DATE : 02/02/2026

RAPPORT HEX

DUCREUX Clémence, GENTIL William, JANNAIRE Clément,
KARA-MOSTEFA Riad, VAISSE Alistair

Sommaire

Introduction

Dans le cadre de ce projet de BUT3, nous avons réalisé un jeu de plateau implémenté en Java en utilisant l'API fournie par le département. Le jeu choisi est Hex, un jeu de stratégie à information parfaite, opposant deux joueurs dont l'objectif est de relier deux côtés opposés du plateau par une chaîne continue de pions.

L'objectif principal de ce projet était de concevoir un moteur de jeu fonctionnel, accompagné de tests, ainsi que de bots capables de jouer automatiquement, dont au moins un bot utilisant une approche algorithmique avancée (alpha-bêta, Monte Carlo ou fonction d'évaluation). Le projet devait également respecter des contraintes de qualité logicielle telles que la documentation du code, l'organisation du dépôt Git et la capacité à déployer et tester facilement l'application sur une machine de l'IUT.

Le jeu Hex se joue à deux joueurs sur un plateau hexagonal. Chaque joueur dispose d'une couleur et joue à tour de rôle en plaçant un pion sur une case libre du plateau. Chaque joueur cherche à relier deux côtés opposés du plateau correspondant à sa couleur à l'aide d'un chemin continu de pions. Il n'y a pas de match nul possible dans Hex : la partie se termine obligatoirement par la victoire de l'un des deux joueurs dès qu'un chemin continu est formé.

Méthodologie

Notre groupe est composé de cinq membres. Afin de travailler efficacement et de limiter les conflits lors du développement, nous avons choisi de diviser le projet en quatre grandes tâches, dont l'une a été réalisée par deux personnes en collaboration. Cette organisation nous a permis de répartir le travail de manière équilibrée et de travailler en parallèle sur différentes parties du projet.

Pour faciliter la collaboration, nous avons utilisé Git comme outil principal de gestion de versions. Chaque tâche disposait de sa propre branche, permettant à chaque membre (ou binôme) de travailler de manière indépendante sans impacter directement le travail des autres. Les fonctionnalités étaient ensuite intégrées progressivement dans la branche principale après validation.

Le dépôt Git reflète cette organisation à travers l'utilisation de branches dédiées ainsi que la présence de tâches/tickets permettant de suivre l'avancement du projet. Les classes principales sont documentées à l'aide de Javadoc, et un README explique clairement comment lancer les démonstrations et tests du programme sur une machine de l'IUT.

Tout au long du projet, nous avons adopté une approche itérative, en privilégiant d'abord une version fonctionnelle du moteur de jeu avant d'ajouter progressivement des améliorations, notamment sur les bots et les outils de test. Cette démarche nous a permis d'identifier rapidement les problèmes de conception et de valider les choix techniques étape par étape, plutôt que de chercher à implémenter toutes les fonctionnalités dès le départ.

Cette méthodologie présente plusieurs points positifs, notamment une bonne séparation des responsabilités, une réduction des conflits lors des fusions et une meilleure lisibilité globale du projet. En revanche, certaines intégrations ont nécessité des ajustements, et certaines décisions techniques prises tôt dans le projet ont parfois limité les possibilités d'évolution sans modifications importantes du code.
Dans l'ensemble, cette organisation nous a permis de mener le projet à son terme de manière structurée, tout en acquérant une première expérience concrète de travail collaboratif sur un projet de développement logiciel.

Les tâches

Le projet a permis de développer un jeu de Hex fonctionnel en Java, en s'appuyant sur l'API fournie par le département. Le moteur de jeu est principalement implémenté dans les classes HexBoard et HexPly. Il gère correctement la représentation du plateau, la validité des coups, l'alternance des joueurs ainsi que la détection des conditions de fin de partie.
Un affichage textuel du plateau a été mis en place via la méthode toString() de HexBoard. Cet affichage, utilisé en console, a servi d'outil de debug tout au long du projet afin de visualiser l'état du jeu à chaque tour et de faciliter la vérification du comportement du moteur et des bots.

Des tests fonctionnels et démonstrations ont été réalisés à travers plusieurs méthodes main, notamment dans la classe HexMain. Le programme peut être lancé en mode interactif, avec un joueur humain, ou en mode automatique grâce à l'option autoplay, permettant de simuler des parties complètes sans intervention humaine. Ces tests ont permis de valider la stabilité générale du moteur et le bon enchaînement des coups.

Concernant les bots, un bot simple ainsi qu'un bot plus avancé basé sur une recherche Minimax à profondeur limitée ont été implémentés. Ce dernier est développé dans la classe Simulation et repose sur deux fonctions récursives (explMAX et explMIN) qui explorent l'arbre des coups possibles jusqu'à une profondeur maximale fixée à 6. Le meilleur coup est sélectionné à la racine de la recherche et stocké afin d'être joué. L'évaluation des positions reste volontairement simple et se base principalement sur les états terminaux, ce qui permet d'obtenir un comportement cohérent tout en conservant des temps de calcul raisonnables.

Le projet respecte également les exigences liées à l'organisation du dépôt Git. Le travail a été réparti en tâches avec des branches dédiées, des commits réguliers, une documentation des classes principales via Javadoc et un README décrivant la procédure pour lancer les démonstrations et tests du programme.
Certaines fonctionnalités prévues ou envisagées n'ont toutefois pas été entièrement réalisées. En particulier, le bot Minimax ne dispose pas d'optimisation par élagage alpha-bêta et sa fonction d'évaluation pourrait être enrichie. Une approche basée sur Monte Carlo a été envisagée mais n'a pas été finalisée dans le temps imparti. Ces éléments constituent des pistes d'amélioration naturelles du projet.

Les bots

Plusieurs bots ont été développés afin de tester le moteur de jeu et de comparer différentes approches de prise de décision. Ces bots vont d'un comportement aléatoire à des algorithmes de recherche plus avancés. Parmi ces différentes implémentations, le bot basé sur une recherche Minimax avec élagage alpha-bêta est le plus abouti et constitue le meilleur bot du projet.

Notre meilleur bot : recherche Minimax à profondeur limitée

Le bot principal du projet est implémenté dans la classe Simulation. Il s'appuie sur une recherche de type Minimax : à partir d'un état de plateau, il simule les coups possibles en alternant deux comportements :

- explMAX(...) cherche le meilleur scénario pour PLAYER1 (maximise le score),
- explMIN(...) cherche le meilleur scénario pour PLAYER2 (minimise le score).

La recherche est limitée par une profondeur maximale (MAXDEPTH = 6) afin d'éviter une explosion du nombre de positions explorées. À chaque nœud, le bot parcourt les cases du plateau et ne considère que les coups légaux (position.isLegal(...)). Le coup choisi au niveau racine est stocké dans bestmove et joué ensuite pendant la simulation.

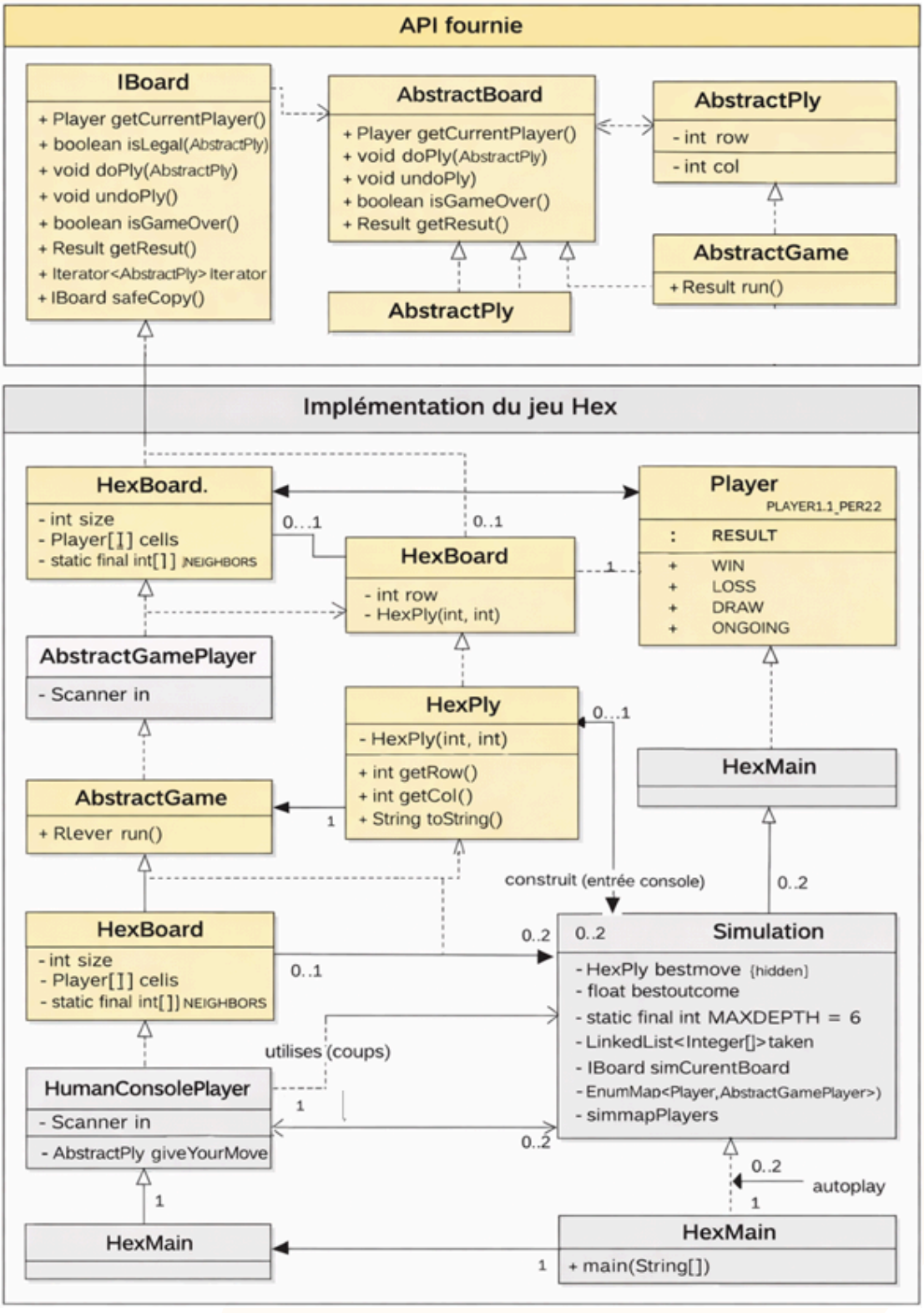
Fonction d'évaluation utilisée

Dans l'état actuel, l'évaluation est volontairement simple et se base surtout sur les fins de partie :

- si la position est gagnante (du point de vue de PLAYER1) : score +1,
- si elle est perdante : score -1,
- si la profondeur limite est atteinte : score 0.

Cela permet déjà d'obtenir un bot cohérent sur des plateaux de taille raisonnable : il privilégie les lignes de jeu menant à une victoire rapide ou évite des pertes évidentes. En contrepartie, l'absence d'une heuristique plus fine (ex : connectivité, distance aux bords, "ponts" typiques de Hex) rend le bot moins performant dès qu'il faut départager des positions non terminales.

Diagrammes de classes



Diagrammes de classes

Le projet s'appuie sur l'API fournie (GameAPI) qui impose une structure classique : un plateau (IBoard / AbstractBoard), des coups (AbstractPly), des joueurs (AbstractGamePlayer) et une boucle de jeu (AbstractGame). Le jeu de Hex est ensuite implémenté en spécialisant ces abstractions avec des classes concrètes.
La classe centrale côté "modèle" est HexBoard, qui représente l'état du plateau (taille size et grille cells). Elle implémente les opérations essentielles du moteur : vérifier la légalité d'un coup (isLegal), appliquer/annuler un coup (doPly, undoPly), produire les coups possibles via l'itérateur (iterator) et déterminer la fin de partie (isGameOver, getResult). L'affichage de debug repose directement sur toString(), ce qui permet d'inspecter rapidement une position en console.

Les coups sont représentés par HexPly, une classe simple qui stocke les coordonnées (row, col). Elle sert d'objet de transfert entre les joueurs (humains ou bots) et le moteur, et elle est utilisée partout où l'API attend un AbstractPly.
Pour les joueurs, HumanConsolePlayer hérite de AbstractGamePlayer et fournit une implémentation concrète de giveYourMove(IBoard). Ce joueur lit une entrée console, vérifie que le format est correct et retourne un HexPly. Cela a été particulièrement utile pendant le développement pour tester le moteur "à la main".

Enfin, la classe Simulation est votre partie "bot". Elle hérite de AbstractGame mais redéfinit run() pour piloter la partie de manière automatique. Concrètement, au lieu d'appeler giveYourMove des joueurs, Simulation calcule un coup avec GiveBestMove, qui déclenche une recherche Minimax via explMAX et explMIN (profondeur limitée par MAXDEPTH = 6). Le choix du meilleur coup est conservé dans bestmove au niveau racine, puis appliqué sur simCurrentBoard. Une liste taken garde une trace des coups joués (utile pour debug/trace console).

La classe HexMain sert de point d'entrée : elle instancie un HexBoard, construit la map de joueurs (EnumMap<Player, AbstractGamePlayer>) et lance soit une partie "classique" avec AbstractGame, soit une partie automatique via Simulation quand l'argument autoplay est fourni.

Conclusion individuelle

Clémence : Ce projet m'a permis de mieux comprendre la mise en place d'un moteur de jeu et l'implémentation d'un bot. Il m'a également aidé à progresser sur l'organisation du code et le travail collaboratif avec Git.

William :

Clément : Ce projet m'a permis de mieux comprendre les aspects théoriques liés à la représentation d'un jeu comme Hex, notamment la gestion du plateau et des règles. Il m'a aussi montré l'importance d'une conception rigoureuse pour garantir la cohérence du jeu et faciliter le travail des autres parties du projet.

Riad :

Alistair :

Conclusion collective

Ce projet a permis à notre groupe de mettre en pratique les notions vues en cours, aussi bien sur le plan algorithmique que sur le plan méthodologique. Malgré certaines difficultés liées à la coordination et à l'intégration des différentes parties, nous avons réussi à produire un projet fonctionnel, documenté et conforme aux attentes.

La répartition des tâches et l'utilisation de branches Git dédiées ont été des éléments clés dans la réussite du projet. Ce travail nous a permis de mieux appréhender les enjeux d'un développement collaboratif sur un projet de taille moyenne.