Blocus.io

Table des matières :

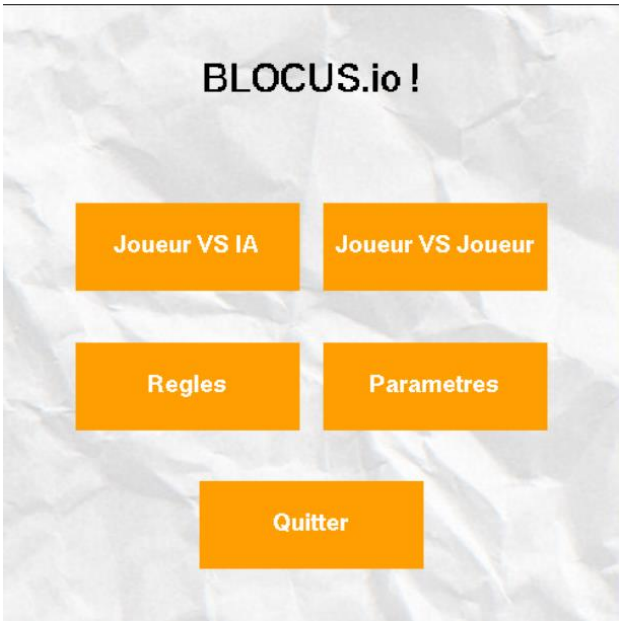
Introduction :

Dans le cadre de ce projet, nous avons développé un jeu intitulé Blocus.io, un jeu de stratégie où deux joueurs s'affrontent sur une grille carrée en tentant de se bloquer mutuellement. Ce jeu met en avant des mécaniques simples mais engageantes : chaque joueur déplace son pion sur une case adjacente et condamne une autre case, avec pour objectif de forcer son adversaire à l'immobilisation. Le premier joueur qui ne peut plus se déplacer perd la partie.

Le projet est réalisé en langage C89, sans dépendances extérieures, à l’exception de la bibliothèque graphique fournie par l’IUT. L'interface du jeu est entièrement contrôlée à la souris.

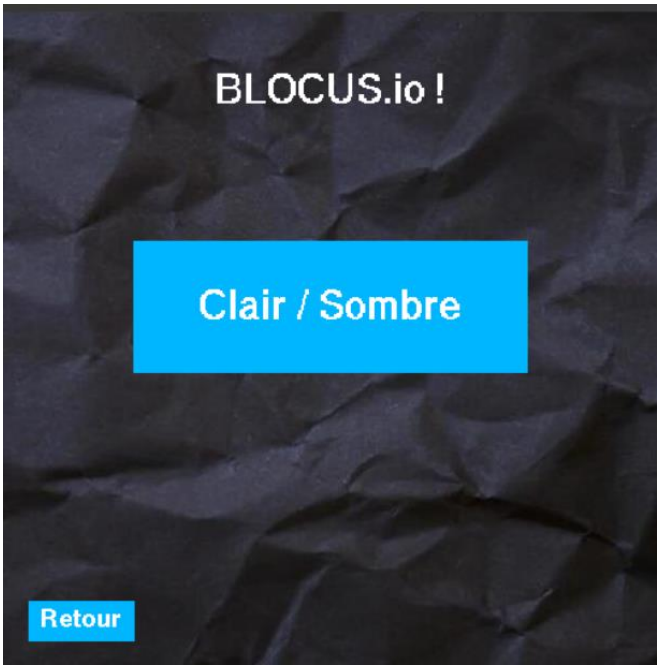
Description des fonctionnalités :

1. Écran d'accueil :



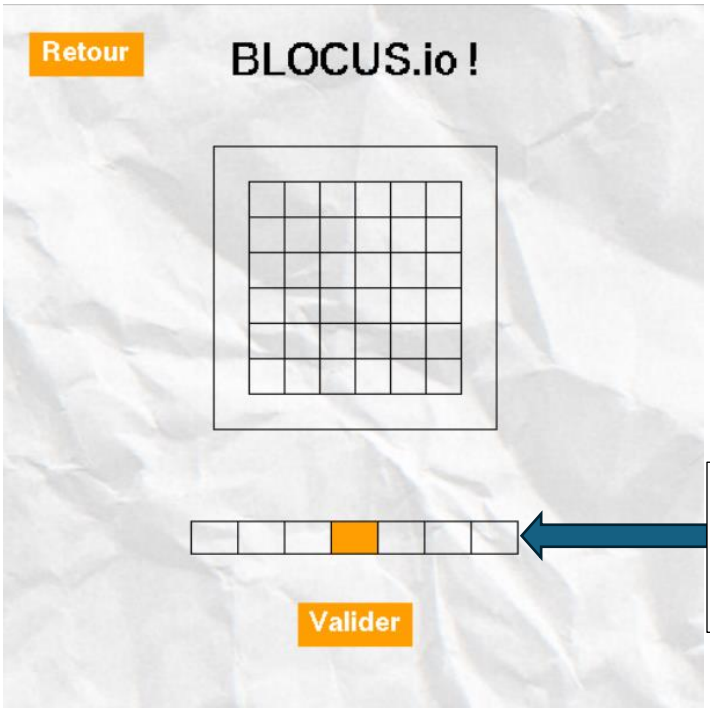
- Offre le choix entre une partie à un joueur (contre l'IA) ou à deux joueurs.
- Permet de connaître les règles.
- Accédez au réglage du jeu.
- Quitter le jeu.

2. Paramètres :



- Permet de choisir si l'on souhaite le jeu en thème clair ou sombre.

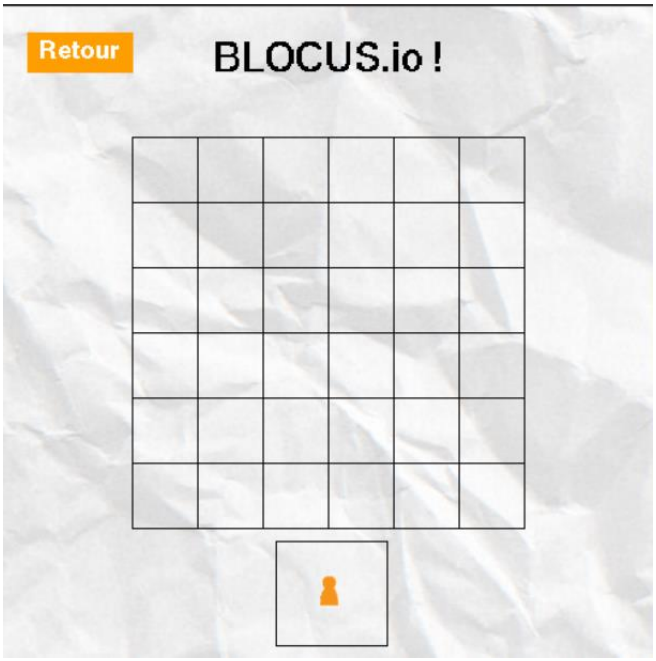
3. Écran de configuration :



En cliquant sur l'une des casse ça change la taille de la grille.

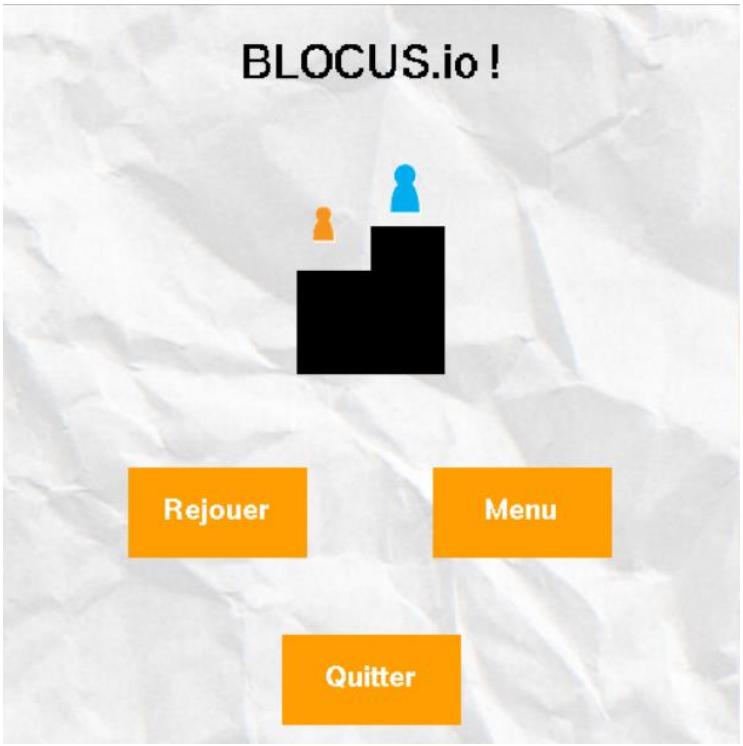
- Permet de choisir la taille de la grille (de 3x3 à 9x9).

4. Écran de jeu :



- Affiche la grille et l'état de la partie en cours.
- Les joueurs peuvent interagir à l'aide de la souris pour :
  - Déplacer leur pion vers une case adjacente.
  - Condamner une case libre.
- Les déplacements et condamnations respectent les règles établies.

5. Écran de fin :



- Indique le gagnant.
- Propose de relancer une nouvelle partie, aller au menu ou de quitter le jeu.

Structure du programme :

Le programme est organisé de manière modulaire pour une meilleure séparation des responsabilités. Chaque fichier source (.c) et d'en-tête (.h) correspond à une fonctionnalité ou un ensemble de fonctionnalités spécifiques. Voici une présentation de la structure du projet :

---

1. Fichiers principaux

- **main.c**
  - Point d'entrée du programme.
  - Gère la boucle principale et la navigation entre les différents écrans (menu, jeu, fin de partie).
- **Makefile**
  - Permet la compilation automatisée du programme.
  - Contient les cibles all (pour compiler), run (pour exécuter), et clean (pour supprimer les fichiers générés).

---

2. Gestion de l'interface utilisateur

- **menu.c / menu.h**
  - Gère l'affichage du menu principal et les interactions associées (démarrer une partie, lire les règles, modifier les options, quitter).
- **options.c / options.h**
  - Implémente les réglages personnalisables du jeu (par exemple : taille de la grille, mode de jeu).
  - Gère l'affichage et les choix dans l'écran des options.
- **regles.c / regles.h**
  - Affiche les règles du jeu sous forme graphique.

---

3. Gestion de la partie

- **partie.c / partie.h**
  - Implémente la logique principale de la partie.
  - Gère le déroulement du jeu : déplacement des pions, condamnation des cases, détection de la fin de la partie.

- **robot.c / robot.h**
    - Implémente les choix et actions du joueur simulé (robot) dans le mode un joueur.
    - Gère le déroulement du jeu : déplacement des pions, condamnation des cases, mais l'algorithme est modifié de sorte a ce que le robot puisse être simulé.

---

4. Outils et structures

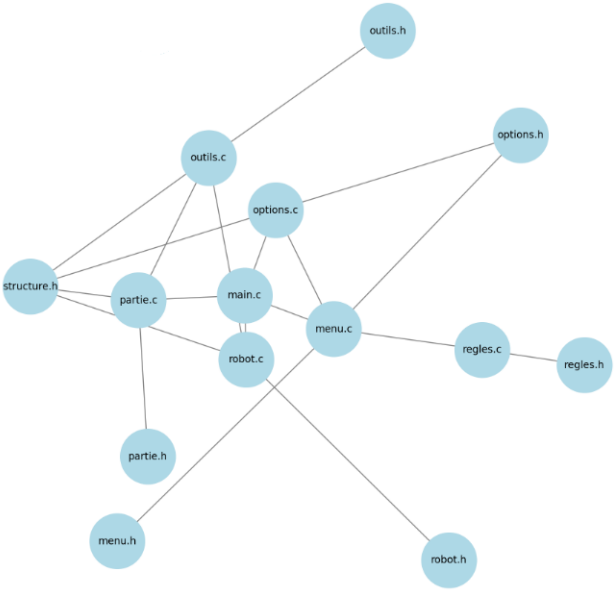
- **outils.c / outils.h**
    - Contient des fonctions utilitaires réutilisables, par exemple pour manipuler les bouton (il peut nous servir dans ce projet pour la grille ou vérifier la validité des coups).
    - Supporte des fonctionnalités transversales nécessaires au fonctionnement global.
- **structure.h**
    - Définit les structures de données utilisées dans le programme, comme :
        - La grille de jeu.
        - Les coordonnées des pions.
        - Les variables représentant l'état global de la partie.
        - Et autres...

---

5. Répertoire des ressources

- **images/**
    - Contient les fichiers graphiques utilisés pour l'interface du jeu (boutons, arrière-plans, icônes, etc.).

Diagramme de structure :



Données représentant l'état de la partie :

L'état d'une partie est représenté par les structures suivantes :

- **Les 2 joueurs de type Joueur :**
  - Position des joueurs dans la map
  - Image du pion et image de sa croix
  - Variable booléens servant à savoir si le joueur gagne
- **Grille de Jeux :**
  - Elle est représenté sous la forme d'une liste de type Entier ou chaque élément peut contenir :
    - Une case prise par le joueur 2 = -2
    - Une case prise par le joueur 1 = -1
    - Une case libre = 0
    - Une case prise par une croix du joueur 1 = 1
    - Une case prise par le joueur 2 = 2
  - Elle est représenté sous la forme d'une liste de type Rect (structure spécifique au projet)
    - Chaque élément contient un sprite selon la valeur indiqué et si c'est 0 il posséde aucune sprite

- **Tour de jeu :**
    - Elle est composées en deux temps :
    - Le premier tour est pris sur une liste de deux élément de type de char prenant la forme de 0 ou de 1 l'indice du 1 représente le tour du joueur (tour + 1).
    - Le deuxième est sous la forme d'une liste de 4 elements de type char prenant la forme de 0 ou de 1 ou (pion_joueur1 / croix_joueur1 / pion_joueur2 / croix_joueur2)

La partie prend donc fin lorsque un des deux joueurs ne possède aucun 0 dans ces cases adjacentes.

Conclusion personnelle :

Amine **:**
Ce projet a été une expérience enrichissante qui m'a permis de renforcer mes compétences en programmation C et en structuration de projet. J'ai particulièrement apprécié la réflexion sur les interactions entre l'interface graphique et la logique du jeu. Dans mon passé j'ai pu coder des projets en python et j'aime particulièrement l'organisation du code en C qui est totalement différentes. J'affectionne le fait que l'on ait utilisé des structures.

Lucas **:**
Ce travail m'a permis d'améliorer ma rigueur dans l'écriture de code et ma capacité à collaborer efficacement dans un cadre structuré. Les défis techniques rencontrés, notamment pour implémenter l'intelligence artificielle, ont été stimulants.