

JEU QUORIDOR

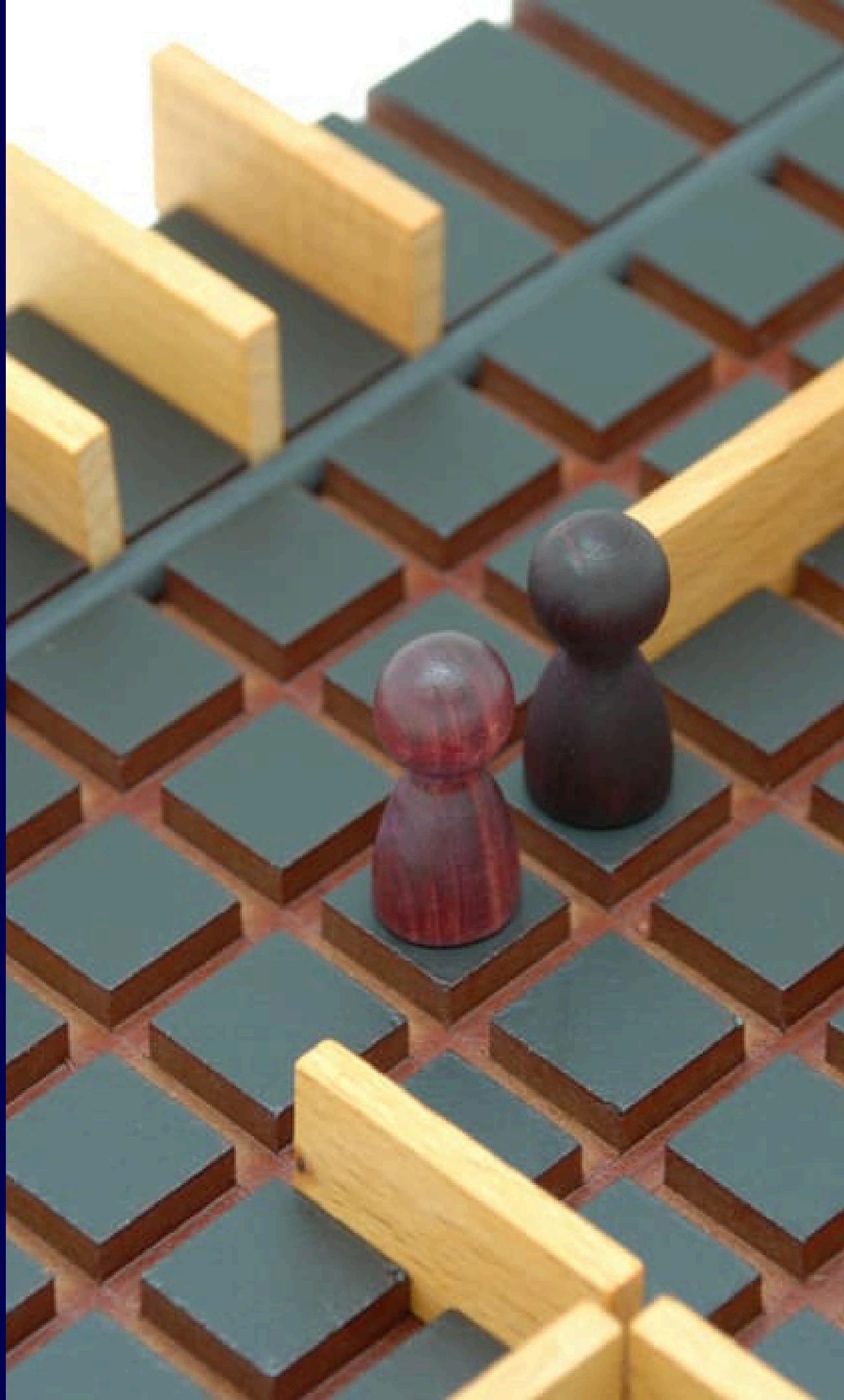
Projet Programmation en Language C - ING3

Al-Amine Maouloud - Jeremy Chen - Ilyes Boukerma - Imane Boussoughi-nahid

S03-GRP : 78

SOMMAIRE

- 1 SYNTHÈSE DU SUJET
- 2 STRUCTURES DE DONNÉES
- 3 EXIGEANCE FONCTIONNELLES PRIORITAIRES
- 4 DIAGRAMME FONCTIONNEL
- 5 EXIGEANCE TECHNIQUE
- 6 ILLUSTRATION DES FONCTIONNALITÉS
- 7 ALGORITHME
- 8 RÉPARTITION DES TACHES
- 9 GRAPHE D'APPELS
- 10 VERSIONNING GIT
- 11 SCREENSHOTS DES RÉSULTATS
- 12 BILAN
- 13 SOURCES DOCUMENTAIRE



1. SYNTHÈSE DU SUJET

Le projet consiste à développer une version console du jeu de société "**Quoridor**", mettant en œuvre des compétences en programmation C et gestion de projet en équipe. Le jeu se joue sur un plateau de 9x9 cases avec pour objectif d'atteindre le bord opposé avant ses adversaires, tout en utilisant des barrières pour ralentir leur progression.

‘**objectif** est de développer un jeu jouable à 2 ou 4 joueurs, humains ou intelligences artificielles (IA), de mettre en place une interface console, incluant un menu pour gérer les parties, consulter les scores et afficher une aide et enfin d’implémenter les règles du jeu, incluant le déplacement des pions, le placement des barrières et la gestion des situations particulières (saut de pion, blocages partiels).

2. STRUCTURES DE DONNÉES

CETTE STRUCTURE REPRÉSENTE UN JOUEUR DANS LE JEU, AVEC SES ATTRIBUTS PRINCIPAUX.

```
typedef struct {  
    char nom[MAX_NAME_LENGTH];  
    char type; // type pour soit un humain soit une IA  
    char jeton; // Jeton du joueur  
    int mursRestants; // Nombre de barrières restantes  
    int score; // Score du joueur  
} Joueur;
```

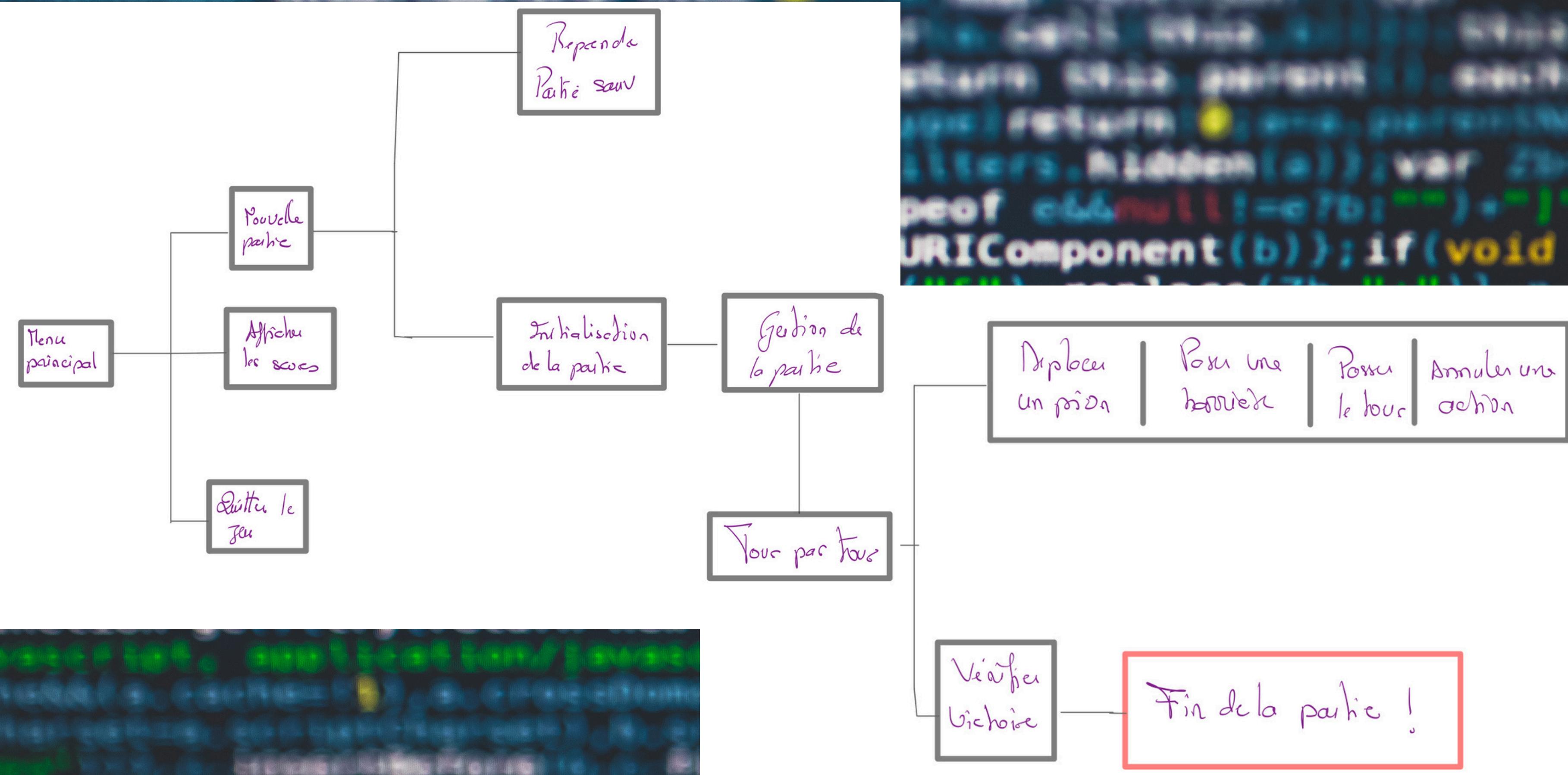
REPRÉSENTE UNE ACTION DANS L'HISTORIQUE, UTILISÉE POUR ANNULER LA DERNIÈRE ACTION.

```
typedef struct {  
    int x;  
    int y;  
    int type;  
} Action;
```

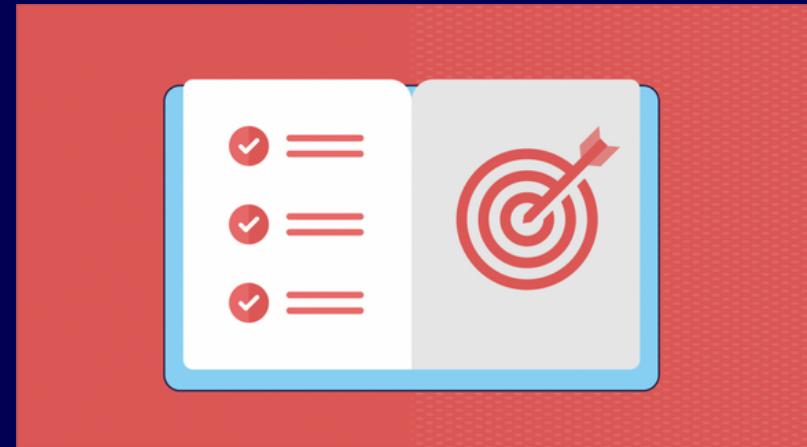
3. EHIGENCES FONCTIONNELLES



ID	Exigence fonctionnelle	Description	Priorité
1	Affichage du plateau	Afficher un plateau de 9x9 cases avec espaces pour les barrières. Permettre l'interaction sur les cases.	Très haute
2	Gestion des joueurs	Supporter 2 ou 4 joueurs avec : nom, type (Humain/IA), jeton unique, nombre de barrières disponibles.	Très haute
3	Déplacement des pions	Permettre aux joueurs de déplacer leur pion selon les règles : cases adjacentes ou saut valide.	Très haute
4	Placement des barrières	Placer des barrières entre les cases : orientation correcte, vérification des contraintes.	Très haute
5	Conditions de victoire	Déetecter la victoire lorsqu'un joueur atteint le bord opposé du plateau.	Très haute
6	Menu principal	Proposer un menu avec les options : nouvelle partie, reprendre partie, afficher l'aide, afficher scores, quitter.	Très haute
7	Affichage et interaction utilisateur	Afficher le plateau, les informations des joueurs, et proposer les actions possibles (déplacement, etc.).	Haute
8	Gestion des scores	Sauvegarder et charger les scores, mettre à jour les points après chaque partie.	Haute
9	Annulation des actions	Permettre d'annuler la dernière action effectuée (déplacement ou barrière).	Moyenne



5. EXIGENCES TECHNIQUES :



- Gestión du plateau : Gère l'affichage du plateau, les déplacements, et les barrières, plateau 2D
- gestion des joueurs : Gère les actions des joueurs, leur état (position, barrières restantes)
- gestion du score : Gère l'attribution des points et le suivi des scores des joueurs.
- Interaction (menu, saisie utilisateur, etc.) : Gère le menu principal, la navigation dans le jeu et l'entrée des actions du joueur (fichier.h et .c)

6. EXEMPLES DE FONCTIONNALITÉS :

```
int poserBarriere(char plateau[ROWS][COLS], Joueur joueurs[], int joueurActif) {
    if (joueurs[joueurActif].mursRestants <= 0) {
        printf(format: "Vous n'avez plus de barrières disponibles.\n");
        return 0;
    }

    char saisie[10];
    int x1, y1, x2, y2;
    char direction;

    while (1) {
        printf(format: "%s, entrez la position de la barrière que vous voulez poser (ex : D1 D2 G) : ", joueurs[joueurActif].nom);
        scanf(format: "%[^\\n]", saisie);

        if (validerSaisieBarriere(saisie, &x1, &y1, &x2, &y2, &direction)) {
            // Vérification selon la direction et la règle spécifique
            if (direction == 'H') {
                if (x1 - 1 < 0 || plateau[x1 - 1][y1] == 'B' || plateau[x2 - 1][y2] == 'B' ||
                    plateau[x1 - 1][y1] == ' ') { // Règle : pas de case blanche
                    printf(format: "Erreur : emplacement invalide pour une barrière horizontale.\n");
                    continue;
                }
                plateau[x1 - 1][y1] = 'B';
                plateau[x2 - 1][y2] = 'B';
            } else if (direction == 'B') {
                if (x1 + 1 >= ROWS || plateau[x1 + 1][y1] == 'B' || plateau[x2 + 1][y2] == 'B' ||
                    plateau[x1 + 1][y1] == ' ') { // Règle : pas de case blanche
                    printf(format: "Erreur : emplacement invalide pour une barrière verticale basse.\n");
                    continue;
                }
                plateau[x1 + 1][y1] = 'B';
                plateau[x2 + 1][y2] = 'B';
            } else if (direction == 'G') {
                if (y1 - 1 < 0 || plateau[x1][y1 - 1] == 'B' || plateau[x2][y2 - 1] == 'B' ||
                    plateau[x1][y1 - 1] == ' ') { // Règle : pas de case blanche
                    printf(format: "Erreur : emplacement invalide pour une barrière gauche.\n");
                    continue;
                }
                plateau[x1][y1 - 1] = 'B';
                plateau[x2][y2 - 1] = 'B';
            }
        }
    }
}
```

```
// Demander le pion du joueur
char pion;
int pionValide;
do {
    pionValide = 1; // Réinitialisation de la validité
    printf(format:"Choisissez un pion : \n");
    scanf(format:"%s", buffer);

    // Vérifier si l'utilisateur a entré plus d'un caractère
    if (strlen(buffer) != 1) {
        printf(format:"Erreur : vous devez entrer UN seul caractere. Veuillez reessayer.\n");
        pionValide = 0;
        continue;
    }

    // Vérifier si le pion est déjà pris
    for (int j = 0; j < i; j++) {
        if (joueurs[j].jeton == buffer[0]) {
            printf(format:"Ce pion est deja pris. Veuillez en choisir un autre.\n");
            pionValide = 0;
            break;
        }
    }
} while (!pionValide);
```

2. ALGORITHMES DE FONCTIONNALITÉS

Titre : Algorithme de déplacement du pion

- Demande la saisie du déplacement
- Vérifier la direction du déplacement (haut, bas, gauche, droite)
- Vérifier s'il existe une barrière sur le chemin ou qu'il ne dépasse pas la limite du tableau
- Vérifie si le pion peut sauter sur un autre pion
- Afficher l'état du plateau après le mouvement.

2. ALGORITHMES DE FONCTIONNALITÉS

Titre : Algorithme de placement de barrière

- Demander la saisie des coordonnées des deux cases adjacentes
- Demander la direction (haut, bas, gauche, droite) pour placer la barrière
- Vérifie si la barrière est valide
- Vérifier que la barrière n'entraîne pas un blocage total du chemin de l'adversaire
- Placer la barrière sur le plateau.
- Mettre à jour la zone de stockage des barrières.

8. RÉPARTITION DES TACHES

Al-Amine

- Initialisation du plateau
- Sauvegardes des scores
- Sauvegarde de la partie

Jeremy

- L'ensemble des blindages
- Déplacement des pions
- Gestions des tours

Ilyes

- Placement des barrières
- Annulation d'un coup
- Gestion des Interactions

Imane

- Affichage du scores
- Affichage de l'aide
- Menu Principal

8. GRAPHE D'APPELS

```
// Fonction pour choisir un joueur au hasard  
> int choisirJoueurAuHasard(int nombreJoueurs){...}  
  
// Fonction pour initialiser le générateur de nombres aléatoires  
> void initialiserAleatoire(){...}  
  
> void Color(int couleurDuTexte, int couleurDeFond){...}  
  
// Fonction pour charger les scores depuis un fichier  
> int chargerScores(Joueur joueur[], int *nombreScores){...}  
  
// Fonction pour sauvegarder les scores dans un fichier  
> void sauvegarderScores(Joueur joueur[], int nombreScores){...}  
  
// Fonction pour trouver un joueur existant dans les scores  
> int trouverJoueur(Joueur joueur[], int nombreScores, const char *nom){...}  
  
// Fonction pour mettre à jour ou ajouter un score  
> void mettreAJourScore(Joueur joueur[], int *nombreScores, const char *nom, int points){...}  
  
// Fonction pour afficher les scores  
> void afficherScores(Joueur joueur[], int nombreScores){...}  
  
> void afficherScoresDepuisFichier(){...}  
  
// Fonction pour gérer les scores en fin de partie  
> void gestionScoresEnFinDePartie(Joueur joueurs[], int nombreJoueurs){...}  
  
> void afficherPlateauEtInfos(char plateau[ROWS][COLS], Joueur joueurs[], int nombreJoueurs, int joueurActif){...}  
  
> void sauverPartie(char plateau[ROWS][COLS]){...}  
  
> void reprise_partie(char plateau[ROWS][COLS], Joueur joueur[], int nombrejoueurs){...}  
  
> void sauvejoueurs(Joueur joueur[], int nombrejoueurs){...}
```

```
> void reprise_partie(char plateau[ROWS][COLS], Joueur joueur[], int nombrejoueurs){...}  
  
> void sauvejoueurs(Joueur joueur[], int nombrejoueurs){...}  
  
> void reprise_joueurs(Joueur joueur[], int *nombrejoueurs){...}  
  
> void sauvegarderpartie(char plateau[ROWS][COLS], Joueur joueur[], int nombrejoueurs){...}  
  
> void reprendrePartie(char plateau[ROWS][COLS], Joueur joueur[], int *nombrejoueurs){...}  
  
> void initialiserPlateau(char plateau[ROWS][COLS], Joueur joueurs[], int nombreJoueurs){...}  
  
> int verifierVictoire(char plateau[ROWS][COLS], Joueur joueurs[], int joueurActif, int nombreJoueurs){...}  
  
> int validerDeplacement(int x1, int y1, int x2, int y2, char plateau[ROWS][COLS]){...}  
  
> int déplacerPion(char plateau[ROWS][COLS], Joueur joueurs[], int joueurActif){...}  
  
> int validerSaisieBarriere(char* saisie, int* x1, int* y1, int* x2, int* y2, char* direction){...}  
  
> int poserBarriere(char plateau[ROWS][COLS], Joueur joueurs[], int joueurActif){...}  
  
> /* void annulerAction(char plateau[ROWS][COLS], Joueur joueurs[], int joueurActif, int nombreJoueurs){...}  
  
> void passerSonTour(int joueurActif, int nombreJoueurs){...}  
  
> void lancerNouvellePartie(){...}  
  
> void continuer_partie(char plateau[ROWS][COLS], Joueur joueur[], int nombrejoueurs){...}  
  
> void afficherAide(){...}  
  
> void afficherMenu(){...}  
  
▷ > int main(){...}
```

10. VERSIONING GIT

This branch is 21 commits ahead of, 4 commits behind [main](#).

Jeremy-C92 Update main.c · 4594081 · now · 23 Commits

.idea	J'ai modif	last week
.gitignore	j'ai configurer	last week
CMakeLists.txt	j'ai configurer	last week
main.c	Update main.c	now

This branch is 6 commits ahead of, 4 commits behind [main](#).

Jeremy-C92 Update main.c · b49a86e · 2 days ago · 8 Commits

.idea	J'ai fini	last week
.gitignore	j'ai configurer	last week
CMakeLists.txt	j'ai configurer	last week
main.c	Update main.c	2 days ago

This branch is 28 commits ahead of, 3 commits behind [main](#).

Jeremy-C92 Update main.c · 70b9e4b · 1 hour ago · 30 Commits

.idea	J'ai fini	last week
.gitignore	j'ai configurer	last week
CMakeLists.txt	j'ai configurer	last week
main.c	Update main.c	1 hour ago

main · 4 Branches · 0 Tags

alamine-sm jeremy j'ai embellie le menu aussi prend cette version la elle est tro... · 85b3c4b · 1 hour ago · 10 Commits

.idea	J'ai modifié	last week
.gitignore	j'ai configurer	last week
CMakeLists.txt	j'ai configurer	last week
main.c	jeremy j'ai embellie le menu aussi prend cette version la elle e...	1 hour ago

Alamine (a dû retravailler sur le main général car branche supprimé)

11. SCREENSHOTS DES RÉSULTATS

A B C D E F G H I

1			B						
2				B J					
3									
4				B					
5				B					
6									
7				B		B			
8				B	I				
9									

Nombre de joueurs : 2

Joueur 1 : Jeremy (Jeton : J) | Score : 0 | Murs restants : 8 <-- Tour en cours

Joueur 2 : Ilyes (Jeton : I) | Score : 0 | Murs restants : 8

C'est au tour de Jeremy (Jeton : J) de jouer.

Actions possibles :

- 1 - Déplacer son pion
- 2 - Poser une barrière
- 3 - Passer son tour
- 4 - Annuler la dernière action

SCREENSHOTS DES RÉSULTATS



A terminal window showing a 9x9 grid board. The board has a pattern of alternating light gray and dark gray squares. In row 3, column D, there is a white square containing the letter 'A'. In row 7, column E, there is a white square containing the letter 'J'. The text "C'est au tour de Alamine (Jeton : A) de jouer." is displayed above the board.

Actions possibles :

- 1 - Deplacer son pion
- 2 - Poser une barriere
- 3 - Passer son tour
- 4 - Annuler la derniere action (non disponible)
- 5 - Quitter le jeu

Choisissez une action (1, 2, 3, 4 ou 5):

5

Vous avez quitté la partie.

SCORES DES JOUEURS	
Nom	Points
Alamine	5
Jeremy	5

Appuyez sur une touche pour continuer... |

12. BiLAN

	Tâches réalisées	Progression (%)	Compétences acquises	Points d'améliorations
Al-Amine	<ul style="list-style-type: none"> Initialisation du plateau Sauvegardes des scores Sauvegarde de la partie Validité de la victoire 	100% 100% 100% 100%	<ul style="list-style-type: none"> Conception et initialisation d'un plateau interactif en console Stockage pour la sauvegarde des scores et de la partie 	<ul style="list-style-type: none"> Optimisation du stockage des scores
Jeremy	<ul style="list-style-type: none"> L'ensemble des blindages Déplacement des pions Gestions des tours 	99% 95% 100%	<ul style="list-style-type: none"> Gestion des déplacements des pions en respectant les règles Implémentation de la logique des tours Algorithmes pour la validation des blindages 	<ul style="list-style-type: none"> Détection des cas limites pour les pions (ex : saut d'un pion même si une barrière sépare les 2 pions)
Ilyès	<ul style="list-style-type: none"> Placement des barrières Annulation d'un coup Gestion des Interactions 	95% 100% 90%	<ul style="list-style-type: none"> Algorithmes pour le placement des barrières en respectant les contraintes Gestion des interactions entre les joueurs 	<ul style="list-style-type: none"> Renforcement des règles de validation des barrières Gérer l'annulation des actions
Imane	<ul style="list-style-type: none"> Affichage du score Affichage de l'aide Menu Principal 	100% 100% 100%	<ul style="list-style-type: none"> Gestion de l'interface utilisateur : affichage des scores et du menu principal Navigation fluide dans le menu principal 	<ul style="list-style-type: none"> Rédaction d'une aide plus détaillée et illustrée

13. SOURCES DOCUMENTAIRES

Programmation en C

- Documentation détaillée du langage C (Wikibooks)
- Tutoriel vidéo sur GitHub Desktop (YouTube)

Algorithmes utiles pour Quoridor

- Concepts de base des graphes Quoridor(Wikiversité)
- Gestion des structures de données avancées en C

Etudes documentaires

- Programmation des jeux avec le langage C (YouTube)
- Guide sur les jeux de plateau et IA
- Cours et exercices d'algorithmique (internet)

Outils pour visualisation et IDE

- Clion (éditeur pour C)
- Tuto pour créer fichiers .h/.c