



SORBONNE UNIVERSITÉ / CFA DES SCIENCES

Projet tuteuré Quoridor

Auteur : Le groupe “TRYD”

Tamer ABOU KARROUM

Rémi RAJARATNAM

Yoan LE NEVEZ

Devamadushan THEVARANCHAN

Chargé de TD: Sébastien TIXEUIL

Résumé

Le projet s'inscrit dans le cadre de l'UE projet tuteuré LU3IN117 visant à développer un logiciel ludique et stratégique inspiré du jeu de plateau Quoridor. Ce projet offre l'opportunité de combiner plusieurs compétences techniques, notamment la conception d'interfaces graphiques modernes, et l'implémentation d'une intelligence artificielle avancée.

1.1 But du Jeu

Le but fondamental du jeu Quoridor est de traverser le plateau en déplaçant son pion depuis sa position de départ jusqu'à atteindre la ligne opposée. Chaque joueur commence la partie positionnée sur une ligne spécifique (par exemple, l'un en haut et l'autre en bas dans une partie à deux joueurs) et doit trouver le chemin le plus rapide et stratégique pour parvenir à l'objectif.

Chaque tour, le joueur peut choisir de déplacer son pion d'une case adjacente (horizontalement ou verticalement) pour progresser vers la ligne d'arrivée. Dans certaines situations, lorsqu'un adversaire bloque directement le chemin, le pion peut effectuer un saut pour contourner l'obstacle.

En plus de déplacer son pion, le joueur dispose d'un nombre limité de murs qu'il peut placer sur le plateau. Ces murs servent à créer des barrières qui ralentissent la progression de l'adversaire. Le challenge consiste à bien doser entre l'avancement personnel et le blocage de l'itinéraire adverse, sans compromettre son propre parcours.

La limitation du nombre de murs impose une gestion stratégique de ces ressources. Placer un mur au mauvais moment ou dans une mauvaise position peut se retourner contre le joueur, car il pourrait se retrouver lui-même bloqué.

Avant chaque coup, le joueur doit analyser le plateau, anticiper les mouvements adverses et envisager plusieurs itinéraires possibles pour atteindre la ligne opposée. La capacité à prévoir plusieurs coups à l'avance est cruciale pour se dégager des impasses.

L'état du plateau évolue constamment en raison du placement des murs. Ainsi, même une stratégie soigneusement planifiée doit pouvoir être adaptée en fonction des actions de l'adversaire. L'anticipation et la réactivité sont des atouts majeurs pour réussir.

Une règle essentielle du jeu est que, même avec le placement de murs, il doit toujours rester au moins un chemin viable pour chaque joueur. Cette contrainte garantit que le jeu reste dynamique et accessible, tout en forçant les joueurs à optimiser l'utilisation de leurs murs.

Synthèse

En résumé, le but du jeu Quoridor est de parvenir le premier à atteindre la ligne opposée du plateau, tout en utilisant intelligemment le déplacement du pion et le placement des murs pour contrer son adversaire. Chaque partie est une véritable épreuve de réflexion et de tactique.

1.2 Options retenues

Afin de concevoir le jeu du Quoridor nous avons choisis deux options.

Une interface graphique moderne

L'interface utilisateur est développée en **JavaFX**, une technologie adaptée aux applications graphiques interactives en Java. Ce choix est motivé par plusieurs facteurs :

Expérience et maîtrise de l'équipe : Tous les membres du groupe ont déjà utilisé Java dont JavaFX, ce qui réduit la courbe d'apprentissage et optimise le temps de développement.

Ergonomie et accessibilité : L'objectif est d'offrir une interface fluide, intuitive et agréable à utiliser, où les joueurs peuvent interagir aisément avec le plateau et les options de jeu.

Flexibilité et modularité : JavaFX permet une séparation claire entre la logique et l'affichage grâce à l'utilisation de fichiers FXML, facilitant ainsi la maintenance et l'amélioration future de l'interface.

Jouer contre un ordinateur intelligent

L'un des axes majeurs du projet est l'intégration d'une intelligence artificielle pour offrir un mode de jeu contre l'ordinateur. L'IA sera développée en Python.

Richesse des bibliothèques IA : Python est reconnu pour sa vaste gamme de bibliothèques spécialisées en intelligence artificielle et en optimisation de décisions, facilitant l'implémentation d'un adversaire performant.

Flexibilité et rapidité de développement : Grâce à sa syntaxe claire et concise, Python permet de tester rapidement différentes approches stratégiques sans nécessiter une lourde implémentation. *Expérience et expertise de l'équipe* : Plusieurs membres du groupe maîtrisent déjà Python et ses outils d'IA, ce qui permet une intégration efficace de l'intelligence artificielle sans difficulté technique majeure.

Adaptabilité et évolutivité : L'utilisation de Python pour l'IA permet de futurs développements basés sur l'apprentissage automatique, offrant ainsi la possibilité d'adapter l'adversaire virtuel au niveau des joueurs en fonction de leurs performances.

1.2.1 Architecture et Technologies Adoptées

Nous avons conçu une architecture modulaire permettant une séparation claire des différentes couches du projet :

Backend: moteur de jeu en Java Le cœur du jeu repose sur un backend développé en Java, qui gère :

La logique du jeu et l'application des règles définies

La gestion des mouvements et des positions des entités (joueurs et murs)

La validation et la vérification des placements de murs

Ce backend assure un fonctionnement fiable du jeu en vérifiant en temps réel la validité des actions effectuées par le joueur ou l'IA.

Frontend JavaFX pour l'affichage L'affichage du jeu est géré par JavaFX permettent d'implémenter l'interface et d'interagir avec la logique du jeu en backend.

Gestion des modules et dépendances Le projet sera organisé en modules et utilisera Maven pour la gestion des dépendances (définies dans **pom.xml**).

1.2.2 Perspectives d'amélioration

Une amélioration envisageable serait le déploiement via Docker pour simplifier la distribution et la gestion des versions de l'application.

En résumé, nous avons opté pour une combinaison de **Java** pour le backend et la logique de jeu car tous les membres ont déjà utilisé ce langage, Python l'intelligence artificielle, et JavaFX pour une interface moderne et réactive.

1.3 Le groupe

Notre équipe se compose de plusieurs membres aux compétences complémentaires. Bien que les noms et rôles définitifs soient à confirmer, nous envisageons la répartition suivante :

Développement de l'intelligence artificielle (Python)

Devamadushan THEVARANCHAN

Tamer ABOU KARROUM

Développement de l'interface graphique (JavaFX)

Yoan LE NEVEZ

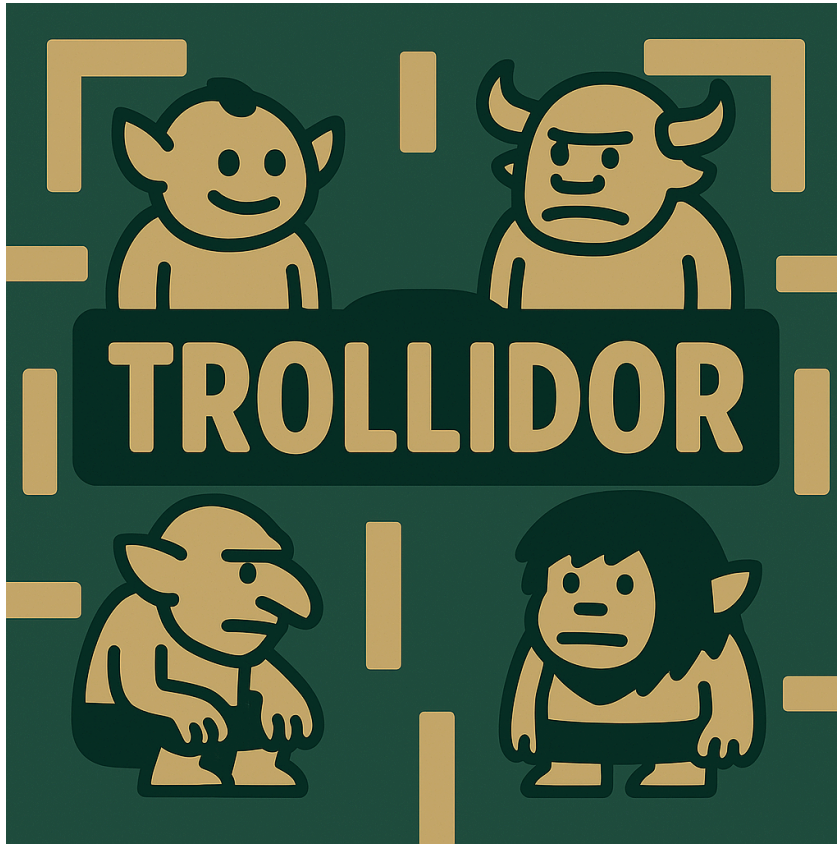
Rémi RAJARATNAM

1.3.1 Mini-CV

Voici nos minis CV avec les projets et expériences pertinentes vis-à-vis du projet se trouvent ci-dessous.

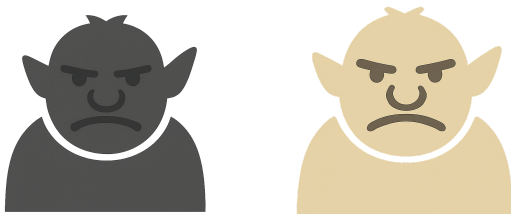
Note: Les rôles pourront être ajustés en fonction de l'évolution du projet.

1.4 Spécification fonctionnelle



L'application “**Trollidior**” de notre groupe **TRYD** est une adaptation moderne du jeu Quoridor dans le monde mythique des Trolls, pensée pour être **simple à utiliser**, **interactive** et **moderne** . Cette spécification décrit **ce que peut faire l'utilisateur**, **ce qui se passe lorsqu'il clique sur un bouton**, la **logique derrière l'interface**, ainsi que les **comportements attendus en cas d'erreurs**.

Le jeu aura comme support visuel le monde mythique des trolls avec comme pions:



Inspiré du minimalisme du jeu d'échecs “chess.com”.

Logique de l'application

1. Lancement de l'application

- L'interface JavaFX démarre en 1024x1024
- Le **menu principal** s'affiche avec les boutons "options", "quitter" et "jouer"

2. Création de partie

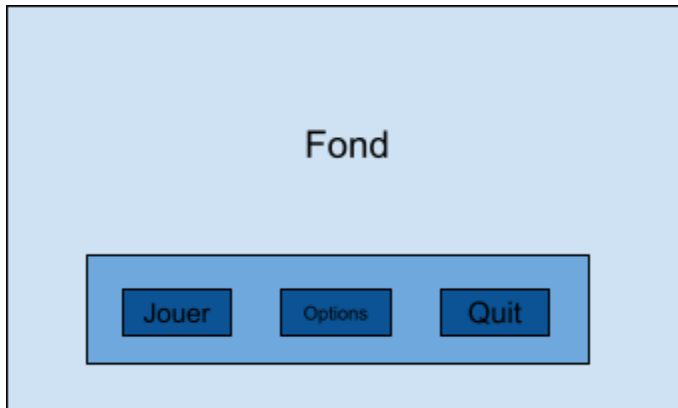
- En cliquant sur "Jouer", une **boîte de paramétrage** permet de configurer :
 - Nombre de joueurs
 - IA (et son niveau de difficulté) ou non
 - Murs par joueur
- En cliquant sur "Confirmer", le jeu démarre

3. Déroulement de la partie

- Les joueurs jouent chacun leur tour
- À chaque tour :
 - Les déplacements possibles sont mis en surbrillance
 - Le joueur peut se déplacer ou placer un mur
- Le jeu vérifie la validité des actions
- Fin de partie si un joueur atteint la ligne opposée

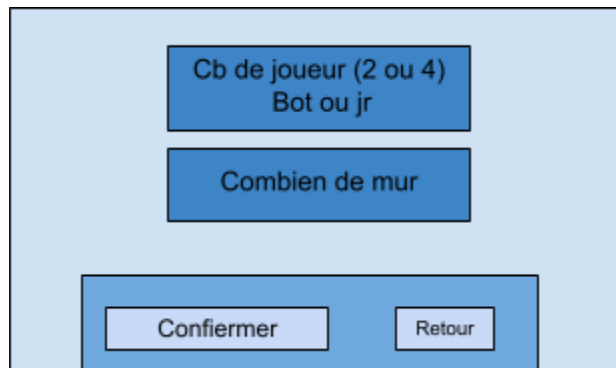
Ce que l'utilisateur peut faire

L'utilisateur peut interagir avec l'application via une interface graphique **cliquable**. Voici les éléments accessibles et leurs effets :

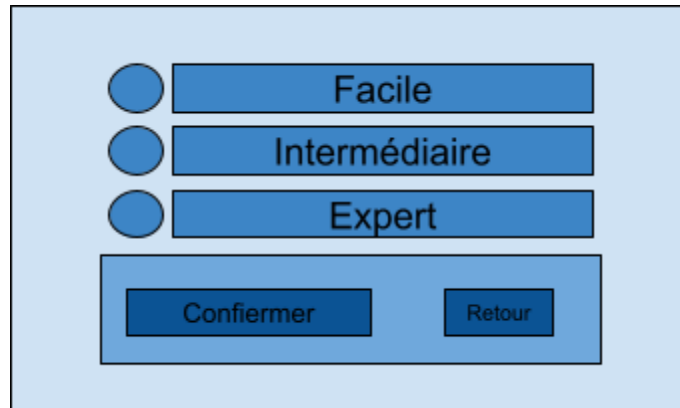


Depuis le menu principal :

- **Jouer :**
 - Ouvre une **fenêtre de configuration de partie**.



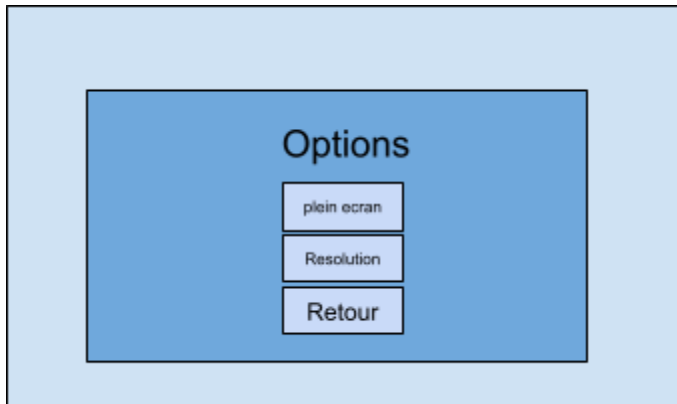
- L'utilisateur peut choisir :
 - Nombre de joueurs (2 ou 4)
 - Jouer contre une IA ou non
 - Si oui ouvre une fenêtre où l'utilisateur doit choisir le niveau de difficulté.



The image shows a graphical user interface for selecting difficulty. It features three radio buttons on the left, each next to a text label in a blue box: 'Facile', 'Intermédiaire', and 'Expert'. Below these options is a container with two buttons: 'Confirmer' and 'Retour'.

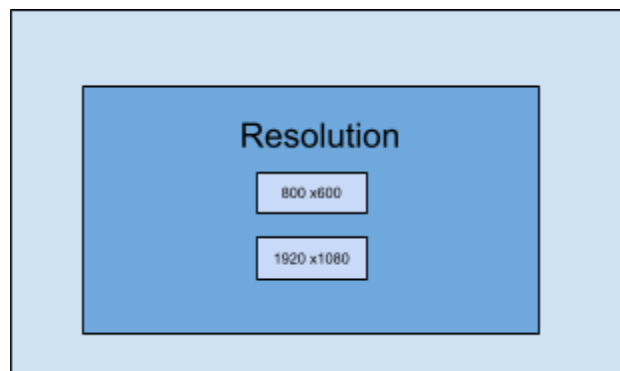
- Nombre de murs par joueur qui doit être saisi
- Un bouton "**Confirmer**" lance la partie avec les paramètres choisis.

- **Options :**



- Accès à un menu permettant :

- De changer la résolution (800x600 ou 1920x1080)



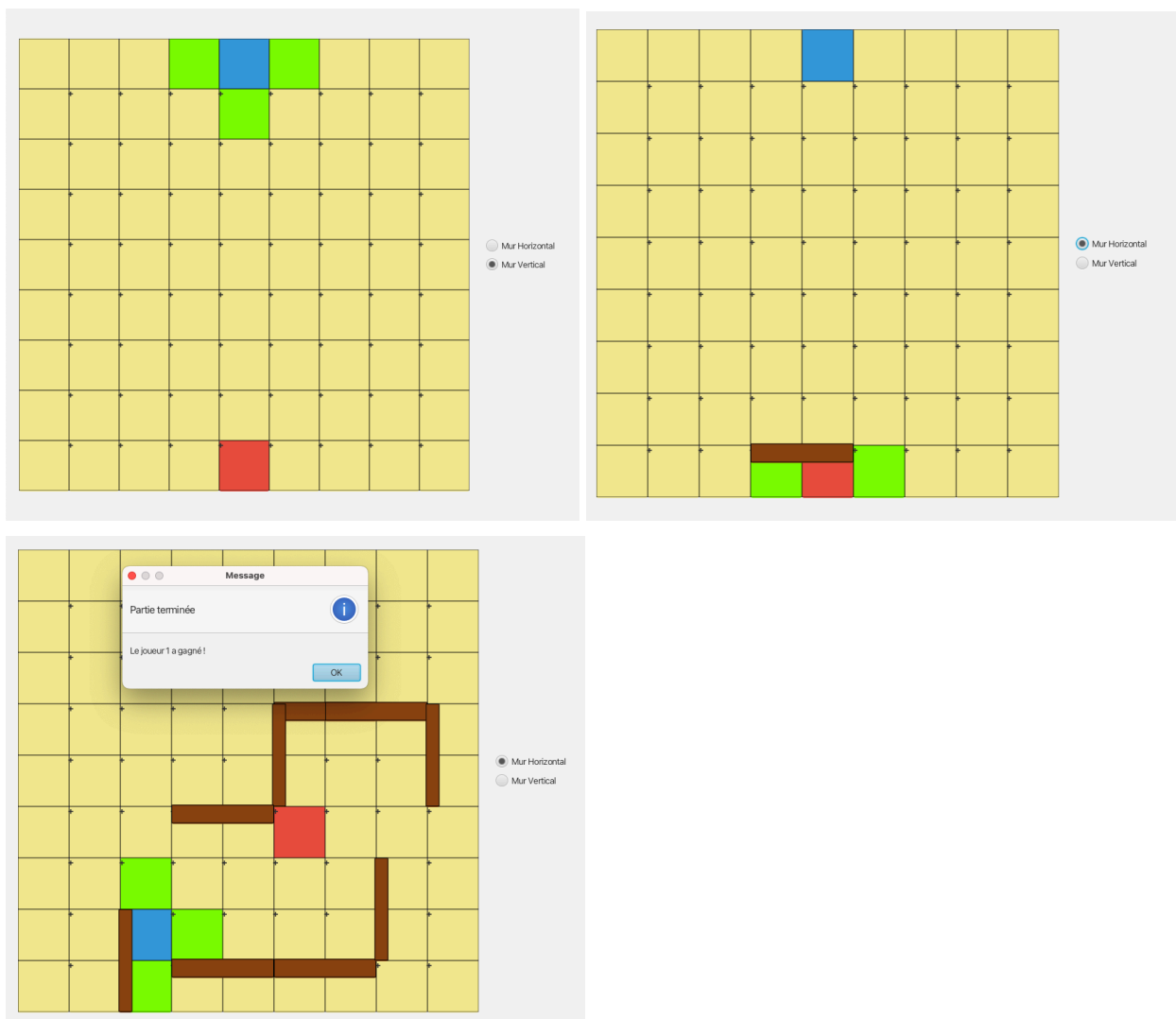
- D'activer/désactiver le plein écran
 - De retourner au menu principal

- **Quitter :**

- Ferme l'application proprement

Depuis l'écran de jeu :

- Cliquer sur une **case valide** (en surbrillance verte) → déplace le joueur.
- Cliquer sur un emplacement mur (sur le screen représenté par un +) choisir orientation (horizontal/vertical) → place un mur.
- Le plateau s'actualise automatiquement après chaque action.
- Une **fenêtre de victoire** s'ouvre quand un joueur atteint son objectif.



Logique de l'application

4. Lancement de l'application

- L'interface JavaFX démarre en 1024x1024
- Le **menu principal** s'affiche avec les boutons "options", "quitter" et "jouer"

5. Création de partie

- En cliquant sur "Jouer", une **boîte de paramétrage** permet de configurer :
 - Nombre de joueurs
 - IA ou non
 - Murs par joueur
- En cliquant sur "Confirmer", le jeu démarre

6. Déroulement de la partie

- Les joueurs jouent chacun leur tour
- À chaque tour :
 - Les déplacements possibles sont mis en surbrillance
 - Le joueur peut se déplacer ou placer un mur
- Le jeu vérifie la validité des actions
- Fin de partie si un joueur atteint la ligne opposée

Cas d'erreurs, exceptions et bugs prévus

Situation	Ce qui se passe	Message/Comportement
Fichier FXML/CSS/Image manquant	L'interface ne charge pas correctement	alerte visuelle si possible
Clic sur case non valide	Aucune action	Silencieux, pas de déplacement
Mur déjà placé	Placement bloqué	Alerte : <i>"Un mur est déjà présent ici."</i>
Mur illégal (bloque passage)	Placement refusé	Alerte : <i>"Chaque joueur doit pouvoir atteindre l'objectif."</i>
Plus de murs disponibles	Blocage de l'action	Alerte : <i>"Vous n'avez plus de murs disponibles."</i>
Changement d'écran rapide (ex. spam du bouton "Retour")	Peut faire planter la navigation	Boutons désactivés temporairement pendant les transitions
Mauvais format saisi dans un champ (ex. murs)	Crash évité	Alerte : <i>"Veuillez saisir un nombre valide."</i>
IA bloquée (aucun coup possible)	La partie se termine	Affichage : <i>"Victoire du joueur humain - IA bloquée."</i>

Comment lancer l'application

Pour démarrer l'application localement :

1. Avoir **Java 21** installé
2. Ouvrir le projet dans un IDE (IntelliJ recommandé)
3. Lancer la classe **JeuQuoridor.java**
4. L'interface s'ouvre directement sur le **menu principal**