Connect4Unix

introduction

Connect4Unix est une implémentation du jeu populaire Connect 4 en C++. Le jeu est joué entre deux joueurs, l'un étant l'utilisateur et l'autre étant contrôlé par l'ordinateur. Les joueurs prennent des tours pour placer leurs jetons dans une colonne de leur choix sur un plateau de jeu 7x6. Le but du jeu est d'être le premier à aligner quatre de ses jetons horizontalement, verticalement ou en diagonale. Le joueur contrôlé par l'ordinateur utilise actuellement une fonction heuristique simple pour placer ses jetons. Après chaque partie on est capable de verifier le score dans un fichier txt et on peut choisir de jouer une autre partie.

Utilisation des tubes : Alec

Pour echanger des informations entres processus, nous avons utiliser 3 tubes :

```
// main.cpp lignes 40-46
// Initialisation des tubes
int fds1[2]; // Utilisateur -> Ordinateur
int fds2[2]; // Ordinateur -> Utilisateur
int fds3[2]; // Utilisateur/Ordinateur -> Père
pipe(fds1);
pipe(fds2);
pipe(fds3);
```

fds1 et fds2 ont comme but d'envoyer la colone du nouveau jeton au joueur opposé après qu'il à été placé. Ces tubes sont aussi utilisé pour envoyer le signal pour quitter le jeu lorqu'on remarque que la partie est terminer.

```
// main.cpp lignes 97 à 102, pris de la section de l'utilisateur
// Lire la colonne choisie par l'ordinateur
read(fds2[0], &colonne, sizeof(colonne));
if (colonne == -1) // signal kill pour finir le jeu
{
    exit(0);
}
```

Le tube fds3 est utiliser pour envoyer le résultat de la partie au processus parent afin qu'il puisse ajuster les scores.

```
// main.cpp lignes 254 à 264
// Lire le résultat de la partie
char resultat;
read(fds3[0], &resultat, sizeof(resultat));
if (resultat == 'X')
```

```
{
    scoreUtilisateur++;
}
else if (resultat == 'O')
{
    scoreOrdinateur++;
}
```

Détection de fin de parties : Mathieu

Pour Détecter la fin d'une partie on utilise la méthode vérifier. Elle analyse chaque rangée, colone et puis chaque diagonale afin de déterminer s'il y à une suite de 4 jetons. Elle s'assure aussi que la grille n'est pas pleine. Cette méthode est exécuté après chaque coup. Elle prend en paramètre la colone et la rangé du dernier jeton placé et retourne un charactère qui représente le résultat : X pour l'utilisateur, O pour l'ordinateur, F pour une grille pleine, et * pour un jeu non terminer.

Gestion de fin de parties : Alec

l'affichage du vainquant est laissé au fils, mais l'affichage du score et la gestion de match retour est fait par le père. À la fin d'une partie le fils qui reconnais l'état finale fait apparaitre la grille et affiche le vainquant, en même temps il envoie le signal -1 à son frère pour lui informé que la session est fini et un message au père dans la forme d'un charactère pour lui informé du résultat du match. les deux processus effectuons un exit et le controlle est retourné au processus père.

```
// main.cpp lignes 109 à 120
// Vérifier si l'ordinateur a gagné
if (resultat == 'O')
{
    std::cout << "L'ordinateur a gagné!" << std::endl;
    jeuUtilisateur.afficher_grille();

    // Envoyer un signal kill pour terminer le jeu
    colonne = -1;
    write(fds1[1], &colonne, sizeof(colonne));
    write(fds3[1], &resultat, sizeof(resultat));

    exit(0);
}
```

Le père, maintenant en controlle, vérifie le résultat et change les scores en conséquant. Il fait un sauvegarde du score avant de demandé à l'utilisateur s'il aimerais continuer, si oui une nouvelle partie est débuté.

Intelligence de l'ordinateur : Mathieu

Pour la section de l'IA, on se sert d’un algorithme heuristique pour que l’IA soit capable de savoir où placer son jeton. Pour ce faire, on commence par générer une copie de la grille et on y place un jeton dans la première colonne. Ensuite, nous évaluons cette nouvelle grille de façon à vérifier l'impact de ce nouveau

jeton sur les chances à l’IA de gagner. On calcule l'impact sur les lignes diagonales horizontale et verticale affectées par ce jeton et l'on donne un score total à la grille. Ceci est répété pour tous les jetons possibles et l’on sélectionne la grille avec le plus haut score. La grille est ensuite retournée aux joueurs de façon qu'il peut jouer son tour.

A noter qu'en ce moment l'algorithme heuristique ne fonctionne pas pour la diagonale. Cependant il en est très proche.

Plus de détails sur qui à fait quoi dans les messages commit