Rapport de projet : IF2B

JEU HEX

Arthur BELLEMERE et Edgar BARRIOS
UTBM

Table des matières

Intro

Dans le cadre de l'UV IF2B, nous avons dû recréer en langage C un jeu intitulé hex. Ce jeu de plateau consiste à opposer deux joueurs avec chacun une couleur de pion, plaçant tour à tour un pion sur le plateau de jeu pour former une suite de points reliant deux bords du plateau.

L'objectif était d'offrir une simulation de ce jeu intuitive pour l'utilisateur et contrôlable en rentrant des caractères via un clavier.

Nous présenterons les différents aspects de la création de ce projet en commençant par la structure générale de notre code, puis nous décrirons plus en détails les fonctionnalités effectives permises par ce dernier et enfin nous offrirons un regard critique sur notre projet.

Pour cela nous avons réparti les explications entre 3 parties : une première partie "structure" où on explique grossièrement le rôle de chaque fonction et l'ordre dans lequel elles sont appelées, une seconde partie "fonctionnement" dans laquelle nous précisons certains en détail sur les solutions et choix de code utilisés pour faire fonctionner le programme, et une dernière appelée "difficultés rencontrées" expliquant les étapes sur lesquelles nous sommes restées bloqué et la solution que nous avons trouvé à ces problèmes.

Structure du code

1ER BLOC : MENU DE DEMARRAGE :

La première étape de notre programme, la fonction **menu** permet de démarrer, reprendre ou quitter la partie. Elle partage sa structure avec d'autres fonctions récurrentes demandant un unique caractère à l'utilisateur avec vérification du caractère entré.

Nous avons fait le choix pour toutes les fonctions de ce type, d'indiquer un unique caractère à entrer et non un mot (chaine de caractères) pour simplifier l'interaction. Cependant nous avons fait en sorte que chaque caractère puisse être rentré en majuscule ou en minuscule.

Cette fonction englobe toutes les autres dans le fichier main.c car le déroulement de la partie dépend de celle-ci.

2E BLOC DE FONCTION : LE PLATEAU DE JEU :

Ce bloc est constitué premièrement d'une fonction demandant à l'utilisateur la taille souhaitée pour le plateau de jeu (entre 9*9 et 14*14) : **la fonction taille_plateau.**

Ensuite nous avons défini un tableau appelé « Plateau » servant de plateau de jeu pendant la partie. C'est l'une des seules variables que nous définissons après le bloc menu car la fonction **taille_plateau** doit fonctionner une première fois au préalable.

Puis pour une meilleure visibilité, on utilise la fonction **initialisation_plateau** qui remplace chaque caractère du tableau plateau par « - » pour indiquer au joueur que la case est vide et qu'il peut y placer un pion.

Enfin nous avons créé une fonction **affichage_plateau** permettant d'imprimer le plateau de jeu. Il est constitué d'un quadrillage, chaque colonne correspond à une lettre et chaque ligne à un chiffre. De plus chaque ligne est légèrement décalée sur la droite par rapport à la précédente pour donner un effet de plateau hexagonal comme sur le jeu d'origine.

3E BLOC DE FONCTION : LE MODE DE JEU :

Il est ensuite demandé le mode de jeu souhaité parmi trois choix (standard, aléatoire, spécial). La majorité de notre travail s'est porté sur le mode standard que nous avons réadapté ensuite. Le mode de jeu est demandé à l'utilisateur avec la fonction **mode_jeu** calquée en grande partie sur la fonction **menu**.

Le déroulement d'une partie standard fonctionne depuis le main en appelant les différentes fonctions donnant lieu à la partie.

Nous avons également choisi de définir le mode aléatoire comme : partie normale + fonction **mode_aleatoire** (mettre schéma)

Ainsi mode de jeu demande le mode. Les 3 choix mènent au même déroulement utilisant la partie standard comme base pour les 2 autres modes. Mais si le mode aléatoire est choisi, des pions sont placés aléatoirement au lancement de la partie. Et dans le cas du mode spécial, 2 coups spéciaux sont proposés.

4E BLOC DE CODE : LES CONDITIONS INITIALES D'UNE PARTIE :

Dans un 4e bloc, nous avons ajouté les dernières informations nécessaires pour les joueurs avant de commencer une partie. La première est une simple impression des règles, indiquant également le chemin que chaque joueur doit emprunter en fonction de la couleur de pion qui leur sera attribué.

Puis nous avons créé une fonction récupérant le prénom des deux joueurs et leur assigne une couleur de pion : la fonction **tirage_aleatoire_couleur**. En effet la couleur attribuée est importante car dans les règles le joueur avec les pions blancs joue en premier et donc le joueur noir en deuxième.

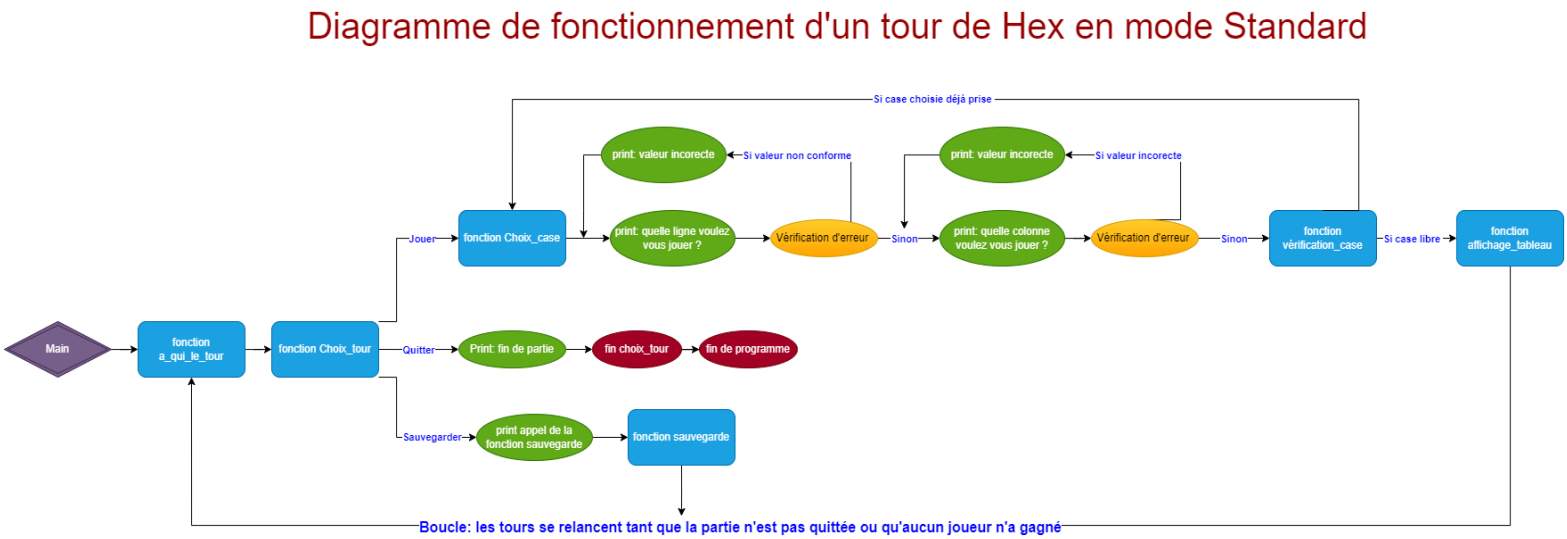
5E BLOC DE CODE : LES FONCTIONS DE FONCTIONNEMENT DU JEU :

Une fois les règles expliquées et les éléments nécessaires à une partie mis en place la partie commence. Le premier élément dont nous avions besoin pour commencer un tour était de savoir quel joueur est censé jouer. Nous avons répondu à cette question via la fonction **a_qui_le_tour**. Pour cela on compte le nombre de pions blancs et noirs posés sur le terrain. Comme les blancs jouent d'abord, et les noirs en deuxième on peut déduire qui doit jouer. Alors, s'il y a autant ou plus de pions blancs que noir c'est au tour des blancs, s'il y a moins de noirs c'est au tour des noirs.

Puis on propose trois choix à l'utilisateur à chaque tour via la fonction **choix_tour**. On propose 4 choix dans le cas du mode spécial. Cette fonction reprend également la structure de **menu**, elle est associée à un caractère rentré par l'utilisateur. Les trois choix sont alors : jouer le tour, sauvegarder la partie et quitter la partie. Et jouer un coup spécial dans le cas du mode spécial.

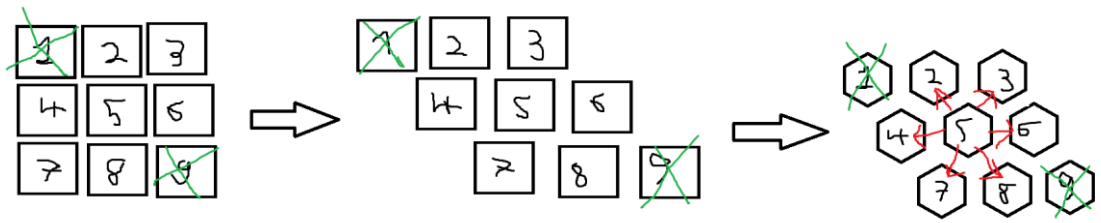
Ces 2 fonctions, a_qui_le_tour et choix_tour, ainsi que 2 autres qu'on va expliquer après sont alors utilisées en boucle tant qu'aucun joueur ne gagne ou tant que les joueurs ne quittent pas la partie.

La fonction **choix_tour** mène à l'utilisation d'autres fonctions selon la lettre rentrée. La plus complexe est celle correspondant à « jouer ». Appelé depuis la fonction **choix_tour**, la fonction **choix_case** demande à l'utilisateur les informations permettant de poser son pion dans la case souhaitée (ligne et colonne). Puis une autre fonction appelée depuis **choix_case** vérifie que la case est vide puis imprime le pion dans la case du plateau : la fonction **verification_case**. Enfin le plateau est affiché avec la fonction **affichage_plateau** évoquée précédemment.



6E BLOC DE CODE : LA VERIFICATION DE VICTOIRE :

Une dernière fonction est appelée depuis le main indépendamment de **choix_tour** à chaque tour. En effet une fois toutes les opérations liées à cette fonction effectuées, la fonction **test_gagner** permet de vérifier à la fin de chaque tour si l'un des deux joueurs a gagné. Une fois cette fonction appelée un tableau de la même dimension que le plateau appelé tableau_test est créé et initialisé à 0. Puis avec une boucle on vérifie la première colonne du plateau pour les blancs ou la première ligne pour les noirs. Pendant le tour des blancs, si un caractère 'B' est trouvé sur la 1ere colonne alors le 0 situé à la place correspondante dans tableau_test est remplacé par '1'. Idem avec 'N' pendant le tour des noirs avec la 1ere ligne. De plus deux variables « numerote_au_moins_1_case_N » et « numerote_au_moins_1_case_B » prennent respectivement la valeur 1 quand le caractère correspondant est trouvé.



Puis on a créé une boucle pour parcourir le plateau de jeu, d'abord pendant le tour des blancs, on parcourt chaque ligne de la deuxième colonne de haut en bas puis de bas en haut puis idem pour la 3e colonne etc… A chaque case ou le caractère 'B' apparait, on vérifie les 6 cases adjacentes au 'B' trouvé. Si, dans le tableau_test, un chiffre est présent parmi ces cases alors une variable appelée minimum prend la valeur la plus petite. Cette valeur est ensuite incrémentée, et mise à la place du B trouvé, dans le tableau_test. De cette manière on trouve les chemins de pions blancs et on les numérote en passant par le chemin le plus court.

Pendant le tour des noirs on parcours chaque colonne de la deuxième ligne de gauche à droite puis de droite à gauche et on répète la numérotation comme pour les pions blancs.

Une fois qu'un joueur a posé les pions d'une manière lui permettant de gagner, le tableau_test sera rempli avec un chemin de nombres croissants reliant deux extrémités. Or comme le chemin de nombre est écrit de manière croissante en partant du 1 posé alors :

Pendant la vérification des pions blanc, si un nombre est posé dans la colonne la plus à droite du plateau, alors le joueur blanc a gagné. Cette condition est vérifiée lorsque l'entier « colonne » utilisé pour poser le nombre à pour valeur taille-1. Si cette condition est vérifiée une variable de type entier appelé case_numerote_derniere_colonne prend une valeur de 1. Et quand cet entier vaut 1 un message de victoire s'affiche pour le joueur blanc.

Pendant la vérification des pions noirs, si un nombre est posé dans la ligne la plus basse du plateau, alors le joueur noir a gagné. Cette condition est vérifiée quand la valeur de la variable de type entier nommé ligne vaut taille-1. Dans ce cas c'est un entier appelé case_numerote_derniere_ligne qui prend la valeur de 1 et mène au message de victoire du joueur noir.

7E BLOC DE FONCTION : LA SAUVEGARDE ET LECTURE DE SAUVEGARDE :

Une fois la fonction **choix_tour** appelée, 3 choix s'offrent à l'utilisateur, 4 dans le cas du mode spécial. Les fonctions décrites précédemment, dans le 5eme bloc, sont appelés lorsque l'utilisateur entre 'J' pour jouer. Mais il peut également sauvegarder le plateau de jeu pour reprendre sa partie plus tard. Pour sauvegarder ces informations nous avons créé une fonction **sauvegarde**.

Dans un premier temps cette fonction copie 5 paramètres de la partie dans un fichier texte appelé « sauvegarde.txt », sans séparateur. Ces paramètres sont : le mode de jeu, la couleur de chaque joueur, le nom du joueur 1, le nom du joueur 2 et la taille du tableau.

Une fois ces paramètres copiés dans le fichier, on parcourt le plateau et on recopie chaque caractère à la suite dans le fichier txt.

Grâce à cette fonction les informations nécessaires ont été enregistrées. Ainsi nous avons créé une deuxième fonction appelée **reprendre_partie** permettant de lire ces informations lorsque l'on veut reprendre la dernière partie sauvegardée. Ces informations sont récupérées une par une, caractère par caractère.

On rappelle alors aux joueurs la couleur de leurs pions avec leur prénom et enfin on récupère chaque caractère du fichier texte correspondant aux cases du plateau et on les réécrit dans le tableau nommé plateau pour reprendre la partie comme elle était au moment où elle a été sauvegardée.

8E BLOC DE FONCTION : LE MODE SPECIAL :

A l'issue de la fonction **mode_jeu**, il est également possible de sélectionner un mode spécial. Ce mode spécial donne accès à deux coups spéciaux. On y accède au moment de l'appel de **choix_tour**, cela nous amène à un second menu offrant trois choix étant : le 1er coup spécial permettant de retirer un pion adverse, le 2eme coup spécial permettant de poser deux pions, et des messages d'informations expliquant le fonctionnement des coups spéciaux aux utilisateurs.

Le premier coup spécial s'active grâce à la fonction **retirer_pion**, qui reprends la structure de **choix_case**. Une fois la case choisie, une autre fonction **verification_retirer_pion** est appelé, elle vérifie que la case choisie contient un pion adverse et renvoie un message d'erreur si ce n'est pas le cas ainsi que renvoie à la fonction **retirer_pion** pour refaire son choix, sinon il remplacera le pion adverse par une valeur numérique. Cette valeur rend la case inutilisable pendant 3 tours.

Le deuxième coup spécial s'active grâce à la fonction **placer_2_pions**. Il permet de placer deux pions non adjacents en un seul tour mais empêche le joueur de jouer son prochain tour. Cette fonction reprend deux fois la structure de **choix_case** et garde en mémoire dans des variables de type entier, la ligne et la colonne des deux cases choisies. On vérifie ensuite que les deux cases ne sont pas adjacentes en additionnant ou soustrayant 1 au numéro de la colonne et de la ligne de la première case choisit pour vérifier que la 2e case choisie ne fait pas partie des 6 cases adjacentes à la première.

Une fois ces conditions vérifiées, les pions sont placés dans les cases prévues.

Fonctionnalités

1ER BLOC : BLOC MENU DE DEMARRAGE :

Toutes les fonctions partageant leur structure avec **menu**, sont constituées d'une acquisition avec message erreur. Formé d'un printf qui affiche une question ainsi que d'un scanf qui récupère la réponse de l'utilisateur. Il y a ensuite vérification d'erreur grâce à un while qui affiche un message d'erreur, puis le même printf et scanf quand la réponse ne fait pas partie des options proposées.

2E BLOC : BLOC PLATEAU DE JEU :

La fonction **taille_plateau** demande à l'utilisateur, puis retourne, une valeur entière. Car la taille du plateau est souvent utilisée dans d'autres fonctions. Donc nous avons créé un entier « taille » dans le main qui prend la valeur de cette fonction pour y faire référence dans d'autres fonctions.

Il est notable que le plateau de jeu est défini de manière dynamique, sa taille est définie par une variable, le programme risque de ne pas fonctionner sur les versions antérieures à C99.

La fonction **initialisation_tableau** est constituée d'une simple boucle for passant sur chaque case du tableau et remplaçant son contenu colonne par colonne puis ligne par ligne.

Pour afficher le tableau de jeu **affichage_tableau** parcourt le tableau avec deux

3E BLOC DE FONCTION : LE MODE DE JEU :

Une fois la fonction **mode_jeu** utilisée elle est associée à un char (appelé « mode ») comme la fonction **menu**.

Si le mode est “A”, donc aléatoire, un if appelle la fonction **mode_aléatoire** constituée de 2 boucles for qui place les N pions, de chaque couleur, aléatoirement puis déroule une partie normale. Cette fonction n’est donc pas appelée dans le cas du mode standard ou du mode spécial.

4E BLOC DE CODE : LES CONDITIONS INITIALES D’UNE PARTIE :

La fonction **tirage_aleatoire_couleur** récupère les prénoms des joueurs puis leur associe une couleur de pion. Au démarrage les joueurs ont 50% de chance de se faire attribuer une couleur ou l’autre. Pour cela on a créé une variable entière prenant comme valeur 0 ou 1 puis avec la fonction srand on tire une des deux valeurs. Si la valeur associée est 0 c’est le joueur 1 qui prend les pions noirs sinon c’est l’inverse. Cette attribution est affichée par un printf et réutilise le nom rentré précédemment par l’utilisateur pour plus de clarté.

5E BLOC DE CODE : LES FONCTIONS DE FONCTIONNEMENT DU JEU :

Pour compter le nombre de pions de chaque couleur sur le plateau dans la fonction **a_qui_le_tour** on utilise 2 boucles for pour parcourir le plateau colonne par colonne puis ligne par ligne et on incrémente un entier appelé nb_blanc à chaque caractère ‘B’ et idem pour un entier appelé nb_noir pour chaque caractère noir rencontré puis on les compare pour déterminer a qui est le tour.

Puis pour poser un pion via la fonction **choix_case** l’utilisateur rentre une lettre et un chiffre. Le chiffre est récupéré et stocké dans une variable entière appelé ligne. On soustrait ensuite 1 à la valeur de ligne étant donné que le tableau commence à la case [0,0] et non [1,1]. La valeur entrée est vérifiée avec message d’erreur via une boucle while, qui vérifie que la valeur est comprise entre 1 et la taille du tableau.

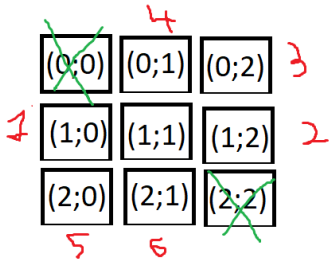
La même opération est ensuite effectuée pour la colonne, cependant la valeur récupérée est un char. Donc on créé une variable de type char appelée « lettre_colonne » et une variable de type entier appelée colonne. Puis en soustrayant le char « a » à la variable lettre colonne on obtient un nombre entier correspondant à la distance entre le caractère ‘a’ et le caractère entier entré. Par exemple la distance entre a et a est 0, entre a et b est 1, entre a et c est 2 etc… On associe ensuite cette valeur à la variable colonne.

La fonction **vérification_case** est ensuite appelée dans la **fonction choix_case**. Avec un if elle vérifie que le caractère présent dans la case choisie est bien « - » ce qui montre qu’elle est vide. Une fois ce paramètre vérifié, cette fonction remplace la valeur de la case choisie, par la valeur associée à « tour ». Pour rappelle tour prend la valeur ‘B’ pendant le tour des blancs et ‘N’ pendant le tour des noirs

6E BLOC DE CODE : LA VERIFICATION DE VICTOIRE :

Pour trouver le chemin suivi par les pions on trouve l'emplacement d'un pion, on trouve l'équivalent de sa case dans tableau_test puis on vérifie la présence de chiffre dans les 6 cases autour de cette dernière. Le code que nous avons utilisé pour vérifier ces cases est le suivant :

Pour une case du tableau choisie arbitrairement ici [1 ;1], les 6 cases qui lui sont adjacentes sur un plateau hexagonal sont celles non barrées. Ainsi pour définir leurs coordonnées on utilise



Case 1 : tableau[1][1-1] Case 2 : tableau[1][1+1] Case 3 : tableau[1-1][1+1]
Case 4 : tableau[1-1][1] Case 5 : tableau[1+1][1-1]Case 6 : tableau[1+1][1]

De la même manière, quand un pion est trouvé sur une case, les coordonnées des cases autour de lui sont définit de la même, manière : on prend les coordonnées ligne et colonne de la case et on teste les 6 variantes de cette coordonnée avec -1, +1 ou +0. Puis lorsque qu'un chiffre est trouvé l'entier appelé minimum prend la valeur du plus petit nombre trouvé dans ses cases et l'incrémente pour changer la valeur de la case vérifiée.

7E BLOC DE FONCTION : LA SAUVEGARDE ET LECTURE DE SAUVEGARDE :

Pour noter la couleur associée à chaque joueur, on réutilise la variable appelée « couleur » utilisée précédemment. Lorsque cet entier vaut 0, alors le nom enregistré comme nom_1 joue les pions noirs et si couleur vaut 1 alors nom_1 joue les pions blancs.

Les noms des joueurs sont recopiés avec un espace dans le fichier txt car ce sont les seuls paramètres de plus d'un caractère, l'espace sert donc de séparateur pour récupérer le string lorsque le fichier est lu.

Depuis la fonction **reprendre_partie**, on a besoin de récupérer deux informations directement dans le main. La première est le mode de jeu, qui est utilisé à plusieurs reprise pendant la partie. Pour le récupérer on définit le mode de jeu comme un pointeur et non comme un simple char pour pouvoir le sortir de la fonction. La deuxième information sortie de la fonction est la taille du tableau, qui contrairement à mode, est retournée à l'issue de la fonction, pour pouvoir être récupéré dans le main puis utilisé comme dans une partie normale.

Toutes les informations sont récupérées grâce à la fonction fscanf. Cette fonction est plus simple à utiliser que son alternative strtok mais ne récupère qu'un seul caractère à la fois. De ce fait nous avons utilisé deux boucles pour récupérer les prénoms caractère par caractère jusqu'au caractère espace placé au préalable dans **reprendre_partie**.

8E BLOC DE FONCTION : LE MODE SPECIAL :

Pour différencier les parties normales des parties spéciales on passe par mode_jeu. Une fois le mode spécial choisi, la condition mode_jeu = Z est remplie, c'est alors un If différent qui s'active dans choix_tour. Elle permet alors d'accéder au quatrième choix étant les coups spéciaux, avec le caractère 'C'.

Une fois le pion adverse retiré par la fonction **retirer_pion**, le caractère est remplacé par la valeur 4. Nous avons choisi cette écriture car lorsque la partie se déroule, la nouvelle fonction s'active au moment de l'affichage du plateau de jeu. Cette fonction, **tour_de_case_inutilisable**, parcours le plateau avec une double fonction for, ignore les caractères '-', 'B' et 'N', retire 1 à tout nombre présent sur le plateau et remplace le caractère 1 par '-'. Grace à cela une fois un pion retiré du terrain, il prend la valeur 4, puis diminue à chaque tour joué (4-3-2-1), et une fois que sa valeur est 1 la case redevient utilisable. Si on essaie de poser son pion sur la case inutilisable, comme la fonction verification_case vérifie que la case est un '-' alors si c'est un chiffre ça affichera le même message d'erreur que si on essayait de poser sur une case déjà prise par un pion et ce jusqu'à la case redevient un '-' après 3 tours.

Difficultés rencontrées :

1E BLOC MENU DE DEMARRAGE :

La première difficulté rencontrée était d'éviter la répétition de la question dans la fonction **menu** car le caractère « \n » étant relevé comme une réponse, le programme affichait un message d'erreur et posait deux fois la question systématiquement. Pour pallier ce problème nous avons utilisé « getchar () » pour se débarrasser de ce caractère. Nous avons utilisé getchar très fréquemment car il est nécessaire à chaque acquisition avec message d'erreur.

2E BLOC FONCTION : LE PLATEAU DE JEU :

Le tableau plateau de jeu n'étant définit que lorsque la fonction **taille_plateau** est appelé, il ne se définissait que lorsque l'utilisateur démarrait une partie. Il a donc fallu penser à redéfinir ce tableau dans le cas où l'utilisateur reprend une partie.

3E BLOC DE FONCTION : LE MODE DE JEU :

Le mode aléatoire et le mode spécial font partie de la même suite d'instructions que standard. Mais pour le mode aléatoire, la fonction **mode_aleatoire** s'ajoute au standard pour poser aléatoirement N nombre de pions de chaque couleur sur le plateau. De plus il a fallu faire en sorte que le mode aléatoire ne place pas deux pions au même endroit ce qui arrivait de temps à autre dans les premières versions du projet. Et pour le mode spécial, on a du ajouter un nouveau choix au menu choix_tour, on a réussi a le faire avec un boucle if qui test si on est en mode spécial.

4E BLOC DE CODE : LES CONDITIONS INITIALES D'UNE PARTIE :

La difficulté rencontrée avec la fonction **tirage_aleatoire_couleur** résidait dans l'acquisition du prénom des joueurs. En effet le prénom de chaque joueur est récupéré dans un string via la fonction fgets. Or le prénom récupéré par fgets est de la forme « prénom\n\0 ». De ce fait le nom des joueurs s'affichait toujours avec un retour à la ligne. Or cela réduisait la lisibilité de l'interface car lorsque le nom était imprimé, la phrase s'affichait puis était « coupée » par un retour à la ligne.

Pour pallier ce problème on cible l'avant dernier caractère du string (« \n ») et on le remplace par (« \0 »). Pour cela on « compte » le nombre de caractères du prénom du joueur avec la fonction strlen qu'on associe à un entier « len_nom ». Puis on désigne l'avant dernier caractère grâce à « len_nom -1 » qu'on remplace par le \0 pour terminer la chaine de caractère et « supprimer » le retour à la ligne.

6E BLOC DE CODE : LA VERIFICATION DE VICTOIRE :

En jouant au jeu, on s'est rendu compte que la fonction **test_gagner** ne pouvais pas vérifier une ligne qui allais dans le sens contraire que le sens dans lequel la fonction vérifie. Donc par exemple pour les blancs, pour une colonne, si la ligne montait dans cette colonne et que ça vérifie de haut en bas alors comme ceux en dessous n'était pas numérote par la fonction alors forcément les cases remplis 'B' au-dessus dans le plateau ne serait pas numérote parce que ceux en dessous ne sont pas numérotes dans le tableau_test. Il faut alors vérifier de haut en bas mais aussi de bas en haut pour pouvoir bien vérifier. Il reste cependant une faille dans cette fonction étant : si on revient sur ses pas avant de continuer sa ligne, donc en sorte de serpent, alors la fonction ne pourra le détecter bien que le joueur ait bien réussi à faire une ligne d'un côté a l'autre.

8E BLOC DE FONCTION : LE MODE SPECIAL :

Lorsque l'on utilise la fonction **poser_2_pions**, il est prévu que le joueur saute un tour. Cependant, comme le choix du tour se fait en comparant le nombre de pions de chaque couleur sur le plateau, le saut de tour se fait automatiquement la différence de pion sur le terrain ayant augmenté avec l'arrivée des deux pions à la suite

Conclusion

Ainsi, à l'issue de ce projet nous avons réussi à recréer le jeu Hex en langage c. avec les tests que nous avons effectué aucun dysfonctionnement n'a été trouvé cependant il reste des voies d'amélioration. La première est l'interface visuelle. En effet les caractères B et N imitent la présence de pions sur le plateau cependant avec des cases ou des caractères affichés en couleur le jeu serait plus intuitif. De plus avec chaque case définie par un hexagone affiché il serait plus simple de différencier chaque case. On pourrait aussi renommer le mode standard, « classique » pour rendre plus intuitive la lettre entrée à la sélection de mode de jeu. On pourrait également appeler le mode standard « classique » pour simplifier le choix de mode de jeu.