|  |
| --- |
| JOURNAL DE BORD |
| Xavier DEGRAEVE  Enzo DERO  **Max M’BEY**  **Felix Cloerec**  [Site web] |

Journal de bord de la réalisation et des tests du projet Mille Bornes

**Semaine du 25/10 :**

**Xavier :** J'ai fait le choix de commencer par créer les assets du jeu, pour pouvoir concevoir l'interface en sachant déjà à quoi elle ressemblerait. J'ai utilisé le logiciel gratuit piskel pour faire l'entièreté des visuels du jeu.

**Enzo :** J'ai commencé par créer toutes les classes avec leurs attributs et leurs méthodes (vides) en me basant sur le diagramme de classes conçu quelques semaines auparavant. Cela m'a permis de poser les bases pour débuter la programmation du jeu. Ensuite, j'ai commencé à développer l'interface du menu principal, qui comporte pour l'instant un bouton "Jouer", un bouton "Quitter" et un logo.

**Max :**

**Felix :** Mise en place du git et apprentissage de ses commandes.

**Semaine du 08/11 :**

**Xavier :** suite de la réalisation des visuels

**Enzo :** J'ai poursuivi en affichant la fenêtre de jeu, en commençant par intégrer les cartes de la main du joueur sous forme de boutons cliquables pour permettre leur sélection. J'ai ajouté une zone de texte destinée à afficher toutes les actions et événements du jeu, ainsi qu'un bouton "Retour au menu" et un bouton "Nouvelle partie". Par ailleurs, j'ai réorganisé le projet pour l'adopter au modèle MVC (Modèle-Vue-Contrôleur) :

- La classe \*\*FenetreJeu\*\* représente la Vue,

- La classe \*\*Partie\*\* représente le Modèle,

- Une nouvelle classe \*\*Controlleur\*\* a été ajoutée pour gérer les interactions entre la Vue et le Modèle.

**Max :**

**Felix :** Début d’implémentation des méthodes des classes Player, CPUFast et CPU Agro et de leur logique, tel que savoir si une carte est jouable ou non ou quelle carte les CPU vont vouloir poser en premier.

**Semaine du 15/11 :**

**Xavier :** fin de la réalisation des visuels

**Enzo :** J'ai implémenté un système de sauvegarde grâce à la classe `java.io.Serializable`, permettant de stocker les objets dans des fichiers au format `.ser`. Cependant, le système de sauvegarde n'était pas optimal au départ : les fichiers de sauvegarde devenaient de plus en plus volumineux. Ce problème a été corrigé par Xavier par la suite.

**Max :**

**Felix :** Suite de l’implémentation de ces méthodes.

**Semaine du 22/11 :**

**Xavier :** Une fois le circuit et les voitures dessinés et animés j'ai commencé par travailler sur la correction de certains bugs d'affichages.

J'ai géré le cas où on lance le jeu pour la première fois et lorsque la sauvegarde n'existe pas.

J'ai ensuite implémenté le système de tour dans le modèle et utilisé la fenêtre de log implémenté par Enzo.

Dans un premier temps, j'ai implémenté le système de tour dans une boucle while qui appelait les joueurs tant qu'il n'y avait pas de gagnants.

Cette implémentation m'a posé des problèmes, notamment de performances et entrainait beaucoup de problèmes avec le tour de l'utilisateur.

j'ai donc opté pour un simple système de boolean qui autorisé le joueur à jouer (à cliquer sur les boutons), et lorsque son tour prends fin, son boolean est mit à false et la méthode d'action des bots est appelée. La méthode d'action des bots appelle une méthode qui met le boolean du joueur à true.

Il y avait un problème concernant la sauvegarde qui prenait de plus en plus de place au fur et à mesure des parties (alors que la sauvegarde de partie n'enregistre que la dernière partie si elle a été laissée en suspens). Le problème venait du fait que la liste de joueurs n’était pas réinitialisée

Il y avait aussi un problème qui venait de la manière dont on initialisé les boutons. On ajoutait les actionListeners en boucle sans jamais les enlever, il y avait donc des soucis quand on cliquait sur les boutons, car les méthodes étaient appelés plusieurs fois au lieu d'une. J'ai réglé le souci en forçant les boutons à n'avoir qu'un seul ActionListener (supprimer tous les listeners avant de les rajouter lors de l'initialisation)

**Enzo :** Les interfaces des fenêtres "Menu" et "Jeu" ont été modifiées pour s'adapter à un affichage en plein écran. J'ai ajusté les dimensions des éléments en fonction de la taille de l'écran. Ensuite, j'ai commencé à implémenter la première voiture pour qu'elle avance et se positionne correctement sur le circuit en fonction de sa distance parcourue (en kilomètres), sans utiliser pour le moment les GIF animés qui seront ajoutés ultérieurement, tout comme les deux autres voitures.

Un problème est survenu avec la mise à jour des cartes lorsque le joueur appuyait sur "Fin de tour" : elles ne se renouvelaient pas. Le souci a été corrigé en supprimant les anciennes images des cartes, en les rendant invisibles, puis en les réaffichant comme lors de l'initialisation de la partie.

**Max :**

**Felix :** Implémentation de nouvelles méthodes dans la classe player pour que le player et les CPU puisse jouer leurs cartes et que cela soit affiché en fonction du type de carte.

**Semaine du 29/11 :**

**Xavier :** J'ai ensuite commencé à travailler sur les interactions entre le joueur et l'interface graphique, notamment lorsque le joueur joue une carte. Le contrôleur est devenu un peu complexe, j'aurais pu mieux gérer la distinction entre contrôleur et modèle pour faciliter la compréhension du code.

Lorsque les éléments les plus importants ont été implémenté sur le joueur, j'ai implémenté le système de jeu des CPU, basé sur un arbre de décision en utilisant les méthodes faites par Felix vérification () et jouerCarte() en les adaptant en fonction du type de joueur. L'architecture MVC a facilité le travail. A la fin, les bots pouvaient piocher, défausser, finir leur tour, et jouer en fonction de leur personnalité. Les attaques sont lancées sur le joueur le plus avancé (si l'attaque est possible). En cas de joueur rendu à la même borne, la cible est choisie au hasard.

Il fallait ensuite permettre au joueur de suivre la partie, j'ai donc implémenté un système complet qui informe des coups joués par le joueur et les bots. Le système de log permet aussi de savoir pourquoi une action du joueur n'est pas possible.

**Enzo :** J'ai commencé à implémenter les premiers tests sur les cartes et leur utilisation. Ensuite :

- J'ai ajouté les deux autres voitures et fait en sorte qu'elles puissent parcourir le circuit, tout comme la première, tout en s'adaptant dynamiquement à la taille du circuit, lui-même ajusté à la résolution de l'écran.

- J'ai intégré les premiers GIF animés des voitures qui dérapent dans les virages.

- J'ai également ajouté les premiers effets sonores : un klaxon lorsqu'on attaque une voiture en cliquant dessus, ainsi qu'une ambiance sonore de foule en arrière-plan.

**Max :**

**Felix :** Modification de certaines méthodes pour aider à l’Display des cards qui soundt jouées par chaque player.

**Semaine du 06/12 :**

**Xavier :** J'ai implémenté le système de partie, le joueur qui commence à jouer est choisi au hasard. J'ai aussi commencé à implémenter le système de sauvegarde des parties. J'ai fait en sorte que certains logs importants soient stockés dans nouveau un fichier .txt à chaque partie. Lorsqu'on reprend une partie, le fichier .txt associé à la manche que l'on reprend est chargé dans la console de log.

**Enzo :** J'ai continué à améliorer les déplacements des voitures en ajoutant un GIF d'animation au démarrage de chaque mouvement, grâce à l'utilisation de timers synchronisés avec la durée des animations. Ensuite :

- J'ai enrichi le système sonore avec un bouton permettant d'activer ou de couper les sons.

- J'ai continué à fluidifier les déplacements des voitures, car elles se "téléportaient" dans les virages.

- Enfin, j'ai ajouté des tests sur le comportement des cartes jouées par les CPU, en tenant compte de leur main et de leur stratégie (rapide ou agressive). J'ai également intégré des timers entre chaque tour des CPU pour ralentir le rythme du jeu, ce qui permet aux joueurs de mieux suivre les actions.

**Max :**

**Felix :** Implémentation de l’affichage des bottes et attaques pour que le joueur ait un visuel des bottes et attaques en cours des CPU et de lui-même.

**Semaine du 20/12 :**

**Xavier :** Le jeu était globalement opérationnel, j'ai donc commencé à ajouter des éléments visuels secondaires (des cartes personnalisées en pixel art, des couleurs différentes pour chaque voiture, une interface entièrement en pixel art pour plus de cohérence visuelle, des meilleures animations de voitures avec des ombres). J'ai pris soin que l'interface s'adapte à la taille à l'écran sur lequel on affiche le jeu.

J'ai continué l'implémentation du système de manche et de partie qu'avait commencé Felix. Il faut finir la manche à pile 700 bornes pour gagner une manche. Lorsqu'un joueur gagne le jeu s'arrête quelques secondes en annonçant le gagnant, et lance ensuite une autre manche en annonçant les points des joueurs. Si la partie est gagnée, le jeu annonce le gagnant de la partie et relance une partie. Les fichiers de sauvegardes de manche sont tous mit dans un nouveau fichier .txt.

J'ai ensuite travaillé à la résolution de bug en faisant de nombreuses de parties tests et en utilisant les classes de test.

J’ai passé le code en anglais (une partie de ce qu’on avait écrit était en français.

**Enzo :** Le jeu étant globalement fonctionnel, j'ai ajouté des commentaires détaillés en anglais dans l'ensemble du code du projet. Cela vise à faciliter l'internationalisation, en particulier pour les classes et les méthodes. J'ai également amélioré le système de gestion du son.

**Max :**

**Felix :** Implémentation du comptage des pts pour chaque joueur dans la classe Game et implémentation des coups fourrés dans la classe joueur. Réalisation du diagramme de classe final.

**Piste d’amélioration pour les prochaines versions :**

-Implémenter les coups fourrés

-Faire des visuels en pixel art pour le menu d’accueil et des historiques pour plus de cohérence

-Pouvoir choisir son pseudo