Chacón José - Meneses Carlos
Tarazona Javier - Liang Tianyi

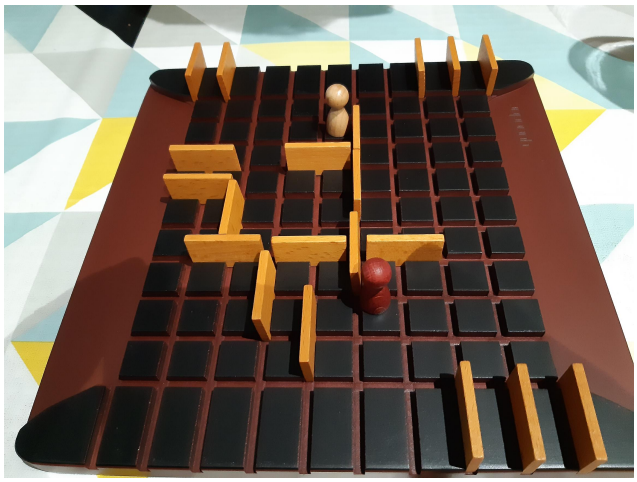
Proposition de Projet IN204 :

Quoridor (C++/SFML) avec IA Heuristique et Architecture POO Moderne

Contexte

Quoridor est un jeu de stratégie à deux joueurs sur une grille 9×9. À chaque tour, le joueur avance son pion d'une case ou place un mur entre deux cases. Le but est d'atteindre la rangée opposée en respectant la règle clé : toujours laisser au moins un chemin possible à l'adversaire.

Objectif. Réaliser un **jeu 2D jouable** en C++ avec **SFML**, présentant une IA simple mais efficace et une **architecture POO** claire. Le projet valorise la conception logicielle, le rendu, la gestion d'événements et les tests.



Plateau de Quoridor (illustration).

Approche et algorithmes

Les jeux à information parfaite sont propices à l'étude de la prise de décision et de l'évaluation des positions. *Quoridor* est pertinent car le plateau *évolue* avec les murs : cela met en avant la **modélisation de graphes** et la **recherche de chemins**.

Idée générale. À chaque tour, choisir entre **progresser** vers la ligne d'arrivée ou **placer un mur** pour ralentir l'adversaire, selon des critères simples : distance estimée, disponibilité de chemins et gestion d'un stock de murs limité.

IA (décision). Nous utiliserons une recherche de type **Minimax** (ou *Négamax*) avec élagage α–β et profondeur bornée pour évaluer les coups prometteurs, tout en gardant l'approche modulaire afin de tester des variantes si nécessaire.

Chemins. Le calcul d'itinéraires s'appuiera sur **A*** (et/ou BFS selon le cas) pour estimer efficacement la progression de chaque joueur et l'impact potentiel des murs.

SFML et structure. Utilisation de `sf::RenderWindow`, `sf::Event`, `sf::Text/Sprite` pour le rendu et les interactions. Architecture **MVC** : Modèle (règles & plateau), Vue (rendu 2D), Contrôleur (souris/clavier, boucle de jeu).

- **POO/encapsulation** : `Board`, `Rules`, `State`, `Game`, `AI`.
- **STL et opérateurs** : conteneurs standards, opérateurs pour logs et replays.
- **Gestion des erreurs** : validations de coups, exceptions simples si besoin.
- **Qualité** : tests unitaires de règles, clarté du code, documentation courte.

Résultats attendus

Livrables principaux. (1) Jeu 2D **fonctionnel** (Humain vs IA/Humain). (2) IA **configurable** et facile à faire évoluer. (3) **Code** C++/SFML structuré avec tests.

Difficulté du CPU. L'utilisateur pourra régler la difficulté (*Facile/Normal/Difficile*). Selon le niveau choisi, le jeu **adaptera** : (a) la profondeur et/ou l'étendue de la recherche Minimax, (b) la **sélection** des coups (activation/désactivation de certains placements de murs), et (c) une **légère randomisation** entre coups de score proche, afin de varier le style de jeu et éviter un comportement trop déterministe.