Projet informatique semestre 4

Groupe C : REVERSO

Introduction

Ce document est le document explicatif du jeu Reverso proposé par le Groupe C. Le but de ce document est d'expliquer la conception du jeu, le fonctionnement de ce dernier et des pistes d'améliorations du jeu. En somme ce document devrait permettre à quelqu'un qui veut continuer le développement de ce jeu de le faire dans de bonnes conditions.

Conception du jeu

Comme demandé dans les consignes. Ce projet est séparé en deux parties, une dédiée à la gestion du serveur et du tournoi, l'autre au client se connectant au serveur.

Le client

Nous avons du réfléchir judicieusement au choix de la définition de notre terrain. En effet ce choix va totalement influencer la totalité de notre projet.
Nous avons donc choisi de définir une double liste (grille) de dimension égale à celle du terrain de jeu (6×6). Nous avons nommé cette grille **tjeu**. Dans cette grille nous pouvons avoir 3 état. Un 0 correspond à une case vide (en réalité la case n'est pas vide, le pion est dessiné mais de la même couleur que le fond donc vide), un 1 correspond à une case avec un pion blanc (allié) et un 2 correspond à une case avec un pion noir (adverse). **tjeu** est initialement vide (définie à la fin de *clientC.py*), elle est remplie grâce à la fonction *initJeu* cette fonction fait partie de la classe "fenetre" qui permet d'initialiser le jeu c'est à dire remplir **tjeu** avec des 0 sauf les cases centrales comme expliquées dans les règles du jeu. Cette fonction fait appel à *traceGrille* (qui fait elle même appel à *affichePion*) qui permet de transformer les valeurs de **tjeu** en pion sur le plateau.
Concernant la classe "fenetre" elle permet de définir la fenêtre générale de jeu, c'est à dire celle où apparait le titre du jeu, le bouton d'aide (permettant d'afficher les règles du jeu). Sur la droite nous avons le bouton permettant de passer, le terrain définit précédemment, le nombre de jetons blancs, de jetons noirs, le nombre de jetons restant. Sur la gauche nous avons la gestion du tournoi.
Nous avons aussi implémenté des fonctions utiles dans cette classe, parmi elle nous avons *comptePions* qui permet de compter le nombre de pions et de l'afficher sur la fenêtre du jeu. Nous avons une autre fonction utile *clearCanvas* qui permet simplement d'effacer la totalité du plateau.
Finalement cette classe héberge l'une des fonctions les plus importante de notre jeu la fonction *mouseClic* qui permet de vérifier si lorsque l'on clique sur une case (id est que l'on veut poser une pièce) l'emplacement de

cette pièce est valide ou non et d'afficher ce pion tout en transformant les pions qu'il doit transformer. C'est grâce à cette fonction nous pouvons permettre au joueur de jouer en respectant les règles du jeu.

Grâce à ces quelques fonctions nous pouvons définir notre terrain de jeu, le modifier à notre guise et vérifier si un coup est jouable ou non.

La classe client permet de définir toutes les fonctions relative à l'utilisateur. Lors de son exécution (dans un invite de commande), un client se connecte au serveur et permet alors l'exécution de plusieurs commandes. Parmi ces dernières nous avons une fonction permettant de confirmer la connexion à un serveur, de commencer la boucle de jeu, d'accepter ou de refuser un pseudo, de quitter le jeu, d'afficher les autres joueurs connectés, d'envoyer une demande de partie, le refus une demande, de jouer un coup, d'empêcher le joueur de jouer avec soi même, de donner des informations au joueur (comme l'adversaire qui passe etc...),de gérer la fin du jeu etc... Hormis les fonctions triviale nous avons deux fonctions essentielles, la fonction *Networkcoup* qui permet au joueur d'effectuer un coup et d'envoyer les informations relative à ce coup au serveur. Et la fonction *NetworkfinJeu* qui permet de communiquer l'information sur la victoire ou la perte du joueur et d'afficher cette information.

Nous avons aussi établit dans le client quelques fonctions utiles, une fonction permettant de demander le pseudo de l'utilisateur lorsqu'il se connecte (et de le refuser si il n'est pas bon), une fonction permettant d'afficher les règles, une fonction permettant au joueur de passer et une fonction permettant de tester si la partie est finie ou pas. Ces différentes fonctions seront utilisées dans le déroulement du jeu.

Le serveur

Ce fichier est divisé en trois classes, une classe permettant d'établir et d'entretenir le lien avec le client, une classe permettant le bon fonctionnement du serveur qui héberge les différentes parties et une classe qui gère la partie en elle même.

La classe *ChannelServer* est la classe permettant d'entretenir le lien entre le serveur et le client en somme de gérer le tournoi. Nous pouvons trouver à l'intérieur des fonctions comme celle qui teste si le pseudo est correct qui dans le cas échéant refuse le pseudo, qui définit les points initiaux. Cette classe permet aussi de mettre en relation deux clients (de demander une partie, de gérer l'acceptation ou le refus). Évidemment, cette classe permet de transmettre toutes les informations entre les différents clients et le serveur, comme le coup qu'un client à joué, si il a passé, si il est vainqueur, si il est perdant etc... Cette classe est donc centrale dans notre jeu elle permet la communication globale entre nos deux fichiers, une bonne conception de cette dernière simplifie grandement le travail.

Le serveur est là pour gérer ce qui ne communique pas directement avec le client mais qui permet le bon déroulement d'une partie, c'est à dire de constituer la liste de joueur connecté, de creer une partie entre deux joueurs, d'actualiser l'état, de prendre en compte qu'un joueur passe, de tester si le jeu est fini ou non.

La classe partie permet de gerer la partie a proprement parler. Elle permet d'initialiser cette dernière en connectant deux joueur, de gérer la fin de partie et les scores et de prendre en considération la fin de partie.

Déroulement d'une partie

Nous allons maintenant expliquer succinctement comment se déroule une partie. Nous admettons que les joueurs sont connectés au serveur. Chaque joueur peut alors inviter une autre joueur à jouer. Lorsqu'il clique sur le bouton "inviter" le client envoie au serveur une demande d'information (*demanderPartie*, *partieRefus*, *partieAccepte*, *requetePartie*, *refusDemande*). Si les deux joueurs ont une occupation qui est égale à 0 (ils sont libres), le serveur initialise une partie entre ces deux joueurs (*creerPartie*,*initialiserPartie*). Une fois que le lien entre les deux joueurs est établie le serveur dit qui doit jouer en premier. Ce dernier clic ce qui teste si la case où il a cliqué est valide, si c'est le cas, il pose sa pièce et retourne les pièces qu'il a capturé (*mouseClic*). Après cela, les informations sur ce coup sont envoyées au serveur (*coup*) puis distribuée à l'autre joueur (*actualiserJoueurs*). C'est ensuite à l'autre joueur de jouer est ainsi de suite.
Il est à noter que à chaque coup on teste si le jeu est fini ou non, si c'est le cas le joueur envoie l'information au serveur qu'il a terminé, le jeu et qu'il est le vainqueur ou le perdant, il actualise aussi sont occupation pour le tournoi, le serveur distribue ensuite l'information au vainqueur et au perdant (*end*, *testFin*,*finJeu*). Il envoie ensuite la disponibilité des joueurs à tous les joueurs.

Idées d'améliorations

Si quelqu'un voulait améliorer se jeu il pourrait :

- Créer un outil de détection des zones libres où poser ses pièces, cet outil réutiliserai la fonction qui nous permet de savoir si un coup est valide ou pas, à chaque nouveau tour, on teste sur toutes les cases si le coup est valide, si oui on met un petit rond de couleur au centre de cette case indiquant au joueur qu'il peut poser un pion ici.
- Mettre en place un temps pour éviter que les joueurs s'éternisent devant le jeu, un temps de 1 minute de réflexion avant de forcer le joueur à passer.

Appréciation du cours d'informatique

Ce cours d'informatique nous offre la possibilité de voir l'enseignement d'une manière différente, nous avons un objectif à réaliser et nous devons nous procurer les connaissances nécessaires pour réaliser ce projet par nos propres moyens. Cette méthode pédagogique peut être très performante, nous avons engrangé une quantité de connaissance conséquente et avons réalisé quelque chose de concret mais cela est très perturbant quand nous sommes habitués à avoir des cours très conventionnel.
De plus nous offrir la possibilité de créer un jeu est une idée très enrichissante pour nous. Cela nous a permis d'approcher des concepts d'informatique comme la manipulation de tkinter, le concept de classe, le concept de communication client-server de manière ludique et agréable. Ce projet nous a demande de nombreuses recherches sur PodSixNet et Tkinter de plus nous avons du changer plusieurs fois de jeu (le Blokus, puis le jeu de Dame, puis le Reverso) car nous ne savions pas quelle difficulté un jeu pouvais représenter.
Ce projet est au final une vision inédite de l'apprentissage à CPBx, cela est agréable et nous aurions aimer avoir plus de temps pour le terminer car nous avons mit beaucoup de temps à nous familiariser avec les outils proposés.