

M1 MIAGE APP - 19/12/2023

# Morpion Solitaire Rapport



Guillaume HENRIQUES PEREIRA  
Nathan ZAFIZARA-BENETRIX

# Architecture du Projet

L'architecture de notre projet de jeu du morpion solitaire est organisée en plusieurs packages distincts, chacun ayant un rôle spécifique dans l'ensemble du projet.

**Package 'components'** : Au cœur de l'application, ce package contient les éléments fondamentaux du jeu.

- **Classe Point** : Représente les coordonnées et le score des coups joués sur le plateau.
- **Classe Alignment** : Gère les alignements de points selon les règles du jeu.
- **Enum Direction** : Décrit les directions possibles pour les alignements sur le plateau.
- **Classe Board** : Gère l'état actuel du plateau, y compris les points placés, les alignements formés, et le score du joueur. Joue un rôle central dans le contrôle du déroulement du jeu, en validant les mouvements et en gérant les états du jeu.

**Package 'game'** : Ce package gère la logique de jeu et l'interaction avec l'utilisateur.

- Le sous-package '**gui**' est crucial pour l'interface utilisateur. Il contient des classes de contrôleurs comme **MainTitleController** et **JoinFiveController**, qui gèrent respectivement la configuration initiale du jeu et le déroulement des parties. **EndOfGameController** prend en charge l'affichage des résultats à la fin d'une partie, et **RankingController** affiche le classement des joueurs.

La classe **MusicPlayer** y est aussi présente et gère tous les sons et musiques

diffusés au cours de la partie

- L'enum **Mode** dans le package '**game**' spécifie les quatre modes de jeu (5T, 5D, 5T# et 5D#), influençant la façon dont les points et les alignements sont gérés.

# Architecture du Projet (suite)

**Package 'ranking'** : La classe GameHistory dans ce package représente le résumé d'une partie jouée, et permet la connexion à une base de données pour enregistrer et récupérer l'historique des parties.

**Package 'solver'** : Ce package introduit une capacité de résolution automatique du jeu. L'interface Solver définit un modèle pour les algorithmes de résolution, et la classe RandomSolver implémente cette interface pour générer des solutions aléatoires.

**Package 'main'** : Contenant la classe Main, ce package sert de point d'entrée pour l'application. Il lance l'interface utilisateur JavaFX, démarrant ainsi le jeu.

# Fonctionnalités Implémentées

## Moteur de Jeu pour les Modes 5T, 5D :

- Permet aux utilisateurs de jouer des parties en modes 5D (Disjoint) ou 5T (Touching), avec des règles distinctes pour chaque mode.

## Méthode de Recherche de Solution Automatique Aléatoire :

- Intègre une fonctionnalité qui permet de générer automatiquement des solutions pour le jeu, utilisant un algorithme aléatoire. Cette fonctionnalité enrichit le jeu en offrant une option de résolution stratégique.

## Interface Graphique Interactive :

- Fournit une interface utilisateur visuellement attrayante et intuitive pour interagir avec le jeu, incluant des éléments graphiques pour représenter le plateau, les points et les alignements.

# Extensions et Améliorations Réalisées

## Tableau des Scores et Meilleures Grilles

- Permet aux joueurs de visualiser un tableau des scores affichant pour chaque mode les 10 meilleures parties, améliorant l'aspect compétitif du jeu.

## Variété des Plateaux de Jeu

- Propose différentes configurations de plateaux de jeu, dénommées 5T# ou 5D#, offrant ainsi une variété et un renouvellement dans l'expérience de jeu.

## Effets Sonores

- Ajout d'effets sonores pour améliorer l'immersion et l'expérience utilisateur lors des actions de jeu comme la formation d'un alignement ou encore la demande d'indices.

## Interface Graphique Inspirée par Zelda

- Développement d'une interface graphique basée sur le thème de Zelda, rendant le jeu visuellement plus attrayant et engageant.

# **Non-implémentation de l'Algorithme NMCS et du Multi-threading**

## **Contraintes de Temps**

- N'ayant pas encore abordé le chapitre au moment de la réalisation du projet et afin de respecter les délais, l'ajout de ces fonctionnalités avancées aurait pu entraîner des retards significatifs, compromettant la livraison en temps voulu du projet.

## **Stabilité de l'Application**

- L'introduction du multi-threading, malgré ses avantages potentiels pour les performances, aurait pu introduire des problèmes de conditions de concurrence et de deadlocks, affectant la stabilité de l'application.

## **Focalisation sur les Fonctionnalités Clés**

- La décision a été prise de se concentrer sur les fonctionnalités essentielles déjà implémentées, telles que le moteur de jeu, la méthode de recherche de solution aléatoire, et l'interface graphique interactive, ainsi que sur les extensions comme le tableau des scores et les variétés de plateaux de jeu, pour assurer la qualité et la ponctualité du projet.

# Défis Techniques et Solutions

## Défi de Logique dans la Détection d'Alignment

- **Problème :** Dans la classe Board, spécifiquement dans la méthode hasAlignment(Point point, Direction direction), nous avons rencontré un problème de logique. Initialement, notre approche pour détecter un alignement se basait sur la vérification de la formation d'un alignement à partir d'un point donné et dans une direction spécifique. Cependant, cette méthode ne testait le point que dans une position unique de l'alignement potentiel, souvent le début, ce qui limitait la capacité à détecter tous les alignements possibles.
- **Solution :** Pour résoudre ce problème, nous avons introduit une boucle imbriquée dans la méthode. La boucle externe permet de tester le point donné dans toutes les positions possibles au sein d'un alignement, tandis que la boucle interne vérifie si un alignement complet est formé. Cette modification a permis de détecter avec précision tous les alignements potentiels créés par la pose d'un nouveau point, renvoyant un ensemble d'alignements possibles au lieu d'un seul.

# Défis Techniques et Solutions (suite)

## Défi de la Représentation Graphique du Plateau

- **Problème :** La création d'une représentation graphique intuitive et claire du plateau de jeu dans l'interface utilisateur était un défi majeur. Il était essentiel d'assurer une visualisation précise des points et des alignements pour une expérience utilisateur optimale.
- **Solution :** Pour relever ce défi, nous avons opté pour une approche innovante en utilisant un script Python fourni par ChatGPT. Ce script a joué un rôle crucial dans la conceptualisation et la génération de la grille du plateau. Il a permis la complétion du fichier FXML qui définit la structure de la grille, y compris les lignes et les points de la manière suivante :
  - Il utilise une boucle for pour l'espacement des lignes et des points, assurant une distribution uniforme sur le plateau.
  - Il calcule également un décalage pour centrer la grille dans le panneau (Pane) JavaFX, en tenant compte des dimensions spécifiques du panneau et de la grille.
  - Chaque ligne verticale et horizontale est ajoutée via une boucle itérative, positionnée avec précision en fonction de l'espacement et du décalage calculés.
  - Des cercles représentant les points de jeu sont également générés avec des identifiants uniques, permettant une interaction interactive avec l'utilisateur lorsqu'il clique sur ces points.
  - L'intégration de ce fichier FXML dans notre interface utilisateur JavaFX a facilité la création d'une grille de jeu fonctionnelle et esthétiquement plaisante.

# Répartition des Tâches dans le Binôme

## Guillaume

### Conception du Jeu de Base

- Développement du package 'component', permettant de jouer au Morpion Solitaire en ligne de commande avec un affichage du plateau dans le terminal. Cette étape a posé les fondations du jeu.

### Affichage du Classement

- Responsable de l'intégration de l'affichage du classement dans l'interface graphique, en utilisant les données issues de la base de données mise en place par Nathan.

### Fonctionnalités Supplémentaires

- Apport de fonctionnalités telles que le solveur aléatoire (random solver) et les indices (hints), enrichissant l'expérience de jeu.

### Mise en place de la charte graphique

- Conception d'une charte graphique inspirée du jeu Zelda, rendant l'interface plus attrayante et engageante pour les utilisateurs.

### Tests unitaires

- Création de tests unitaires permettant de vérifier le bon fonctionnement du code.

## Nathan

### Architecte globale et reviewer

- Mise en place de l'architecture du code et gestion des dépendances, reviewer général du code et du respect des conventions Java.

### Résolution de Problèmes et mise en forme du projet

- Comme la détection d'alignement et a appliqué les meilleures pratiques de programmation pour structurer l'architecture du projet.

### Développement de l'Interface Graphique

- Prise en charge la majorité de la conception de l'interface graphique, en transformant le jeu jouable en ligne de commande en une expérience utilisateur visuelle et interactive.

### Connexion à la Base de Données

- Implémentation de la connexion à la base de données pour gérer le classement des joueurs.

### Effet sonore et charte graphique

- Ajout d'effets sonores pour améliorer l'immersion dans le jeu.

## **Optimisations et Améliorations du Code**

Une amélioration envisageable pour notre code aurait été d'intégrer le multi-threading. Toutefois, étant donné notre manque de familiarité avec cette technique en Java, un sujet non abordé dans nos cours et nouveau pour nous deux (en Java), nous avons choisi de ne pas l'incorporer dans ce projet. Cette décision prend également en compte notre compréhension des défis liés à la concurrence et à la stabilité, acquis grâce aux cours de systèmes d'exploitation de l'année précédente.

Par ailleurs, profitant de l'expérience professionnelle de Nathan en développement Java, nous avons opté pour une approche collaborative. En pratiquant le pair programming sur l'ordinateur de Nathan, Guillaume a pu bénéficier directement de son expertise. Cette méthode nous a permis de nous concentrer sur l'optimisation, la qualité du code, et le respect des conventions de programmation, enrichissant ainsi notre projet.

# Réflexions et Apprentissages

Ce projet a été une expérience enrichissante qui a renforcé nos compétences en programmation Java et en résolution de problèmes complexes. La collaboration et la répartition des tâches ont amélioré notre capacité à travailler en équipe et à communiquer efficacement. La gestion du temps et la planification stratégique se sont avérées cruciales, nous enseignant l'importance de l'organisation dans le respect des délais. Ces apprentissages nous seront utiles pour de futurs projets, notamment en termes de développement technique, de conception d'interface utilisateur, et de travail d'équipe.

Nous souhaitons également remercier notre professeur de programmation objet, Khitem BLIDAOUI, pour son soutien et ses conseils pertinents tout au long du projet.

