ProjetP4pse

Projet Puissance 4

Ce projet implémente un jeu de Puissance 4 en utilisant des sockets pour permettre la communication entre un serveur et plusieurs clients. Le jeu supporte les modes de jeu Humain vs Humain, Humain vs Robot et Robot vs Robot. Le niveau des robots est choisi par l'utilisateur.

Structure du Projet

```
.
├── client
│   ├── client
│   ├── client.c
│   ├── client.h
│   └── Makefile
├── game
│   ├── jeu.c
│   └── jeu.h
├── include
│   ├── dataspec.h
│   ├── datathread.h
│   ├── erreur.h
│   ├── ligne.h
│   ├── msgbox.h
│   ├── msg.h
│   ├── pse.h
│   └── resolv.h
├── lib
│   └── libpse.a
├── Makefile.inc
├── modules
│   ├── datathread.c
│   ├── erreur.c
│   ├── ligne.c
│   ├── Makefile
│   ├── msgbox.c
│   ├── msg.c
│   └── resolv.c
├── ordinateur
│   ├── ordinateur.c
│   └── ordinateur.h
├── packages.microsoft.gpg
├── README.md
└── serveur
    ├── Makefile
```

```
    ├── serveur.c
    └── serveur.h
```

Compilation

Pour compiler ce projet, utilisez `gcc` :

1. Dans un premier terminal serveur, se placer dans le dossier serveur. Pour exécuter le fichier, utilisez :

   ```
   make
   ```

2. Dans un premier terminal client, puis un second, se placer dans le dossier client. Pour exécuter le fichier, utilisez:

   ```
   make
   ```

3. L'application gère plusieurs partie simultanément, répéter l'étape 2 pour lancer d'autres parties répéter l'étape 2 dans de nouveaux terminaux.

Démarrage du serveur

Pour démarrer le serveur, exécutez la commande suivante :

```
./serveur <port>
```

Où `<port>` est le numéro de port sur lequel le serveur écoutera les connexions des clients.

Démarrage du client

Pour démarrer un client, exécutez la commande suivante :

```
./client localhost <port>
```

Où `<port>` est le numéro de port sur lequel le serveur est en écoute.

Exemple

Démarrez le serveur :

```
make
./serveur 2000
```

Démarrez deux clients (dans deux terminaux différents) :

```
make
./client localhost 2000
```

Fonctionnalités

Mode de Jeu

Le serveur demande aux clients de choisir un mode de jeu (classique ou contre un robot). Et si mode robot, choisir le niveau de ce dernier.

Grille de Jeu

La grille de jeu est affichée et mise à jour après chaque coup.

Vérification de Victoire

Le jeu vérifie automatiquement les conditions de victoire après chaque coup.

Mode Robot

Un client peut choisir de jouer contre un robot. Le robot utilise l'algorithme Minimax pour choisir ses coups. La profondeur de parcours de cet algorithme régit la difficulté du jeu.

Application client/serveur

Le jeu est lancé via un terminal serveur serveur et se joue avec deux terinaux clients.

Multithread

On peut lancer plusieurs parties en simultanée.

Fichiers et Fonctions Clés

serveur.c

- `main` : Initialisation du serveur, gestion des connexions des clients, et orchestration des parties.
- `changerModeJeu` : Demande aux clients de choisir le mode de jeu et le niveau du robot si applicable.

- `sessionClient` : Gère une session de jeu entre deux clients.
- `handleClient` : Permet de gérer la connection de clients, les attribuant deux à deux à des parties.

client.c

- `main` : Initialise le client, se connecte au serveur et gère l'interaction avec le serveur.
- `choisirModeDeJeu` : Permet au client de choisir le mode de jeu.
- `jouer` : Gère le déroulement d'une partie pour le client.

jeu.c

- `afficherGrille` : Affiche la grille de jeu.
- `initialiserGrille` : Initialise la grille de jeu.
- `ajouterPion` : Ajoute un pion dans la grille.
- `verifierVictoire` : Vérifie les conditions de victoire.
- `grillePleine` : Vérifie si la grille est pleine. Les autres fonctions permettent de tester le jeu en local.

ordinateur.c

- `meilleurCoup` : Utilise l'algorithme Minimax pour choisir le meilleur coup pour le robot.
- `minimax` : Algorithme Minimax pour évaluer les coups.
- `evaluerPosition` : Parcours l'ensemble des coups possibles.
- `evaluerScore` : Évalue le score de la grille actuelle.
- `jeuTermine` : Vérifie si le jeu est terminé.
- `jouerBot` : Fait jouer le bot, utiliser pour le parcours en profondeur Minimax.
- `enleverBot` : Enlève les cases jouées par le bot dans le parcours en profondeur pour simuler la suite de la partie.
- `estPleine` : Vérifie si une colonne est pleine.
- `jouerJoueur` : Place un pion en tant que joueur pour simuler l'adversaire du Bot.
- `enleverJoueur` : Enlève les cases simulées pour le joueur dans le parcours en profondeur.
- `verifierLigne` : Vérifie si un joueur a complété une ligne de 4, utilisée par `jeuTermine` .

Auteur

Projet scolaire pour l'école des Mines de Saint-Etienne.
Martin RABIER @MartinRabier Tristan Panhelleux @tristanplx