Boscher Leni
Domergue Tanguy
Groupe 6

Compte Rendu Projet Tutoré 1.1 : Le blocus

Table des matières

Boscher Leni
Domergue Tanguy
Groupe 6

Introduction :

Le blocus est notre premier projet à réaliser dans la matière Algorithmique, Programmation et Langages. Il s'agit d'un jeu que nous avons dû programmer en langage C, avec l'aide de la bibliothèque graphique de l'IUT.

Les règles du jeu sont simples, après avoir initialisé les dimensions de jeu qui peuvent aller de 3 à 9, vous jouerez à tour de rôle ; votre premier pion peut être placé dans n'importe quelle case, il vous sera ensuite demandé de placer une croix sur n'importe quelle case, rendant cette case condamnée, l'objectif est de bloquer votre adversaire pour qu'il ne puisse plus déplacer son pion. Vous aurez le droit de placer vos croix n'importe où, votre pion, quant à lui, ne pourra que se déplacer sur des cases adjacentes à sa position (diagonales comprises).

Fonctionnalités :

Notre programme est composé de :

- Un menu principal

Qui sera affiché en premier lors de l'exécution du programme, celui-ci permettra à l'utilisateur de lire les règles, et à l'aide du clic gauche de souris de faire :

(Exit) : Quitte le jeu

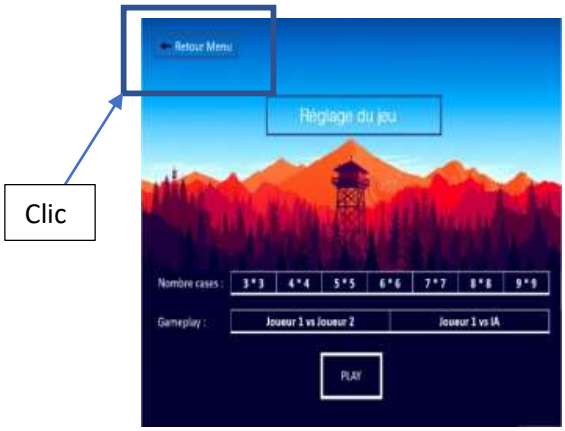
(Jouer) Menu qui permet de gérer les dimensions, le type de partie et le lancement de la partie) :

Boscher Leni
Domergue Tanguy
Groupe 6

Lorsque l'utilisateur appuie sur jouer, une nouvelle image de fond est générée avec de nouvelles options. C'est le menu réglage.

- Un menu réglage

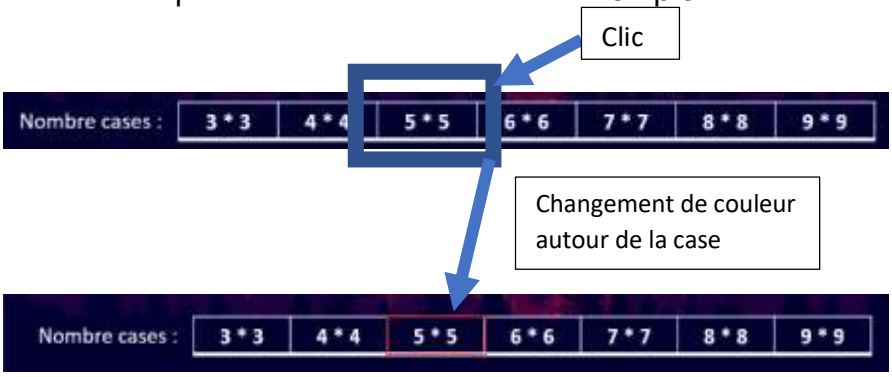
(Retour Menu) qui permet de retourner au menu principal vu précédemment.



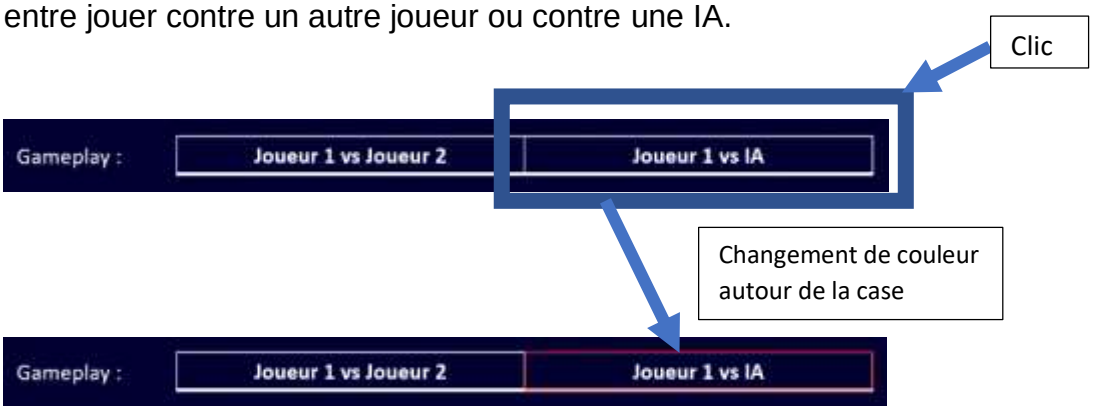
Retour au menu principal

Boscher Leni
Domergue Tanguy
Groupe 6

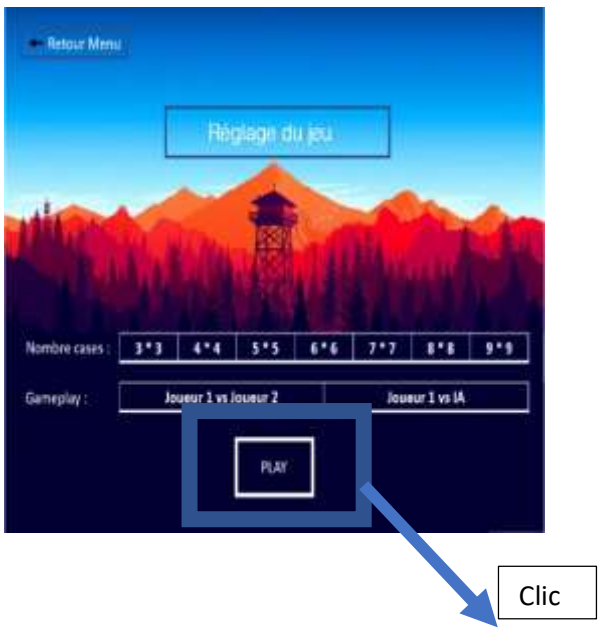
(Nombres cases) permet de choisir les dimensions de la grille, en appuyant sur les dimensions proposées celles-ci seront entourés d'un rectangle rouge pour montrer à l'utilisateur qu'il les a bien sélectionnés. Exemple :



(Gameplay) permet de sélectionner contre qui vous allez jouer, vous avez le choix entre jouer contre un autre joueur ou contre une IA.



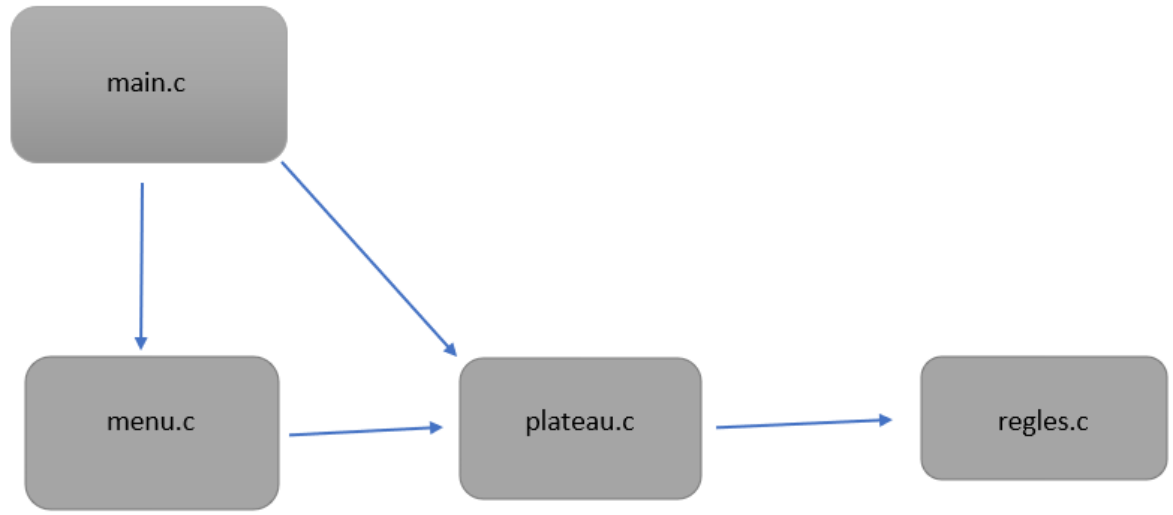
(Play) permet enfin de lancer le jeu selon les options que vous avez choisies.



Boscher Leni
Domergue Tanguy
Groupe 6

Présentation - Structure interne du programme

Notre programme est découpé en plusieurs fichier .c, qui exécute différentes fonctions, le programme principal les appelle les unes après les autres et permet de faire fonctionner notre jeu.



Le programme principal (le main) appelle le menu qui, via ses fonctions appelle le plateau de jeu qui possède ses restrictions (regles.c).

Main.c

```
int main(void){
    int number, versus;
    InitialiserGraphique();
    CreerFenetre(10,10, LargeF, HautF);
    if (accueil(&number, &versus)==0)return EXIT_SUCCESS;
    while(1) {
        if(SourisCliquee()) {
            if (_X>=322 && _X<=426 && _Y>=453 && _Y<=511){
            FermerGraphique();
            fflush(stdout);
            plateau(number, versus);
            break;
            }
        }
    }
 return EXIT_SUCCESS;
}
```

Boscher Leni
Domergue Tanguy
Groupe 6

Menu.c

Le fichier menu.c est composé de plusieurs fonctions :

La fonction accueil va permettre d'afficher le menu principal.

```
int accueil(int* number, int* versus){
    ChargerImageFond("./img/menu.png");
```

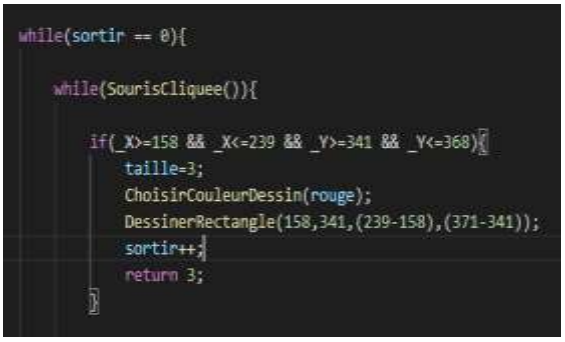
Elle va ensuite faire appel à une boucle qui contiendra des instructions en fonction de position en cas de clic.

Le premier if a pour condition la position du bouton « Exit » qui permet de quitter le programme lorsque l'on est dans le menu principal, d'où l'appel à la fonction « FermerGraphique ».

Le deuxième if possède comme condition la position du bouton « Jouer » il permet d'accéder au menu réglage, le menu qui permettra de déterminer les dimensions et le type de la partie, c'est pour cela que l'on fait appel aux fonctions dimension et gameplay, on stockera le retour de ces fonctions dans des pointeurs qui permettront de lancer le jeu en fonction des désirs du joueur.

Le troisième if a pour condition la position du bouton « Retour Menu » qui se trouve dans le menu réglage, il permet de retourner au menu principal lorsqu'on est dans le menu réglage.

La fonction **dimensions** va créer un int taille et un int sortir = 0.



```
while(sortir == 0){

    while(SourisCliquee()){

        if(_X>=158 && _X<=239 && _Y>=341 && _Y<=368){
            taille=3;
            ChoisirCouleurDessin(rouge);
            DessinerRectangle(158,341,(239-158),(371-341));
            sortir++;
            return 3;
        }
```

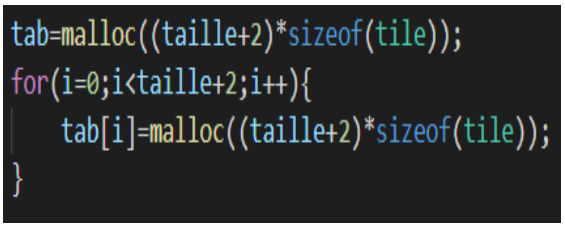
Dans le while(sourisClique()), il y a 9 if qui possèdent en conditions des positions, si il y a un clic sur une de ces positions la variable sortir est ajoutée par 1, on sort donc de la boucle, la fonction va renvoyer une valeur (entre 3 et 9) en fonction du clic de l'utilisateur.

La fonction **gameplay** fonctionne comme la fonction dimension, cependant elle permet de renvoi soit un 1, soit un 2 selon le clic de l'utilisateur, ces chiffres seront interprétés par une autre fonction qui lancera une partie à 2 joueurs ou un seul.

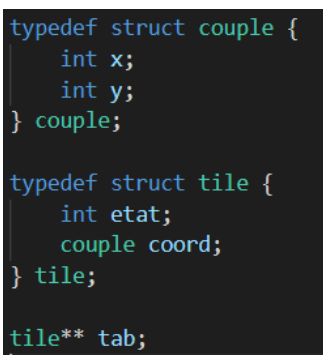
Plateau.c

Plateau.c n'est composé que d'une fonction qui déroule la partie en fonction de la taille de grille demandée et du nombre de joueurs.

- Prend deux arguments de type int.



```
tab=malloc((taille+2)*sizeof(tile));
for(i=0;i<taille+2;i++){
    tab[i]=malloc((taille+2)*sizeof(tile));
}
```



```
typedef struct couple {
    int x;
    int y;
} couple;

typedef struct tile {
    int etat;
    couple coord;
} tile;

tile** tab;
```

- Une des parties les plus importantes de la fonction est la structure de base du code du jeu.
- Un tableau dynamique est d'abord généré dans le but de stocker les informations de la grille entière.

- L'objectif est de, grâce aux structures, pouvoir retrouver l'état de la partie à tout moment, ainsi que les coordonnées et les indices de chaque case.

- En plus des informations de chaque case, le tableau génère des -1 à l'extérieur de la grille afin de ne pas pouvoir effectuer d'action à ces endroits.
- La fonction initialise d'abord la grille de jeu en fonction de la taille choisie par l'utilisateur.
- Le mode de jeu choisi par l'utilisateur définit quel type de partie sera joué, soit 2 joueurs en 1 contre 1, soit 1 joueur contre un ordinateur.
- Si le mode de jeu choisi est un 1 contre 1 entre 2 joueurs, le code se répète tour par tour, en changeant seulement les couleurs des pions et des croix.
- La variable "jouer" permet de savoir à quel tour nous sommes, si jouer vaut 1, le joueur 1 doit poser ou déplacer son pion, s'il vaut 3, c'est au joueur de 2 de poser ou déplacer son pion. Si jouer vaut 3, le joueur 1 doit poser sa croix, tandis que s'il vaut 4, c'est le joueur 2 qui pose sa croix. Cette variable prend une place dans le tableau dynamique en fonction de la case sur laquelle on a cliqué.

```
jouer=1;
while (1) {
    while(SourisCliquee()==1) {
            if(jouer==1) {
                SourisPosition();
                indice = calculIndice(_X, _Y);
                pion = chercherPion(jouer);
```

- Au clic, les éléments suivants sont déclenchés pour le déplacement des pions, on appelle les fonctions calculIndice et chercherPion décrites dans regles.c (ci-après).
- Pour le placement des croix, il n'y a pas besoin de la fonction chercherPion.

```
if(winCondition(chercherPion(1), chercherPion(3))==1) {
    break;
}
```

- A chaque fin de tour, la condition de victoire est vérifiée, et si c'est le cas, la boucle globale se termine et met fin au jeu.

```
while (jouer==3) {
        if(jouer==3) {
            _X=((rand()%DIMENSIONS)+decalage);
            _Y=((rand()%DIMENSIONS)+decalage);
            indice = calculIndice(_X, _Y);
            pion = chercherPion(jouer);
```

- Le fonctionnement pour l'ordinateur est un peu différent et fait appel à des coordonnées aléatoires et dès que les coordonnées respectent les restrictions, il effectue le coup en mémoire. Cela vaut autant pour les pions et les croix.

- La fin du fichier se résume à libérer les locations d'espaces du tableau dynamique puis de fermer la fenêtre après un clic et enfin de renvoyer sur le menu principal.
- Une particularité dans le code de tableau.c est qu'il nous a fallu effectuer de nombreuses conditions à chaque coup en fonction de la taille définie par l'utilisateur. En effet, nous avons dû charger des images de différentes tailles étant donné que nous avons décidé d'avoir des dimensions fixes pour la création de la grille.

Regles.c

La fonction **chercherPion** :

- Prend un argument de type int.
- Renvoie les indices du pion du joueur en train de jouer.
- Cette fonction sert à trouver sur quelle case se trouve le pion du joueur pendant son tour pour pouvoir déplacer son pion sans problèmes.

La fonction **calculIndice** :

- Prend deux arguments de type int.
- Renvoie les indices de la case sur laquelle l'utilisateur clique grâce aux coordonnées de la bibliothèque graphique _X et _Y.
- Cette fonction sert à la vérification et au déplacement des pions et des croix.

```
if((x>decalage+DIMENSIONS) || (x<decalage) || (y>decalage+DIMENSIONS) || (y<decalage)) {
    couple indice;
    indice.x=-1;
    indice.y=-1;
    return indice;
}
```

- Cette partie vérifie si l'on a cliqué en dehors de la grille, et si c'est le cas renvoie des indices égaux à -1 afin de ne pas pouvoir s'y déplacer grâce aux restrictions.

```
int caseVerif(couple indice) {
    if(indice.x == -1  || indice.y == -1) {
        return -1;
    }
    if(tab[indice.x][indice.y].etat==-1){
        return -1;
    }
    if (tab[indice.x][indice.y].etat==0){
        return 1;
    } else {
        return 0;
    }
```

La fonction **caseVerif** :

- Prend un argument de type couple.
- Renvoie -1, 0 ou 1, seul 1 permet de continuer le déroulement de la partie
- La première fonction de restriction de déplacement, qui vérifie si la case cliquée par l'utilisateur n'est pas déjà utilisée ou en dehors de la grille.

- Cette fonction est appelée au moment du clic et si la case n'est pas pleine elle autorise l'utilisateur de jouer son coup, sinon il doit cliquer autre part.

La fonction **deplacAdj** :

- Prend deux arguments de type couple.
- Renvoie 1 ou 0.
- La deuxième fonction de restriction, elle, vérifie que la case cliquée soit bien adjacente au joueur. Elle vérifie chaque case adjacente dont les diagonales autour du joueur.
- Cette fonction est appelée au même moment que la fonction caseVerif ci-dessus et le coup n'est autorisé à être joué que si les deux fonctions renvoient 1.

La fonction **winCondition** :

- Prend deux arguments de type couple.
- Renvoie 1 ou 0, si 0 le programme continue sans problèmes, si 1 le programme met fin à la partie avec l'écran de fin.
- La condition de victoire est de bloquer totalement son adversaire en faisant en sorte que toutes ses cases adjacentes soient bloquées. Ainsi, si l'état des cases adjacentes d'un joueur sont toutes bloquées au moment de jouer alors l'autre joueur a gagné.
- La fonction chercherPion est donnée en argument lors de l'appel de la fonction afin d'en faire les vérifications.

La fonction **AfficherTableau** :

- Ne prend pas d'argument.
- Ne renvoie rien
- Cette fonction est un soutien et permet d'afficher l'état de la grille entière dans le terminal à chaque coup joué.

Conclusions personnelles

- Leni BOSCHER

Pour un premier projet, il est évident que nous devons en tirer des leçons, et par là notamment une meilleure gestion du temps, et peut-être une meilleure organisation. Je me suis occupé du jeu tandis que Tanguy s'est occupé du menu. Cela ne nous a pas empêché de s'entraider en cas de problèmes.

Malgré cela, ce projet m'a grandement aidé à la compréhension du langage utilisé et je pense que j'en ressors grandi dans plusieurs domaines. Effectivement, je considère que ce projet m'a aidé à m'améliorer en C, et dans l'organisation de groupe pour la programmation.

Pour parler du contenu, je suis conscient que mon code n'est pas optimisé mais j'ai eu des difficultés pour les fonctions à l'origine. Malgré ça je suis fier de ma structure de données de base, que je trouve solide, c'est la partie sur laquelle j'ai eu le plus de mal, structurer mes données n'était pas aisé, mais je considère que je m'en suis bien sorti finalement.

Le temps m'a réellement fait défaut durant ce projet car je m'y suis pris trop tard, je reste positif en me disant que j'ai au moins réussi à le terminer.

Finalement, c'était un premier projet intéressant, qui m'a appris, j'en ressors avec de nouvelles connaissances et grandi en organisation de projet informatique.

- Tanguy DOMERGUE

Comme Leni l'a précisé précédemment, le temps n'a pas été en notre faveur, cependant, il ne nous a pas empêchés de rendre le devoir à temps. Avec mon camarade nous nous sommes répartis les tâches, il s'est occupé de la partie pure du jeu, quant à moi, je me suis occupé de l'aspect graphique du menu et des réglages, et du stockage des données en fonction des choix des utilisateurs.

Nous avons été très complices et à l'écoute l'un de l'autre, ce qui nous a permis d'avoir une très bonne entraide malgré la distance qui nous séparait.

Le développement du jeu m'a énormément aidé sur un point qui est l'organisation de code, qui était un défaut qui m'empêchait de comprendre mieux mes codes en me relisant, j'ai appris à maîtriser mieux mes fonctions contenues dans divers fichiers.

Je pense comme mon camarade que notre code pourrait être plus optimisé et plus court. Mais nous sommes quand même fiers de notre production.