Compte rendu - 5ème séance PJS4

Histoire

Au cours de cette séance de PSJ4, outre les évolutions dans les différents domaines techniques, nous avons commencé à définir l'histoire et le background de notre jeu. Voici ce qui a été décidé :

Leta Lestrange, une auror (un inspecteur de police du ministère de la magie) a été assassinée, c'est pourquoi le ministère de la magie vous met vous et 7 autres aurors sur l'enquête de sa mort. Thésée Scamander, son mari, a pu assister à la scène du crime mais a été emmené de force par le meurtrier et, pour pouvoir être sauvé, laisse des indices sur les conditions du meurtre aux aurors qui mènent l'enquête. Le ministre de la magie vous a fourni une liste de suspects qui sont capables de commettre ce meurtre et qui ne possèdent pas d'alibi, et c'est à vous de trouver les conditions du meurtre (meurtrier, arme, lieu). A la fin, la majorité devra juger de la culpabilité des meurtriers et la personne désignée sera envoyée à Azkaban jusqu'à la fin de sa vie.

Règles du jeu :

Thésée Lestrange laisse des indices sur votre passage, ils permettront de déterminer le véritable crime qui se sera déroulé. Cependant, chacun des aurors vont d'abord tenter de deviner un lieu, une arme et un meurtrier selon les indices qui leur sont adressés par Thésée. A la fin de la première phase (tous les aurors ont trouvé leur coupable, leur lieu du crime et l'arme du crime), ceux-ci vont procéder au vote afin de déterminer le véritable meurtre. La majorité décide et le coupable sera envoyé à Azkaban.

Technique

Logique d'application :

Au début de la séance, nous nous sommes penchés sur la solidité du code afin de le rendre plus facilement maintenable et évolutif. Pour cela, nous avons supprimé les dépendances nuisibles entre les classes Partie et MJ en créant une interface *IMaitreJeu*. De plus, afin de résoudre les problèmes de solidités liés aux implémentations des cartes de jeu, nous avons modifié la classe Partie en supprimant les listes des différentes classes et en créant une map qui à chaque type de carte y associe une liste. Par ailleurs, nous avons travaillé sur la phase 2 du jeu, à savoir, la phase de vote pour désigner le coupable. Il y a donc les méthodes permettant à chaque joueur de désigner le coupable (le MJ étant le seul à donner le vrai coupable) et les tests associés.

Android :

Nous avons développé différentes vues gérant l'attente du joueur durant la sélection des personnages, ainsi qu'un écran permettant aux joueurs de saisir l'adresse IP du PC ayant le rôle de serveur (et donc de plateau de jeu). Enfin, nous avons commencé à mettre en commun le code Android gérant la partie serveur et communication entre clients, ainsi que le code gérant les vues et activités.

Communication réseau :

Coté client (Android) : Nous avons créé deux classes *ServiceEnvoyer* et *ServiceRecevoir* qui seront respectivement utilisées pour l'envoi des données au serveur et leur réception. Ces classes étendent la classe abstraite *AsyncTask* de l'API Android qui permet à ces services de s'exécuter sur un Thread en arrière-plan de celui de l'UI. Nous avons aussi codé la classe *ChoixPersos* qui permet à chaque joueur de communiquer au serveur le personnage qu'il a choisi (en utilisant les deux classes codées

précédemment). Pour vérifier notre travail nous avons dû fusionner notre travail avec celui de l'équipe s'occupant d'Android, ce qui pose quelques problèmes de conflits, sans doute dus à Android Studio. Nous avons résolu ces conflits grâce au client graphique Git-Fork et à l'éditeur de texte Visual Studio Code.

Coté Serveur (ordinateur) : Nous avons codé les classes utiles à la communication des joueurs lors du choix de leurs personnages, notamment la classe *ChoixPersos* qui prévient un joueur lorsque c'est à son tour de choisir un personnage et lui envoi la liste des choix qui lui sont disponibles, et empêche donc les joueurs suivant de choisir le même personnage qu'un autre. Pour ce faire nous avons dû fusionner notre travail précédant avec celui de l'équipe s'occupant de la logique d'application car leur code était nécessaire au développement de nos classes.

Interface JavaFX :

Réalisation d'un prototype de menu sur la version ordinateur du logiciel, création de l'écran de nouvelle partie. Nous avons eu une réflexion afin que l'application s'adapte à la taille de la fenêtre et de l'écran de l'utilisateur afin d'assurer la meilleure expérience possible de sa part mais n'avons pas encore implémenté cette fonction. De plus, nous avons commencé à étudier le FX-CSS, ce qui nous a permis de tester en conditions réelles le composant développé précédemment, appelé *ViewManager*, nous a permis de détecter et de réaliser les différentes modifications liées à l'implémentation de cette classe (ajout notamment de la fonction *previous()* qui permet d'aller à la vue précédence, là où *next()* permet d'aller à la suivante). Nous avons pris en main les *Anchorpane* qui sont un type de pane au positionnement relatif, ce qui nous permettra d'adapter l'application à la taille de l'écran.