

# Rapport sur le jeu de dame

Wassim Saidane, Aurélien Authier

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Protocole choisis</b>	<b>1</b>
<b>3</b>	<b>Implémentation</b>	<b>1</b>
3.1	Serveur . . . . .	1
3.2	Client . . . . .	2
3.3	Le jeu . . . . .	2
3.3.1	Constante prédéfini . . . . .	2
3.3.2	Le pion . . . . .	2
3.3.3	La fonction start . . . . .	3
3.3.4	La fonction choisir_pion . . . . .	4
3.3.5	La fonction deplace_pion . . . . .	5
3.3.6	La fonction affiche_plateau . . . . .	6
3.3.7	La fonction game_over . . . . .	6
<b>4</b>	<b>Conclusion</b>	<b>6</b>

## 1 Introduction

Le but de ce projet était de modéliser un jeu de dame en réseau, où deux joueurs peuvent s'affronter, en respectant les règles du jeu, et peuvent être observé par une tierce personne. Nous devons donc mettre en place un serveur, un client, un protocole et, si le temps le permettait, une représentation graphique.

## 2 Protocole choisis

Nous avons choisi d'utiliser le protocole TCP car il garantit l'émission ainsi que la réception de paquets de données.

## 3 Implémentation

Nous avons repris le code du TP5 permettant de d'établir un tchat avec les clients.

### 3.1 Serveur

Le plateau et un objet de structure pion (voir 3.3.2) sont initialisés.

```
int sockfd, len, connfd[2], plateau[TAILLE * TAILLE];
Pion p;
```

On demande aux clients le pion qu'ils veulent déplacer.

```
char buf[100] = "Quel_pion_voulez_vous_deplacer_?_";
```

Le plateau est ensuite initialisé avec start, le message est envoyé au client et le serveur affiche le déplacement.

```
start(plateau);
while (1)
{
    for (int i = 0; i < 2; i++)
    {
        write(connfd[i], (const char *)&plateau, sizeof(plateau));
        printf("Le_Joueur_%d_deplace_un_piont\n", i + 1);
        write(connfd[i], (const char *)&buf, sizeof(buf));
        recv(connfd[i], &p, sizeof(p), 0);
        printf("Le_joueur_%d_a_deplacer_le_pion\n", i + 1, p.x, p.y);
        printf(".....:\\n_x:_%d\\n_y:_%d\\n", i + 1, p.x, p.y);
    }
}
```

## 3.2 Client

Nous initialisons un message de taille 100 et un pion (voir 3.3.2).

```
char message[100];
Pion p;
```

Le client va exécuter dans une boucle infini l’affichage du client.

```
while (1)
{
    recv(sockfd, &plateau, sizeof(plateau), 0);
    recv(sockfd, &message, sizeof(message), 0);
    printf("%s\n", message);
    p=choisir_pion(plateau);
    deplace_pion(p, plateau);
    write(sockfd, (const char *)&p, sizeof(p));
}
```

Remarque 1 :

Ici la boucle est infinie car nous avons pas terminé le sujet il aurait fallu créer une fonction `game_over()` [voir 3.3.7] qui renvoie un booléen si un joueurs n’a plus de pion.

## 3.3 Le jeu

### 3.3.1 Constante prédéfini

Pour afficher le plateau nous affichons des nombres simples.

```
#define J2_PION 2
#define J2_DAME 4
#define J1_PION 1
#define J1_DAME 3
#define VIDE 0
#define TAILLE 10
#define J2 0
#define J1 1
```

### 3.3.2 Le pion

Nous avons créer une structure pion.

```
typedef struct pion
{
    int x;
    int y;
} Pion;
```

Lorsqu’un pion devient dame il reste un pion.

### 3.3.3 La fonction start

La fonction start va initialiser le plateau qui est un tableau en 2 dimension en fonction de la taille. On initialise le plateau avec des cases vides puis on place les pions.

```
void start(int plateau[])
{

    for (int i = 0; i < TAILLE; i++)
    {
        for (int j = 0; j < TAILLE; j++)
        {
            plateau[i * TAILLE + j] = VIDE;
        }
    }

    for (int i = 0; i < 4; i++)
    {
        for (int j = (i + 1) % 2; j < TAILLE; j += 2)
        {
            plateau[i * TAILLE + j] = J1_PION;
        }
    }

    for (int i = TAILLE - 1; i > TAILLE - 5; i--)
    {
        for (int j = (i + 1) % 2; j < TAILLE; j += 2)
        {
            plateau[i * TAILLE + j] = J2_PION;
        }
    }
}
```

### 3.3.4 La fonction choisir\_pion

Cette fonction prend un plateau en paramètre et permet au joueur de choisir un pion avec les coordonnées x et y. La fonction retourne un pion.

```
Pion choisir_pion(int plateau[])
{
    affiche_plateau(plateau);
    printf("Choisissez_un_pion!\n");
    Pion p;
    printf("coordonnee_X:_");
    scanf("%d", &p.x);
    while (p.x < 0 || p.x > TAILLE)
    {
        printf("\nLa_coordonnee_x_n'est_pas_valide");
        scanf("%d", &p.x);
    }
    printf("coordonnee_Y:_");
    scanf("%d", &p.y);
    while (p.y < 0 || p.y > TAILLE)
    {
        printf("\nLa_coordonnee_y_n'est_pas_valide");
        scanf("%d", &p.y);
    }
    return p;
}
```

### 3.3.5 La fonction `deplace_pion`

La fonction prend un paramètre un pion et un plateau. On crée un pion temporaire afin de ne pas écraser le pion principale. Ensuite les coordonnées du pion sont effacées pour être remplacées par celles du pion temporaire (cette section de code est en commentaire car elle ne fonctionne pas).

```
void deplace_pion(Pion p, int plateau[]){
    Pion temp;
    printf("Saisir_nouvelle_coordonnee_X:_");
    scanf("%d", &temp.x);
    while (temp.x < 0 || temp.x > TAILLE)
    {
        printf("\nLa_coordonnee_x_n'est_pas_valide");
        scanf("%d", &temp.x);
    }
    printf("Saisir_nouvelle_coordonnee_Y:_");
    scanf("%d", &temp.y);
    while (temp.y < 0 || temp.y > TAILLE)
    {
        printf("\nLa_coordonnee_y_n'est_pas_valide");
        scanf("%d", &temp.y);
    }
    /* plateau[p.x][p.y]=VIDE;
    plateau[temp.x][temp.y]=J1_PION;*/
    affiche_plateau(plateau);
}
```

Remarque :

Nous aurions voulu faire la vérification du coup pour voir si il est réglementaire dans une fonction `verif_coup()` appelé dans cette fonction.

### 3.3.6 La fonction `affiche_plateau`

La fonction affiche le plateau

```
void affiche_plateau(int plateau[])
{
    indice(TAILLE);
    for (int i = 0; i < TAILLE; i++)
    {
        printf("%d_ ", i);
        for (int j = 0; j < TAILLE; j++)
        {
            printf("%d|", plateau[i * TAILLE + j]);
        }
        printf("\n");
    }
    printf("\n");
}
```

### 3.3.7 La fonction `game_over`

Nous avons malheureusement pas implémenter cette fonction.

```
Booleen game_over(int plateau[])
{
    if(nb_blanc <= 0 || nb_noir <= 0)
        return true;
    return false;
}
```

## 4 Conclusion

Le projet n'étant pas terminé, le code ne fonctionne que pour deux joueurs et le joueur peut seulement demander le pion qu'il souhaite déplacer et entrer ses nouvelles coordonnées