Rapport sur le jeu de damme

Wassim Saidane, Aurélien Authier

Table des matières

1 Introduction

2 Protocole choisis

Nous avons choisis d'utiliser le protocole TCP car 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 initialisé.

```
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 depalce 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 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 renvoit 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 simples nombres.

```
#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 dimenssion 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 paramétre et permet au 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 paramétre un pion et un plateau. On crée un pion temporaire afin de ne pas écraser le pion principale. Ensuite les coordonée du pion sont effacé pour être remplacé par le 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 coordonee 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 joueurs peut seulement demander le pion qu'il souhaite déplacer et entrez ces nouvelles coordonées