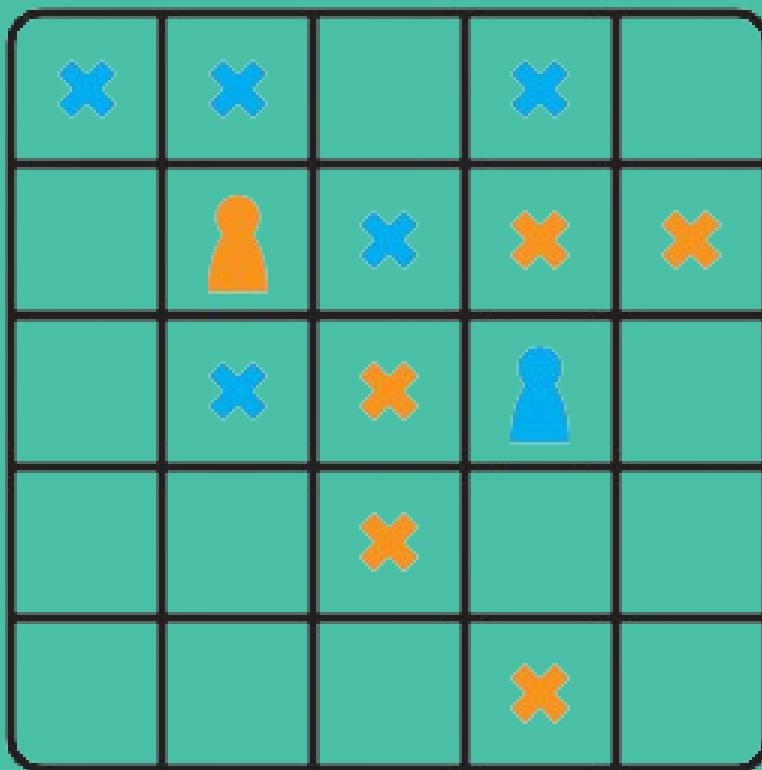


# Rapport Blocus

2024 - 2025

Réalisé par ATIK Wael et  
SRIVASTAVA-TIAMZON Emmanuel



# Sommaire

- 1. Introduction (page 3)**
- 2. Commencement de l'idée de création (page 4)**
- 3. Création du premier écran (page 5)**
- 4. Démonstration du menu (page 6)**
- 5. Création de la grille (page 7)**
- 6. Gestion du jeu (page 8)**
- 7. Démonstration du jeu (page 9)**
- 8. Menu de fin (page 10)**
- 9. Structuration du jeu (page 11)**
- 10. Structuration des fichiers (page 12)**
- 11. Donnée de l'état d'une partie (page 13)**
- 12. Les résultats du projet (page 14)**
- 13. Conclusion (page 15)**

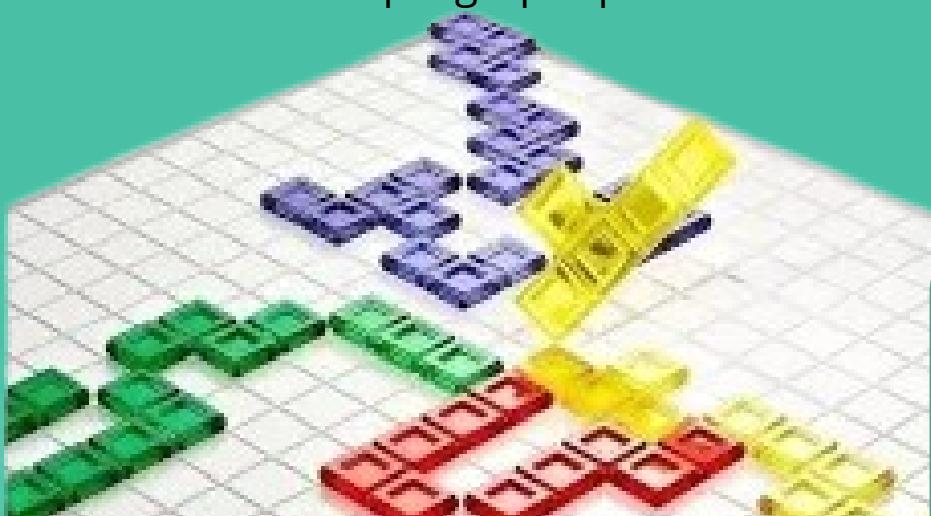
# Introduction

Pour le premier semestre de notre BUT 1 à l'IUT de Fontainebleau, notre première situation d'apprentissage et d'évaluation était de réaliser un jeu où deux joueurs se déplacent sur une grille en tentant de se bloquer mutuellement. Plus communément appelé blocus, il est souvent associé au jeu de société Blokus :

Blokus est un jeu de plateau abstrait créé en 2000 par Bernard Tavitian, un mathématicien et artiste français. Le jeu est édité par Sekkoïa avant d'être racheté par Mattel. Il s'est rapidement imposé comme un classique des jeux de stratégie grâce à ses règles simples mais profondes, idéales pour des parties rapides et tactiques.

Le plateau est une grille de 400 cases (20x20). Les pièces doivent être placées de manière à ce qu'elles se touchent uniquement par les coins, et jamais par les côtés. L'objectif est de poser un maximum de pièces sur le plateau tout en bloquant les adversaires.

Assez similaires aux règles imposés pour notre projet. Ici notre objectif était donc de créer un jeu qui dispose de trois écrans, le premier le menu de sélection qui consiste à choisir la taille de la grille/terrain de jeu et de choisir si l'on souhaite jouer avec une autre personne ou une IA, le deuxième écran est le jeu en lui même avec la grille, les pions et les croix à placer pour bloquer l'adversaire et gagner. C'est un jeu qui a l'air au premier abord simple et facile à faire, mais regorge de nombreuses difficultés et défis que l'on a du surmonter en C89 par le biais de la bibliothèque graphique de l'IUT de fontainebleau.



# Commencement de l'idée de création du projet

On avait d'abord eu pour idée de définir un tableau de 3x3 jusqu'à 9x9 par rapport à l'entrée du joueur. Ensuite créer avec la bibliothèque graphique trois

écran le premier qui nous permets de choisir la taille du tableau ensuite le deuxième sera le terrain qui fera la taille du tableau et ici les grilles seront faites avec la fonction dessinerrectangle puis un sprite d'un pion sera placé en fonction du choix du joueur, puis l'adversaire pourra se placer. Il faudra déplacer le pion à la limite du possible donc à ses cases adgacentes et sera effacé à sa dernière case pour définitivement être dans sa nouvelle case. des sprite seront aussi utilisé pour le remplissage des cases ou on peut utiliser la fonction remplirectangle. Le troisième écran affichera donc le gagnant (celui qui a pu se déplacer en dernier avant d'être enfermé).

L'idée est que le premier écran soit comme ça :

Veuillez choisir la taille de l'écran :

3x3	4x4	5x5	6x6	7x7	8x8	9x9
-----	-----	-----	-----	-----	-----	-----

Voulez-vous démarrer une partie à un ou deux joueurs?

1 joueur	2 joueur
----------	----------

On avait initialement penser à séparer le code en trois partie un fichier main.c qui allait tout coordonnée, un fichier graphique.c qui allait s'occuper de toute les fonctions qui consiste à l'affichage et enfin un fichier jeu.c qui allait s'occuper de la logique du jeu en lui même.

# Création du premier écran

Le premier écran pour rappel consiste au choix de la taille du terrain de jeu ensuite on aura le choix de jouer contre une IA qui placera son pion et ses croix par lui même et on a aussi le choix de jouer contre un adversaire qui utilisera la même souris que le joueur initial.

Pour l'affichage des différentes options de taille de grille on utilise une structure car dans le jeu, il y aura beaucoup de boutons qui seront assez similaires. Ensuite on utilise des fonctions que ça soit pour créer et dessiner les boutons ou même pour ce qui se passe après avoir cliqué sur un bouton.

Donc le premier écran qui est le menu consiste :

- De sept boutons qui permettent de choisir la taille du terrain de jeu que ça soit une grille de 3x3 à 9x9
- De deux autres boutons qui définira si l'on joue contre une IA ou une personne réel
- D'un texte qui s'affiche et qui s'adapte au choix du joueur
- D'un bouton "Confirmer" pour pouvoir confirmer le choix avant de commencer la partie

On comprend donc que c'est une interface graphique intuitive avec des couleurs et des boutons interactifs.

**Veuillez choisir la taille de la grille :**

3x3

4x4

5x5

6x6

7x7

8x8

9x9

**Voulez-vous démarrer une partie à un ou deux joueurs?**

IA

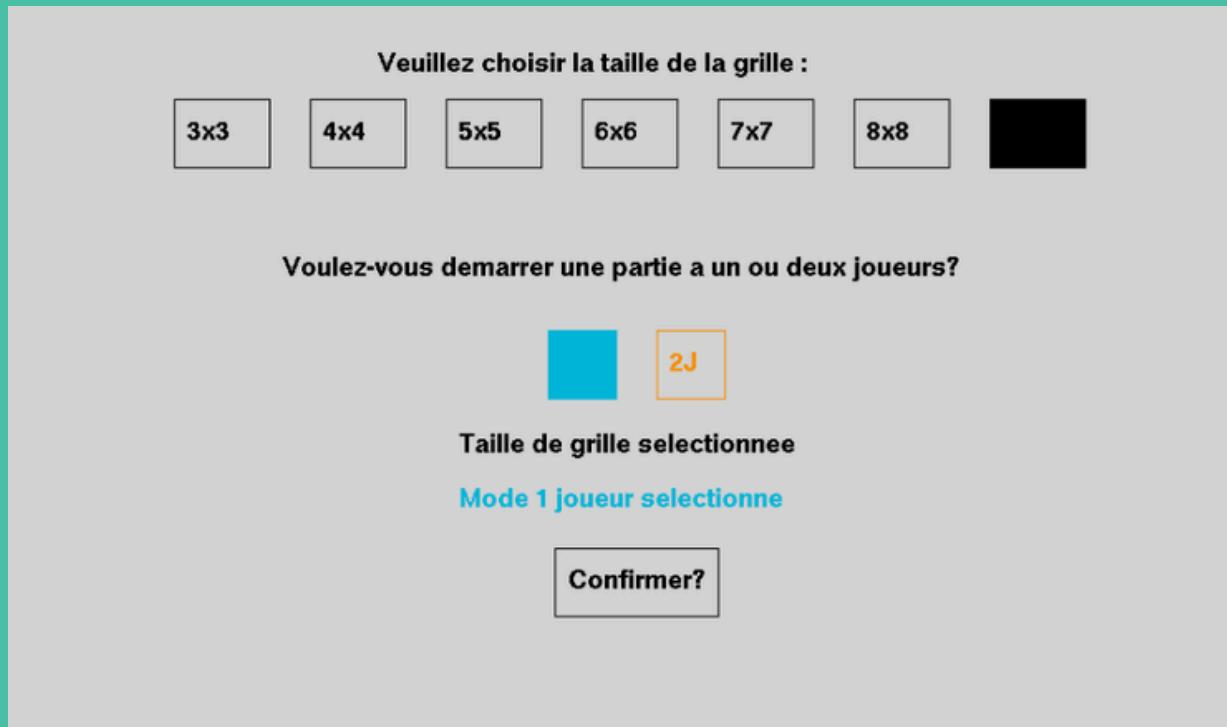
2J

# Démonstration du menu

Notre objectif pour notre menu à interface graphique intuitive est de pouvoir changer de choix en temps réels par exemple si l'on souhaite jouer dans une grille de 3x3 mais qu'au final on change d'avis et que l'on souhaite jouer dans une grille de 9x9 alors on peut facilement grâce au bouton confirmer.

L'affichage du menu consiste en plusieurs fonctions distinctes :

- Une structure qui instancie les boutons voulus,
- DessinerBouton qui explicitement dessine des boutons avec un rectangle, un texte centré et une couleur qui sont repris de la structure,
- EstClique est une fonction qui vérifie parmi les coordonnées du bouton si il y a eu un clic dans la zone,
- MenuGraphique est la fonction qui réunit les trois fonctions précédentes pour pouvoir afficher tout ce dont on a besoin pour Afficher un menu interactif permettant à l'utilisateur de configurer les paramètres de la partie (taille de la grille et mode de jeu).



Ainsi après avoir choisi la taille de la grille et le nombre de joueur présent dans cette partie alors on définit :

taille\_grille = 9 et nombre\_joueurs = 2;  
pour ensuite passer à l'affichage de la grille.

# Création de la grille

L'affichage de la grille a pour objectif de permettre au joueur de jouer que ça soit contre une IA, contre un adversaire ou même contre lui même, ça ne change rien aux règles du jeu et au terrain de jeu.

La fonction principale du terrain de jeu est AfficherGrille qui a pour objectif de dessiner une grille carrée composée de cellules de taille égale.

La taille (3x3, 4x4, jusqu'à 9x9) est déterminée par la sélection dans le menu.

## Explication des paramètres :

- taille\_grille : Nombre de cellules par ligne ou colonne (ex. 3 pour une grille 3x3) déterminé par le menu.
- largeur\_case : Dimension en pixels de chaque cellule,
- offset\_x et offset\_y : Décalages horizontaux et verticaux, calculés pour centrer la grille dans la fenêtre.

## Fonctionnement en détail :

- Deux boucles imbriquées parcouruent les lignes (i) et les colonnes (j) de la grille tel un tableau.
- La fonction calcule les coordonnées du coin supérieur gauche de chaque cellule.  
offset\_x et offset\_y recentrent la grille.
- j \* largeur\_case et i \* largeur\_case déplacent la position de chaque cellule.
- Chaque cellule est dessinée comme un rectangle vide avec la fonction DessinerRectangle par le biais de la bibliothèque graphique de l'IUT de fontainebleau

En bref la grille est comme une matrice qui a trois colonnes et trois lignes.

Voici un exemple avec une grille de 3x3 :

Donc on a largeur\_case = 500

/ 3 ≈ 166 pixels

offset\_x = (1000 - 3 \* 166) / 2 ≈ 167 pixels

offset\_y = (600 - 3 \* 166) / 2 ≈ 67 pixels

(2, 0)	(2, 1)	(2, 2)
(1, 0)	(1, 1)	(1, 2)
(0, 0)	(0, 1)	(0, 2)

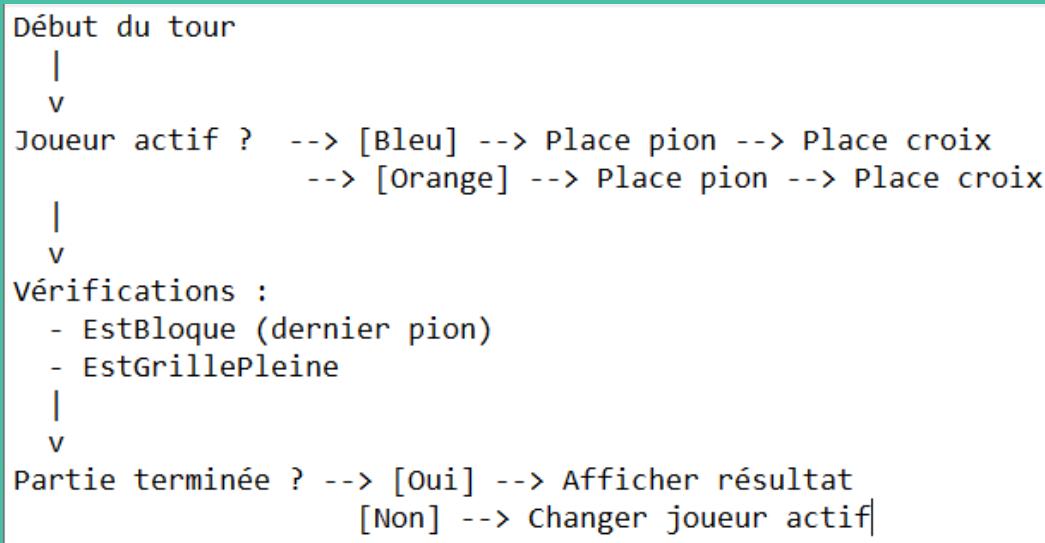
# Gestion du jeu

Le jeu est entièrement automatisé pour gérer les tours des joueurs et les règles et elle consiste en deux fonctions qui dépendent du choix fait dans le menu avec tout d'abord le

mode de jeu contre une IA, dont la fonction GestionJeuIA. Son objectif : Permettre au joueur humain de jouer contre une IA. Elle permet à l'utilisateur de placer un pion bleu et une croix pour bloquer l'IA. L'IA joue de manière automatique, en respectant les règles elle place un pion orange dans une case vide proche de son précédent emplacement parmi ses cases adjacentes ensuite place une croix orange pour bloquer l'utilisateur. Le jeu alterne les tours entre le joueur humain et l'IA. Cette fonction prend en compte la gestion des conditions de victoire ou de blocage.

La deuxième fonction, GestionJeu qui a pour objectif : permettre une partie compétitive entre deux joueurs humains. A tour de rôle le joueur 1 place un pion bleu et une croix bleu ensuite le joueur 2 place un pion orange et une croix orange. Lorsque les règles de placement des pions et croix sont respectées alors un pion doit être placé dans une cellule voisine et une croix peut être placée dans n'importe quelle cellule vide.

Le placement de pion et de croix est en harmonie avec les règles données par exemple, un pion ne peut être placé que dans une cellule adjacente au dernier pion du joueur actif. Une croix peut être placée dans n'importe quelle cellule vide.



Un schéma est toujours une bonne façon de résumer et interpréter les choses.

# Démonstration du jeu

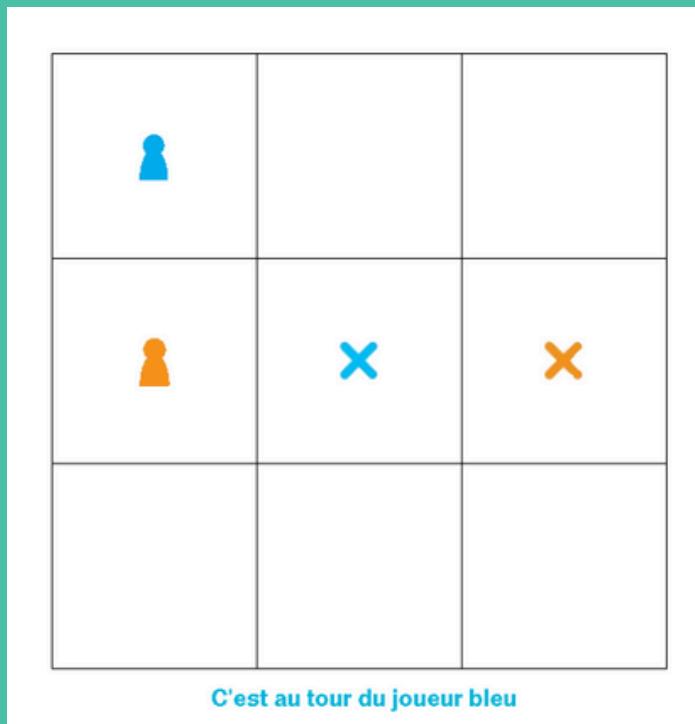
La gestion des tours quant à lui utilise une boucle for qui augmente (i) de 1 à chaque tour et après chaque placement de croix on vérifie s'il est pair pour que le jeu fonctionne avec les joueurs qui alternent :

Joueur bleu (ou humain) joue pendant les tours pairs.

Joueur orange (ou IA) joue pendant les tours impairs.

Après chaque action on écrit en dessous du terrain de jeu à qui est le tour suivant ensuite le jeu passe automatiquement au tour suivant.

Capture d'écran de notre jeu :



La gestion du jeu, bien que compliquée et composée de nombreuses fonctions pour pouvoir être en marche, elle se résume en quelques étapes simples, après le commencement du début du jeu et l'affichage du menu, alors soit le mode à deux joueurs se lance soit celui contre une IA. Ensuite à chaque tours il y a une vérification à faire : si l'on peut placer le pion/la croix à un tel endroit ou non mais surtout si le pion est bloqué, que ça soit par ses propres croix par les côtés de la grille, bloqué par l'adversaire ou même si la grille est pleine. Cette logique de jeu a été un réel défi pour notre groupe car elle se montrait coriace. C'était probablement l'un de nos plus grands défis, mais malgré tout on a su la surmonter et pouvoir réunir nos idées et travailler ensemble pour réaliser la logique du jeu en lui même.

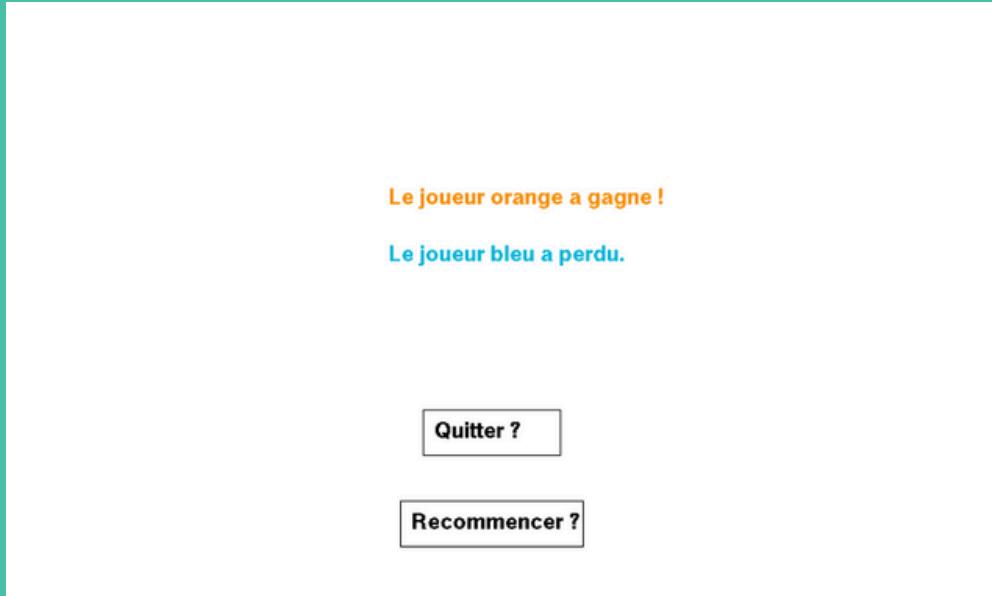
Pour revenir à la gestion du jeu après chaque vérification les joueurs peuvent enfin placer leur pions et ensuite leur croix pour subvenir à une mise à jour visuelle. Pour enfin finir avec le menu de fin qui affiche le résultats de l'affrontement et l'option de soit quitter le jeu en fermant la fenêtre graphique ou de recommencer via des bouton.

```
DebutJeu
|
+--> GestionJeu / GestionJeuIA
|
+--> DeplacementValide (pour vérifier les mouvements)
+--> EstBloque (pour vérifier les blocages)
+--> EstGrillePleine (pour vérifier la fin de partie)
|
+--> AfficherPion / AfficherCroix (mise à jour visuelle)
|
+--> MenuFin (affiche les résultats en cas de fin)
```

# Menu de fin

La vérification de fin de jeu a pour objectif de vérifier si le pion d'un joueur est bloqué ou pas et d'Informer les joueurs du résultat de la partie et permettre de quitter ou de recommencer. Il y a l'affichage du gagnant :

"Le joueur bleu a gagné !" ou "Le joueur orange a gagné !" selon les résultats, un bouton "Quitter" qui permet de fermer l'application en un clic et un bouton recommencer qui nous ramène directement au menu et réinitialise les données précédemment enregistrée.



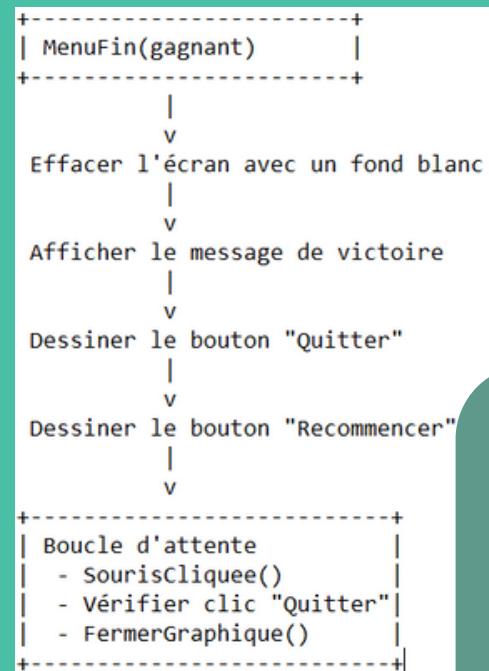
Le menu de fin est une partie importante pour conclure la partie en offrant un retour visuel sur le résultat et en permettant une interaction (comme quitter le jeu ou de rejouer). Donc notre menu de fin a deux objectifs principaux Informer le résultat de la partie : Afficher un message indiquant le gagnant de la partie (Joueur bleu ou Joueur orange) et permettre une conclusion claire et visuellement distincte du jeu. Mais aussi permettre à l'utilisateur de quitter ou rejouer avec un bouton interactif. La fonction utilisée pour ce troisième et dernier écran est MenuFin qui attend en paramètre soit le chiffre un ou deux pour pouvoir déclarer le gagnant.

int gagnant définit le joueur gagnant de la partie :

1 : Le joueur bleu a gagné.

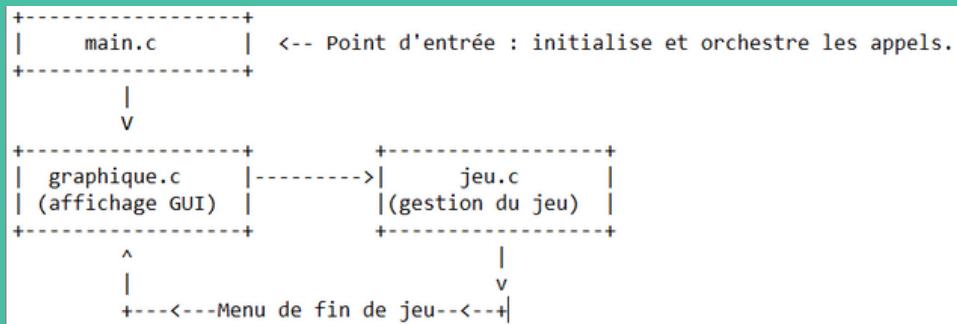
2 : Le joueur orange (ou IA) a gagné.

Voici un schéma expliquant le parcours de la fonction de menu de fin. Le plus important dans un menu de fin est de délaisser le jeu et faire comprendre que c'est fini, on a donc opté pour une façon simple d'exécution en effacant l'écran entièrement en blanc, pour ensuite afficher le message de victoire et célébrer le gagnant. Ensuite nous donnons deux options, soit de quitter le jeu soit de recommencer et pour cela l'écran de fin reste comme il est jusqu'à que l'on clique sur l'un des deux boutons donnés. Le menu de fin était l'une des étapes les plus facile de notre projet, car réunissant les connaissances que l'on a aquises avec le jeu et le menu alors il nous montre une telle simplicité.



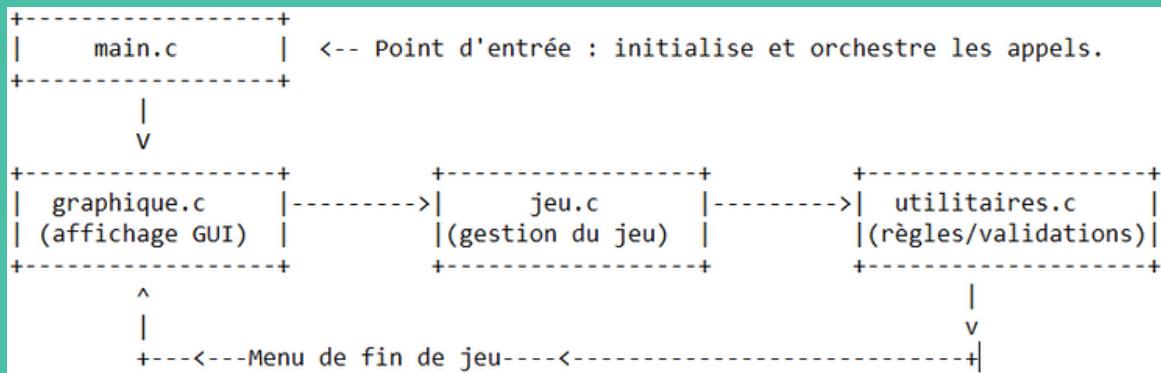
# Structuration du jeu

La structuration du jeu était une étape majeure et compliquée d'exécution surtout vers le début de notre projet. Il y a eu beaucoup de compliquation vis à vis la logique du jeu et comment répartir le code, car à la base nous avions opté pour un seul fichier que l'on allait séparer plus tard. Malgré les défis présentés, nous avons su les surmonter. Comme je l'ai mentionné au début de ce rapport de progression, nous avions initialement pris l'idée de structurer le jeu en trois fichiers distincts. Le premier "main.c" qui s'occupait de lancer le début du jeu et d'initialiser des sprites et même de les libérer à la fin du jeu. En second temps un fichier "graphique.c" qui allait être structurer de toutes fonctions qui gère l'affichage, l'interface graphique et toute sortes de modification graphique faites au cours du jeu. Enfin un troisième fichier "jeu.c" qui allait gérer les fonctions comme GestionJeu, GestionJeulA et DebutJeu auquel j'ai mentionné auparavant.



Mais après réflexion et après avoir questionné le professeur sur la structuration des codes, alors on a décidé de raccourcir un fichier. Le fichier "jeu.c" présentait beaucoup de lignes de code, ce qui allait être encombrant et assez barbant à scroller, car nous avions initialement pensé mettre dans ce fichier la logique du jeu comme la fonction DéplacementValide, EstBloqué et EstGrillePleine. Ce qui allait être beaucoup trop. Nous avions donc décidé de créer un quatrième fichier appelé "utilitaires.c" qui allait s'occuper de ses fonctions utiles et fondamentales à la logique du jeu.

Voici donc le processus de lancement de notre programme :



# Structuration des fichiers

Pour mieux comprendre la structuration du jeu alors je vais vous l'expliquer en détail. Le fichier main.c : Initialise la bibliothèque graphique (graph.h) et charge les ressources (sprites) pour ensuite appeler la fonction DebutJeu pour gérer le jeu et le commencer. Il libère enfin les ressources et termine le programme proprement. Le

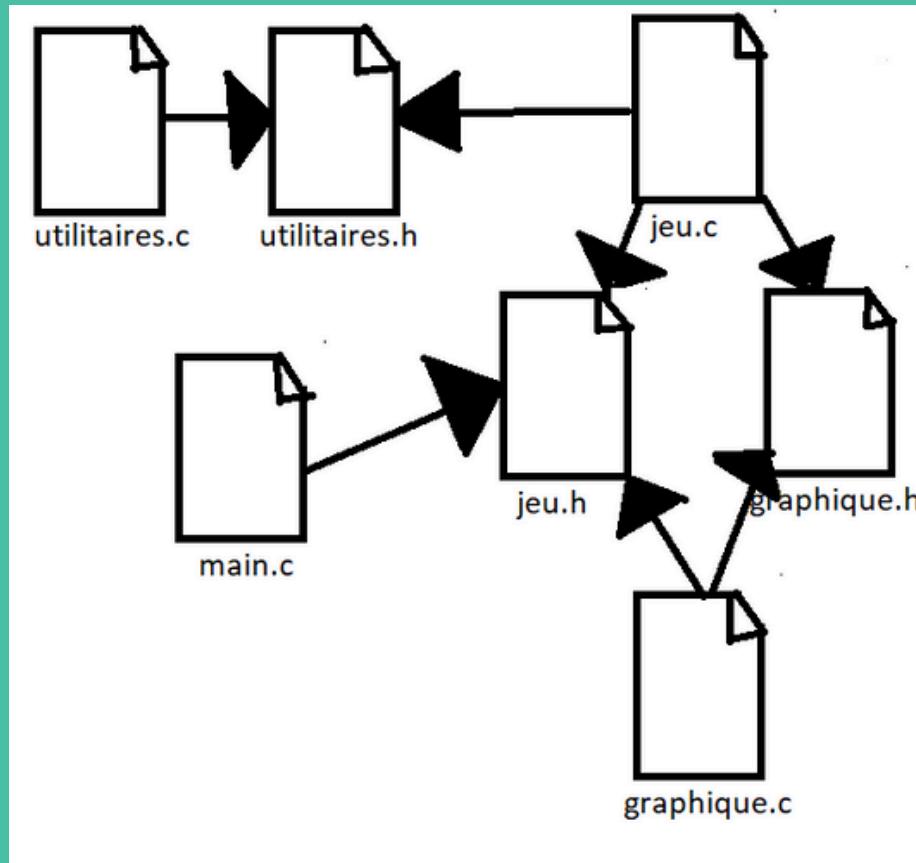
fichier graphique.c : Responsable des interactions visuelles et a pour fonctions principales: MenuGraphique qui affiche le menu de configuration du début du jeu,

AfficherGrille qui dessine la grille selon la taille choisie, MenuFin qui affiche le message de victoire et de fin de partie, RedessinerCase, AfficherPion et AfficherCroix gèrent les éléments visuels dans la grille. Le fichier jeu.c contient la logique principale

du jeu avec comme fonctions principales : GestionJeu qui est le mode 2 joueur,

GestionJeuIA qui est le mode joueur contre IA et CommencerPartie qui combine menu et démarrage de la partie. Le fichier utilitaires.c fournit des outils pour valider et gérer l'état de la partie, et a pour fonctions principales : EstGrillePleine qui vérifie

si toutes les cases sont occupées, EstBloque qui vérifie si un joueur ne peut plus bouger, donc terminer la partie, DeplacementValide qui vérifie si un déplacement respecte les règles.



# Données de l'état d'une partie

Les données représentant l'état d'une partie sont un détail crucial au rapport de progression pour vous montrer que l'on a bel et bien réussi notre projet et pour vous expliquer notre logique en profondeur. Commençons donc par la structure des données principales qui contient la grille du jeu, la taille de la grille, la position des derniers pions, l'état d'une partie et le tour de jeu. Commençons donc par la suite logique : la grille de jeu, elle est représentée par une matrice 2D int grille[9][9], chaque cellule de la matrice peut avoir l'une des valeurs suivantes : 0 = Cellule vide, 1 = Cellule occupée par un pion bleu (Joueur 1), 2 = Cellule occupée par un pion orange (Joueur 2 ou IA), 3 = Cellule marquée par une croix bleue et enfin 4 = Cellule marquée par une croix orange.

Ensuite la taille de la grille, une étape cruciale du projet, pour se sentir dans un environnement de jeu qui n'a pas l'air d'une simple grille. Elle est représentée par un entier int taille\_grille et elle définit la dimension carrée de la grille (entre 3 et 9). Pour poursuivre la position des derniers pions, un élément important qui nous montre un déplacement et une réelle progression d'une partie. Elle comprend les coordonnées des derniers pions joués par chaque joueur avec int dernierXbleu, int dernierYbleu qui sont les dernières positions du pion bleu. Et quant à l'adversaire : int dernierXorange, int dernierYorange qui sont les dernières positions du pion orange. En avançant dans la progression du projet, notre logique a su se développer et nous a fait réfléchir qu'il devait y avoir une étape pour chaque faits et gestes c'est pour cela qu'il y a un état de la partie, il comporte l'indicateur de l'étape actuelle du tour, qui sont :

int etat = 0 : Placement d'un pion.

int etat = 1 : Placement d'une croix. C'est grâce à ces états que l'on a pu simplifier certaines étapes. Bien sûr, dans un jeu de tour par tour il est essentiel qu'il y ait une donnée représentant le tour de jeu. Il contient un entier int tour, incrémenté après chaque action (pion + croix) qui permet de distinguer le joueur actif : Joueur 1 (tour pair), Joueur 2/IA (tour impair).

En réunissant toute ces données acquises au fil du jeu alors on a une interaction des données dans le programme qui se présente lorsqu'un joueur clique sur une case, ses coordonnées sont calculées à partir de la position de la souris, et les validations (EstBloque, DeplacementValide) mettent à jour la matrice grille en conséquence. Les données sont ensuite utilisées pour dessiner les pions et croix ou vérifier les conditions de victoire.



# Les résultats du projet

Lors du commencement de ce projet, notre objectif était de maîtriser une nouvelle base de données qui est la bibliothèque graphique fournit par l'IUT de Fontainebleau/Sénart, pour ensuite recréer un jeu nommé **Blocus**, ce qui a été un succès. Notre programme a su surmonter chaque étape cruciale demandé. De part les trois écrans distincts qui représente le début, le jeu et la fin, notre programme a éventuellement pu accomplir et faire de notre logique une réalité. Nous évaluons donc que notre programme répond aux objectifs fixée comme le fait que le terrain de jeu prend la forme d'une grille carrée dont la taille est choisie à chaque partie entre 3 et 9, le système de tour par tour, l'alignement des pions/croix, etc.

Nous sommes fier de pouvoir vous présenter notre projet surtout avec les fonctionnalités implémentées avec succès comme par exemple : le menu interactif, les modes de jeu (1 joueur contre IA, 2 joueurs), les conditions de victoire, etc.

Les points forts de notre programme est la fluidité de l'interface graphique, l'intuitivité pour les utilisateurs et la dispersion sur plusieurs fichiers qui fonctionnent en harmonie.

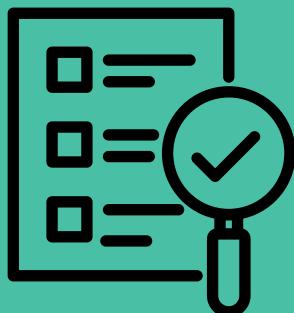
Au fil de ce projet nous avons rencontrés bien des limites et défis comme, l'alignement du pion et de la grille au milieu de l'écran, merci à la logique de ATIK Wael nous avons pu réussir ce défi de part de son expérience dans le développement de jeux vidéos en python et de l'idée des "offset".

La vérification des cases adjacentes était tout autant un défi pour nous car nous avions initialement pris la logique que l'on a appris en python qui est la boucle "for i in range" pour pouvoir vérifier une liste entière une par une, c'est ce que l'on a fait avec les cases adjacentes.

C'est donc grâce à notre expérience en python que la logique du jeu s'est développé.

Si l'on aurait eu plus de temps et que l'on se serait mieux organisé alors on aurait pu développer par exemple une IA améliorée, ou certaines animations qui pourraient rendre l'expérience utilisateur plus engageante, un différent choix de langue, un bouton pause ou autre.

La répartition des tâches est une étape importante dans l'organisation d'un projet, mais pour nous il s'est réalisé assez naturellement et rapidement. ATIK Wael était le responsable de l'interface graphique du jeu et de l'architecture. Quant à SRIVASTAVA-TIAMZON Emmanuel, il était responsable de la logique



# Conclusion

## SRIVASTAVA-TIAMZON Emmanuel :

Durant ce projet, j'ai eu l'opportunité de travailler sur la logique du jeu, l'interface graphique, et l'organisation du code, j'ai contribué à la conception du menu interactif et à la gestion des règles du jeu, tout en veillant à produire un code clair et maintenable ce qui a grandement facilité la collaboration.

Durant cette période de première situation d'apprentissage et d'évaluation j'ai appris à mieux m'organiser en groupe, à diviser et pouvoir gérer un grand programme, m'adapter à une bibliothèque graphique imposée et m'adapter dès qu'un défis se présentait. Un des principaux défis a été dans la logique du jeu, le fait de pouvoir vérifier les cases adjacentes pour vérifier si l'on pouvait placer un pion à un tel ou tel endroit mais aussi le fait de pouvoir vérifier une condition de fin de jeu.

Ce projet m'a permis de renforcer mes compétences dans le langages de programmation C, tout en découvrant l'importance de concevoir une interface utilisateur intuitive et d'en apprendre plus sur la structuration de données, surtout avec le Makefile et les fichiers d'en têtes car je me suis surtout concentré sur cela.

J'ai particulièrement apprécié l'aspect de développement et de maîtrise d'une bibliothèque graphique imposée surtout en travaillant en équipe. L'utilisation de différents fichiers et l'optimisation constante du jeu. J'ai déjà pu créer des jeux en python avec une bibliothèque graphique, certes c'est une expérience bien différentes mais qui a quand même pu m'être utile pour mon apprentissage. Cette situation d'apprentissage et d'évaluation a su me charmer .

## ATIK Wael :

Durant ce projet, j'ai eu l'opportunité de travailler sur la logique du jeu, l'interface graphique, le débuggage , la structure du code. J'avais déjà travaillé avec Emmanuel sur un projet en NSI donc la séparation du travail a été plutôt simple , j'avais aussi déjà fait un jeu donc j'ai pu réutiliser des concepts que je connaissais comme le offset par exemple.

En tant que gérant de la logique du jeu , j'ai eu à faire le EstBloqué et DeplacementValide qui furent très compliquées .

Réparer les erreurs qu'a produit la commande make Run, fut ardu car j'avais oublié de TOUT coder en C89 ce qui a fait que lorsque j'ai fait make run , 200 erreurs approximativement se sont affichés. J'ai donc rencontré de nombreuses difficultés à essayer tant bien que mal les erreurs durant 2 jours et 2 nuits.

Il y a eu des moments où je me disais si réellement un coefficient 40 était important mais au final j'ai réussi à faire un projet qui tient la route et qui **marche** et j'en suis très fier.