Différentiel du premier rapport

Algorithme général

Le programme prend d'abord en compte l'existence d'options s'il y en a –grace à l'utilisation de getopt qui parse les commandes entrées en argument du main-, dans ce cas il remplit la structure correspondante avec les infos rentrées si jamais elles rentrent dans les règles données du morpion.
Le programme récupère la valeur, imposée à 5 s'il n'y a pas eu de précisions dans les options, corrigée de la grille et créée un tableau dynamique de *int* avec un *malloc*.
La grille est entièrement initialisée avec la valeur VIDE de type FAUX correspondant à la valeur 0.
Durant tout le reste du programme le tableau sera transmis entre les fonctions par pointeurs, de même que pour la structure *t_game* contenant les options et le nombre de tours joués.
Le programme est composé d'une boucle vérifiant le retour de fonction de FINISH qui arrive lorsque la grille a été intégralement remplie.
A chaque tour de boucle le programme demande la valeur de case à jouer, vérifie son existence et sa disponibilité.
Dans le cas où tout est bon alors la case est remplacée par le numéro de joueur correspondant au tour de jeu. La difficulté ici fut de trouver comment déterminer à quel joueur correspondait le tour de jeu.
Si jamais la case n'existe pas (inférieure ou supérieure à la taille de la grille) alors le programme reste bloqué dans la fonction récupérant les coordonnées de la case tant qu'aucune valeur correcte n'as été rentrée.

Une fois la case validée le programme part dans la vérification de l'alignement de 5 cases de manière soit horizontale, soit verticale, soit diagonale droite soit diagonale gauche. Dans le cas où 5 cases sont alignées alors le joueur auquel correspond la couleur gagne un point, la valeur est enregistrée dans un tableau situé dans la structure *t_game*.

Le programme reboucle au début et revérifie donc si toutes les cases de la grille ont été jouées. Si c'est le cas il affiche les scores et détermine qui a obtenu le score le plus haut pour afficher un gagnant.

Dans le cas de l'utilisation de l'option –c la boucle est sensiblement différente, la limite de joueur se trouve à 2 et dans ce cas la gestion du tour de jeu se fait avec un modulo 2. Le joueur gardera le même schéma algorithmique que vu précédemment, lorsque le tour de l'ordinateur arrive il utilise une fonction qui détermine quelle case il peut jouer en random.
On initialise préalablement srand avec time.
Une boucle check si la valeur random trouvée se situe dans la grille puis si la case est déjà remplie, dans le cas où la case est disponible le programme retourne dans les mêmes fonctions de remplissage que celles utilisées pour le joueur.

Conclusion

Il a été difficile de programmer de manière à pouvoir changer les règles au fur et à mesure du projet.

Une erreur au niveau de l'algorithme random m'as fait perdre énormément de temps, à savoir une inversion de x et y dans le remplissage de tableau qui menait à la réécriture sur des cases déjà remplies.

Dans la première version de l'ia prévue j'avais prévu un algorithme min-max sur un arbre, afin de mettre des niveaux de jeux il était prévu de donner sa construction en 3 niveaux de profondeur différents selon la demande du joueur.

Afin de ne pas reproduire tout le temps le même schéma de jeu le min-max devait calculer, dans le cas où il y aurait au moins deux évaluations égales sur un niveau, un random entre les deux branches concernées.

Il s'est avéré que la fonction d'évaluation était plus complexe que prévu et le second choix d'algorithme s'est porté sur un random défensif qui aurait réutilisé les fonctions calculant combien de points ont été alignés. Dans le cas de 4 points alignés 'libres' donc non compris les bords de la grille ou un autre joueur ayant joué à côté, l'ia devrait placer sur un des deux emplacements son pion. De même que dans le cas de 3 points alignés. Dans le cas où l'ia ne trouve plus, ou alors moins de 3 points alignés elle joue en random.

Eventuellement ajouter un calcul pour tenter de faire marquer des points dans ces cas-là à l'ia aurait été intéressant. Cependant l'erreur précédente ne m'as pas permis de finaliser cette version de l'ia.

Lors de ce projet j'ai pu découvrir la librairie optget qui m'as fait gagner énormément de temps sur le parsing que j'avais prévu, au début, de faire à la main.