible et dans un temps imparti, ajoutant ainsi un élément de suspense et de défi à chaque partie.

Ainsi, afin de structurer notre projet de PRS dans ce rapport intermédiaire, nous allons détailler dans un premier temps le contexte et les objectifs de notre projet, puis nous exposerons en détail les cas d'utilisation du jeu Mastermind multijoueur. Ensuite, nous décrirons les spécifications fonctionnelles nécessaires à son développement, ainsi que les solutions techniques envisagées pour sa réalisation.

github : https://github.com/NexxFire/Mastermind-PRS

CHAPITRE 1 : PRÉSENTATION DES CAS D'UTILISATION

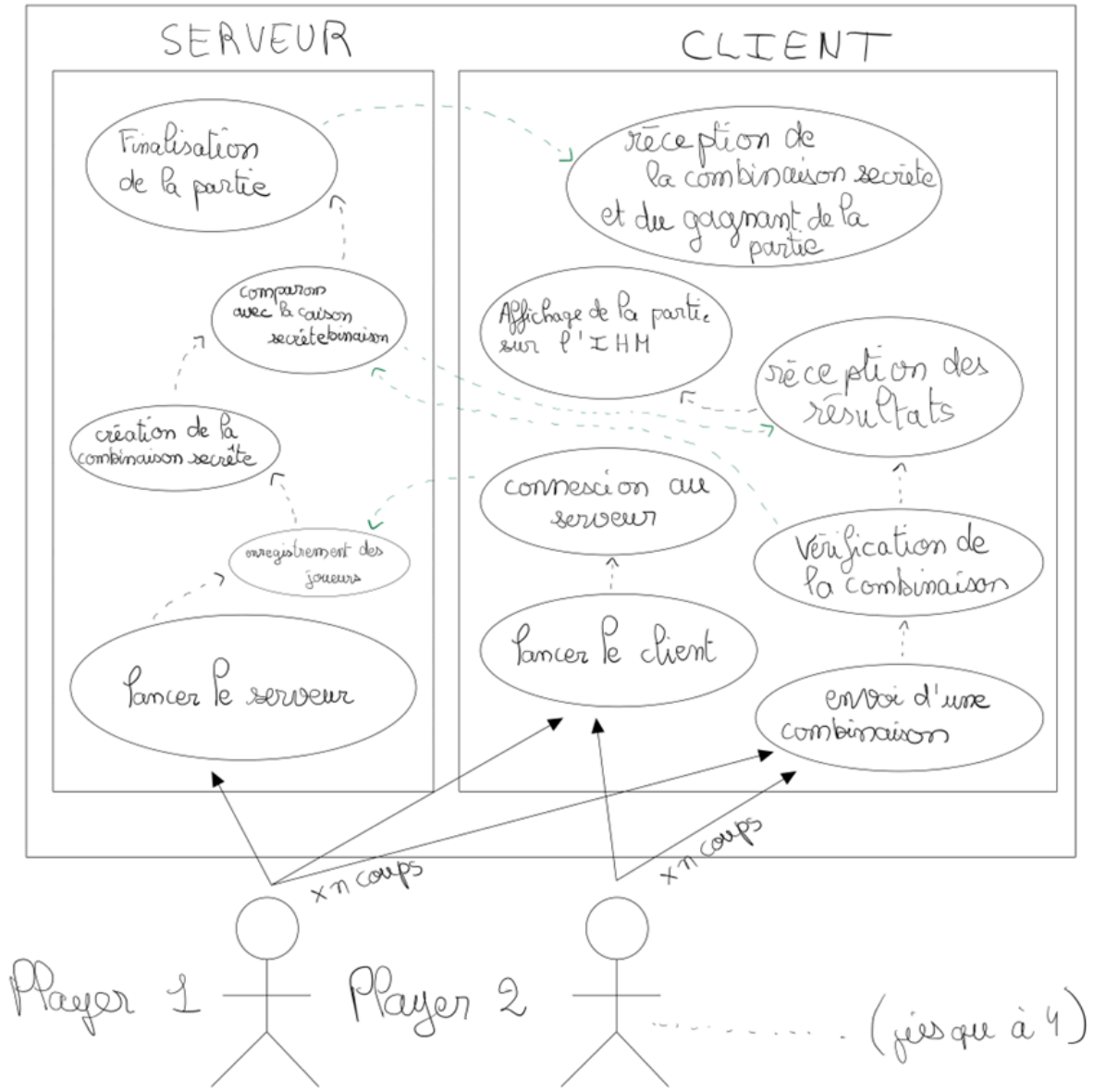


Figure 1 : diagramme des cas d'utilisation

1.1 Explication du Diagramme UML

Le diagramme UML des Cas d'Utilisation ci-dessus présente les interactions entre les différents acteurs du système, à savoir les clients et le serveur, ainsi que les fonctionnalités principales du jeu. Nous avons modélisé ici deux clients, qui représentent deux joueurs sur une même partie. Le maximum de clients pouvant se connecter sur cette partie est de 4.

Les flèches continues représentent les fonctions principales, tandis que les flèches pointillées représentent les fonctions secondaires. Les fonctions principales établissent directement la liaison entre les actions de l'utilisateur et le système. Les fonctions secondaires découlent des fonctions principales, étant créées après l'interaction principale entre le client et le système. Les flèches vertes en pointillées représentent les flux de données échangé par les différente partie de l'application

1.2 Présentation des cas d'utilisations

1) Lancement et connexion au Serveur :

Lorsque le client numéro 1 souhaite jouer une partie, il lance le serveur puis se connecte à celui-ci automatiquement en lançant le client.

2) Préparation de la Partie :

Une fois connecté au serveur, les clients peuvent indiquer qu'ils sont prêts à commencer la partie. Le démarrage de la partie n'a lieu que lorsque tous les joueurs connectés sont prêts.

Le serveur communique ensuite aux clients des informations utiles à la partie puis crée ensuite la combinaison secrète.

3) Proposition de Combinaison :

Les clients proposent des combinaisons de couleurs pendant le déroulement de la partie en saisissant leur proposition via l'IHM.
Une fois la proposition effectuée, le client l'envoie au serveur pour évaluation.

4) Réception et Traitement de la Proposition :

Le serveur reçoit les propositions de combinaison des clients, les évalue par rapport à la combinaison secrète, et envoie les résultats au client pour affichage.
Les résultats incluent le nombre de bonnes couleurs à la bonne place et le nombre de bonnes couleurs à la mauvaise place.

5) Fin de la Partie :

Le serveur détecte la fin de la partie, soit lorsque l'un des joueurs a trouvé la combinaison secrète, soit lorsque le nombre maximum de tentatives est atteint pour tous les joueurs.
À la fin de la partie, le serveur annonce le vainqueur et la combinaison secrète. Il réinitialise ensuite le jeu pour une nouvelle partie.

1.3 Gestion du projet

Planning actuel et prévisionnel

Pour mener à bien notre projet de jeu Mastermind multijoueur, nous avons élaboré un planning prévisionnel et un planning actuel, chacun détaillant les différentes phases du projet et leur répartition sur les semaines.

Actuel	AVRIL		MAI				JUIN	
	SEMAINE 16	SEMAINE 17	SEMAINE 18	SEMAINE 19	SEMAINE 20	SEMAINE 21	SEMAINE 22	SEMAINE 23
ANALYSE	2h							
SPECIFICATIONS FONCTIONNELLES	2h	2h						
CONCEPTION TECHNIQUE		4h	4h					
REALISATION				10h	10h	5h		
VALIDATION						2h		
DOCUMENTATION	2h	2h	2h			4h	4h	1h
PRESENTATION-RENDU FINAL								1h

Previsionnel	AVRIL		MAI				JUIN	
	SEMAINE 16	SEMAINE 17	SEMAINE 18	SEMAINE 19	SEMAINE 20	SEMAINE 21	SEMAINE 22	SEMAINE 23
ANALYSE	4h							
SPECIFICATIONS FONCTIONNELLES	2h	2h						
CONCEPTION TECHNIQUE		4h	4h					
REALISATION				10h	10h	10h	heures +	
VALIDATION							2h	
DOCUMENTATION	2h	2h	2h			2h	2h	2h
PRESENTATION-RENDU FINAL								1h

Figure 2 : Planning actuel et prévisionnel du projet

Le planning prévisionnel a été établi en début de projet pour structurer et organiser les différentes tâches à accomplir. Voici les principales phases définies dans ce planning :

- Analyse : Pendant cette phase, nous avons consacré 2 heures au choix du projet et à vérifier s'il correspondait aux spécifications du cours. Cette étape était cruciale pour assurer que notre projet répondrait aux exigences académiques et techniques.

- Spécifications Fonctionnelles : Nous avons passé 2 heures par semaine à vérifier si le choix du projet permettait l'utilisation efficace de concepts tels que les mutex, les boîtes aux lettres et les signaux. Cette vérification nous a aidés à valider l'intérêt et la faisabilité de notre projet.

- Conception Technique : La conception technique a été réalisée sur deux semaines avec 4 heures allouées chaque semaine pour développer les premières idées de création de structures et de fonctions, ainsi que pour planifier la gestion du code.

- Réalisation : La phase de réalisation, qui est la plus intensive, s'est étalée sur quatre semaines avec des sessions de 10 heures par semaine, et une extension en semaine 22 pour couvrir les heures supplémentaires nécessaires.

- Validation : Deux semaines ont été prévues pour la validation du projet avec des sessions de 2 heures par semaine. Cette phase a consisté à vérifier que le projet répondait à toutes les spécifications et qu'il était pleinement fonctionnel.

- Documentation : La documentation a été réalisée sur trois semaines, avec 2 heures allouées chaque semaine pour s'assurer que toutes les étapes et les décisions du projet étaient bien documentées.

- Présentation et rendu final : Correspond à la date finale du rendu du projet.

Le planning actuel reflète l'exécution réelle du projet, avec les ajustements apportés en cours de route pour répondre aux imprévus et aux besoins émergents. En résumé, le planning actuel est similaire en tout point au planning prévisionnel. Cependant, nous avons été plus performants lors de la phase de réalisation, réussissant à compléter les tâches plus rapidement que prévu. Cela nous a permis de consacrer du temps supplémentaire à la validation et à la documentation, assurant ainsi une meilleure qualité du projet final.

Répartition des Tâches et Collaboration

Tout au long du projet, nous avons appris à gérer efficacement notre temps et à répartir les tâches entre nous. Cette gestion de projet nous a permis de :

- Planifier et organiser les différentes phases du projet.
- Répartir les tâches en fonction des compétences de chacun.
- Collaborer de manière étroite pour respecter les délais.
- Adapter nos plans en fonction des imprévus et des besoins émergents.

CHAPITRE 2 : SPÉCIFICATIONS FONCTIONNELLES

2.1 Création de la partie :

Les joueurs doivent se connecter au serveur, lorsqu'ils cochent qu'ils sont prêts, la partie démarre. Lorsqu'un seul joueur est connecté au serveur, il peut faire une partie seul. Lorsque plusieurs joueurs sont connectés, il faut attendre que tous les joueurs mentionnent qu'ils soient prêts pour démarrer la partie.

2.2 Interface de jeu :

La partie commence lors du lancement du client et du serveur. Une fois connecté au serveur, le client envoie au serveur qu'il est "ready", c'est-à-dire qu'il est prêt à commencer

la partie. L'interface signale ensuite au client qu'il attend la réponse des autres joueurs connectés au serveur avant de démarrer la partie. L'interface informe le nombre de joueurs dans la partie et l'identifiant de celui-ci.

Une fois la partie lancée, l'interface utilisateur est créée sur le terminal de la façon suivante :

Figure 3 : IHM au début de la partie

Les 12 lignes de 4 carrés représentent l'espace pour chaque tentative du joueur. Lorsque le joueur propose une combinaison de quatres caractères, il est mémorisé et inséré sur la première ligne. Les deux colonnes à côté représentent les indices pour le joueur : la première indique le nombre de couleurs correctes et **bien** placées, présentes dans la combinaison proposée par le joueur. La seconde indique le nombre de couleurs correctes et **mal** placées de la combinaison.

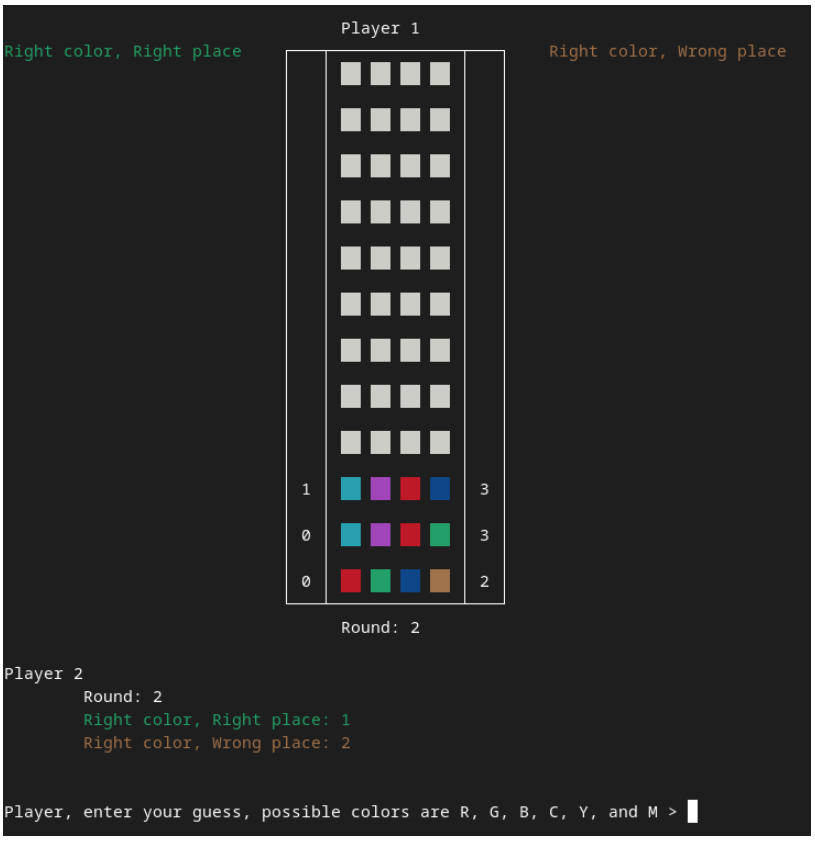


Figure 4 : IHM durant la partie

Le tableau en bas du terminal montre les performances des autres joueurs actualisées à chaque tour, avec leur nombre de tentatives effectuées, le nombre de couleurs trouvées, et le nombre de couleurs situées à la bonne position. Ainsi, le joueur peut déterminer son positionnement par rapport aux autres joueurs et accélérer ou non son jeu. Le but étant de trouver la combinaison le plus rapidement possible.

Les joueurs interagissent avec le jeu en utilisant des commandes simples basées sur l'entrée en majuscule des premières lettres de chaque couleur en anglais. Par exemple, pour proposer une combinaison de couleurs rouge, bleu, jaune et vert, le joueur entre simplement "RBYG". Le jeu reconnaît automatiquement ces commandes et les traite en conséquence.

2.3 Client :

Le client est chargé de l'IHM, et ainsi de la gestion des entrées et sorties de l'utilisateur. Il vérifie aussi que la combinaison proposée par l'utilisateur est valide avant de l'envoyer au serveur. À la fin de la partie, le client se ferme automatiquement, on peut donc relancer le programme pour jouer une autre partie.

2.4 Serveur :

Le serveur quant à lui va prendre en charge la synchronisation avec tous les clients, au début de la partie, il génère le code secret et attend que les clients lui communique les coups choisis par l'utilisateur. Après traitement des coups choisi par l'utilisateur, il renvoie les résultats aux clients. À la fin de la partie, le serveur envoie aux clients le vainqueur et se réinitialise en vue de démarrer une nouvelle partie.

Gestion des parties :

Le serveur gère la création, la gestion et la clôture des parties une fois terminées. Le programme aura l'implémentation des règles du jeu du Mastermind, y compris la génération aléatoire de la combinaison secrète et la validation des coups des joueurs.

La partie se termine soit lorsqu'un joueur a trouvé la combinaison secrète, soit lorsque tous les joueurs ont atteint le maximum de 12 tentatives. A la fin de la partie, le numéro du gagnant est affiché ainsi que la combinaison secrète.

CHAPITRE 3 : SPÉCIFICATIONS TECHNIQUES

3.1 Technologies Utilisées

Langage de programmation :

Nous avons choisi d'utiliser le langage C pour le développement de notre jeu Mastermind multijoueur. Ce choix a été motivé par plusieurs raisons :

Performance et contrôle : Le langage C offre un contrôle précis sur la gestion des ressources système, ce qui est essentiel pour optimiser les performances de notre application et assurer une expérience utilisateur fluide.

Interopérabilité avec les IPC : Le langage C est bien adapté à l'utilisation des mécanismes de communication inter-processus (IPC) tels que les boîtes aux lettres et les signaux, ce qui nous permettra de mettre en place facilement la communication entre les différents processus du jeu.

Compatibilité et portabilité : Le langage C est largement utilisé dans le développement système et est pris en charge par la plupart des systèmes d'exploitation, ce qui garantit une bonne compatibilité et portabilité de notre application sur différentes plateformes.

Gestion des connexions :

Nous avons mis en place la communication entre les clients et le serveur en utilisant des boîtes aux lettres et des signaux. Chaque client et le serveur disposent de leur propre boîte aux lettres pour recevoir les messages des autres processus. Lorsqu'un client envoie une proposition de combinaison au serveur, il envoie un signal au serveur pour lui indiquer qu'un message est disponible dans sa boîte aux lettres. Le serveur vérifie régulièrement les boîtes aux lettres des clients pour recevoir les propositions et envoyer les résultats aux clients.

Gestion de l'interface utilisateur :

Pour l'interface utilisateur, nous avons conçu une interface simple et intuitive permettant aux joueurs d'interagir facilement avec le jeu. Nous utilisons le terminal comme support d'affichage et d'interaction, garantissant une portabilité maximale sur différentes plateformes. Voici comment nous gérons l'interface utilisateur :

Interface de proposition de combinaison : Chaque joueur peut proposer une combinaison de couleurs en saisissant les lettres correspondant aux couleurs dans le terminal. Par exemple, "RBYG" représente une proposition avec les couleurs rouge, bleu, jaune et vert. Les joueurs utilisent des commandes simples basées sur l'entrée en majuscule des premières lettres de chaque couleur en anglais.

Affichage des résultats : Après avoir proposé une combinaison, le joueur reçoit un retour du serveur sur le nombre de couleurs correctes à la bonne place et à la mauvaise place. Ces résultats sont affichés dans la fenêtre du terminal du joueur pour lui permettre d'ajuster ses futures propositions.

Affichage des performances des autres joueurs : Le terminal affiche également les performances des autres joueurs à chaque tour, y compris le nombre de tentatives effectuées, le nombre de couleurs correctes trouvées et le nombre de couleurs correctes à la bonne position. Cela permet à chaque joueur de suivre le progrès des autres participants et d'adapter sa stratégie en conséquence.

Architecture Client-serveur :

Le jeu suit une architecture client-serveur, où un serveur centralisé gère la logique du jeu et la communication entre les joueurs. Le client est uniquement chargé de l’IHM.

3.2 Protocole applicatif : diagramme de séquences des requêtes/réponses

Dans ce protocole, tous les envois de données sont suivis d’un acquittement avec un code de validation spécifique à chaque envoie, cela permet de garantir une bonne synchronisation client-serveur.

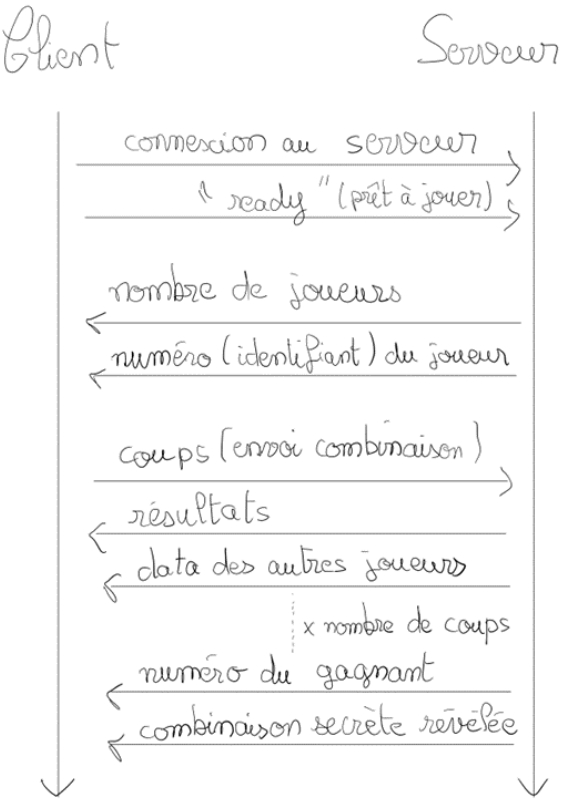


Figure 5 : Diagramme de séquence, requêtes/réponses

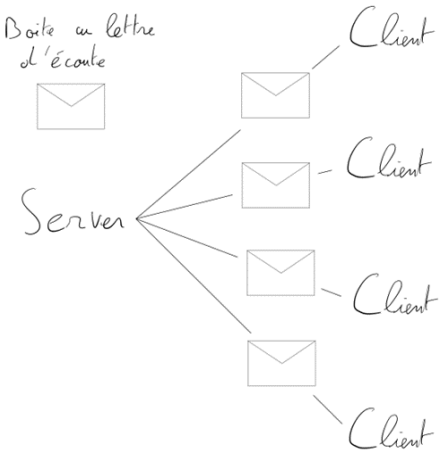


Figure 6 : Schéma d'architecture des boites au lettre

3.3 Description fonctionnelle des traitements des requêtes

1) Traitement de la connexion des clients au serveur :

- Le serveur écoute les demandes de connexion entrantes des clients.
- Lorsqu'un client se connecte, le serveur lui attribue un identifiant unique pour la session.
- Le serveur envoie le nombre de joueur et son numéro d'identifiant (numéro de joueur)
- Le serveur crée la combinaison secrète et la garde en mémoire tout au long de la partie.

2) Analyse des propositions de combinaison des joueurs :

- Le serveur reçoit les propositions de combinaison des joueurs.
- Il analyse les combinaisons proposées et les compare à la combinaison secrète : Le serveur calcule le nombre de bonne couleur et de bonne position dans la combinaison pour déterminer les indices à renvoyer aux joueurs.
- Le serveur arrête la partie d'un joueur au bout de 12 tentatives.

3) Mise à jour de l'interface utilisateur des joueurs avec les informations reçues du serveur :

- Après chaque action, le serveur envoie des mises à jour aux clients. Les mises à jour incluent les résultats des autres joueurs.

CHAPITRE 4 : SOLUTIONS TECHNIQUES

4.1 Schéma d'architecture logicielle

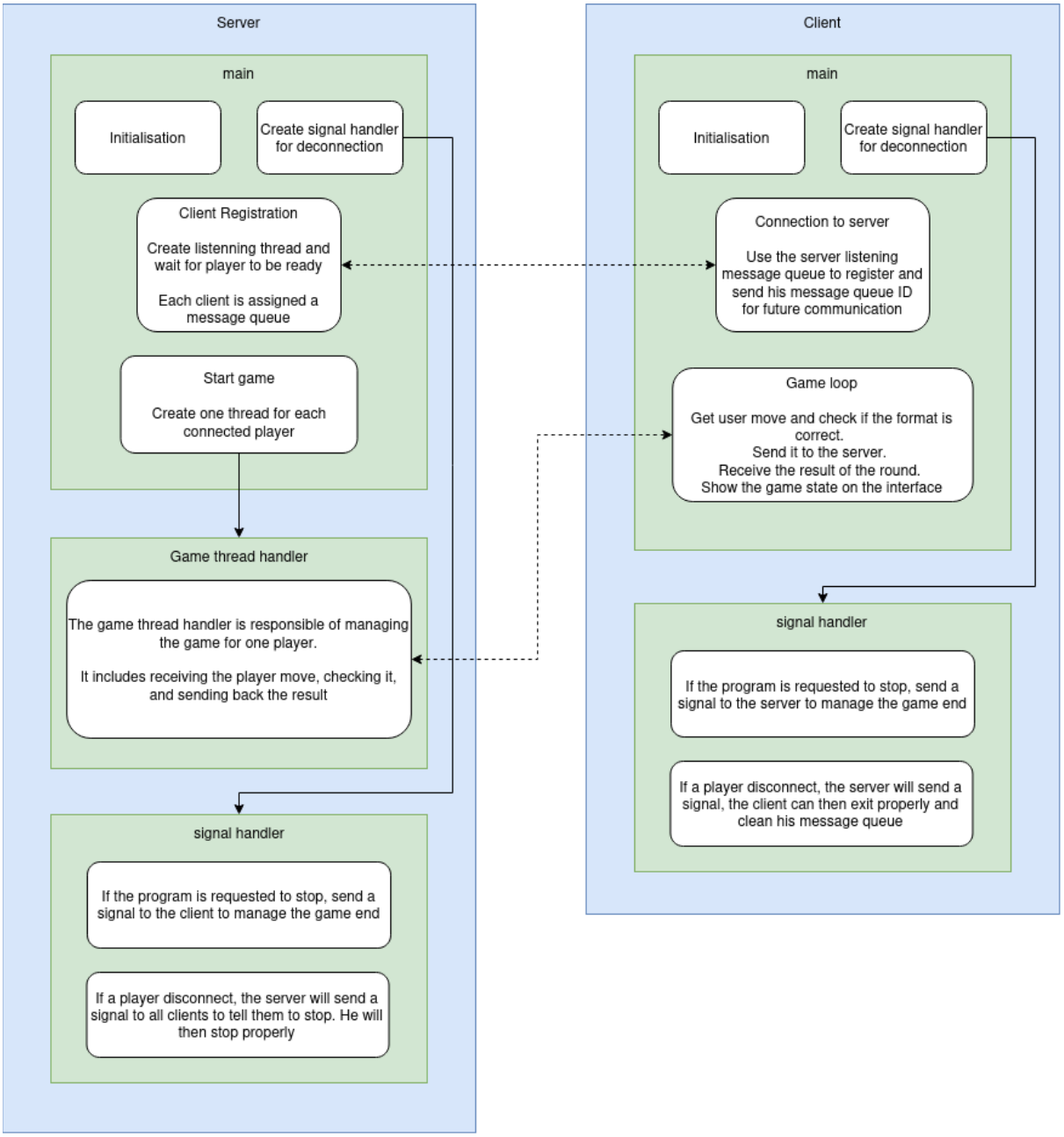


Figure 7 : Schéma d'architecture logicielle

4.2 Structure de données utilisées

Client :

Les structures de données côté client sont assez simples. Une structure *game* enveloppera toutes les données nécessaires à la partie, soit le plateau, les résultats de chaque tour, ainsi que d'autres champs utiles à la gestion de la partie.

En plus de cette structure, nous avons défini une autre structure *otherPlayer*, qui permet de stocker toute les informations des autres joueurs de la partie. Cette structure sera aussi comprise dans la structure game

Server :

Pour le serveur, il y a un peu plus d'informations à stocker car nous avons décidé de réunir les informations des 4 joueurs dans une seule structure de donnée pour simplifier les communications entre les threads. En faisant cela, nous exploitons la mémoire partagée des threads pour écrire et lire dans la structure de jeu afin de partager les informations nécessaires aux threads

Nous avons donc défini les structure *gameData*, *playerList* et *player*

gameData est la structure principale du serveur, elle contient toutes les informations nécessaires pour le fonctionnement du serveur et des threads gérant les clients.

player contient les informations d'un joueur de la même manière que dans la structure de donnée *game* coté client, ce qui permet une communication plus efficace entre le client et le serveur.

playerList sert uniquement à répertorier une liste de 4 joueurs maximum et le nombre de joueurs. Elle sert plutôt d'intermédiaire entre les deux structures pour rendre les données plus claires.

4.3 Architecture multi threads

Pour simplifier les traitements réalisés par le serveur, nous avons adopté une architecture multi threads pour différentes parties du jeu, notamment pour l'enregistrement et la gestion des clients durant la partie.

Enregistrement client :

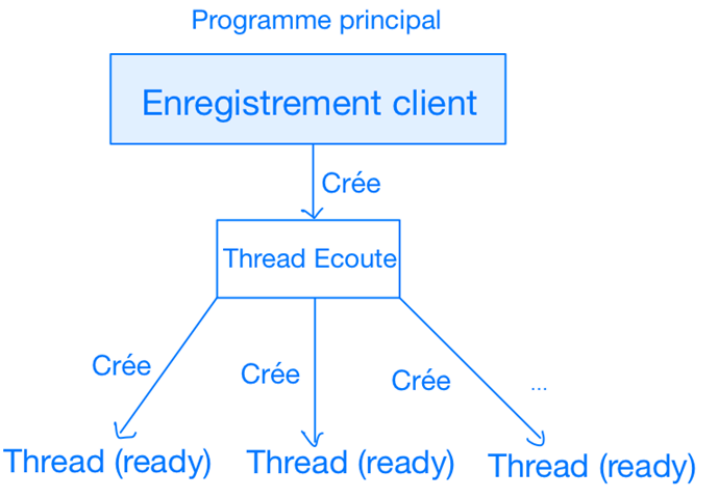


Figure 8 : Schéma d'architecture multi threads pour l'enregistrement de client

Lors de la réception d'une demande de connexion par le thread écoute, le serveur enregistre le client dans le système et crée un thread ready dédié au client qui vient de se connecter. Ce thread a pour rôle d'attendre un message du client indiquant qu'il est prêt à commencer la partie. La partie peut commencer uniquement si tous les threads ready sont fermés.

Gestion de la partie :

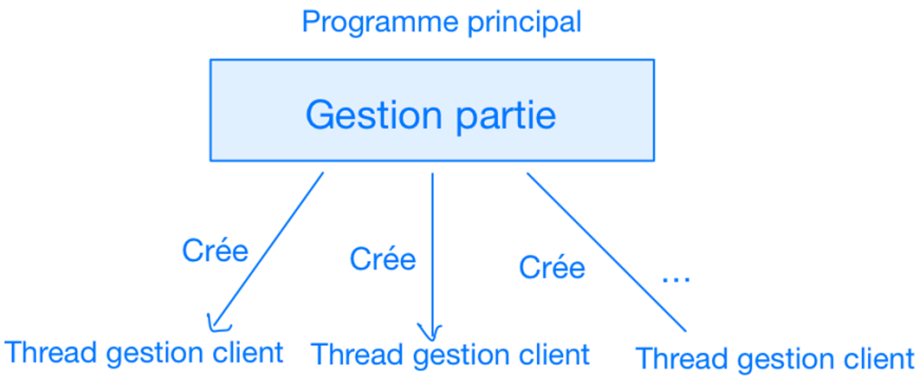


Figure 9 : Schéma d'architecture multi threads pour la gestion de la partie

Afin de gérer plusieurs clients de manière simultanée, on crée un thread côté serveur pour gérer chaque client. Cela nous permet de simplifier les traitements côté serveur en ayant pour chaque joueur un morceau de programme dédié.

4.4 Signaux et fermeture du programme

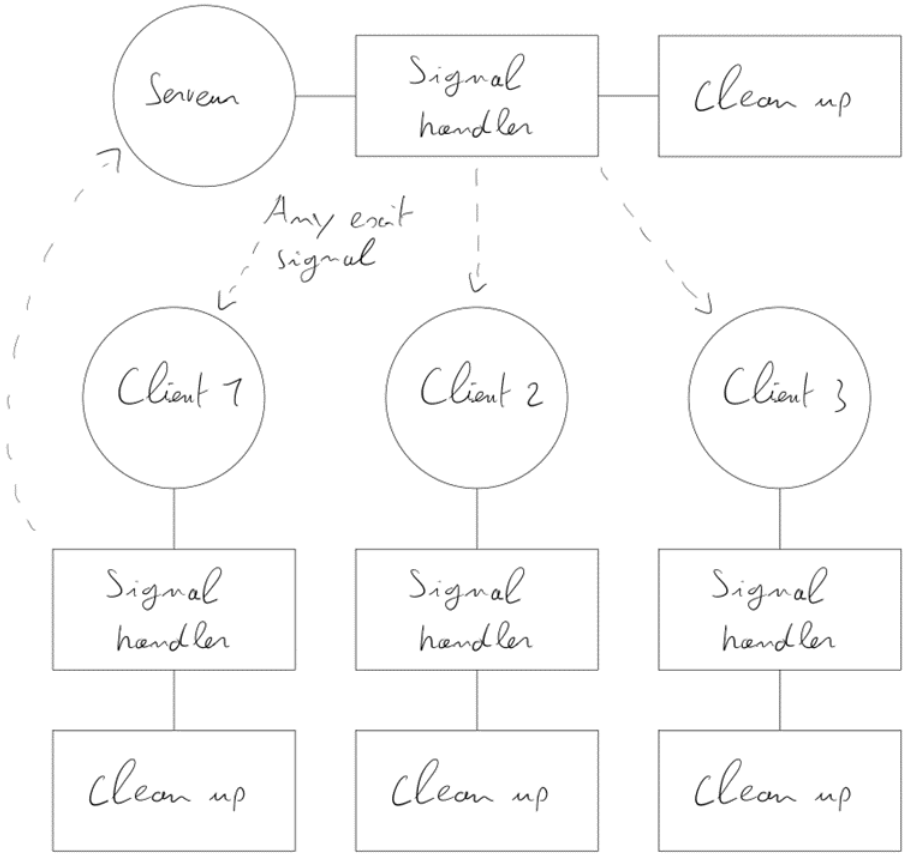


Figure 10 : Architecture du gestionnaire de signal

L'objectif ici est de mettre en place un gestionnaire de signaux qui permet de fermer le programme proprement et d'avertir le reste des utilisateurs et le serveur en cas de fermeture. La déconnexion d'un client ou du serveur entraîne donc la fermeture de tous les autres utilisateurs pour éviter un manque de cohérence sur l'affichage des autres clients.

CHAPITRE 5 : BILAN

5.1 Conclusion sur l'Avancée du Projet

Nous avons terminé et rendu fonctionnel notre projet de jeu Mastermind multijoueur, en respectant les spécifications techniques demandées. Nous avons utilisé l'architecture multi threads pour gérer les différentes tâches simultanées, les mutex pour assurer la gestion des ressources critiques et les signaux pour les interruptions et les arrêts du programme. La communication entre les clients et le serveur a été mise en place en utilisant des boîtes aux lettres et des signaux. Chaque client et le serveur ont leur propre boîte aux lettres pour recevoir les messages des autres processus. Le serveur vérifie régulièrement les boîtes aux lettres des clients pour recevoir les propositions et envoyer les résultats aux clients.

Pour l'interface utilisateur, nous avons conçu une interface simple et intuitive permettant aux joueurs d'interagir facilement avec le jeu. Le terminal est utilisé comme support d'affichage et d'interaction, garantissant une portabilité maximale sur différentes plateformes. Les fonctions fondamentales nécessaires au fonctionnement du projet, telles que la gestion des entrées utilisateurs et la communication, ont été créées et intégrées. Les signaux et les mutex ont été mis en place pour gérer les interruptions et les arrêts du programme ainsi que pour assurer une synchronisation et une protection efficaces des sections critiques du code.

5.2 Idées d'améliorations

Bien que le projet soit fonctionnel, nous avons identifié plusieurs points d'amélioration pour enrichir l'expérience utilisateur et augmenter la compétitivité du jeu. Tout d'abord, nous pourrions améliorer l'interface utilisateur en utilisant des bibliothèques telles que ncurses pour offrir une interface graphique plus riche et interactive. Cela rendrait l'expérience utilisateur plus agréable et intuitive.

Ensuite, nous pourrions implémenter la fonctionnalité de jouer plusieurs parties en même temps sur le même serveur, permettant à différents groupes de joueurs de participer à des sessions de jeu distinctes simultanément. De plus, il serait bénéfique d'ajouter la possibilité de relancer une partie automatiquement avec les mêmes joueurs une fois qu'une partie est terminée. Cela maintiendrait l'engagement des joueurs et faciliterait la continuité des sessions de jeu.

Enfin, nous pourrions mettre en place un système de points pour suivre les scores entre les parties. Ce système ajouterait un élément de compétition supplémentaire. Les scores pourraient être cumulés et affichés sur un tableau de classement, augmentant ainsi l'engagement des joueurs.

5.3 Notre point de vue

Nous sommes satisfaits des progrès réalisés dans le développement de notre jeu Mastermind multijoueur. Le projet s'est révélé être à la fois ludique et stimulant, nous permettant de mettre en pratique les notions théoriques vues en cours. L'utilisation des concepts de programmation système, de communication inter-processus et de gestion des ressources partagées a été essentielle pour réaliser ce projet. Cette expérience nous a permis d'approfondir notre compréhension et notre maîtrise de ces sujets. En plus des aspects techniques, nous avons également amélioré notre gestion de projet en apprenant à répartir les tâches efficacement, à collaborer étroitement et à respecter les délais pour créer un jeu fonctionnel et agréable à jouer.