

Rapport projet système et réseaux

Khmarou Fatima-Ezzahra
Abounaim Yassine

2024-2025

1 Introduction

Le projet présenté dans ce rapport consiste à développer une version en langage C d'un jeu inspiré de The Mind. Ce jeu collaboratif repose sur la synchronisation et l'intuition des joueurs, qui doivent coopérer pour atteindre un objectif commun sans communiquer directement.

Dans notre implémentation, le jeu s'appuie sur une architecture client-serveur, où les Clients représentent les joueurs, le Serveur coordonne les échanges, et le Robot simule un joueur automatisé ou effectue des tâches spécifiques dans le jeu. Ce projet recrée l'essence du gameplay de The Mind tout en explorant des concepts de programmation en réseau et en structuration.

L'objectif est de proposer une expérience de jeu interactive tout en respectant les contraintes techniques d'une application en C, en mettant l'accent sur la communication entre les différents composants. Cette introduction établit les bases avant d'explorer en détail la conception et la réalisation du projet.

2 Le gestionnaire du jeu

Le gestionnaire de jeu permet à plusieurs joueurs de se connecter au serveur grâce à une architecture basée sur des sockets TCP. Pour cela, la fonction `socket()` est utilisée pour créer des sockets, tandis que `bind()` associe chaque socket à une adresse IP et au port 4242. Le serveur est ensuite mis en mode d'écoute avec `listen()`, permettant d'attendre des connexions entrantes. Les clients sont acceptés via `accept()` et enregistrés dans la structure des joueurs. Chaque joueur connecté est géré dans un thread distinct grâce à `beginthread`, ce qui permet une gestion simultanée des interactions avec plusieurs joueurs en parallèle.

2.1 Communication entre le serveur et les joueurs

Fonctionnalité : Envoyer et recevoir des messages entre le serveur et les joueurs.
`send()` : Pour envoyer des messages aux clients, comme les cartes distribuées ou les résultats des manches.

recv() : Pour recevoir les messages des joueurs, notamment leurs réponses ("oui") ou leurs cartes jouées.

2.2 Validation des cartes jouées

Vérifier si les cartes jouées respectent l'ordre croissant pour valider la manche.

Tableaux (carteJeu[]) : Stocke les cartes jouées par les joueurs dans la manche.

Comparaison séquentielle dans une boucle : Vérifie si chaque carte est plus grande ou égale à la précédente.

```
bool verifOrdre()
{
    for (int i = 1; i < cartesJouees; i++)
    {
        if (carteJeu[i - 1] > carteJeu[i])
            return false;
    }
    return true;
}
```

2.3 Gestion des manches

Passer à la manche suivante en cas de réussite ou réinitialiser le jeu en cas d'échec.

Variables globales (manche, cartesJouees) : Suivi de l'état de progression du jeu.

Réinitialisation : Réinitialisation des cartes et compteurs via memset.

2.4 Gestion des threads pour les joueurs

Permettre à plusieurs joueurs d'interagir simultanément avec le serveur.

beginthread() : Création d'un thread par joueur pour gérer indépendamment les échanges avec chaque client.

2.5 Synchronisation et gestion des états

la synchronisation et la gestion des états des joueurs s'appuient sur des champs comme jeu dans la structure Joueur, ainsi que des boucles et des compteurs comme joueursPrets pour vérifier que le nombre minimal de joueurs prêts est atteint avant de commencer une manche. Ces fonctionnalités travaillent ensemble pour offrir une expérience de jeu fluide et interactive.

3 Joueur Humain

Le client implémenté dans notre programme permet à un joueur de se connecter à un serveur pour participer à un jeu en réseau. Une fois connecté, le client reçoit des cartes distribuées par le serveur, les affiche, puis permet au joueur de jouer les cartes une par une en les envoyant au serveur. Le jeu fonctionne sur un mode de manches, où le joueur doit jouer une carte à chaque tour. Si le joueur souhaite quitter, il peut le faire à tout moment en saisissant "exit". Le programme gère également la communication entre le client et le serveur en envoyant des messages et en recevant des réponses, tout en maintenant une interaction continue jusqu'à ce que le joueur décide de se déconnecter.

3.1 Connexion au serveur

Le programme initialise le réseau avec WSASStartup et crée un socket TCP (socket()). Il se connecte ensuite au serveur, qui est supposé être en écoute sur l'adresse IP 127.0.0.1 et le port défini (4242).
Si la connexion réussit, le client reçoit un message de bienvenue du serveur.

```
// Initialisation de Winsock
WSADATA wsaData;
if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0)
{
    printf("Erreur WSAStartup\n");
    return 1;
}

// Création du socket
descClient = socket(AF_INET, SOCK_STREAM, 0);
if (descClient == INVALID_SOCKET)
{
    printf("Erreur socket\n");
    WSACleanup();
    return 1;
}

addrServeur.sin_family = AF_INET;
addrServeur.sin_port = htons(PORT);
addrServeur.sin_addr.s_addr = inet_addr("127.0.0.1");
```

3.2 Réception des cartes (figure1)

Le serveur envoie une liste de cartes au client, qui les affiche à l'écran.
Le client compte le nombre de cartes reçues en décomposant la chaîne reçue en tokens (via strtok()).

3.3 Jouer des cartes (figure2)

Le joueur est invité à entrer une carte à jouer, qui est envoyée au serveur via send(). Une fois la carte jouée, le compteur de cartes diminue, et le joueur est informé du nombre restant.
Si l'utilisateur entre "exit", il quitte la partie.

figure1

```
// Découper la chaîne pour obtenir les cartes
char *token = strtok(buffer, " "); // Utilise un espace comme délimiteur

// Compter les cartes
while (token != NULL)
{
    // Incrémenter directement pour chaque token trouvé
    carte++;
    token = strtok(NULL, " ");
}
```

figure2

```
while (carte > 0)
{
    printf("Entrez une carte a jouer (ou 'exit' pour quitter) : ");
    fgets(buffer, sizeof(buffer), stdin);
    buffer[strcspn(buffer, "\n")] = '\0';

    if (strcmp(buffer, "exit") == 0)
    {
        printf("Deconnexion...\n");
        break;
    }

    send(descClient, buffer, strlen(buffer), 0);

    carte--;

    printf("Carte envoyée. Il reste %d cartes a jouer.\n", carte);
}
```

3.4 Boucle de manches

Après avoir joué toutes ses cartes pour une manche, le client attend une réponse du serveur (par exemple, "Manche gagnée" ou "Manche perdue").

Le jeu recommence avec une nouvelle manche ou se termine si l'utilisateur décide de quitter.

3.5 Déconnexion propre

Le programme gère la déconnexion proprement en fermant le socket client avec `closesocket` et nettoie les ressources réseau avec `WSACleanup`.

```
closesocket(descClient);
WSACleanup();
return 0;
}
```

4 Joueur Robot

Notre robot joueur est un système autonome conçu pour interagir avec un serveur de jeu de cartes dans un environnement client-serveur. Ce robot remplace un client manuel traditionnel en automatisant entièrement le processus de jeu, depuis la réception des cartes jusqu'à leur choix et leur envoi stratégique au serveur.

Voici quelques différences entre notre joueur humain et le joueur robot

4.1 Jouer les cartes

Les cartes sont jouées automatiquement à partir du tableau `carte[]`, en commençant par la plus petite après le tri.

Une fois une carte jouée, toutes les cartes restantes sont décalées à gauche pour maintenir un ordre cohérent.

Le robot insère un délai aléatoire (`Sleep(delai)`) entre les coups pour simuler un comportement humain.

```
while (c > 0)
{
    printf("Entrez une carte a jouer\n");
    srand(time(NULL));
    int delai = (rand() % 6 + 5) * 1000;
    Sleep(delai);

    snprintf(buffer, sizeof(buffer), "%d", carte[0]);

    printf("Carte jouee : %d\n", carte[0]);

    // Décaler les cartes restantes vers la gauche
    for (int i = 0; i < c - 1; i++)
    {
        carte[i] = carte[i + 1];
    }

    send(descRobot, buffer, strlen(buffer), 0);

    c--;

    printf("Carte envoyee. Il reste %d cartes a jouer.\n", c);
}
```

4.2 Structure du code

Plus modulaire et automatisé, avec l'utilisation de fonctions spécifiques comme `trierCarte` pour des tâches distinctes.

Les cartes sont manipulées à l'aide d'un tableau dynamique, ce qui permet un traitement programmatique.

```

void trierCarte(int carte[], int manche)
{
    for (int i = 0; i < manche - 1; i++)
    {
        for (int j = 0; j < manche - 1; j++)
            if (carte[j] > carte[j + 1])
            {
                int temp = carte[j];
                carte[j] = carte[j + 1];
                carte[j + 1] = temp;
            }
    }
}

```

5 Déroulement du jeu

Le jeu se déroule en plusieurs manches. Chaque joueur reçoit des cartes numérotées de 1 à 100 aléatoirement et doit les jouer dans l'ordre croissant, sans communiquer ni établir d'ordre de jeu. Celui qui pense avoir la plus petite carte la joue, suivi du prochain, et ainsi de suite. Si les joueurs réussissent, ils reçoivent une carte supplémentaire pour la manche suivante. Sinon, ils recommencent à la première manche.

Pour illustrer tout ça, on débute par un test par une partie jouée par deux joueurs humains :

5.1 Joueur1 et joueur2 (manche perdue):

Dans cette manche les joueurs ont joués leurs cartes, mais ce n'était pas dans l'ordre alors on affiche un message annonçant qu'ils ont perdus et on lance une nouvelle partie.

```

C:\Windows\System32\cmd.e x + v
Microsoft Windows [version 10.0.22631.4460]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\33760\Downloads>serveur0
En attente
Nouvelle connexion !
Nombre de joueurs connectes : 1
En attente
Nouvelle connexion !
Nombre de joueurs connectes : 2
En attente
Le jeu peut commencer !
Le joueur 1 recoit la carte : 8
Le joueur 2 recoit la carte : 96
Le joueur 2 a joue : 96
Le joueur 1 a joue : 8
Manche perdue !
Le joueur 1 recoit la carte : 95
Le joueur 2 recoit la carte : 63

20 clock_t tempsDebut, tempsFin;
21
22 void debutTemps()
23 {
24     tempsDebut = clock();
25 }
26
27 void finTemps()

```

```

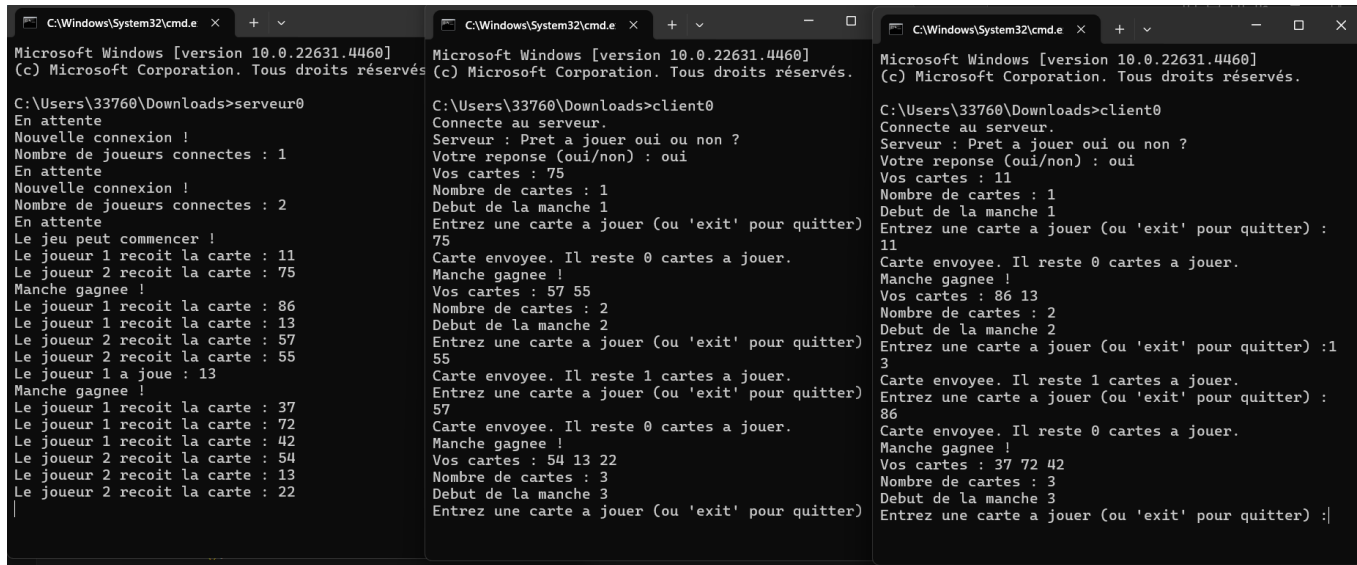
C:\Windows\System32\cmd.e x + v
Microsoft Windows [version 10.0.22631.4460]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\33760\Downloads>client0
Connecte au serveur.
Serveur : Pret a jouer oui ou non ?
Votre reponse (oui/non) : oui
Vos cartes : 96
Nombre de cartes : 1
Debut de la manche 1
Entrez une carte a jouer (ou 'exit' pour quitter) : 96
Carte envoyee. Il reste 0 cartes a jouer.
Manche perdue !
Vos cartes : 63
Nombre de cartes : 1
Debut de la manche 1
Entrez une carte a jouer (ou 'exit' pour quitter) : |

```

5.2 Joueur1 et joueur2 (manche gagnée):

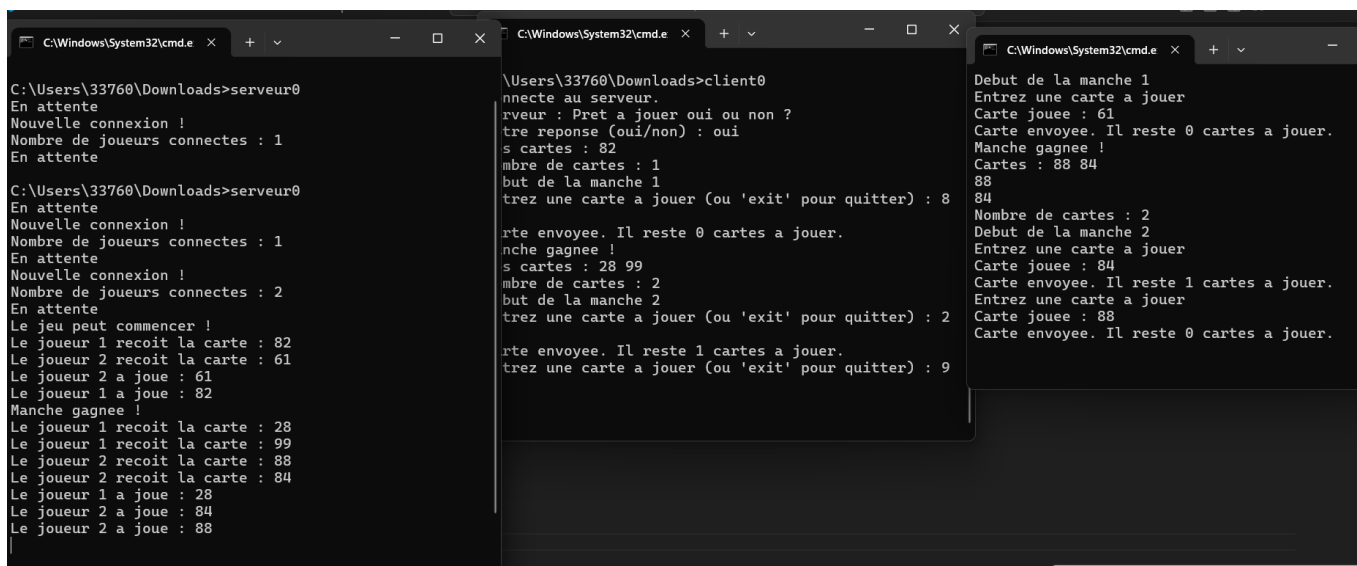
Contrairement à la première partie les joueurs ont posé les cartes dans le bon ordre du coup on a une manche gagnante , on affiche un message et on envoie les cartes pour la manche suivante .



The image shows three side-by-side Windows command prompt windows. The first window, titled 'C:\Windows\System32\cmd.e', shows the server (serveur0) output: 'En attente', 'Nouvelle connexion !', 'Nombre de joueurs connectes : 1', 'En attente', 'Nouvelle connexion !', 'Nombre de joueurs connectes : 2', 'En attente', 'Le jeu peut commencer !', 'Le joueur 1 recoit la carte : 11', 'Le joueur 2 recoit la carte : 75', 'Manche gagnée !', 'Le joueur 1 recoit la carte : 86', 'Le joueur 1 recoit la carte : 13', 'Le joueur 2 recoit la carte : 57', 'Le joueur 2 recoit la carte : 55', 'Le joueur 1 a joué : 13', 'Manche gagnée !', 'Le joueur 1 recoit la carte : 37', 'Le joueur 1 recoit la carte : 72', 'Le joueur 1 recoit la carte : 42', 'Le joueur 2 recoit la carte : 54', 'Le joueur 2 recoit la carte : 13', 'Le joueur 2 recoit la carte : 22'. The second window, titled 'C:\Windows\System32\cmd.e', shows the client (client0) output: 'Connecte au serveur.', 'Serveur : Pret a jouer oui ou non ?', 'Votre reponse (oui/non) : oui', 'Vos cartes : 75', 'Nombre de cartes : 1', 'Debut de la manche 1', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 75', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 57 55', 'Nombre de cartes : 2', 'Debut de la manche 2', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 55', 'Carte envoyee. Il reste 1 cartes a jouer.', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 57', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 54 13 22', 'Nombre de cartes : 3', 'Debut de la manche 3', 'Entrez une carte a jouer (ou \'exit\' pour quitter) :'. The third window, titled 'C:\Windows\System32\cmd.e', shows the client (client0) output: 'Connecte au serveur.', 'Serveur : Pret a jouer oui ou non ?', 'Votre reponse (oui/non) : oui', 'Vos cartes : 11', 'Nombre de cartes : 1', 'Debut de la manche 1', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 11', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 86 13', 'Nombre de cartes : 2', 'Debut de la manche 2', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 13', 'Carte envoyee. Il reste 1 cartes a jouer.', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 86', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 37 72 42', 'Nombre de cartes : 3', 'Debut de la manche 3', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 42'.

5.3 Joueur et robot:

Dans ce test nous avons un joueur qui joue avec un robot et ce dernier pratique le jeu d'une façon autonome capable de participer sans intervention humaine.



The image shows three side-by-side Windows command prompt windows. The first window, titled 'C:\Windows\System32\cmd.e', shows the server (serveur0) output: 'En attente', 'Nouvelle connexion !', 'Nombre de joueurs connectes : 1', 'En attente', 'Nouvelle connexion !', 'Nombre de joueurs connectes : 2', 'En attente', 'Le jeu peut commencer !', 'Le joueur 1 recoit la carte : 82', 'Le joueur 2 recoit la carte : 61', 'Le joueur 2 a joué : 61', 'Le joueur 1 a joué : 82', 'Manche gagnée !', 'Le joueur 1 recoit la carte : 28', 'Le joueur 1 recoit la carte : 99', 'Le joueur 2 recoit la carte : 88', 'Le joueur 2 recoit la carte : 84', 'Le joueur 1 a joué : 28', 'Le joueur 2 a joué : 84', 'Le joueur 2 a joué : 88'. The second window, titled 'C:\Windows\System32\cmd.e', shows the client (client0) output: 'Connecte au serveur.', 'Serveur : Pret a jouer oui ou non ?', 'Votre reponse (oui/non) : oui', 'Vos cartes : 82', 'Nombre de cartes : 1', 'Debut de la manche 1', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 8', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 28 99', 'Nombre de cartes : 2', 'Debut de la manche 2', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 2', 'Carte envoyee. Il reste 1 cartes a jouer.', 'Entrez une carte a jouer (ou \'exit\' pour quitter) : 9'. The third window, titled 'C:\Windows\System32\cmd.e', shows the client (client0) output: 'Debut de la manche 1', 'Entrez une carte a jouer', 'Carte jouée : 61', 'Carte envoyee. Il reste 0 cartes a jouer.', 'Manche gagnée !', 'Vos cartes : 88 84', 'Nombre de cartes : 2', 'Debut de la manche 2', 'Entrez une carte a jouer', 'Carte jouée : 84', 'Carte envoyee. Il reste 1 cartes a jouer.', 'Entrez une carte a jouer', 'Carte jouée : 88', 'Carte envoyee. Il reste 0 cartes a jouer.'.

6 Statistique du jeu

Pour les statistiques de notre jeu ,on a opté pour plusieurs options :

6.1 Gestion du temps d'une manche

On a utilisé une fonction `debutTemps()` qui démarre un chronomètre pour mesurer la durée d'une manche. Ensuite la fonction `finTemps()` qui calcule la durée écoulée depuis l'appel de `debutTemps` et l'écrit dans un fichier texte (fin de chaque manche). Le résultat est stocké dans un fichier `text` . `Resultatmanche()` enregistre dans ce fichier le résultat de la manche (gagnée , perdue) ainsi que toutes les cartes qui ont été jouées durant cette manche .

```
void debutTemps()
{
    tempsDebut = clock();
}

void finTemps()
{
    tempsFin = clock();
    int tempsReponse = (tempsFin - tempsDebut) * 1000 / CLOCKS_PER_SEC;

    FILE *fich = fopen("temps_manches.txt", "a");

    fprintf(fich, "La duree de la manche %d : %d ms\n", manche, tempsReponse);

    fclose(fich);
}

void resultatManche(bool res)
{
    FILE *fich = fopen("temps_manches.txt", "a");

    fprintf(fich, "Cartes jouees : \n");
    for (int i = 0; i < cartesJouees; i++)
    {
        fprintf(fich, "%d ", carteJeu[i]);
    }

    if (res)
    {
        fprintf(fich, "\nRésultat : Gagnée\n");
    }
    else
    {
        fprintf(fich, "\nRésultat : Perdue\n");
    }
}
```

Voici un fichier qui interprète les statistiques des manches qui ont été effectuées précédemment .

Cartes jouées :

96 8

Résultat : Perdue

La durée de la manche 1 : 56221 ms

Cartes jouées :

63 95

Résultat : Gagnée

La durée de la manche 1 : 56965 ms

Cartes jouées :

111 775

Résultat : Gagnée

La durée de la manche 1 : 23048 ms

Cartes jouées :

13 555 557 886

Résultat : Gagnée

La durée de la manche 2 : 77104 ms

Cartes jouées :

10 1

Résultat : Perdue

La durée de la manche 1 : 6130 ms

Cartes jouées :

30 50

Résultat : Gagnée

La durée de la manche 1 : 12448 ms

Cartes jouées :

61 82

Résultat : Gagnée

La durée de la manche 1 : 13894 ms