

Projet langage C Semestre 1

Le jeu du Quoridor



Table des matières

Description du jeu.....	2
Règle du jeu.....	2
Préparation du jeu	2
Principe du jeu.....	3
Menu.....	4
Fin de partie.....	4
Contraintes	4
Exemple illustré en console.....	4
Barrières	5
Interruption de partie.....	5
Sortie du jeu.....	5
Versioning de votre projet : GIT	5
Organisation et Planning	6
Soutenance	6
Annexes	7
1- Table ANSI	7
2- La manipulation du curseur sous Windows.....	8
3- La gestion de la couleur sous Windows	8

Description du jeu

Quoridor (marque déposée) est un jeu de société dit jeu de stratégie combinatoire pour 2 ou 4 joueurs. Il a été conçu par Mirko Marchesi et édité en 1997. Ce jeu se joue sur un plateau.

Le but est d'atteindre le premier le bord opposé à sa ligne de départ

Règle du jeu

Les règles du jeu de ce projet sont inspirées des sites suivants :

- ❖ [Quoridor ,Jeu de réflexion et de société ,Gigamic](#)
- ❖ Illustré avec des exemples : [Quoridor : règles \(letempledujeu.fr\)](#)

Préparation du jeu

Participants

Pour jouer, il est prévu soit :

- 2 joueurs dont les pions sont placés l'un en face de l'autre sur le plateau.
- 4 joueurs dont les pions sont positionnés sur un bord de plateau chacun.
- Le choix du nombre de joueurs est à faire avant de lancer la partie.
- Chaque joueur est soit humain soit une IA de l'ordinateur, à choisir avant de lancer la partie.
- Chaque joueur saisit son nom.

Matériel

- Plateau de 9 cases par ligne et 9 cases par colonne. Entre les lignes et entre les colonnes il y a un emplacement prévu pour planter les murs du corridor (ou Qoridor) !
- 20 barrières.
- Chaque barrière a une longueur de 2 cases et se positionnera entre deux lignes ou deux colonnes.
- 1 zone de stockage par joueur des barrières en début de partie (10 barrières par joueur pour 2 joueurs ou 5 barrières pour 4 joueurs).
- Un pion par joueur avec un jeton à choisir parmi un caractère de son choix, qui peut être aussi semi-graphique : voir annexe [table ANSI](#).
- Il est possible de colorier les cases du plateau, les barrières, les pions (jetons) pour une visualisation plus agréable du jeu : voir annexe [La gestion de la couleur sous Windows](#).

Chaque joueur aura au départ un pion et des barrières à sa disposition dans sa zone de stockage. Les barrières sont distribuées équitablement entre les joueurs.

Chaque joueur pose son pion au bord et centre de sa ligne de départ (1 des côtés du plateau).

Principe du jeu

Partie

L'ordre de jeu des joueurs est défini par hasard. Le 1^{er} joueur tiré au hasard commencera la partie etc.

Chaque joueur joue à tour de rôle avec son nom affiché. Il choisit de déplacer son pion ou de poser une de ses barrières

Après ce 1^{er} tour, chaque joueur peut soit :

- Déplacer son pion d'une case : verticalement ou horizontalement, en avant en ou en arrière. Un déplacement ne se fera entre deux cases que si aucune barrière n'est présente entre elles. Si le déplacement est possible alors le pion change de case, sinon, le joueur rejoue. Quand le pion est sur le plateau pour une partie, il ne pourra en sortir. Pour déplacer le pion avec le curseur, voir l'annexe [La manipulation du curseur sous Windows](#).
- Poser une barrière entre deux blocs de cases : verticalement ou horizontalement. Ceci est possible tant que le joueur a encore des barrières dans sa zone de stockage. Pour cela, le joueur saisit les coordonnées des 2 cases adjacentes, ainsi que la position de la barrière par rapport à elles. Les barrières ne sont jamais posées à l'extérieur.
NB : le but de la barrière est de créer son propre chemin ou de ralentir un adversaire. Dans ce dernier cas, il est interdit de fermer totalement l'accès à sa ligne de but de son(ses) adversaire(s).
- Passer son tour
- Annuler le coup du joueur. Cette option permet d'annuler uniquement la dernière action faite sur le plateau. Il n'y a qu'une seule annulation possible en cas de demande. La demande se fait par le joueur en cours.

Le joueur saisit l'action qu'il va faire dans le menu.

Situations particulières

✓ Déplacement :

- Deux pions sont l'un en face de l'autre et leur case n'est pas séparée par une barrière. Le pion du joueur passe au-dessus de son adversaire et prend la prochaine case vide. Cette dernière est située dans la même direction si aucune barrière n'est placée après le pion survolé sinon le joueur pourra bifurquer à gauche ou à droite selon son choix et selon les éventuelles barrières présente.
- Dans le cas où la partie se fait avec 4 joueurs, il n'est pas possible de sauter plus d'un pion à la fois.

✓ Blocage de la partie :

- Il n'y aucun gagnant.

Calcul des scores

- Le nom de chaque joueur sera stocké pour la partie. S'il a déjà joué et est connu alors il prendra son score précédemment sauvegardé.
- Le score de chaque joueur est calculé à la fin de chaque partie. Si je joueur a déjà joué alors on mettra à joueur son ancien score en lui ajoutant celui de la partie jouée pour obtenir le score final.
- Chaque partie rapporte au gagnant 5 points. En cas de blocage, aucun point n'est attribué.
- Le nom de chaque joueur sera stocké pour la partie. S'il a déjà joué et est connu alors il prendra son score précédemment sauvegardé.
- Si le joueur est nouveau alors le score final sera celui de la partie
- Dans les parties bonus, les cases primées rapportent au joueur le nombre de points indiqué. Ce sera ajouté à son score.
- Dans tous les cas, le score final sera sauvegardé dans un fichier texte. Aucun score existant avant la partie ne sera perdu.

Timer

En tournoi, il est possible d'allouer à chaque joueur un temps limité, ce qui ajoute à la difficulté.

Menu

Le menu du jeu permettra aux joueurs de :

- Lancer une nouvelle partie
- Reprendre une partie sauvegardée
- Afficher l'aide
- Afficher les scores des joueurs
- Quitter le jeu

Fin de partie

Le premier joueur qui atteint une des 9 cases de la ligne opposée à sa ligne de départ gagne la partie (= bord opposé de celui de départ sur le plateau).

Contraintes

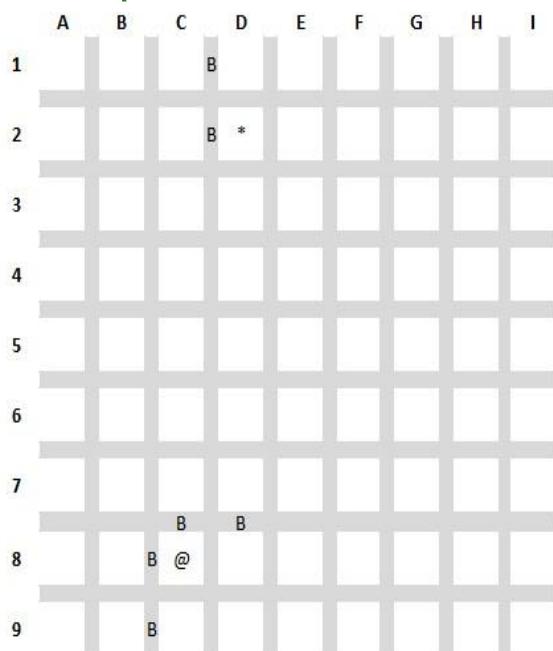
Le plateau de jeu

Le jeu est présenté en mode console.

Pour le jeu de base est constitué de 9 lignes et 9 colonnes (soit 81 cases pour les jetons).

Ces lignes et ces colonnes ont un espace suffisant pour placer horizontalement ou verticalement des murs entre eux. La longueur des murs est celle de 2 cases de plateau.

Exemple illustré en console



Nombre de joueurs : 2

Joueur : Eren

Score partie : 0

jeton : @

Mur restant : 9

Actions possibles :

- 1 - déplacer son pion
- 2 - poser une barrière
- 3 - passer son tour
- 4 - annuler la dernière action

Barrières

- ✓ Les barrières se mettent entre deux lignes ou deux colonnes. Leur longueur est de 2 cases.
- ✓ Elles sont utilisées pour tracer son chemin ou ralentir son adversaire.
- ✓ Pour les positionner, il ne faut pas bloquer totalement le chemin de son adversaire. Pour les positionner, il y a les 2 options suivantes, en rappelant que chaque barrière a une longueur de 2 cases :
 - 1) Le joueur saisira les coordonnées de 2 cases voisines et précisera si la barrière est mise en haut, en bas, à droite ou à gauche de ces cases, en vérifiant qu'entre ces 2 cases il n'y a pas déjà une barrière :
Exemple de saisie de case positionnée à gauche des cases D1 et D2 : D1 D2 G
 - 2) Déplacer le début et fin de barrière, en respectant la longueur des 2 cases et qu'elle soit entre 2 cases pas déjà occupée par une barrière. Pour déplacer une barrière, aidez-vous de l'annexe [La manipulation du curseur sous Windows](#).
- ✓ En cas d'impossibilité, le coup n'est pas pris en compte et le joueur rejoue.
- ✓ Elles sont représentées par la lettre 'B' sur le plateau dans l'[Exemple illustré en console](#) ci-dessus.

Interruption de partie

Il est possible d'interrompre la partie en cours. Dans ce cas la partie est sauvegardée dans un fichier. Elle pourra être reprise ultérieurement. Ce sera alors la dernière partie interrompue qui sera chargée.

Sortie du jeu

En fin d'opération (partie ou autre option), l'application sera toujours redirigée sur le menu principal. L'application ne terminera que si l'option 'Quitter' est choisie.

Versioning de votre projet : GIT

Pour ce projet, vous devrez créer votre compte sur GitHub classroom en suivant les instructions de la page BoostCamp [Section : GitHub classroom | Projet Informatique 1 | BOOSTCAMP \(omneseducation.com\)](#).

PS : Pour l'utilisation de git (soit en mode commande sur console comme sur le tuto [Versioning avec Git](#), soit avec des logiciels appropriés comme git Tortoise, github desktop ... plus simples d'utilisation pour des débutants), je vous laisse vous débrouiller en cherchant sur le net. **Pour plus de détails sur Git, consulter les tutoriaux de la page BoostCamp ci-dessus.**

Travail à faire : Jeu de base obligatoire (20 points)

Pour le jeu de base, les fonctionnalités suivantes doivent être respectées tout en respectant aussi dans le chapitre des [Contraintes](#) citées plus haut :

- Plateau de jeu et de la partie droite d'information et d'interaction : voir [Exemple illustré en console](#) plus haut.
- Menu : comme indiqué dans le [Menu](#) plus haut.
- Jeu à 2 ou 4 joueurs comme énoncé dans [Participants](#) plus haut dans le chapitre [Préparation du jeu](#).
- Déplacement de chaque pion, en tenant compte des contraintes citées plus haut dans [Partie](#) et [Situations particulières](#).

- Placement des barrières, en tenant compte des contraintes citées plus haut dans [Barrières](#).
- L'interruption de partie telle que citée dans [Interruption de partie](#) doit être respectée.
De même, la [Sortie du jeu](#) doit être aussi respectée.

Bonus possibles (4 points)

Ces idées de bonus possibles ci-dessous ne comptent que si chacune des fonctionnalités du Travail à faire : Jeu de base obligatoire (20 points) est tout ou en partie fonctionnelle. Si au moins une de ces fonctionnalités n'est pas implémentée même en partie dans le code, les bonus ci-dessous ne compteront pas, question de priorité du jeu de base par rapport au(x) bonus proposés ci-dessous.

- Mise en place d'un chronomètre par joueur comme indiqué dans [Timer](#). Celui qui aura le plus grand temps en fin de partie sera pénalisé de 2 points dans son score.
- Il est possible d'utiliser des caractères de couleur comme montré dans l'annexe [La gestion de la couleur sous Windows](#), comme pour mettre en avant les cases particulières.
- Mise en place d'une partie avec un ou plusieurs joueurs de type ordinateur (type IA = Intelligence de l'ordinateur). Le comportement de ce joueur peut être très basique ou plus évolué.
- Mise en place de conseil de placement d'un pion ou d'une barrière pour un joueur humain
- Possibilités pour un jeu à 2 joueurs d'augmenter la taille du plateau (nous irons au maximum jusqu'à une grille de 12 lignes et 12 colonnes. Pour cela, la réserve totale de barrière sera de 40).
- Possibilité d'améliorer la grille d'origine avec des cases bonus. Dans ce cas, le pion est posé sur cette case (et uniquement dans ce cas), il gagnera la prime (=point bonus) de cette case. Il n'y aura pas plus de 7 cases spécial 'prime'.
- Toutes autres idées que vous aurez imaginées !

Organisation et Planning

Soutenance

Pour faciliter la démo de vos soutenances, prévoyez une partie quasiment terminée qui montre une victoire d'un des joueurs. Pour cela, préparez à l'avance un ou des fichiers de parties entamées que vous pouvez charger.

- Votre travail est à faire en équipe de 3 ou 4 étudiants à l'intérieur du même groupe de TD, en respect des équipes données au préalable.
- Votre travail est à déposer dans le lien de la page BoostCamp du projet qui va être fourni.
- La date et heure limite de chaque rendu sur BoostCamp vous être fournies, avec les consignes spécifiées sur BoostCamp, y compris celles concernant le plagiat en intégrant l'utilisation abusive de chatGPT.

Bon travail à tous 😊



Annexes

1- Table ANSI

La table ANSI ci-dessous vous permet d'afficher des caractères semi-graphiques à l'écran : pour le caractère choisi, afficher **0x** pour le code hexadécimal suivi du chiffre de la colonne et du chiffre de la ligne. Par exemple, le code ANSI en hexadécimal du cœur est 0x03, celui du carré est 0xDB etc.

Page de codes 850

	0	16	32	48	64	80	96	112	128	144	160	176	192	208	224	240
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
0	-0															
1	-1	⌚	◀	!	1	A	Q	»	q	ü	»	i	▀	▀	D	ß
2	-2	⊕	‡	‘	2	B	R	b	r	é	Æ	ó		—	E	o
3	-3	♥	!!	#	3	C	S	c	s	à	ð	ú			E	ó
4	-4	♦	§	\$	4	D	T	d	t	å	ð	ñ	—	—	È	ø
5	-5	♣	¤	%	5	E	U	e	u	à	ð	N	À	+	¹	ð
6	-6	♠	—	&	6	F	V	f	v	à	û	²	À	à	1	µ
7	-7	*	↓	‘	7	G	W	g	w	ç	ù	²	À	À	1	b
8	-8	█	1	(8	H	X	b	x	é	ÿ	í	○	└	1	p
9	-9	O	↓)	9	I	Y	i	y	é	ö	ø	¶	¶	—	º
10	-A	█	→	*	:	J	Z	j	z	è	ó	¬			0	·
11	-B	σ	←	+	:	K	l	k	l	í	ø	%	¬	¬	█	û
12	-C	♀	↶	,	<	L	\	l	l	i	£	%	¶	¶	█	ÿ
13	-D	♪	↔	-	=	M]	m]	i	ø	i	=	—	I	ÿ
14	-E	♫	▲	.	>	N	^	n	-	À	x	*	¥	‡	I	*
15	-F	☀	▼	/	?	O	_	o	Δ	À	f	*	¬	□	█	·

Exemple en C :

```
int main ()
{
    // exemple pour afficher un cœur suivi d'un carré
    printf("%c %c", 0x03, 0xDB); // 0x pour le codage hexadécimal
    return 0;
}
```

2- La manipulation du curseur sous Windows

Pour ce projet, vous aurez besoin d'une fonction de manipulation du curseur à l'écran. La fonction **gotoligcol** sur Windows ci-dessous vous permet de positionner le curseur à la ligne "lig" et à la colonne "col". Vous devez copier les lignes suivantes avant votre main (l'exemple des instructions du main ci-dessous ne sont bien sûr pas à copier, sauf la dernière toujours obligatoire).

```
#include <windows.h>

void gotoligcol( int lig, int col )

{
    // ressources
    COORD mycoord;

    mycoord.X = col;
    mycoord.Y = lig;
    SetConsoleCursorPosition( GetStdHandle( STD_OUTPUT_HANDLE ), mycoord );
}

int main ()

{
    gotoligcol(5,15); // place le curseur ligne 5 colonne 15 à l'écran
    printf("bonjour"); // écrit bonjour à la position du curseur

    return 0;
}
```

3- La gestion de la couleur sous Windows

Vous aurez aussi besoin d'une fonction de gestion de couleur à l'écran. La fonction **Color** sur Windows ci-dessous vous permet de changer la couleur du texte dans la console, ainsi que la couleur du fond pour la ligne. Vous devez copier les lignes suivantes avant votre main (l'exemple des instructions du main ci-dessous ne sont bien sûr pas à copier, sauf la dernière toujours obligatoire).

```
#include <stdio.h>
#include <stdlib.h>
#include <windows.h>

////////// PROTOTYPES DES SOUS-PROGRAMMES
void Color(int couleurDuTexte,int couleurDeFond);

///////////////////////////////
// Nom du sous-programme : COLOR
// Rôle : change la couleur du texte dans la console, ainsi que la couleur du fond pour la ligne
// Compatibilité : Xindows
/////////////////////////////
void Color(int couleurDuTexte,int couleurDeFond) // fonction d'affichage de couleurs
{
    HANDLE H=GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(H,couleurDeFond*16+couleurDuTexte);
}
```

```
int main()
{
    /*
    0 : Noir
    1 : Bleu foncé
    2 : Vert foncé
    3 : Turquoise
    4 : Rouge foncé
    5 : Violet
    6 : Vert caca d'oie
    7 : Gris clair
    8 : Gris foncé
    9 : Bleu fluo
    10 : Vert fluo
    11 : Turquoise
    12 : Rouge fluo
    13 : Violet 2
    14 : Jaune
    15 : Blanc
    */

    // essai numero 1
    Color(12,3);
    printf("Le texte va avoir la couleur 12 et le Fond la couleur 3\n");

    // essai numero 2 : on repasse au noir et blanc
    Color(15, 0);
    printf("Hello world!\n");

    // essai 3 : un peu de turquoise
    Color(11, 0);
    printf("Hello world!\n");

    // et la dernière couleur reste jusqu'à ce qu'on change de nouveau
    printf("Hello world!\n");

    return 0;
}
```