Projet Informatique Semestre 4
Groupe B : Tetris VS

Introduction : Ce document est un complément d'informations pour le jeu Tetris VS créé par notre groupe. Il sera expliqué la composition générale des différents fichiers ainsi que celle des modules principaux. Ce document peut servir comme support pour quelqu'un souhaitant améliorer ou modifier le jeu. Il se clôturera également par un avis sur le cours d'informatique du semestre 4.

Composition du Document :

I – Gestion et Composition des Fichiers :

Le projet est séparé en deux parties, une dédiée à la gestion du serveur hébergeant le tournois, une autre au client se connectant au serveur.

- **Conception du Terrain et des Pièces :**

Afin de faciliter grandement la sauvegarde du jeu, nous avons modélisé la grille du terrain par une double liste (grille) de dimensions celles du terrain de jeu. Un 0 dans cette grille correspond à un espace libre/vide, et chaque entier de 1 à 7 correspond à un carré de couleur (une partie d'une pièce). Nous utilisons ces entiers afin de connaître la couleur des cases : chaque entier de 1 à 7 correspond à une couleur : voir *docCouleur* dans *constantes.py*.

De même chaque pièce est, au préalable, enregistrée dans une bibliothèque, par le biais de plusieurs doubles listes, afin de représenter chacune de ses rotations possibles (voir *docRotation* dans *constantes.py*).

- **La partie Client :**

- *clientB.py* : Ce fichier comporte la *class Client*. Lors de son exécution via une invite de commandes, un objet de classe *Client* est créé et connecté au serveur ouvert précédemment. Ce fichier utilise des objets et commandes d'autres fichiers afin de fonctionner correctement. Il comporte entre autres de nombreuses commandes d'actions comme le déplacement des pièces, la gestion du temps des tours, l'animation de la chute des pièces…

- *display.py* : Ce fichier est un fichier important, il gère l'affichage graphique de la fenêtre. C'est lui qui affiche les différents messages sous le terrain, l'horloge, le stock de pièces et surtout la liste des joueurs connectés et les points leurs correspondants.

- *piece.py* : Ce fichier comporte la classe *Piece* permettant de simplifier la conception du jeu en créant une pièce avec des caractéristiques physique, couleur, position…

- *terrain.py :* Ce ficher est responsable de la gestion de la grille de jeu, des vérifications et contrôles pour savoir si une pièce est jouable, ainsi que de son affichage durant les parties. Sa zone d'affichage se réduit à la partie gauche de la fenêtre.

- *constantes.py* : Ce fichier comporte tous les paramètres utilisés dans l'ensemble des fichiers clients, comme la taille du terrain, celle des pièces…

- **La partie serveur :**

- *serverB.py* : Ce fichier comporte les class de *Channel* et de *Server*. Lors d'une exécution via une invite de commandes, un serveur est ouvert et une fenêtre Tk est ouverte, permettant la gestion en temps réelle du tournois. (Ouverture/Fermeture des inscriptions...)

- *partie.py :* Ce fichier comporte la class *Partie* utilisée dans le fichier serveur. Elle est responsable de la gestion des tours d'une partie jusqu'à sa fin.

À Noter: *Afin de simplifier l'utilisation de nos fichiers, les fonctions ont été regroupées par leurs rôles. Des séparateurs en commentaires permettent de se repérer dans les fichiers.*

II – Mise en place du Tournois :

Afin de mettre en place le tournois, il est nécessaire de lancer (via une invite de commande) le fichier *serverB.py*. Dès lors, une fenêtre Tkinter s'ouvre avec plusieurs options. Les deux premiers boutons bloquent ou autorisent respectivement les connexions et les demandes de match entre joueurs. Afin de commencer un tournois il faut déverrouiller les inscriptions, et dès que tous les joueurs sont en jeu, les matchs peuvent être déverrouillés et les inscriptions verrouillées. (Afin d'éviter une déconnexion/reconnexion d'un joueur pendant le tournois)

Toute cette partie de code est présente en bas de fichier du fichier *serverB.py*

III – Déroulement d'une Partie

En amont d'une partie, un joueur envoie une demande à un autre joueur via les boutons de la liste des joueurs (gérée par *display.py*). Le serveur transmet la demande à l'autre joueur qui peut accepter, refuser ou être obligé selon son nombre de points. Dès lors que le second joueur répond, la partie démarre. Lors d'une demande, les deux joueurs passent d'un *état* 0 (non engagé) à un *état* 3 (en attente).

À Noter: *Il y une grande quantité de vérifications avant la création, et durant une partie afin de contrer les problèmes où un joueur se déconnecterait durant un match ou après avoir fait une demande de match à un autre joueur.*

Les deux joueurs sont ordonnés aléatoirement (joueur n°1 et n°2), un objet de classe *Partie* est créé et est sauvegardé dans les *Channels* des deux joueurs afin de conserver la partie (le joueur

face à eux, le nombre de tours de la partie..). La *Partie* est responsable du début de tours en utilisant le *Send* du *Serveur* afin d'informer les joueurs sur leur *état* (qui passe à 1 dont le tour commence, 2 pour celui qui doit patienter). Ensuite, c'est la partie *Client* qui est responsable du déroulement d'un tour, de sa gestion de temps (voir section correspondante dans le fichier).

Le *Client* envoie chaque modification au *Serveur*, afin qu'il puisse actualiser l'affichage du second joueur en temps réel. Lorsqu'un tour est terminé, le *Client* en informe le *Serveur*, qui appelle les fonctions de la *Partie* sauvegardée dans les Channels des deux joueurs (*self.partie*), afin de démarrer le tour du joueur suivant.

C'est le fichier *Client* qui détecte le cas d'un mauvais coup amenant à une défaite. De même que pour une fin de tour, il en informe le Serveur, puis la Partie se charge de distribuer les points aux deux joueurs, puis de modifier leurs états afin qu'ils puissent faire de nouveaux matchs (Les deux repassent en *état* 0).

IV – Idées d'améliorations/Compléments

Le jeu peut encore être amélioré et complété, voici quelques idées de compléments possibles :

- Gestion de Thèmes/d'apparence de la fenêtre : Personnalisation des couleurs des fenêtres
- Personnalisation des touches de contrôle

V – Appréciation du cours d'Informatique

L'idée de la réalisation d'un jeu est extrêmement intéressante et bien plus concrète que les TD précédents d'informatique. C'est également la première fois que nous sommes amenés à créer et à mener un projet un peu plus conséquent que d'écrire un ensemble de fonctions. C'est une très bonne expérience et une approche ludique pour comprendre ce que peut être un projet informatique.

En revanche, même si la réalisation d'un jeu en tour par tour n'est pas le projet le plus compliqué à réaliser en informatique, il reste tout de même un projet demandant une quantité importante de notions. Autant que dans la communication *Client-Serveur* que dans la maitrise des possibilités amenées par *Tkinter* pour l'affichage de nos modules, ce projet a demandé de nombreuses recherches d'informations sur les documentations *PodSixNet* et *Tkinter* non présentées et non abordées dans les documents de TD. Ainsi, pour certains jeux comme celui, assez différents de celui introduit en TD, la réalisation totale du premier jeu semblait prendre un temps trop important et ne nous aidait pas vraiment pour la réalisation finale de notre jeu.

- **Problèmes rencontrés et Découvertes :**
  - Compréhension de certains modules de *Tkinter* (*grid*, *pack*, *scrollbar*, ...)
  - Création rapide de liste avec *[p for p in liste]*

Cependant, ce projet était une façon assez inédite afin de se familiariser avec le langage de programmation *Python* durant notre parcours à CPBx. C'était un projet vraiment très agréable à réaliser. Merci d'avoir proposé ce projet !