Exigences du projet Quoridor (C++/SFML)

Équipe: Tarazona Javier, Liang Tianyi

Novembre 2025

Introduction

Ce document synthétise les exigences fonctionnelles et non fonctionnelles du projet *Quoridor (C++/SFML) avec IA*, telles que décrites dans la proposition. L'objectif est de réaliser un jeu 2D jouable en C++ avec SFML, doté d'une IA simple mais efficace, et d'une architecture objet claire et modulaire.

Exigences fonctionnelles

F1. Plateau de jeu **9 × 9** cases conforme aux règles de Quoridor.

F2. **Tours de jeu** alternés entre deux joueurs.

F3. À chaque tour, un joueur doit **soit** déplacer son pion d'une case (orthogonalement, avec sauts autorisés par les règles), **soit placer un mur** entre deux cases dans le respect des contraintes.

F4. **Validation des coups** et des placements de murs :

- Respect des limites du plateau et de l'alignement des murs.
- Interdiction de chevauchement de murs et de placements illégaux.
- **Règle clé** : toujours laisser au moins **un chemin possible** à chaque joueur vers sa rangée d'arrivée.

F5. **Condition de victoire** : un joueur gagne lorsqu'il atteint la rangée opposée.

F6. **Rendu 2D** avec SFML : affichage de la grille, des pions, des murs, et des informations d'interface (textes/indications).

F7. **Gestion des entrées** (souris/clavier) via SFML pour sélectionner, déplacer, et placer des murs.

F8. **Boucle de jeu** événementielle : poll des événements, mise à jour de l'état, effacement, dessin, affichage.

F9. **IA adverse** jouable (Humain vs IA) en plus du mode **Humain vs Humain** local.

F10. **Algorithme de décision IA** : Minimax (ou Négamax) avec **élagage** α**–**β et **profondeur bornée**.

F11. **Évaluation et chemins** : utilisation de **A*** et/ou **BFS** pour estimer les distances, valider l'existence de chemins et guider l'heuristique.

F12. **Difficulté IA configurable** : Facile / Normal / Difficile ajustant notamment la profondeur de recherche, l'étendue de la génération de coups et une légère **randomisation** pour départager des coups de score proche.

F13. **Gestion des erreurs** : messages/retours clairs pour coups invalides ; refus des actions non conformes.

F14. *Optionnel – journalisation/relecture* : opérateurs et structures permettant d'archiver une partie (logs/replay).

Exigences non fonctionnelles

NF1. **Architecture POO modulaire** suivant un style **MVC** :
- *Modèle* : règles, plateau, état (`Board`, `Rules`, `State`).
- *Contrôleur* : boucle de jeu, gestion des entrées.
- *Vue* : rendu SFML (`sf::RenderWindow`, `sf::Event`, `sf::Text`/`sf::Sprite`).

NF2. **Qualité du code** : clarté, documentation courte, usage de STL et surcharge d'opérateurs lorsque pertinent.

NF3. **Testabilité** : **tests unitaires** des règles et validations principales.

NF4. **Performance** : réactivité de l'interface ; recherche IA efficace grâce à l'élagage α–β et heuristiques ; calcul de chemins performant (A*/BFS).

NF5. **Fiabilité** : validations systématiques, gestion d'exceptions simples le cas échéant.

NF6. **Évolutivité** : IA configurable et **facile à faire évoluer** (changement d'heuristiques/profondeurs).

NF7. **Portabilité outillage** : projet standard C++ avec dépendance SFML ; compilation prévue dans un environnement Windows équipé de SFML.

NF8. **Expérience utilisateur** : lisibilité du plateau, interactions simples ; éviter un comportement IA trop déterministe via une randomisation contrôlée.

Livrables et critères d'acceptation

L1. **Jeu 2D fonctionnel** : modes Humain vs IA et Humain vs Humain ; règles appliquées et victoire détectée.

L2. **IA configurable** : au moins trois niveaux (Facile/Normal/Difficile) avec effets visibles sur la force et le style de jeu.

L3. **Code C++/SFML structuré** : architecture POO claire, tests unitaires de règles, documentation d'usage succincte.

Contraintes et hypothèses

— Taille du plateau : 9×9 ; nombre de murs limité par joueur (selon règles usuelles de Quoridor).

— L'IA peut utiliser Minimax ou Négamax indifféremment tant que l'interface reste stable.

— Les chemins sont vérifiés par A* et/ou BFS ; l'implémentation choisie doit garantir la règle du chemin restant.

Notes d'implémentation (informelles)

— Utilisation de SFML : `sf::RenderWindow`, `sf::Event`, `sf::Text`/`sf::Sprite` pour le rendu et les interactions.

— Classes pressenties : `Board`, `Rules`, `State`, `Game`, `AI`.