Rapport Projet Puissance 4

Premier jour :

Je ne voyais pas trop par où commencer j'ai donc fait d'abord ma structure Grille avec sept tableaux de taille 7 qui devaient correspondre aux lignes de ma grille. Mais en essayant de faire mes fonctions initialisation (qui me sert à initialiser à ma grille à chaque début de partie) et affiche_grille (qui me permet d'afficher ma grille) ma structure me posé un peu problème je ne voyais pas comment afficher les éléments de ma structure grille avec ma fonction affiche_grille. J'ai donc modifié ma structure en ne mettant qu'un tableau de 7 tableaux de taille 7 qui sur le moment représentait toujours mes lignes.
Une fois mes fonctions faîtes j'ai voulu les tester mais lors de mon compilage j'ai une erreur que je ne comprenais pas. J'ai mis 4 jours avant de me rendre compte que j'avais oublié de mettre #include "jeu.c".

Deuxième jour :

J'ai voulu commencer à faire ma fonction qui me servirait à savoir s'il y a un puissance 4 ou non dans ma grille. En réfléchissant à comment là faire je me suis rendu compte que ma structure Grille serait plus simple à utiliser si mes tableaux dans mon tableau représentaient les colonnes de ma grille au lieu des lignes, j'ai donc fait les changements nécessaires sur mes fonctions déjà existantes. Puis j'ai donc commencé la fonction puissance_4 en faisant tout d'abord des tests à la main pour voir où et comment on pouvait obtenir des puissances 4.
J'ai remarqué que pour les puissances 4 en ligne ou en colonne à partir de la ligne 4 ou de la colonne 4 (respectivement) si on n'avait pas trouvé de puissance 4 avec l'élément d'avant on n'avait pas besoin de continuer parce qu'on ne pourrait avoir au mieux qu'un puissance 3.
Et j'ai eu à peu près le même raisonnement avec les puissances 4 en diagonale : dans une grille on a des diagonales inférieures à 4 emplacements il est donc inutile de vérifier les diagonales en question c'est pour cela que dans mes deux dernières doubles boucles for mes la limites pour mes colonnes est la colonne 4 et pour mes lignes la ligne 4 aussi.
Je n'ai pas rencontré de gros problèmes lors de la création de cette fonction juste des oublis de « ; » et des inversions de variable entre « c » et « l ». Ma fonction puissance_4 contenais aussi au départ plusieurs break mais en faisant ma fonction plus tard en relisant mes fonctions j'ai changé pour mettre des return à la place.

Troisième jour :

J'ai décidé de m'attaquer aux actions que l'on peut effectuer sur la grille lors de la partie.
J'ai donc commencé par essayer de comprendre le fonctionnement des rotations sur une feuille pour trouver une « formule » qui me permettrait de mettre mes jetons à la bonne place après la rotation. Au début j'ai eu peur parce que je ne voyais pas comment faire autrement qu'en utilisant des valeurs absolues mais à force de regarder j'ai pu trouver comment faire autrement.
J'ai ensuite commencé à les coder puis à les tester. Les tests étaient concluant seul petit problème je n'avais pas de graviter ce qui me paraissais logique au vu de ce que j'avais coder, j'ai décidé de me préoccuper de ce problème plus tard et donc de continuer la dernière fonction qui me manquait mettre_jeton. Pour celle-là j'ai décidé de faire la graviter directement dedans parce que ça me paraissait plus simple à faire. Quand j'ai voulu la testait, si je voulais mettre un pion dans une colonne qui était entièrement vide si la colonne suivante n'était pas pleine mon jeton se retrouvé dans la colonne d'à côté, c'est pour cela que j'ai dû rajouter un if avant ma boucle for.

Quatrième jour :

J'ai créé une nouvelle structure Player pour pouvoir affecter plus facilement les jetons aux joueurs, j'ai pensé aussi à faire un union avec les deux types de jetons à l'intérieur mais je n'arrivais pas à savoir comment l'utiliser du coup j'ai abandonné l'idée.
J'ai fait ma fonction graviter qui au début ne me faisait descendre que les pions des 3-4 lignes du bas mais avec persévérance j'ai réussi à trouver le problème et à le modifier.
Puis j'ai fait une fonction action qui étais composé de ce qu'elle a déjà mais aussi de ce que qu'il y a dans la fonction choix_action sauf le 5/ car c'est pendant le cinquième jour que j'ai diviser ma fonction action en deux fonction : action et choix_action car quand j'ai voulu rajouter le choix 5/ quitter la partie ma fonction ne fonctionnais plus je l'ai donc diviser en deux.
J'ai ensuite fait une fonction compte qui compte le nombre de jeton qu'il y a dans la grille parce qu'au départ j'avais prévu de faire pour ma fonction final un compteur qui allait jusqu'à 49 pour le nombre de tour (pour lequel la grille serait complète) mais en y réfléchissant je me suis aperçu que puisqu'on pouvait aussi tourner la grille

dans nos actions possible on pouvait avoir plus de 49 tours avant que la grille soit complète. J'ai donc fait mon compteur en plus pour savoir si la grille est complète ou non.
J'ai aussi fait mes fonctions choix_jeton (qui permet de choisir notre jeton) et qui_commence (qui permet de choisir lequel des deux joueurs commence), ces deux fonctions sont similaires du coup ça n'a pas était très compliqué à les faire non plus. Seule différence j'avais mis dans ma fonction choix_jeton un if et un else pour affecter les jetons aux joueurs mais je les ai enlevés quand j'ai effectué le lendemain ma fonction play que j'ai renommer PvsP plus tard.

Cinquième jour :

J'ai commencé par ma fonction déroulement où j'avais mis plusieurs if avec les conditions nécessaires pour savoir si il y avait puissance 4 ou non et si oui un printf qui printer « puissance 4, le joueur 1 (ou 2) gagne » mais après mure réflexion j'ai préféré faire une fonction gagnant pour savoir si y en a un et si oui lequel. En testant ma fonction déroulement ça n'afficher pas mes grilles, c'était lié à un oubli dans ma fonction graviter que j'ai régler directement. Puis j'ai commencé ma fonction play (qui maintenant est ma fonction PvsP) pour cela j'avais besoin d'un affichage de départ j'ai donc rajouter la fonction affichage_debut et c'est en voulant tester cette fonction que j'ai eu un problème mes jetons n'étais plus des « 0 » et des « X » mais des « @ » et des « I » c'est pour cela que j'ai enlevé l'affectation de mes jetons de la fonction choix_jeton et que je les ai mise dans ma fonction PvsP.

Sixième jour :

Je dois faire mon IA, je décide de la faire en aléatoire parce que je ne sais pas trop comment faire autrement. J'ai fait tout d'abord mes fonction choix_action_M et choix_colonne_M qui doivent me donner des nombres aléatoires pour les actions de mon IA, pour le contenu de ces deux fonctions j'ai cherché sur internet parce que je n'avais absolument aucune idée de comment avoir des nombres aléatoires. Ensuite j'ai fait la fonction PvsM qui est un copié collé de ma fonction PvsP avec les modifications nécessaires pour jouer contre l'IA (la fonction deroulement remplacer par la fonction deroulement_0 que j'ai fait juste après).

J'ai fait une fonction mettre_jeton_ordi qui est similaire en tout point à ma fonction mettre_jeton mais où j'ai rajouté l'emplacement dans les paramètres.
Ma fonction deroulement_0 est un copié collé de la fonction deroulement où j'ai remplacé la partie du deuxième joueur par comment doit réagir l'IA. Au départ l'IA ne faisait vraiment que des choix aléatoires mais en testant de jouer contre elle j'ai remarqué que c'était pas très agréable parce qu'elle ne posait que rarement des jetons donc j'ai rajouter des conditions pour que déjà au départ elle pose forcément un jeton et puis j'ai fait une fonction compteur_jeton (qui compte le nombre de jeton pour chaque jeton et la différence de jeton poser entre les deux types de jetons) pour obliger l'IA à jouer des jetons plus souvent (tous les 2 jetons d'écart avec la fonction doit_mettre_jeton) et une fonction jeton_aligner (qui regarde si il y a une colonne avec trois jetons identiques d'affilais) pour qu'elle puisse au moins bloqué les puissances 4 en colonnes.

Septième jour :

J'ai remarqué qu'on ne savait pas trop ce que l'IA effectuer comme action (à part pour la mise des jetons parce que j'avais déjà prévu un printf pour cela) j'ai donc rajouté une fonction action_effectuer pour que ça nous print à chaque fois ce que l'IA à fait.
J'ai ensuite fait ma fonction cj (je n'avais plus d'inspiration pour les noms) qui nous permet de choisir quels types de partie nous voulons effectuer (PvsP ou PvsM).

Huitième jour :

J'ai essayé de faire les fonctions nécessaires pour pouvoir rajouter le mode MvsM mais ça me faisait des parties avec plus de 6675 tours (à ce niveau-là mon jeu plantait) du coup j'ai rajouté l'obligation de jouer un jeton si le nombre de tour modulo 4 valais 0 ou 1 du coup j'avais beaucoup moins de tour mais ça boguer au tour 29 et ne sachant plus quoi faire j'ai laissé tomber.
Puis j'ai fait ma fonction finale jeu.
Voici ce que j'avais fait si cela vous intéresse :

```
int deroulement_OvsO (Grille G, Player ordi1, Player ordi2, int pj){
    int g;
    int a;
    int jaj;
    int jao;
    int d = 0;
    int nbrjeton = 0;
    int tour = 0;
    affiche_grille(G);
    while(nbrjeton < 49){
        if((pj == 1 && (tour % 2) == 0 ) || (pj == 2 && (tour % 2) == 1)){
            printf("tour %d: c'est au tour de l'ordi 1 \n", tour);
```

```
        jao = jeton_aligner(G, ordi1);
        if(jao != 8){
            G = mettre_jeton_Ordi(G, ordi1.jeton, jao + 1);
        }
        jaj = jeton_aligner(G, ordi2);
        if(jaj != 8){
            G = mettre_jeton_Ordi(G, ordi1.jeton, jaj + 1);
        }
        else{
            a = choix_action_M();
            d = doit_mettre_jeton(G, ordi1, ordi2);
            if( a != 1 && d == 0 && tour != 0 && tour != 1 && ((tour % 4) != 0 || (tour % 4) != 1)){
                G = action(G, ordi1.jeton, a);
                action_effectuer(a);
            }
            else{
                a = choix_colonne_M();
                while(G.GC[a - 1][0] != ' '){
                a = choix_colonne_M();
                }
                G = mettre_jeton_Ordi(G, ordi1.jeton, a);
                printf("L'ordi1 à mis un jeton dans la colonne %d \n", a);
            }
        }
    }
    else{
        printf("tour %d: c'est au tour de l'ordi 2 \n", tour);
        jao = jeton_aligner(G, ordi2);
        if(jao != 8){
            G = mettre_jeton_Ordi(G, ordi2.jeton, jao + 1);
        }
        jaj = jeton_aligner(G, ordi1);
        if(jaj != 8){
            G = mettre_jeton_Ordi(G, ordi2.jeton, jaj + 1);
        }
        else{
            a = choix_action_M();
            d = doit_mettre_jeton(G, ordi1, ordi2);
            if( a != 1 && d == 0 && tour != 0 && tour != 1 && ((tour % 4) != 0 || (tour % 4) != 1)){
                G = action(G, ordi2.jeton, a);
                action_effectuer(a);
            }
            else{
                a = choix_colonne_M();
                while(G.GC[a - 1][0] != ' '){
                a = choix_colonne_M();
                }
                G = mettre_jeton_Ordi(G, ordi2.jeton, a);
                printf("L'ordi2 à mis un jeton dans la colonne %d \n", a);
            }
        }
```

```
        }
        g = gagnant(G, ordi1, ordi2);
        if(g == 1){
            return 0;
        }
        tour ++;
        nbrjeton = compte(G);
    }
    g = gagnant(G, ordi1, ordi2);
    if(g == 0){
        printf("aucun puissance 4, égalité");
    }
    return 0;
}

int MvsM (){
    int d;
    int j;
    int c;
    Grille plateau;
    Player ordi1;
    Player ordi2;
    d = affichage_debut();
    while(d == 1){
        j = choix_jeton_O();
        if(j == 1){
            ordi1.jeton = 'O';
            ordi2.jeton = 'X';
        }
        else{
            ordi1.jeton = 'X';
            ordi2.jeton = 'O';
        }
        c = qui_commence_O();
        plateau = initialisation();
        deroulement_OvsO(plateau, ordi1, ordi2, c);
        fprintf(stdout,"Voulez-vous recommencer? (1/ oui) (2/ non) \n");
        fscanf(stdin, "%d", &d);
    }
    return 0;
}



int choix_jeton_O (){
    srand(time(NULL));
    return rand() % 2 + 1;
}



int qui_commence_O (){
```

```
    srand(time(NULL));
    return rand() % 2 + 1;
}
```