Rapport du projet BLOCUS

Adam Meddahi et Quentin Lacombe

Table des matières

Introduction

Le but de ce projet était de réaliser un jeu nommé Blocus. Le but du jeu est de bloquer son adversaire sur une grille en ne pouvant se déplacer que sur les cases adjacentes à la position actuelle du joueur et de pouvoir condamner une case à chaque tour. Dès qu'un joueur ne peut plus déplacer son pion, la partie s'arrête et le joueur bloqué a perdu. Ce projet a été réalisé à l'aide de C89 et de la bibliothèque graphique de l'IUT. Ce projet fut réalisé en binôme par Adam Meddahi et Quentin Lacombe et devait être réalisé avant le vendredi 15 janvier 2021 à 8h00. Dès le début du projet, nous avons développé ce projet dans un dépôt du serveur du GoGS du département afin de pouvoir gérer convenablement les versions de nos fichiers sources et de centraliser ces derniers.

Règles

Notre jeu respecte les règles établies. En effet, la grille est de forme carrée et est d'une taille située entre 3x3 (inclus) et 9x9 (inclus). Les tours des joueurs s'alternent et comme expliqué précédemment, chaque joueur doit déplacer son pion sur une case adjacente et doit essayer de bloquer l'autre à l'aide de croix qu'il peut poser sur toutes les cases libres de la grille.

Interface

L'interface de notre jeu est divisée en trois écrans, un écran permettant de choisir les paramètres de la partie, un second écran sur lequel la partie se déroulera et un troisième écran affichant le résultat de la partie et laissant la possibilité au joueur de relancer une partie ou bien de quitter le programme. Notre jeu est majoritairement contrôlé à l'aide de la souris, et un peu au clavier.

Fonctionnalités du programme

Fonctionnalités de l'accueil

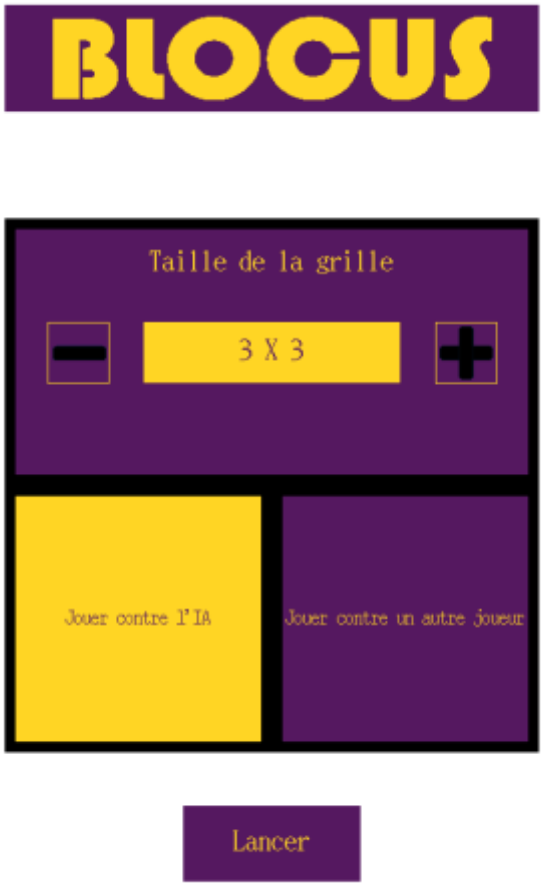
L'écran d'accueil de notre programme permet à l'utilisateur de paramétrer la partie qu'il va jouer. Il peut choisir la taille de la grille sur laquelle la partie se déroulera à l'aide de deux boutons permettant d'augmenter ou de diminuer la taille de la grille. Cette partie de l'écran d'accueil vous est disponible dans l'image ci-dessous.

Les boutons « + » et « - » ne fonctionnent que si leurs limites respectives ne sont pas atteintes ce qui signifie que dès que le joueur a sélectionné une grille de 3x3 il ne pourra pas choisir une taille en dessous et le même principe s'applique au bouton plus qui sera bloqué dès lors que la taille 9x9 sera sélectionné. L'affichage de la

taille de la grille se renouvelle à chaque clic de l'utilisateur sur le bouton « + » ou « - » qui correspond à une taille réglementaire. De base, une taille est déjà sélectionnée et cette dernière est 3x3.

L'écran d'accueil de notre programme permet aussi à l'utilisateur de choisir contre qui il veut jouer, 2 choix s'offrent à lui, soit il joue contre l'IA, soit il joue contre un autre joueur, encore une fois l'utilisateur devra cliquer sur un bouton afin de décider qui sera son adversaire. Le bouton sélectionné aura une couleur de fond jaune et une couleur de texte violette. De base, un choix est déjà sélectionné et ce choix correspond au mode de jeu contre l'IA. L'image ci-dessous est une capture d'écran de la partie de l'accueil en question.

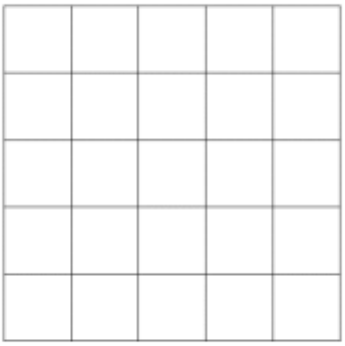
Le dernier bouton de notre écran d'accueil est un bouton permettant de lancer la partie, il est visible sur la capture d'écran de l'ensemble de l'écran d'accueil ci-dessous.



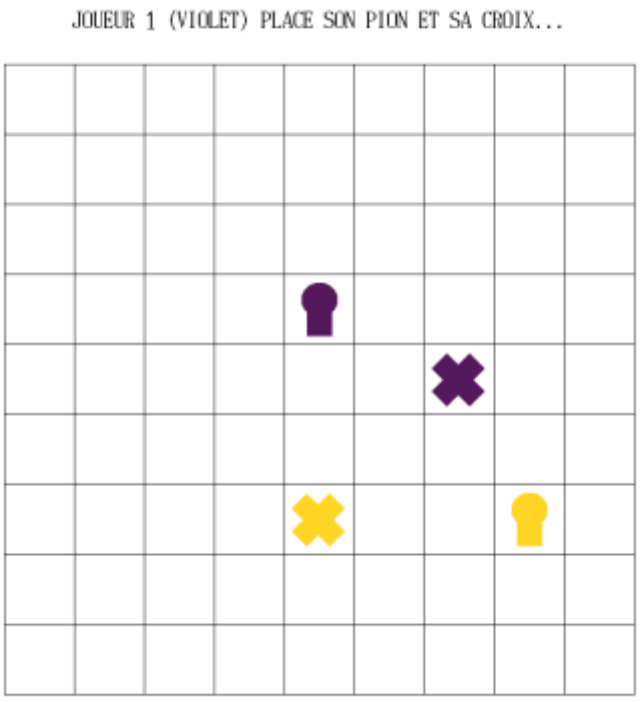
Fonctionnalités de l'écran de jeu

L'écran de jeu permet d'afficher une grille de la taille donnée via l'écran d'accueil, la grille sera centrée dans la fenêtre et chaque cellule est un carré de 80px de côté. Cette taille permet d'avoir la meilleure visibilité possible pour toutes les tailles possibles de la grille tout en évitant un débordement pour une grille de taille 9x9.

La capture d'écran ci-dessous illustre un exemple de grille dessiner par le second écran.

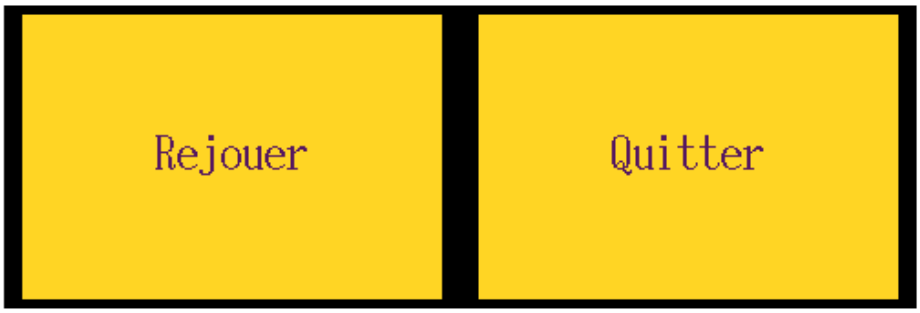


L'écran de jeu permet aussi d'afficher des pions ainsi que des croix, tant que le mouvement essayé par l'utilisateur n'est pas réglementaire, rien ne se passera, car le second écran n'autorise que des mouvements réglementaires. L'interaction avec cet écran se fait à la souris. Une croix représente une case condamnée. Dans un souci de compréhension des joueurs, le second écran informe qui doit jouer à l'aide d'un texte au-dessus de la grille. Le second écran attendra alors qu'un coup réglementaire soit effectué. La capture d'écran si dessous illustre une phase de jeu sur le second écran.



Fonctionnalités de l'écran de fin

L'écran de fin va permettre de renseigner le(s) joueur(s) du résultat de la partie, quatre états sont possibles et ces quatre états sont : **victoire du premier joueur**, **victoire du second joueur**, victoire **de l'IA** et pour finir **match nul entre les joueurs**. L'écran de fin permet aussi de laisser la possibilité au joueur de décider de rejouer une partie ou de quitter le programme. Comme dans la majorité du programme, l'utilisateur interagit avec l'écran de fin à l'aide de la souris. L'image ci-dessous est une capture de l'écran de fin du programme dans le cas d'une partie à 2 ou le joueur 2 a remporté la partie.



Structure du programme

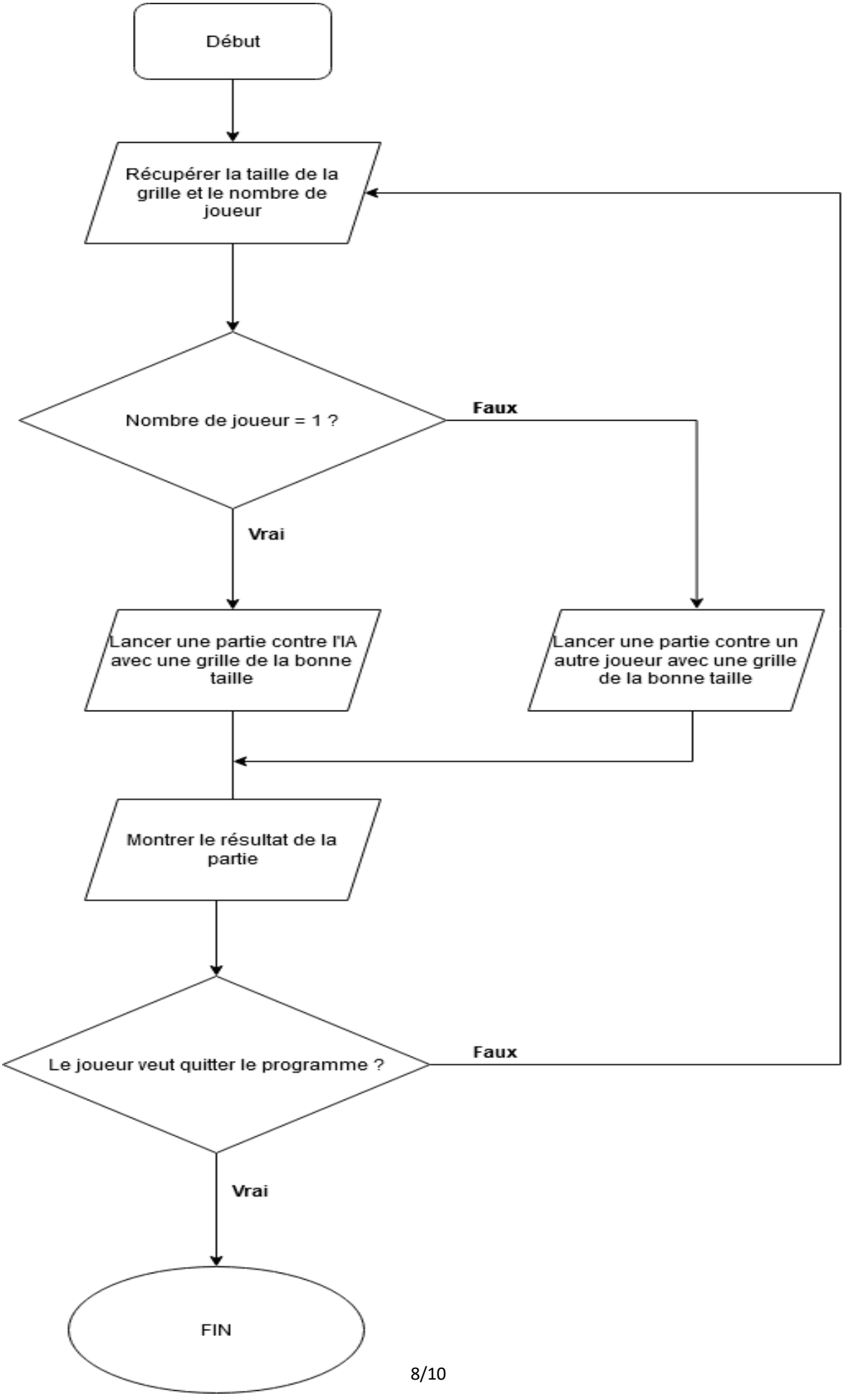
Découpage des sources

Notre projet est découpé en plusieurs fonctions et bibliothèque, le but était de favoriser l'organisation du dossier afin d'évoluer dans un environnement clair et propre tout en créant seulement le nombre nécessaire de bibliothèques afin de garder une certaine logique. Notre projet contient trois bibliothèques : affichage.h, fonctionnementJeu.h et souris.h. Nous avons mis en place un fichier Makefile afin de pouvoir

compiler de façon efficace et simple notre projet. Nous avons aussi mis notre projet sous la forme d'un dossier composé de différentes directions (par exemple « biblio » ou « images ») afin d'ajouter en clarté au projet.

Structure du programme principal

Notre but était de réaliser un fichier main.c qui soit le plus simple et clair possible, cela fut permis grâce au nombreuses fonctions et bibliothèques que nous avons mis en place au début de notre projet et par la réalisation d'un algorigramme afin de savoir à quoi nous voulions que ressemble notre programme. L'algorigramme vous est disponible ci-dessous. Le principe de notre programme est de récupérer les paramètres de la partie dans un premier temps, puis de lancer une partie respectant les paramètres définis et ensuite de laisser le choix au joueur de rejouer ou d'arrêter le programme.


Début
Récupérer la taille de la grille et le nombre de joueur
Nombre de joueur = 1 ?
Faux
Vrai
Lancer une partie contre l'IA avec une grille de la bonne taille
Lancer une partie contre un autre joueur avec une grille de la bonne taille
Montrer le résultat de la partie
Le joueur veut quitter le programme ?
Faux
Vrai
FIN

Explication des données

Les données géométriques

Nous avons défini dans le fichier affichage.h deux structures qui nous ont été utile tout au long du projet, ces structures sont : la structure position qui contient 2 entiers représentants l'abscisse et l'ordonné d'un point, la structure taille qui contient deux entiers représentant la longueur et la largeur d'un rectangle.

Les données sur l'état de la partie en cours

Tout d'abord, il a fallu définir les différentes composantes pouvant représenter l'état de la partie en cours. Nous en sommes arrivés à la conclusion que cette dernière était : l'état et la position des cases de la grille ainsi que la position des pions. La position de la grille est gérée par une fonction spécifique, tandis que la position de chaque case est relative à la position de la grille, car la position d'une case est régie par la formule : (positionXGrille+i*80 ; positionYGrille+i*80). L'état des cases de la grille n'a que deux options, soit la case est libre, soit elle est condamnée. Pour ce qui est de la position d'un pion ou d'une croix, elle ne peut pas être en dehors de la grille, ou une case occupée par un autre élément. La gestion des cases condamnée se fait par le biais d'une liste chaînée nommée « Prison » et qui va stocker une à une les cases condamnées par une croix, mais également permettre de vérifier quelles cases sont prises. Pour éviter qu'un élément soit placé sur un pion, sa position est stockée dans une variable et est disponible à tout moment du programme. Aussi, la fonction « afficherAccueil » renvoie une structure de type « parametrePartie » pour que les paramètres du jeu soient chargés sur le 2ème écran en conséquence et les fonctions « multijoueur » et « ia » renvoient chacun un entier situé entre 1 et 4 compris, afin de pouvoir exploiter les données de la victoire sur l'écran de fin.

Conclusion

Adam Meddahi

Ce projet m'a permis de renforcer mes connaissances du langage C89 et mon aisance avec ce dernier. Je pense que ce projet m'a permis de me mettre dans de vraies conditions de travail, et cela, grâce au travail en binôme avec Quentin, ce qui nous a forcé définir les différentes fonctionnalités à réalisés puis de réfléchir à comment les mettre en œuvre de la meilleure des manières. Je pense aussi que le fait de devoir respecté un planning de développement m'a permis de me mettre dans une situation de développement professionnelle. Je pense avoir beaucoup appris grâce à ce projet et je pense que mon niveau à la suite de ce projet et beaucoup plus important qu'avant du fait des conditions définies dans le sujet (utilisation d'un dépôt git, réalisation du Makefile) ainsi que celle que nous nous sommes imposés (communication régulière). Ce projet m'a aussi permis de me familiariser et d'apprendre énormément sur l'utilisation de git. Je pense refaire ce projet seul dès que j'en aurais le temps afin de pouvoir voir en combien de temps il m'est possible de le réaliser et de mettre en œuvre les différentes choses que m'ont appris ce projet et dès le début cette fois-ci.

Quentin Lacombe

Ce projet m'a été particulièrement bénéfique pour mon apprentissage de la programmation en C89 et ce sur plusieurs points. En effet, pendant la réalisation de ce projet, je suis passé par différents blocages et incompréhensions, mais je les ai toutes surmontées, certaines seul, d'autres avec l'aide d'Adam, et cela m'a permis de prendre des automatismes et de faire de moins en moins d'erreurs dans mon code, en essayant d'adapter un raisonnement le plus simple possible. De plus, j'ai dû apprendre à échanger avec mon camarade, comme le feraient des collègues en entreprise, et ce notamment grâce à nos « mises en commun » les soirs de la semaine vers 22h où nous faisions un point sur ce que nous avions fait dans la journée. Pour ne pas nous gêner mutuellement dans notre avancée, nous avions décidé d'envoyer un message à chaque « git push » que nous faisions, pour nous arranger et éviter de fusionner les branches du git à chaque fois. Pour ce qui est de mon ressentiment sur l'aboutissement du projet, je suis assez satisfait de la forme qu'il a prise, car nous avons corrigé tous les bugs que nous avons remarqué et les graphismes me plaisent. Cependant, le mode « solo » me laisse un goût amer, car le joueur dirigé par l'ordinateur est remarquablement lent, et je pense qu'avec une meilleure gestion du temps, nous aurions pu produire un jeu presque parfait. Ceci dit, j'ai tout de même pris beaucoup de plaisir à la réalisation du blocus, et j'ai pris un peu plus de confiance quant à mes capacités. Enfin, lorsque j'aurai plus d'expérience en programmation, j'envisage de recommencer ce projet entièrement et seul, afin de réaliser la version que j'aurais aimé rendre aujourd'hui.