

Introduction

Le projet est un jeu Blocus dont le but est de bloquer l'adversaire. Il permet de choisir la taille de la grille de jeu entre 7 tailles différentes (de 3 par 3 cases à 9 par 9 cases). IL donne aussi la possibilité de pouvoir jouer seul contre le pc ou à 2. Il est complètement jouable à la souris. Les joueurs peuvent placer librement leurs pions sur la grille de jeu lors du commencement de la partie, puis dans un périmètre d'une case (diagonales comprises) autour d'eux et placer une croix sur une case libre de la surface de jeu. La partie se déroule au tour par tour et se termine lorsque l'un des 2 joueurs ne peut plus se déplacer. L'écran de fin apparaît alors, indique le gagnant et propose de recommencer la partie.

Description

Notre projet s'ouvre sur un écran de menu. Il est composé d'une banderole numérotée de 3 à 9. Celle-ci permet de choisir la taille de la grille de jeu. En dessous, nous pouvons trouver deux cases nommées respectivement :

-Solo

- 2 Joueurs

C'est deux cases servant de boutons pour choisir le mode de jeu de la partie dont la caractéristique est indiquée clairement par leurs appellations.

Taille de la grille :

3	4	5	6	7	8	9
---	---	---	---	---	---	---

Choix de la taille de la grille. Il suffit de cliquer sur la case avec la valeur voulue (il faut faire ce choix en 1^{er}).

Mode de jeu :

Solo

2 Joueurs

Choix du mode de jeu (à sélectionner en 2^{ème}). La partie commence immédiatement après avoir sélectionné les 2 paramètres disponibles


L'interface du menu est plutôt claire et compréhensible sans explications (hormis l'ordre des cliques).

Suite à la sélection faite sur cette 1^{ère} page, le menu disparaît et la grille apparaît avec les paramètres voulus. Par un souci de lisibilité, nous utiliserons la grille de taille 3 pour illustrer notre description.

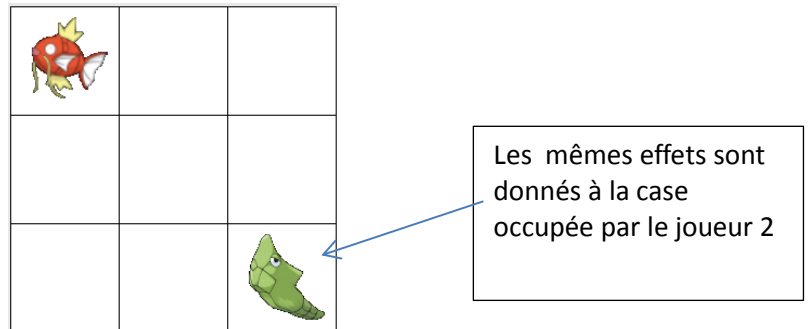


Le jeu commence et le joueur 1 doit cliquer dans la zone jouable proposée par la grille de la taille choisie. Si le joueur clique hors de cette zone rien ne se passe tant qu'il n'a pas sélectionné une case correcte. Lorsque le joueur 1 aura cliqué sur une case voulue, son pion apparaît au milieu de la case. Pour le joueur 1 il sera modélisé par un Magicarpe (c'est la carpe rouge pour ceux qui ne connaissent pas). La case est dorénavant bloquée, c'est-à-dire qu'aucun des 2 joueurs ne peut plus interagir avec celle-ci tant que le pion l'occupe.

Le joueur 1 occupe dorénavant cette case. Aucune action n'est plus faisable sur cette case jusqu'au prochain tour.

Après, c'est au tour du joueur 2 d'effectuer la même opération. Son pion est modélisé par un Chrysacier (le cocon vert avec des yeux). Cependant comme le joueur 1 occupe une case, il n'a plus que 8 cases pour choisir (en restant dans l'exemple de la grille taille 3).

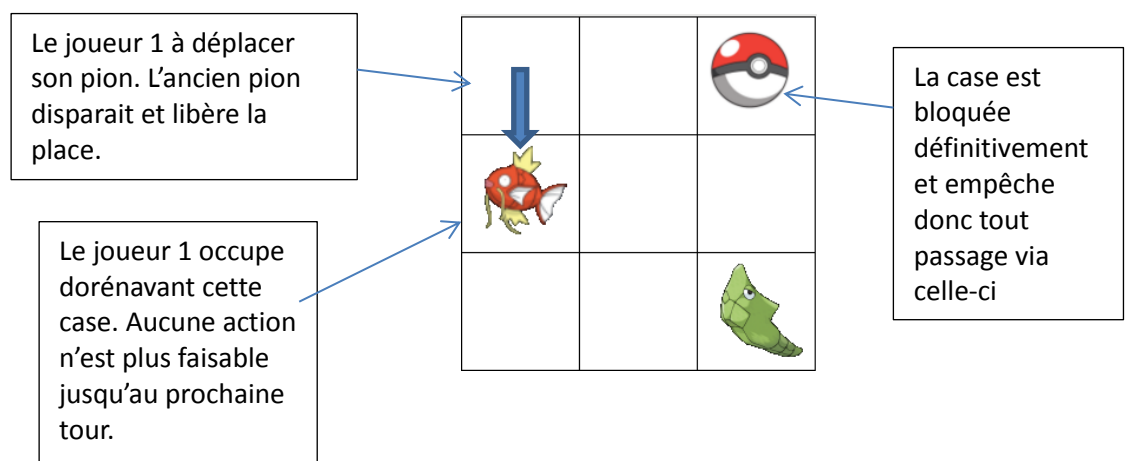


Lorsque cette étape est terminée, le gameplay principale du jeu commence.

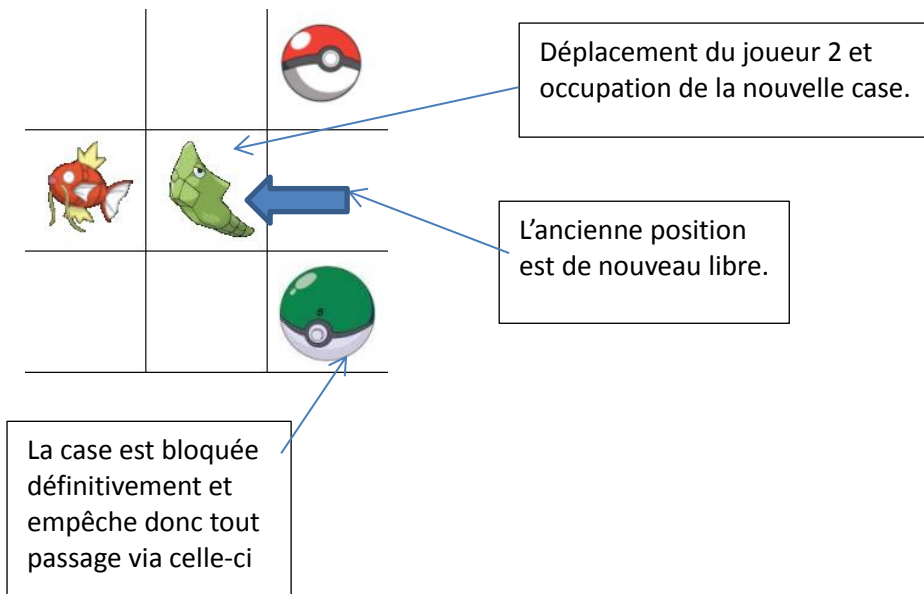
Nous commençons avec le tour du joueur 1. Lors de son tour, il devra effectuer 2 actions :

- Déplacer son pion sur une des 8 cases adjacentes (si elles sont libres). Cela aura pour effet de libérer la case anciennement occupée et de bloquer la nouvelle.

- Placer une « croix » (ici modéliser par une pokeball de la couleur du pion du joueur soit rouge pour le joueur 1 et vert pour le 2) librement sur une case libre du terrain. Cela aura pour effet de bloquer la case jusqu'à la fin de la partie. La case sera bloquée pour les 2 joueurs.

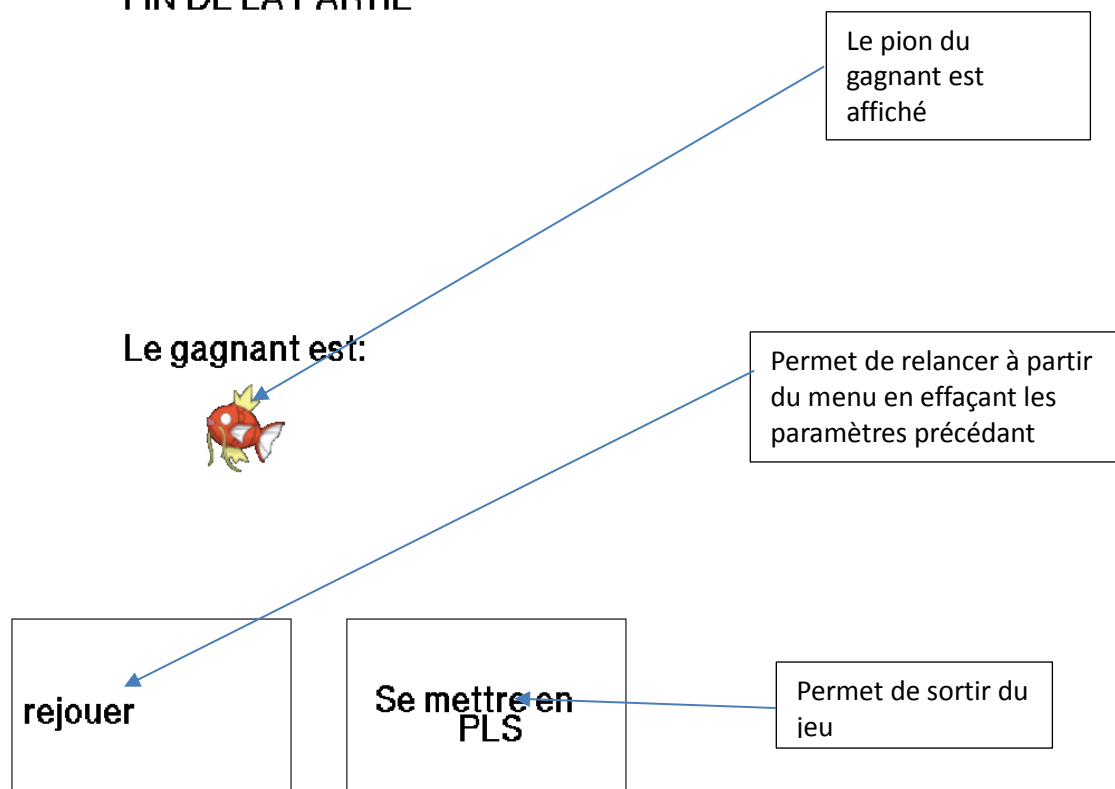


Après le tour du joueur 1 arrive celui du joueur 2 qui a le droit au même action.

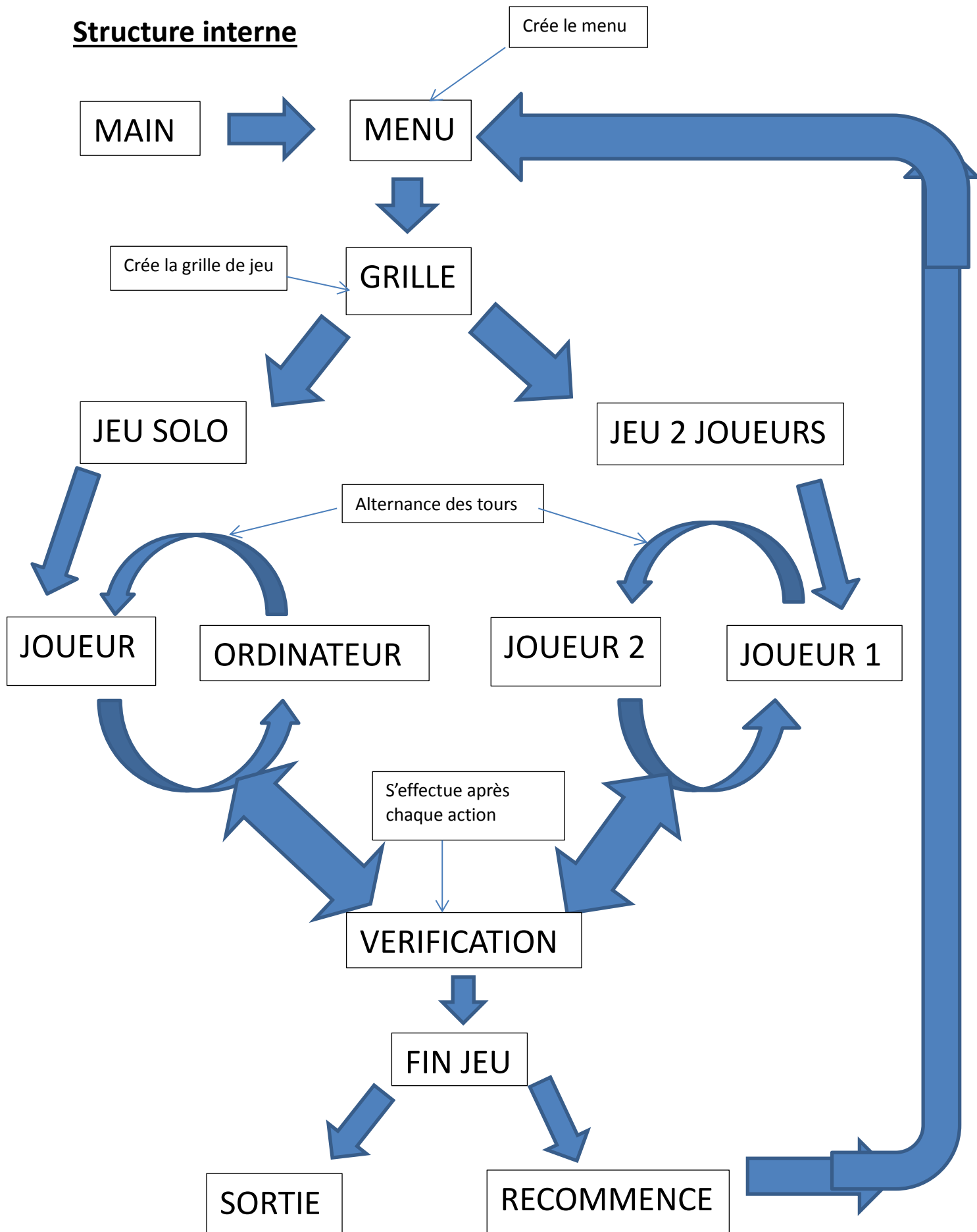


Le jeu continuera à alterner entre les 2 joueurs Jusqu'à se que l'un des 2 soit bloqué. Lorsque cela ce produira, l'écran de jeu cèdera place a celui de fin qui affiche le gagnant et nous affiche un bouton recommencer qui nous renvoi au menu de départ après avoir réinitialisé tous les paramètres de l'ancienne partie.

FIN DE LA PARTIE



Structure interne



Nous avons choisi de découper le fichier en plusieurs fonctions qui ont chacune une tâche clé dans le déroulement du programme. La principale séparation est celle entre l'aspect 2 joueurs et le solo. Elles sont fondamentalement identiques hormis que pour le solo les positions du joueur 2 sont choisies par le pc et non un joueur. La vérification est sous forme de fonction pour une raison évidente, elle est appelée à chaque fin d'action soit 2 fois par tours.

Nous avons regroupé la 1^{er} phase où les joueurs placent leurs pions et celle où ils se déplacent et placent leurs « croix » pour une raison simple, le nombre de variables nécessaires était trop grand.

Données

-Les principales variables manipulées sont celles des coordonnées. Elles sont attribuées lors du placement des 2 joueurs et sont utilisées lors de leurs déplacements. Lorsque le joueur se déplace, des variables de position vérifient via une boucle la position du clic par rapport aux coordonnées des cases. Si le clic est produit à l'intérieur d'une case libre et adjacente, le pion est affiché au milieu de la case grâce aux coordonnées de vérification (elles prennent la forme de x,xx et y,yy). Les anciennes coordonnées sont utilisées pour afficher un fond blanc à la place de l'ancien pion et elles sont remplacées par celles du nouveau pion. Les coordonnées des 2 joueurs sont mémorisées dans des variables différentes.

-Les autres valeurs importantes sont celles contenues par le tableau. La vérification des cases est faite par une boucle. Cela permet de transposer la position des tokens. La valeur 1 (=cases occupées) est attribuée aux cases du tableau selon l'avancement de la boucle. Lorsque la position de clic correspond, les placements se font et la valeur d'avancement de la boucle sert de position dans le tableau. Cette position est mémorisée dans un autre couple de variable pour réinitialiser les anciennes positions du tableau à 0 (= cases libres) lorsque le pion n'est plus sur la case correspondante. Ce tableau sert à définir les cases occupées et les cases libres.

-La variable n sert juste à définir la taille de la grille. Elle est primordiale pour définir la taille des boucles de vérification.

-La séparation des phases de jeu est définie par une variable nommée b. selon la valeur qu'elle prend, les lignes de codes contenues dans les « if » (qui vérifient ça

valeur) s'exécute et lorsque l'action est réalisée (exemple le déplacement) la valeur de b change pour passer au « if » suivant. La dernière action renvoie au 1^{er}.

Conclusion :

MOCRET Killian :

Lors de la réalisation de ce projet, j'ai appris à lire un code méthodiquement, en effet lorsqu'un bug apparaît il m'était très utile d'avoir une parfaite compréhension de ce que j'ordonner à l'ordinateur et de pouvoir retranscrire étapes par étapes pour pouvoir vérifier où se produisait l'erreur.

La chose qui m'aurait été des plus utiles est l'organisation des priorités de codage. En effet lorsque certains groupes commencent par coder les choses les plus faciles et qu'ils savent faire, ils ne peuvent pas les relier, nous, nous avançons pas à pas pour ainsi pouvoir mieux se retrouver dans l'avancement du programme et pouvoir voir si l'action future ne pose pas problème aux actions précédentes et pouvoir utiliser les valeurs précédentes pour la suite.

En conclusion, ce projet a été quelque chose dans lequel j'ai apprécié porter mon intérêt mais si j'avais eu une plus grande maîtrise de langage C en particulier des pointeurs, je suis persuadé que la qualité de ce programme aurait été tirée vers le haut.

DA COSTA Mélissa :

Ce projet m'a permis de mieux m'organiser dans la production de mon code. De réfléchir avant de coder à décomposer les différentes étapes de construction du jeu. De comprendre les erreurs produites, de mieux les anticiper et de les corriger avec plus d'aisance. Il m'a permis aussi de mieux maîtriser certains points vus en TP, comme les séparations en différentes fonctions et la création du Makefile ainsi que la bibliothèque graphique. J'ai appris à gérer un projet, à gérer mon temps, à travailler en équipe et à diviser les tâches à faire. Cependant nous avons tout de même fait des erreurs facilement contournables si nous avions réfléchi à l'étendue du programme dès le départ. En effet lors de la construction de notre code nous ne pensions pas tout de suite à le rendre plus compacte et simple. Aussi, nous n'avons utilisé aucun pointeur, ce qui nous amène à avoir énormément de variables et d'avoir nos deux fonctions principales bien trop longues.