Justification du projet Hex comme travail final de Programmation Orientée Objet

Jair Vasquez
Santiago Florido
IN204 – Programmation Orientée Objet

***Résumé*—Le jeu de Hex mis en place offre déjà une architecture orientée objet claire et extensible : `Board` encapsule le plateau, `GameState` regroupe la logique de victoire et des coups, `Cube` modélise la géométrie hexagonale. Cette base se prolonge naturellement vers une interface graphique SFML qui continuera d'exploiter et d'enrichir les principes de POO. L'architecture intègre aussi deux IA (heuristique et réseau de neurones graphes TorchScript) branchées sur un Négamax modulaire. Ce document présente de façon globale pourquoi ce projet est un candidat pertinent pour le travail final de POO.**

I. Vue d'ensemble orientée objet

L'application console actuelle (`main.cpp`) illustre l'abstraction du domaine Hex en trois classes :

- **Board** : gère la taille, l'état des cases et l'unique point d'écriture (`place`) pour garantir l'intégrité du plateau.
- **GameState** : fournit les représentations linéaires, les coups disponibles et la détection de victoire (BFS sur les connexions), en s'appuyant sur le modèle de données sans se soucier d'affichage.
- **Cube** : porte le système de coordonnées cubiques et les directions, isolant le calcul géométrique.

La responsabilité de chaque classe est nette et rend le projet aisé à maintenir, tester et faire évoluer. Côté IA, toutes les décisions passent par l'interface `IMoveStrategy` (`RandomStrategy`, `MonteCarloStrategy`, `NegamaxHeuristicStrategy`, `NegamaxGnnStrategy`), injectée dans le runner console ou self-play : la POO garantit l'interchangeabilité des stratégies sans changer la logique de jeu.

II. Justification pour le projet final

Le code déjà en place démontre les notions fondamentales de POO (encapsulation, séparation des responsabilités, modélisation d'objets du domaine). Il est suffisamment complet pour une partie jouable contre une IA heuristique ou une IA GNN (TorchScript) pilotée par `NegamaxStrategy` avec table de transposition, killer/history heuristics et valeurs de Zobrist. L'évaluation change uniquement via le polymorphisme (`NegamaxHeuristicStrategy` vs `NegamaxGnnStrategy`) tandis que la recherche reste inchangée, ce qui illustre la solidité de la conception objet. Le passage à une interface SFML enrichira le projet en illustrant l'extension de classes existantes et l'introduction de nouveaux objets dédiés à l'affichage et aux entrées.

III. Apports POO via l'interface SFML

L'intégration d'une interface graphique SFML permet de continuer la démarche objet :

- Création d'une classe `HexView` responsable du dessin du plateau (tuiles, pions, grille), s'appuyant sur `Board` pour lire l'état courant.
- Introduction d'un `InputHandler` SFML pour transformer les clics souris en coordonnées hexagonales et invoquer `Board::place`, conservant l'API existante.
- Possibilité d'un `GameController` qui orchestre `GameState`, `HexView` et les entrées SFML, sans même modifier les classes de logique ni `IMoveStrategy` : la POO est renforcée par la composition d'objets spécialisés.
- Ajout d'animations ou de surlignages en encapsulant ces comportements dans des objets graphiques dédiés, tout en gardant les règles du jeu au sein de `GameState`.

Cette organisation montre comment l'interface visuelle s'appuie sur les abstractions existantes et comment de nouvelles classes prolongent la structure objet.

IV. IA Hex déjà réalisée : heuristique et GNN

A. Stratégies interchangeables et Négamax

- **Interface unique** : `IMoveStrategy` est implémentée par `RandomStrategy`, `MonteCarloStrategy`, `NegamaxHeuristicStrategy` et `NegamaxGnnStrategy`. `GameRunner` injecte deux stratégies pour générer des parties self-play ou pour le binaire interactif.
- **Recherche factorisée** : `NegamaxStrategy` mutualise l'itérative deepening, l'élagage α–β, une table de transposition (hâchage de Zobrist). Les sous-classes ne changent que l'évaluation : c'est le polymorphisme appliqué à l'IA.

B. Evaluation par réseau de neurones graphes

- **Extraction objet** : `FeatureExtractor` construit et met en cache un `Graph` hexagonal par taille, puis aplatit dix caractéristiques par case (occupation, côtés cibles, degré, distances BFS, joueur au trait) compatibles avec le script Python d'entraînement.
- **Encapsulation TorchScript** : `GNNModel` utilise un patron PIMPL pour isoler libtorch, gérer CPU/CUDA et renvoyer une valeur dans $[-1, 1]$ pour le joueur courant. Si le modèle

`scripts/models/hex_value_ts.pt` manque, la stratégie bascule automatiquement vers l'heuristique.
- **Négamax GNN** : `NegamaxGnnStrategy` compose `FeatureExtractor` et `GNNModel` dans la recherche existante ; l'appel à l'IA GNN se fait donc sans coupler la logique du jeu à libtorch.

C. Pipeline déjà utilisé

- **Self-play** : le binaire `selfplay` orchestre deux stratégies via `GameRunner` et stocke chaque état avec `DataCollector` pour générer des JSONL d'entraînement.
- **Apprentissage** : `scripts/train_gnn.py` entraîne un GNN sur ces données et exporte le TorchScript consommé par C++ ; le binaire interactif (`src/ui/main.cpp`) propose déjà de choisir l'IA heuristique (`h`) ou GNN (`g`).

V. Plan de travail orienté POO pour le jeu complet

La structure actuelle sert de socle pour un ensemble cohérent d'étapes, toutes centrées sur des objets spécialisés :

- **Déjà livré** : moteur console, IA heuristique et IA GNN TorchScript interchangeables via `IMoveStrategy`, générateur self-play et script d'entraînement.
- **À poursuivre** : interface SFML (`HexView`, `InputHandler`, `GameController`) en réutilisant les stratégies existantes ; `Ruleset` paramétrable ; enregistrement/replay (`MatchRecorder`) et gestion des scènes ; tests unitaires supplémentaires sur l'heuristique et le contrôleur.
- **Effets visuels modulaires** : overlays graphiques sans altérer `GameState`.

VI. Conclusion

Le projet Hex combine une base jouable, une architecture orientée objet claire et un potentiel d'extension naturel vers SFML et des modules algorithmiques. Cette continuité du modèle vers la décision et l'affichage justifie pleinement son choix comme travail final.