Conclusion

Etat du projet

La phase initiale de développement du projet étant à présent terminée, il convient désormais d'établir un bilan de l'état actuel du projet, relativement aux exigences spécifiées dans le sujet.

Implémenté

Fonctionnalités & règles du jeu

- [x] Création de 3 à 6 joueurs
- [x] Choix d'identité
- [x] Distribution des cartes en fonction du nombres de joueurs
- [x] Accusation d'un joueur
- [x] Révélation de l'identité d'un joueur
- [x] Déclenchement d'un effet *Witch?*
- [x] Déclenchement d'un effet *Hunt!*
- [x] Les 12 cartes *Rumeur* ainsi que leurs effets
- [x] Défausse d'une carte
- [x] Elimination d'un joueur révélé en tant que sorcière
- [x] Passage au tour d'un joueur révélé en tant que villageois
- [x] Gain de score pour le joueur révélant l'identité d'une sorcière, qui peut rejouer un tour
- [x] Gain de score pour le dernier joueur à l'identité secrète, qui commence au prochain *round*
- [x] Conditions de victoire et “duel à mort” en cas d'égalité

Directives de conception

- [x] Utilisation du patron de conception **Singleton**
- [x] Utilisation du patron de conception **Stratégie** pour les décisions des joueurs artificiels
- [x] Utilisation du patron de conception **Visiteur** pour le comptage des points.
- [x] Utilisation du patron de présentation **MVC** pour que le modèle reste indépendant de la vue.
- [x] Gestion de deux interfaces utilisateur concurrentes, l'une en lignes de commandes et l'autre graphique
- [x] Architecture modulaire et extensible
- [x] Documentation du code

À-implémenter

Sauf omission de notre part, le logiciel présente, dans son état actuel, toutes les fonctionnalités et tous les patrons de conception explicitement exigés dans le sujet.

Il manque toutefois la documentation de la classe `fr.sos.witchhunt.view.gui.scenes.game.GamePanel`, que nous ne sommes pas parvenus à rédiger dans les temps.

Bogues

Cette première version stable ne présente à notre connaissance aucun bogue majeur. Voici une liste exhaustive des bogues mineurs répertoriés à ce jour :

- **Choix dans un menu :** Lors du choix d'un élément parmi un menu grâce à l'interface graphique, il arrive que son choix soit perçu comme invalide et qu'il doive cliquer une seconde fois sur le bouton. Il s'agit peut-être d'un problème dû à l'exécution asynchrone des directives concernant des composants Swing.

Perspectives d'amélioration

Bien que ce projet ne couvre pas les phases de maintenance et d'évolution du logiciel, il est adéquat d'envisager quelques perspectives d'amélioration le concernant :

- **Affichage des cartes Identité :** L'interface graphique pourrait afficher les cartes *Identité* des joueurs
- **Plus d'options :** Il serait par exemple utile de pouvoir choisir une valeur pour le délai entre les actions de jeu
- **Musique & effets sonores :** Le jeu est actuellement muet comme une carpe et chiant comme la lune
- **Amélioration des IA :** Il serait intéressant que les joueurs artificiels disposent de plus de comportements, ou de faire en sorte de catégoriser leurs adversaires pour mieux choisir leurs cibles.
- **Redimensionnement des cartes** L'interface graphique donne toujours la même taille aux cartes. Il serait intéressant de pouvoir les redimensionner dynamiquement lorsque la fenêtre change de taille.

Analyse rétrospective sur le code produit

Ce projet a été pour nous l'occasion d'améliorer considérablement notre niveau en langage `java`, ce qui nous permet à présent de porter un

regard critique sur le code rédigé au début de son développement. Force est de constater qu'il présente des classes inélégantes qui mériteraient un *refactoring*.

- **Effets :** Les effets des cartes *Rumeur* ont été représentés par des classes anonymes parfois chargées en code. Avec l'expérience dont nous disposons à présent, nous pensons que les traitements communs à plusieurs effets auraient gagné à être définis chacun sous la forme de `Consumer` ou de `Function` stockés statiquement par la classe `Effet` à des fins de factorisation.
- **Vue console :** Au début du développement, nous connaissions mal le patron de présentation *MVC* et pensions qu'il fallait aussi que la vue ne contienne pas de références au modèle. Nous avons donc créé une vue console dont les méthodes reçoivent uniquement des types primitifs et des chaînes de caractères en argument, rendant le code des classes `StdView` et `DisplayMediator` inutilement complexe.