POLYTECH NICE SOPHIA

UNIVERSITÉ CÔTE D'AZUR

Les Aventuriers du Rail - Projet S5

Equipe B

Jim Lainel - Matvei Maksimenka - Rayan Outili - Louis Rainero

07/02/2025

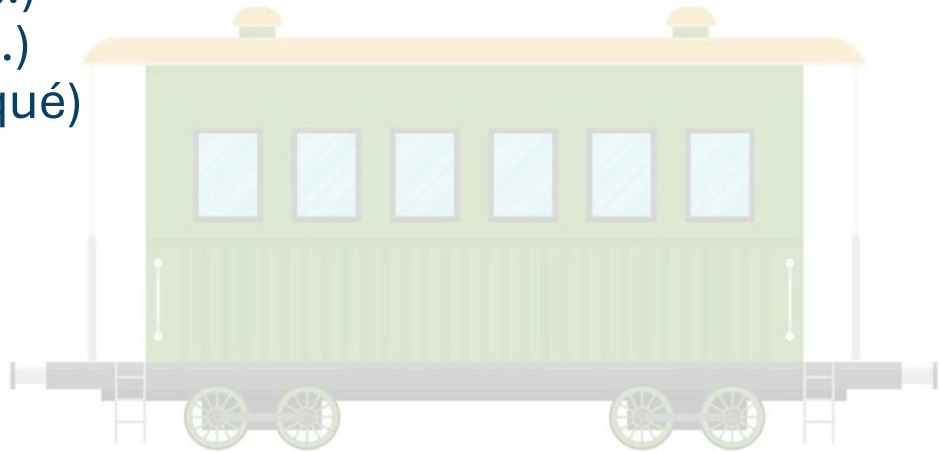
Fonctionnalités attendues:

Fonctionnalités de base :

Représentation du jeu (plateau, routes, wagons, gares, cartes…)

Moteur de jeu (gestion des pioches, calcul des points, bonus…)

Bots (stratégies progressives, du plus simple au plus sophistiqué)



Fonctionnalités supplémentaires:

2 Simulations de 1000 parties

Statistiques sur ces parties

Un 3ème bot respectant la stratégie demandée

Toutes les fonctionnalités demandées ont été réalisées.

Stratégie de gestion de développement

Avant

Une branche par US et plusieurs branches par tâche

Après

Une branche par feature avec possibilité de sous-branches si nécessaire

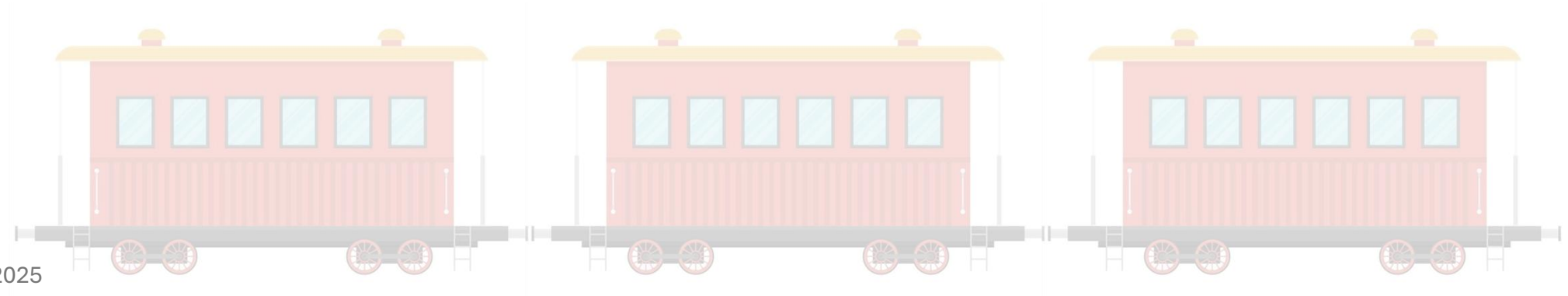
Structure des Commits :

```
prefix: description #issue-id
```

Préfixes :

- `feature:` Nouvelles fonctionnalités
- `fix:` Corrections de bugs
- `refactor:` Restructuration du code
- `test:` Tests
- `docs:` Mise à jour de la documentation

Exemple : `fix: resolve issue with route scoring calculation (#2)`



Architecture et qualité

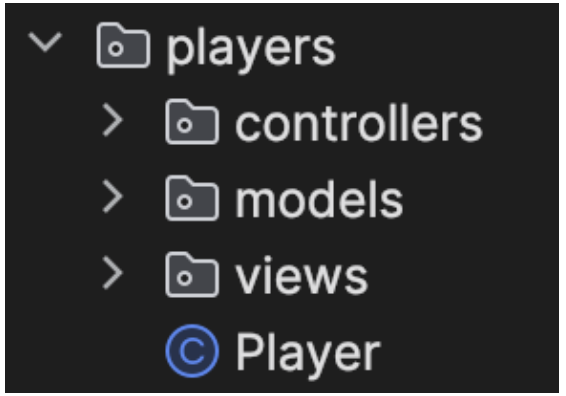
Architecture MVC

- Contrôleurs :
  - Implémente les règles d'une portion du jeu
  - Réalise une action spécifique avec un joueur
  - Manipule les données du jeu
- Modèles :
  - Stocke les données du jeu sans implémenter les règles du jeu
- Vues :
  - Affiche les données du jeu dans des sorties spécifiques

Architecture et qualité

Architecture joueur

- Contrôleur :
  - Implémente le cerveau du joueur
- Modèles :
  - Stocke la main du joueur dans la partie
  - Accès à une partie du plateau
- Vue :
  - Affiche les données du jeu dans des sorties spécifiques



Architecture et qualité

Architecture du moteur de jeu

- Manipule des contrôleurs en trois types :
  - Actions
  - Après le tour du joueur
  - Fin de partie
- N’implémente pas de règle du jeu
- Responsable des tours du jeu et des retours des contrôleurs

Architecture et qualité

Interfaçage et triche

- Séparation de l'implémentation des modèles :
  - Une classe pour le contrôleur
  - Une interface vers son modèle en lecture
- Vérification des règles du jeu dans les contrôleurs
- Interfaces en lecture :
  - Routes
  - Villes

Architecture et qualité

Principes et conception

- Principe SOLID
- Patrons de conception :
  - Factory
  - Composite

Implémentation des Bots

Classes déjà présentes:

Medium/Easy

BotEngine

Description de la stratégie

L'objectif principal du bot sera de complémenter son objectif long, pour cela il voudra passer par des cartes objectifs courts.
Il ne fera pas forcément cela par le chemin le plus direct mais un équilibre entre nombre de points et wagons utilisés, par exemple choisir un chemin avec un peu plus de wagons mais qui rapporte plus de points par wagon placé.
De plus les objectifs remplis devraient plutôt être avec des routes longues dedans (considérer points objectif + longueur route)
En même temps il essaiera de bloquer les actions des autres joueur en essayant de prévoir leurs futures actions en fonction des informations qu'il a sur la partie et les joueurs.

La manière de jouer dépend aussi de l'avancement de la partie. Au début il faut accumuler des cartes wagons et piocher quelques objectifs pour commencer à réfléchir où placer des routes, il est important de ne pas trop se projeter car les blocages arrivent vite, il font donc pas accumuler d'objectifs non réalisés ou réalisables. De plus il vaut mieux placer plusieurs routes d'affilé afin de surprendre les adversaires et éviter les blocages.
Attention ne pas trainer, finir vite peut empêcher les autres de compléter les objectifs longs.

Informations que le bot peut et doit prendre en compte :

- Routes occupées par les autres joueurs
- Les couleurs des cartes visibles prisent par les autres joueurs
- Le nombre de cartes wagons en main des autres joueurs
- Le nombre de wagons restant à chaque joueur
- Toutes les destinations longues possibles
- Toutes les destinations courtes possibles
- Le nombre de joueur (pour savoir à quel point il est important de bloquer)

Ce que le bot doit pouvoir calculer :

- Quelles routes seront essentiels pour son objectif
- Quelles destinations les autres ont l'air de compléter
- Est-ce que une gare est nécessaire à placer
- Niveau d'importance d'une route (facile à prendre, sert à beaucoup d'objectifs)
- L'avancée de la partie en fonction du nombre de wagons/cartes des autres

REFACTOR

Nouvelles classes:

BotEngine

extends

BotEngineControllable

extends

BotEngineWithRandom

extends

Medium/Easy

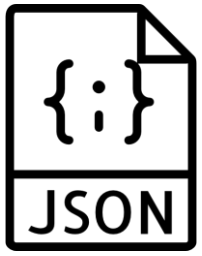
BotEngine

extends

ObjectiveBotEngine

Simulations de parties

AVANT

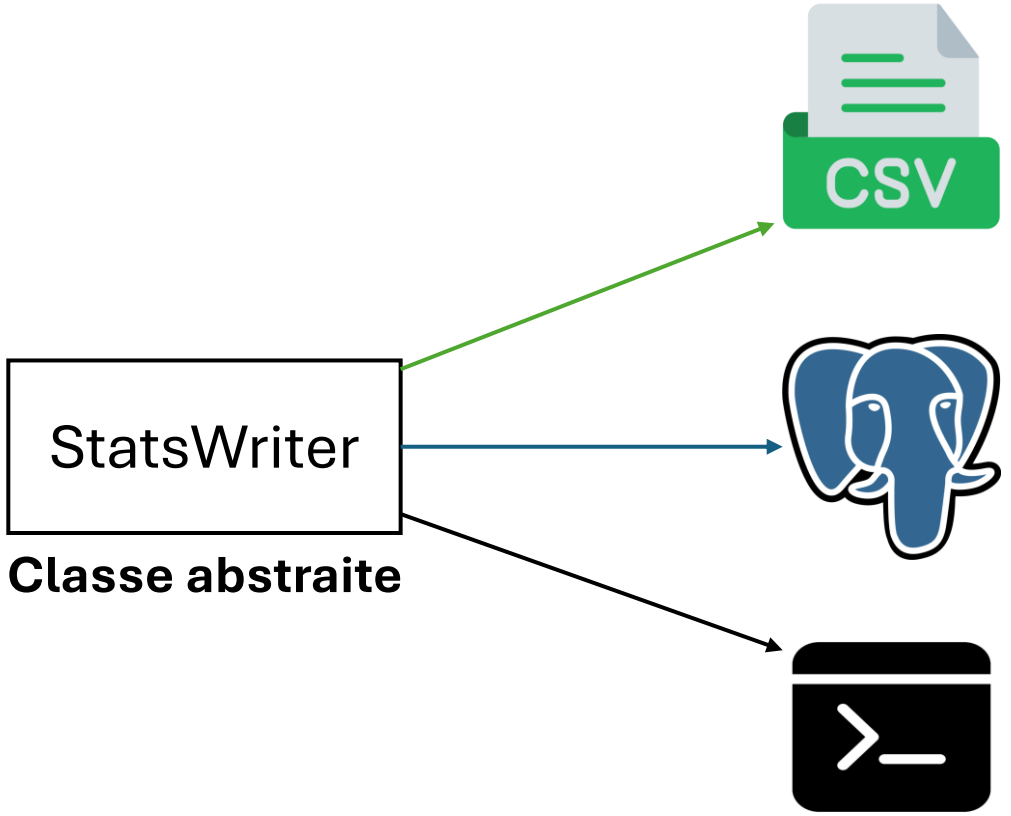


- Formatage en JSON uniquement
- Calcul a la fin de chaque partie
- Taux de victoire uniquement

MAINTENANT

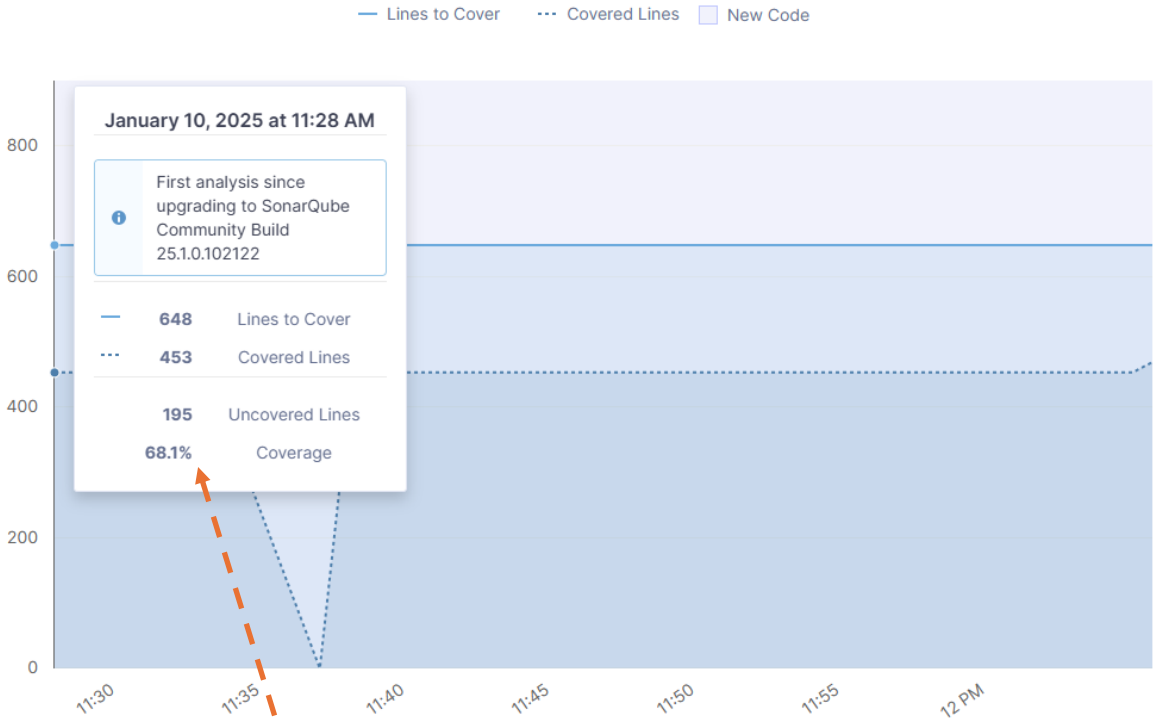
VUE

- Récupération via les vues
- Toutes les actions
- Formatage au choix (StatsWriter)



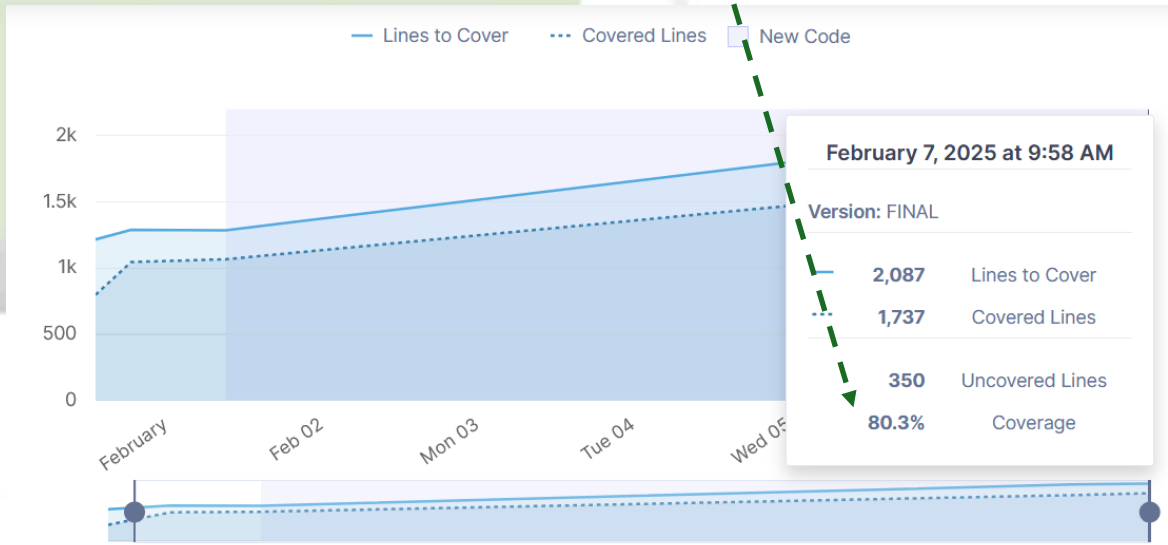
Démonstration

Sonar Avant / Après



68.1% Coverage

80.3% Coverage



Bilan

- Conventions de nommage et organisation (expérience du projet PS5 Poker)
- Stratégie de branchement (plusieurs itérations avant de trouver la bonne)
- Qualité du code : architecture, lisibilité, tests
- Respect des délais
- Gestion des issues, milestones, régularité

POLYTECH NICE SOPHIA

UNIVERSITÉ CÔTE D'AZUR

Merci pour votre attention

Équipe B

Jim Lainel - Matvei Maksimenka - Rayan Outili - Louis Rainero