AVIGNON UNIVERSITÉ

Rapport final du projet interface

Tommy Anfosso

18 mai 2020

Licence d'Informatique
Ingénierie Logicielle

UE Interface graphique et ergonomie

Responsable
Mohamed MORCHID

UFR
SCIENCES
TECHNOLOGIES
SANTÉ

AVIGNON UNIVERSITÉ
CERI

CENTRE
D'**ENSEIGNEMENT**
ET DE **RECHERCHE**
EN **INFORMATIQUE**
ceri.univ-avignon.fr

Sommaire

1 Introduction

Le but du projet est de concevoir et programmer une application permettant de jouer au Morpion et de gérer différents aspects du jeu tel que la gestion d'une intelligence artificielle et des fichiers de sauvegarde.
Nous avions dans un premier temps réaliser la conception de l'interface avec l'outil Balsamiq, il nous était maintenant demandé d'implémenter celle-ci et de satisfaire toutes les fonctionnalités demandées.

Il était initialement prévu que je fasse équipe avec Zouhair cependant nous avons décider de travailler chacun de notre côté car nous avons des cibles différentes. J'ai donc fait le projet tout seul.

Rappel du cahier des charges :

- Gérer les modèles d'IA (suppression, affichage et gestion du fichier de configuration)
- Choix du mode de jeu (homme contre IA ou homme contre homme)
- Identifier clairement le vainqueur via un traitement spécifique des pions composant la ligne du vainqueur
- Régler le niveau de difficulté. Un utilisateur pourra changer ce niveau entre deux parties
- Fournir, si nécessaire, un espace de visualisation de la phase d'apprentissage de l'IA. Il sera notamment ajouté un indicateur de progression de cette phase d'apprentissage dans un processus dédié (Task)
- Proposer à l'utilisateur de modifier les paramètres du jeu via un menu dédié
- Insérer deux processus de transition vu en cours
- Suivre le prototype réalisé durant la phase d'ergonomie

2 Conception et organisation des tâches

2.1 Prototype d'interface réalisé avec Balsamiq

(a) Figure 1 : L'interface principale met en premier plan la grille de jeu. Le panneau d'affichage en dessous de celle-ci indique le tour de jeu. Un menu ouvrable est disponible sur la droite pour accéder à des options avancées. Enfin, un bouton de remise à zéro de la partie est disponible en haut à droite de la grille.

(b) Figure 2 : Le menu propose divers réglages tels que le choix du mode de jeu (Joueur contre IA ou Joueur contre Joueur), le choix de la difficulté (Facile, Normal ou bien Difficile) et enfin un bref récapitulatif des régles du jeu.

(c) Figure 3 : Lorsqu'il y a un gagnant, on indique quelle ligne a été complétée et qui a gagné la partie.

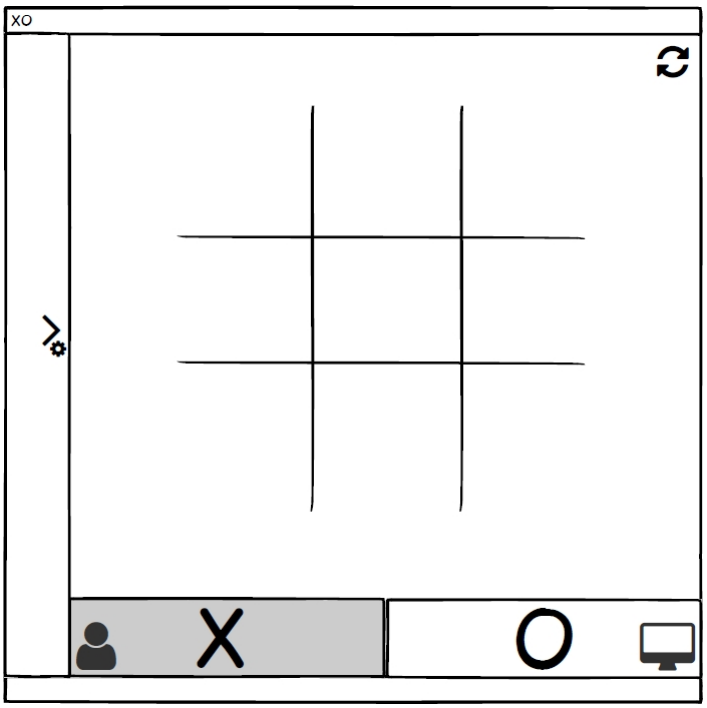

Figure 1. Interface : menu non-déployé

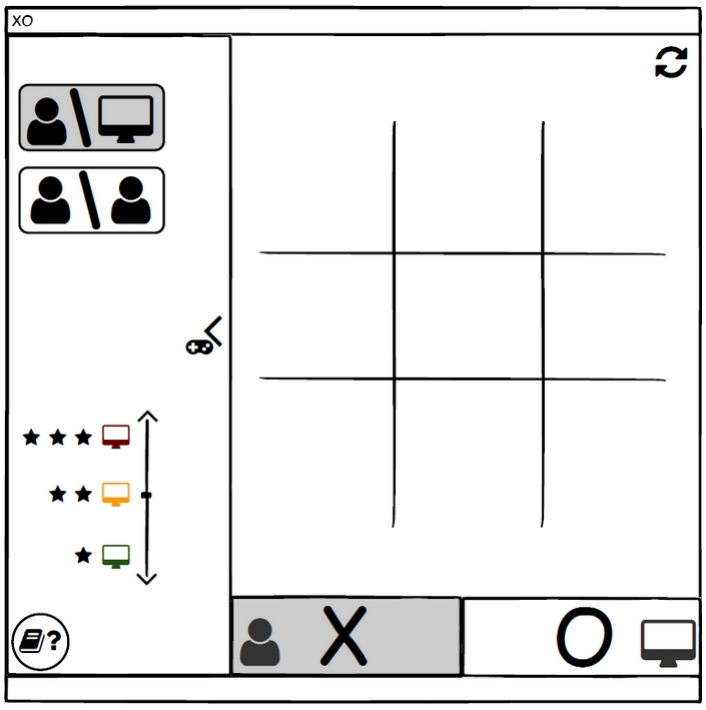

Figure 2. Interface : menu déployé

2.2 Liste des fonctionnalités à implémenter (dans l'ordre)

I. Pouvoir jouer une partie optimale (7h / 40%)

- F1) Prototype d'une interface très simplifiée (aucune animation, utiliser des éléments JavaFX simples, pas d'event handlers,..). Que les éléments essentiels (grille et tour de jeu).
  Temps estimé : 1h

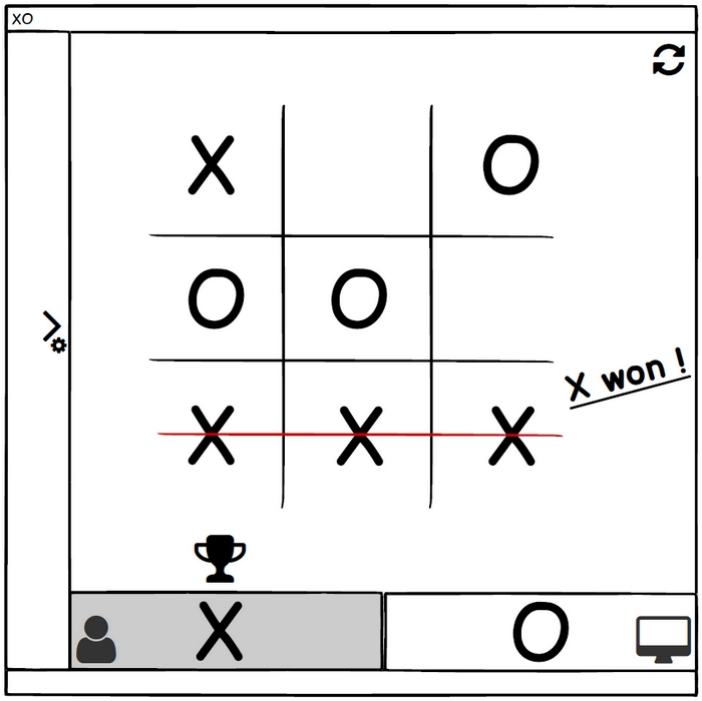

Figure 3. Interface : fin de partie

- F2) On peut dessiner un symbole quand on clique à l'intérieur d'une case
  Temps estimé : 10min

- F3) Système de tour + fin de partie (lorsque grille complète pour l'instant) + reset de partie
  Temps estimé : 30min

- F4) On est capable de déterminer l'issue d'une partie (qui a gagné/ tie)
  Temps estimé : 1h

- F5) Le robot joue aléatoirement
  Temps estimé : 30min

- F6) Le robot suis un modèle IA. Espace de visualisation de phases d'apprentissage de l'IA dans un processus dédié (Task).
  Temps estimé : 4h

II. Ajout des éléments secondaires de l'interface (4h / 20%)

- F7) Pouvoir gérer les modèles d'IA via l'interface
  Temps estimé : 1h

- F8) On peut accéder à un menu (pour des options supplémentaires)
  Temps estimé : 30min

- F9) On peut choisir le mode de jeu (solo ou 1v1)
  Temps estimé : 1h

- F10) On peut choisir la difficulté
  Temps estimé : 1h

- F11) On peut consulter le rappel des règles du jeu
  Temps estimé : 30min

III. Ergonomie (8h / 40%)

- F12) Faire une interface qui ressemble aux dessins prototypes
  Temps estimé : 3h

- F13) Faire les animations qui conviennent lorsqu'on on passe la souris sur les éléments
  Temps estimé : 4h

- F14) Le curseur change en fonction de l'élément sur lequel l'utilisateur pointe (si c'est une case → un feutre, si c'est un bouton de menu → main avec un doigt, ..)
  Temps estimé : 1h

2.3 Diagramme de Gantt

Planning initialement prévu (4)

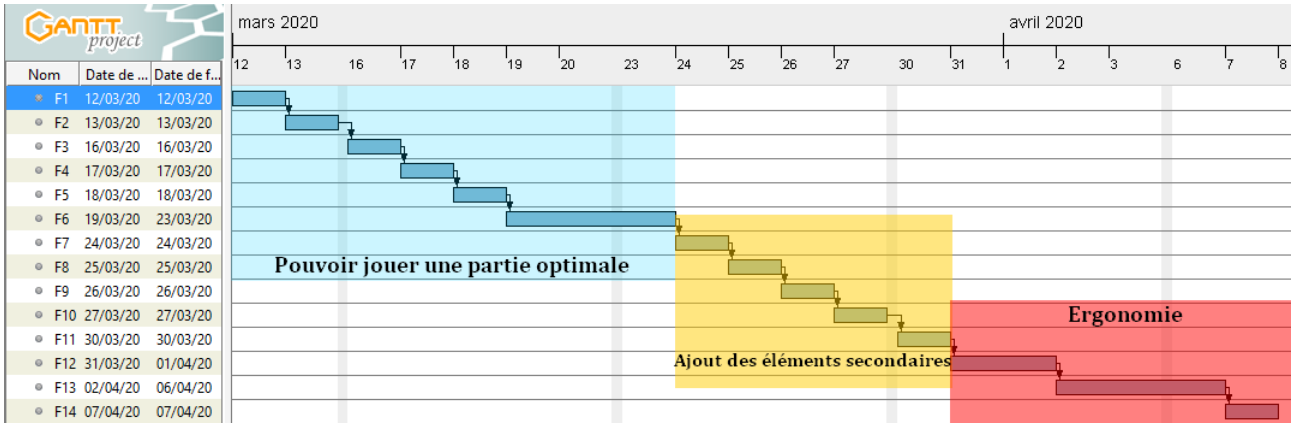

Figure 4. Diagramme de Gantt

3 Organisation du code et des ressources

La JavaDoc complète est disponible dans le code source (générée dans le répertoire 'doc').

3.1 Les packages

- **morpion** : Contient toutes les classes et ressources dédiés à l'application.

- **ai** : Contient des classes pour faire fonctionner un réseau neuronal artificiel de type Perceptron multicouche, on les utilise pour nos modèles d'IA.

- **Mk** : Contient des classes personnelles qui simplifient certains procédés répétitifs au cours des projets de programmation.

3.2 Les classes (package morpion)

- Les classes principales (MVC) :
  Main : Est responsable du lancement de l'application JavaFX.

  Ent : Représente l'entité, c'est à dire l'état d'une partie de morpion à un instant t.

  View : Est responsable de la gestion des événements, fait appel au Controller pour l'execution des cas d'utilisations et le lancement de certaines transitions.

  Controller : Est responsable de l'exection des cas d'utilisations de l'application ainsi que de la synchronisation des données entre l'entité et la vue.

- Les classes secondaires :
  Elles sont utilisées dans les classes principales, ce sont les acteurs principaux des différentes fonctionnalités de l'application.

  Ai : Est responsable de la gestion des modèles d'IA et de l'enregistrement des données de jeu prévues à cet effet.

  AnimationFactory : Est responsable de la création de toutes les animations/-transitions de l'interface et de leur mise en cache.

  TaskLearning : Est responsable de l'apprentissage de nouveaux coups par le modèle d'IA sélectionné.

  TaskConfiguring : Est responsable de détecter en temps réel si le fichier de configuration a été modifié et d'en informer, si c'est le cas, le thread d'apprentissage.

- Les classes 'utilitaristes' :
  Elles ont pour but de simplifier certaines tâches répétitives et utilisées dans diverses situations, elles ne contiennent que des méthodes statiques publiques.

  RES : Est responsable de donner un accès facile aux ressources (images) utilisées dans l'application pendant l'execution.

3.3 Les énumerations

- **Square** : Représente une case de la grille de jeu ('X', 'O' et 'EMPTY').
- **Row** : Représente une ligne de la grille de jeu ('HORIZONTAL_1', ..).
- **Player** : Représente un joueur ('X' et 'O')
- **Mode** : Représente un mode de jeu ('P_VS_AI' et 'P_VS_P')
- **Difficulty** : Représente une difficulté ('EASY', 'NORMAL' et 'HARD')

3.4 Les ressources (fichiers FXML et CSS)

- **View.fxml** : Représente la vue principale de l'application. Elle inclut les sous-vues 'Grid', 'Turn' et 'Menu'.

- **Grid.fxml** : Représente la vue de la grille.
- **Turn.fxml** : Représente la vue du tour de jeu.
- **Menu.fxml** : Représente la vue du menu déployable.
- **application.css** : Définit les classes de styles utilisées dans l'application.

4 Historique des versions/commits

L'historique complet du développement est accessible sur la plateforme Github au lien suivant : `https://github.com/Motyak/S6_Morpion/`

Voici les dates de début et d'aboutissement des étapes importantes du projet :

- Début développement 'Partie optimale' :
  -> 14 avril 2020
- Fin développement 'Partie optimale', début développement 'Partie interface' :
  -> 19 avril 2020
- Fin développement 'Partie interface', début développement 'Partie ergonomie' :
  -> 21 avril 2020
- Fin développement 'Partie ergonomie', début développement 'Partie refactoring' :
  -> 27 avril 2020
- Fin développement 'Partie refactoring' + documentation du code :
  -> 13 mai 2020

5 Démonstration

En ce qui concerne la démonstration, j'ai préféré enregistrer une courte vidéo présentant différents scénarios avec visualisation en temps réel de la console Eclipse et du contenu des fichiers 'data/moves.txt' contenant les coups enregistrés et 'data/config.txt' contenant la configuration.

Je vous prie de bien vouloir la visionner, elle est disponible sur la plateforme YouTube (`https://www.youtube.com/watch?v=Xsn9bwgOyyQ`) ou bien en local dans l'archive de mon rendu sous le nom de 'demo.mkv'.

6 Solutions employées

6.1 Calcul du résultat d'une grille

Une grille est un ensemble de cases. Une case ne peut avoir que trois valeurs possibles : 'EMPTY'(0), 'X'(-1) ou 'O'(1).

Pour calculer le résultat de la grille, on ajoute dans une liste la somme de chaque ligne individuellement (une ligne = trois cases alignées).

Puis on trie le contenu de la liste dans l'ordre descendant en prenant le carré de chaque valeur.
La première ligne de la liste sera donc celle dont la valeur absolue est la plus grande.
Je peux ensuite retourner directement un Joueur à partir de cette valeur : si la valeur est égale à -3, le vainqueur est 'X', si c'est 3, le vainqueur est alors 'O', sinon il n'y a pas de vainqueur (on retourne null).

6.2 Enregistrement des coups de vainqueur

L'intelligence artificielle, afin d'apprendre, aura besoin d'analyser un ensemble de coups jouées qui ont menés vers une victoire. C'est pour cela qu'à chaque partie gagnée, on enregistre tous les coups du vainqueurs à la suite du fichier 'data/moves.txt'.
Le format choisi pour l'enregistrement d'un coup est le suivant :

Coup = GrilleAvant + ';' + GrilleApres
Grille = Case0 + ',' + Case1 + ',' + ... + Case8
Case = 0 | -1 | 1

Ex : "0,0,0,0,0,0,0,0,0 ;-1,0,0,0,0,0,0,0,0"

6.3 Apprentissage et configuration

Un thread tourne en boucle, en parallèle du programme, et permet de lancer des apprentissages pour le modèle d'IA sélectionné lorsqu'il y a des nouvelles données à lui faire apprendre.

L'apprentissage d'un modèle s'effectue en lui faisant apprendre chaque coup inclus dans le fichier 'data/moves.txt', contenant tous les coups qui ont mené vers une victoire. A chaque fois que le fichier est modifié (ajout ou modification de données), le modèle réapprend à zéro à partir des données du fichier.

Le thread Configuration surveille en temps réel si le fichier de configuration est modifié, et si c'est le cas il relance le thread Apprentissage (pour charger le modèle correpondant).

Il est possible d'attribuer un modèle d'IA par mode de difficulté via le fichier de configuration, bien qu'il y ait des valeurs par défaut il est possible de l'éditer via l'application pour tester divers paramètres et voir comment l'intelligence artificielle adapte son jeu.

6.4 Transitions et animations

Les deux animations les plus importantes de l'interface sont l'animation de la ligne gagnante (avec annonce du vainqueur) et l'animation du déploiement du menu permettant d'accéder aux options avancées.

(a) L'animation de la ligne gagnante est une animation de type 'PathTransition' qui fait déplacer un point d'une coordonnée de départ à une coordonée d'arrivée. Pour dessiner un tracé de ce mouvement, il est nécessaire d'écrire une fonction qui va, à chaque changement de position de ce point, dessiner sur le Canvas la ligne de couleur rouge avec une certaine épaisseur. On peut modifier la durée de l'animation pour faire un tracé plus lent ou au contraire plus rapide.

(b) L'animation du déploiement de menu est une animation de type 'ParallelTransition', c'est à dire qu'il y a en réalité plusieurs transitions différentes qui se lanceront parallèlement. Il s'agit principalement de 'TranslateTransition' pour faire glisser le panel du menu d'un côté ou de l'autre, mais aussi une transition personnalisée afin de modifier la largeur du panel contenant la grille (afin qu'elle reste centrée en permanence).
De ce fait les panels adaptent leur taille fluidement et non pas d'un coup.

7 Difficultés rencontrées

La principale difficulté de ce projet a été pour moi la compréhension du fonctionnement du réseau neuronal. Même utilisé sur une grande variété de données, j'ai l'impression que l'IA n'adapte pas sa manière de jouer plus que ça et qu'elle continue de jouer très mal. Le but du projet n'étant pas d'étudier les fonctions de transfert employées ou d'analyser les données enregistrées pour réduire l'erreur, je me suis donc contenté de ces résultats.