Moreira Faria Dylan 17801753

Programmation Réseaux

Rapport de projet : Monopoly

21 Avril 2019

Description du projet/Introduction :

Nous avons décidé, mon groupe et moi, d'effectuer une implémentation du Monopoly dans le cadre de notre projet à rendre pour le cours de programmation réseaux. Nous avons fait plusieurs choix, et parmi ceux-ci nous avons choisi de nous permettre de tordre les règles du jeu et de les adapter pour celles qui nous semblaient compliquées à implémenter. Voici un petit compte rendu :
- Pas de « prison », nous avons un système de permutation de positions entre la case de visite, et la case de punition.
- Pas d'échange entre joueurs.
- Constructions achetables à tout moment, pas besoin d'avoir toutes les propriétés d'un même groupe.
- Si une seule personne fait faillite, alors la partie est terminée, il n'y a qu'un perdant.
- Pour ne pas avoir à côté les innombrables cartes de chance et de communauté, nous avons simplifié la chose en mettant un chiffre aléatoire entre -200 et 200.

Pour ma part, mon Monopoly a été réalisé en Python, grâce aux librairies *tkinter* et *pygame.* Ayant oublié s'il était nécessaire d'installer des dépendances pour une machine externe à la mienne, en dehors de *pygame* évidemment, mais il me semble qu'il s'agit de la seule dépendance nécessaire à installer.

Développement :

Je ne pense pas détailler chaque fonction car cela deviendrait vite fastidieux et contre-productif. Je vais donc plutôt décrire, de la manière la plus précise possible, la structure de mon programme.

A) Mon programme principal tient dans un seul fichier, j'aurais aimé le séparer en plusieurs fichier pour éviter les lignes de codes indigestes, mais étant plutôt débutant en Python, j'ai préféré conserver mon fichier tel quel, car les 3/4 sont en fait des méthodes d'une seule et même classe. J'ignore si je peux les séparer en plusieurs fichiers sans que cela ne me cause de gros problèmes. De fait je m'excuse d'avance si mon code est assez repoussant voire imbuvable.
   Il y a donc dans mon dossier de projet, des sous-dossiers, contenant certains dépendance pour les contenus graphiques et sonores, un dossier ayant été créé en même temps que l'exécutable, 3 fichiers textes contenant les chemins vers certaines images et 3 fichiers.py.

B) - Monopoly.py : Il s'agit du plus gros fichier, et donc du fichier contenant tout le corps de mon programme. Il contient 6 classes :

- La classe Propriete, ayant pour attribut le nom de la propriété, son propriétaire, le nombre de constructions qu'elle contient et son prix d'achat.
- La classe Info, qui contient le nombre de joueurs total et la liste des pseudos de la partie.
- La classe Plateau, qui contient toutes les informations de plateau nécessaires, la position des joueurs, leur argent, l'état des 28 propriétés (propriétaire et constructions), le dé 1, le dé 2, le tour global et le tour du joueur actuel.
-La classe Client, qui comme son nom l'indique, s'occupe de la partie client du programme.
-La classe GUI, mon interface graphique qui contient beaucoup trop d'attributs pour que je les énumère ici.
-La classe Monopoly, qui va donc regrouper une classe queue, la classe Client et la classe GUI.

- serverMonop.py : Nous avons là le fichier qui contient le programme du serveur pour le jeu, avec notre protocole en guise de « langage » de communication avec le client du jeu.

- utils.py : Il s'agit de mon fichier utilitaire, qui regroupe les petites choses dont j'ai besoin et dont la présence n'est pas directement nécessaire dans le programme principal (Des images, des fonctions annexes, etc.)

C) Le fichier principal est donc principalement un « recueil » de méthodes pour la classe GUI, qui contient donc le plus important de mon programme, elle ne se contente pas d'afficher les informations visuelles, elle se charge également de récupérer les informations auprès du client et également de faire tourner la fonction de jeu principale appelée Game. L'organisation du code est un peu bordélique et ne respecte très certainement pas les conventions.
Il y a des méthodes d'affichages, d'animation, de création de boutons, de récupération d'informations comme les propriétés que le joueur possède etc.

Ma classe GUI communique avec ma classe Client par le biais des queues, je me suis fortement inspiré de votre correction faite sur le chat et son interface graphique.
Cela fonction sans encombre, notre programme étant très léger sur la partie réseau. En effet, les données ne s'envoient qu'à chaque fin de tour, les échanges ne sont pas relativement rapides.
Notre protocole est également plutôt court et léger, car ce sont les clients qui gèrent toutes les données de jeu. Le serveur quant à lui ne se charge que de transmettre les données, les seules modifications à sa charge sont le lancement des dés et la positions correspondante du joueur suivant.

Lancement du programme :

Il suffit de lancer le programme avec l'interpréteur de python3, en lançant la commande adéquate avec le fichier Monopoly.py.
Un fenêtre va s'ouvrir avec 2 champs à remplir. Le premier étant l'hôte auquel se connecter, et le second le port de connexion.
Si la connexion s'effectue, une deuxième fenêtre va se lancer pour que nous puissions entrer notre pseudo, celui qu'on désire utiliser en jeu.
Si le nombre de joueurs attendus n'est pas atteint, le programme est alors en attente, sinon la fenêtre de jeu se lance enfin.
Le serveur est moins « user friendly », il doit se paramétrer avec le code source, on doit donc y entrer l'adresse hôte et le port sur lesquels on souhaite héberger le serveur. Il est aussi important de modifier la variable « nb_con » qui correspond au nombre de joueurs attendus pour la partie (nous avons limité ce nombre à 6).
Pour la suite, je pense que mon interface graphique est plutôt intuitive, néanmoins elle comporte tout de même quelques bugs/erreurs graphiques, pas forcément dérangeante pour le déroulement de la partie, à priori.
J'ai fais face à plusieurs segfaults sans savoir quelle(s) en étai(en)t la/les raison(s). En espérant que celui-ci ne vienne pas s'interposer dans vos tests.

Les touches à votre dispositions :
- Les flèches directionnelles : Lorsqu'il ne se passe rien à l'écran (pas d'action contextuelle, pas d'animation), en somme, quand les autres joueurs sont en train d'effectuer leurs choix, vous pouvez par courir les cases du plateau une par une pour afficher les cartes des propriétés et donc vérifier les informations que vous recherchez.
Lors d'une action contextuelle, un menu donc, les flèches de gauche et de droite vous permettent de sélectionner une possibilité (qui varie selon le contexte).
- La touche entrée : utilisée pour valider les choix dans les menus de sélections et pour dévoiler le contenu d'une carte communauté ou chance.
- La touche échap : utilisée pour sortir d'une sélection de carte (vente ou construction)

Protocole :

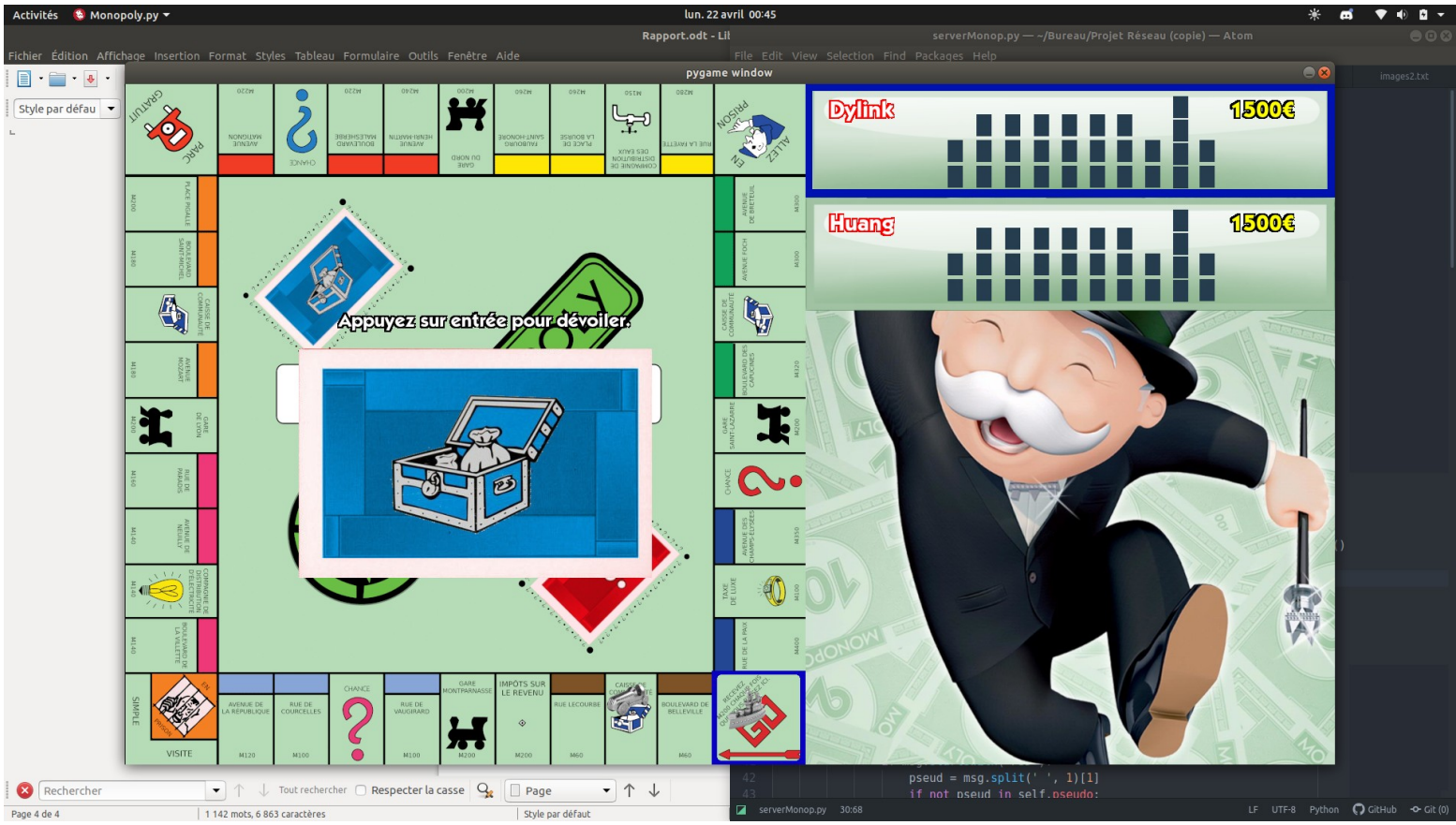
Les commandes du protocole :
- PSD
- PLT
- FIN
- INF

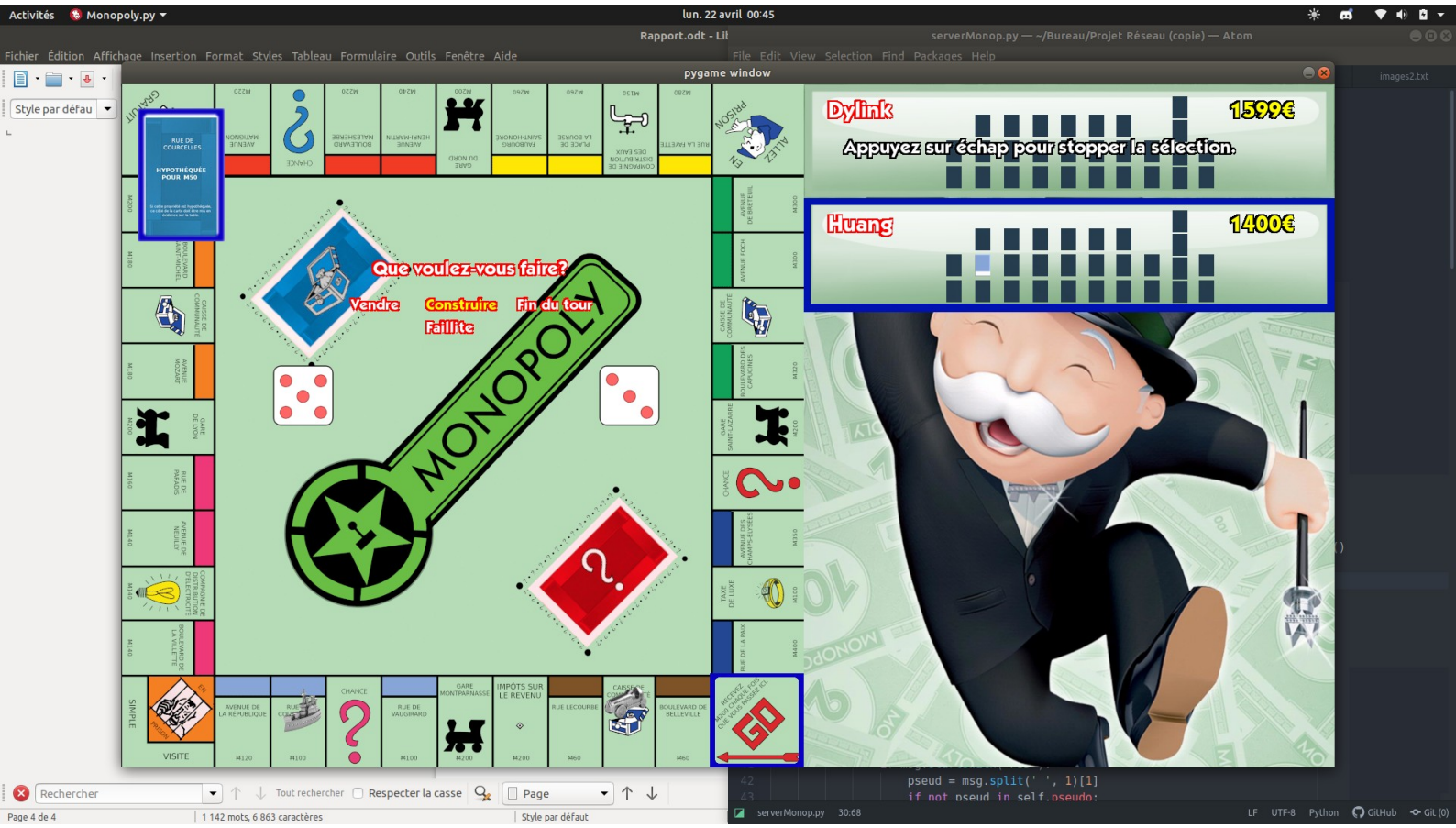
Le protocole se présente comme ceci :
Le serveur attend le nombre de connexions entrés dans le code source. A chaque connexion, il envoie un message ‘PSD’ au client contenant les pseudos connectés (donc vide s’il n’y eu personne de connecté), il reçoit ensuite lui-même un message ‘PSD’ de la part des clients qui se connectent, contenant le pseudo desdits clients. Lorsque le nombre de joueurs requis est atteint, le serveur envoie un message ‘INF’ contenant à la suite : le nombre de joueurs et la liste des pseudos dans l’ordre de connexion.
Dans le même temps, un deuxième message est envoyé, ‘PLT’ suivi de toutes les données de plateau, découpées comme ceci : 6xpositiondesjoueurs + 6xargentdesjoueurs + 28x2xpropriétés (propriétaires et constructions) + dé1 + dé2 + tourglobal + tourjoueur.
Lorsque le client reçoit ‘INF’ et lance la partie.
Le client reçoit ‘PLT’ et répartie les données comme nécessaires.
A chaque fin de tour, le client renvoie PLT avec toutes les données modifiées par son tour, et le tour du joueur incrémenté.
Le serveur reçoit ‘PLT’ et modifie les valeurs correspondantes aux dés et par conséquent la position du joueur suivant.
Si un client décide de faire faillite, il envoie au serveur la commande FIN suivie de son pseudo.
Le serveur reçoit la commande FIN et la renvoie aux autres clients pour les notifier de la fin de partie et du perdant.
Les données de jeux s’échangent donc principalement par la commande ‘PLT’.

Conclusion :

Pour conclure, j'aimerais spécifier que nous avons eu des lacunes en communication au sein de mon groupe. Chacun avançant exponentiellement de son côté sans forcément tenir au courant de la prise de direction etc.
Pour cette dernière semaine, j'ai malheureusement eu de gros soucis de santé, qui m'ont empêché de bien avancer sur ce projet, qui peut donc paraître bâclé, ce qui me désole fortement car j'avais une avance considérable et je voyais même cette semaine supplémentaire comme un bonus. Seulement j'ai été diagnostiqué de plusieurs infections qui m'ont bouffé mon énergie, j'étais soit en repos, soit en déplacement pour mes consultations. Je n'ai donc pas pu rejoindre mes collègues pour tester si mon programme fonctionnait bel et bien avec le leur.
Par contre j'ai pu l'essayer avec celui de Véronique, à priori les deux programmes s'entendent bien, nous avons pu essayer quelques faibles fois pour rencontrer des problèmes et les corriger, mais rien de bien efficace j'en ai bien peur…

Tout ça pour dire que je suis navré de rendre un projet qui peut sembler bâclé, j'y ai mis du coeur et voir ces progrès sans cesse me donnait la motivation de continuer encore et toujours à avancer dessus. Malgré tout je suis quand même à peu près fier du résultat, j'ai réussi à implémenter les choses que je voulais, même celles purement optionnelles mais qui rendent bien graphiquement.
Je tiens à préciser également que la plupart des assets visuels sont faits à la main ou modifiés à la main. Le plateau a été traduit par mes soins, les cartes de propriétés et d'hypothèques ont également étés faites par moi, les vignettes de possessions sur les bannières également etc.
Exemples pour l'interface graphique :


Activités
Monopoly.py
lun. 22 avril 00:45
Rapport.odt - Lil
serverMonop.py — ~/Bureau/Projet Réseau (copie) — Atom
pygame window
Dylink
1500€
Huang
1500€
Appuyez sur entrée pour dévoiler.
pseud = msg.split(' ', 1)[1]
if not pseud in self.pseudo:


Activités
Monopoly.py
lun. 22 avril 00:45
Rapport.odt - Lil
serverMonop.py — ~/Bureau/Projet Réseau (copie) — Atom
pygame window
Dylink
1599€
Appuyez sur échap pour stopper la sélection.
Huang
1400€
Que voulez-vous faire?
Vendre
Construire
Fin du tour
Faillite
MONOPOLY
pseud = msg.split(' ', 1)[1]
if not pseud in self.pseudo: