[SCIF1b] Informatique

TP- Les sept merveilles du monde des sept couleurs

Table des matières

I. Introduction

On cherche à réaliser un jeu codé en C : les 7 couleurs. On pourra jouer entre humain mais aussi avec des intelligences artificielles. Le principe du jeu est le suivant : il y a un tableau de 30 par 30 de cases de 7 couleurs différentes. Les joueurs commencent dans un coin opposé et à chaque tour, un joueur choisit une couleur. Toutes les cases de cette couleur, adjacentes à son territoire, sera conquis par le joueur. Le premier joueur ayant la majorité des cases gagne. Ce projet a pour intérêt d'apprendre à coder en groupe (nous utiliserons Git), avec de la lisibilité (plusieurs fonctions, fichiers et des commentaires) et monter en compétence vis-à-vis du codage en C.

II. Voir le monde en 7 couleurs

Il y a deux joueurs le premier est assimilé au nombre 0, commence en bas à gauche et a pour caractère le "v" et le second au nombre 1, commence en haut à gauche et a pour caractère le "^".

A. Question 1

On génère le tableau de jeu avec des lettres A, B, C, D, E, F et G aléatoirement. Les deux cases de départ ont leur symbole. (fichier : map_gen.c)

B. Question 2

Désolé, on ne sait pas lire donc on a fait directement la question 3.

C. Question 3

Dans le fichier maj.c, on définit la fonction maj_board qui met à jour le tableau lorsque qu'un joueur annonce sa couleur. On a créé un algorithme récursif. On initialise la position de départ et le caractère associé au joueur puis on définit un tableau de visite pour ne pas regarder plusieurs fois la même case. On fait ensuite à la fonction search_cells qui visite une case, regarde autour d'elle. S'il y a une case non-visitée de la couleur annoncée, elle est modifiée et la fonction se relance sur cette case ; s'il y a une case non-visitée appartenant au joueur et la fonction se relance sur cette case. Ainsi, toutes les cases adjacentes au territoire final du joueur sont vérifiées. On a pris soin qu'un joueur ne peut pas jouer la couleur adverse (il absorberait le territoire adverse, gagnant la partie) en vérifiant que la couleur annoncée est bien valide. S'il donne une autre couleur, un message apparaît pour lui indiquer sa faute et il passe son tour. (fichier : maj.c)

III. A la conquête du monde

On cherche maintenant à se faire affronter deux joueurs.

A. Question 4

A ce stade, on fait un jeu qui ne s'arrête jamais. Les joueurs jouent l'un après l'autre en annonçant leur couleur. (fichier : human.c)

B. Question 5 :

La condition de victoire est qu'un joueur ait une majorité de case, c'est-à-dire au moins 30*30/2+1. Il y a une autre condition d'arrêt possible si les deux joueurs ont exactement 30*30/2 cases : c'est une égalité. Pour ce faire, à chaque mise à jour du tableau, on parcourt le tableau et on compte le nombre de cases occupées par le joueur. Par ailleurs, on affiche à chaque tour le pourcentage d'occupation de chacun des joueurs. La condition d'arrêt est définie dans la fonction game_end qui appelle une fonction de calcul de territoire appelée territory. (fichier : stop_condition.c, territory.c)

IV. La stratégie de l'aléa

On implémente un premier type d'intelligence artificielle (IA) basée sur l'aléatoire.

A. Question 6

On cherche à créer une IA la plus basique qu'il soit : à chaque tour, l'IA choisit une couleur tout à fait au hasard (fonction alea1_color). Par ailleurs, on ajoute un bout de code au début du jeu pour choisir les types de joueurs. (fichier : alea1.c)

B. Question 7

On améliore un peu cette IA : elle ne prend au hasard une couleur qui est parmi celles adjacentes au territoire (fonction alea2_color). Pour cela on utilise une fonction appelée search_colors, assez proche de la fonction de recherche de cellules pour la mise à jour, sauf qu'on regarde juste les cases adjacentes et on met à 1 la case correspondante à la couleur dans un tableau ayant une valeur par couleur. Ensuite, on tire des lettres aléatoirement jusqu'à que la lettre obtenue soit une des lettres

répertoriées dans le tableau. On met à jour le début du programme pour choisir le type de joueur. (fichier : alea2.c)

V. La loi du plus fort

A. Question 8

On fait un nouvel algorithme, plus intelligent, qui ajoute à chaque tour la couleur qui lui fait gagner le plus de cases (fonction glouton_color). On reprend la fonction de la question 7, sauf qu'au lieu de simplement regarder s'il y a une couleur, on regarde combien il y en a (fonction count_colors). On ajoute 1 à chaque fois à la variable associée dans le tableau. Enfin, on choisit la lettre avec le nombre d'obtention maximum. (fichier : glouton.c)

B. Question 9

On fait affronter les IA semi-aléatoire et gloutonne. Cette dernière gagne. Pour faire combat juste pour pouvoir comparer les performances entre les IA, il faudrait générer un monde symétrique. On aurait donc le même milieu pour les deux.

C. Question 10

On implémente un choix de nombre de parties et un indicateur de nombres de parties gagnées. On lance 100 parties entre l'IA aléatoire et l'IA glouton, et cette dernière les gagne toutes.

VI. Les nombreuses huitièmes merveilles du monde (bonus)

A. Question 11

On crée une IA qui maxime le périmètre du territoire à chaque coup : son but est de s'étendre vite. Dans la fonction hegemonique_color, on lance la fonction count_perimeter pour chaque couleur. Elle compte chaque case adjacente à l'ensemble du territoire et de la couleur annoncée. On joue ensuite la couleur qui augmente le plus le périmètre. On a essayé le joueur mais on a constaté un problème dans la Figure 1. Dans ce cas, le joueur hégémonique souhaite jouer la lettre G car un autre choix de couleur réduirait son périmètre ; cependant, il n'y a aucune lettre G dans son périmètre ! Il est donc bloqué.

Techniquement, cette IA répond au cahier des charges mais on corrige tout de même ce problème pour lui permettre de jouer à chaque tour. Maintenant, on utilise la fonction search_color (définie dans le fichier alea2.c) pour vérifier que l'IA joue toujours une couleur qui ajoute au moins une case à son territoire. On obtient que le joueur hégémonique est moins performant que le joueur glouton mais un peu meilleur que l'alea2. (fichier : hegemonique.c)

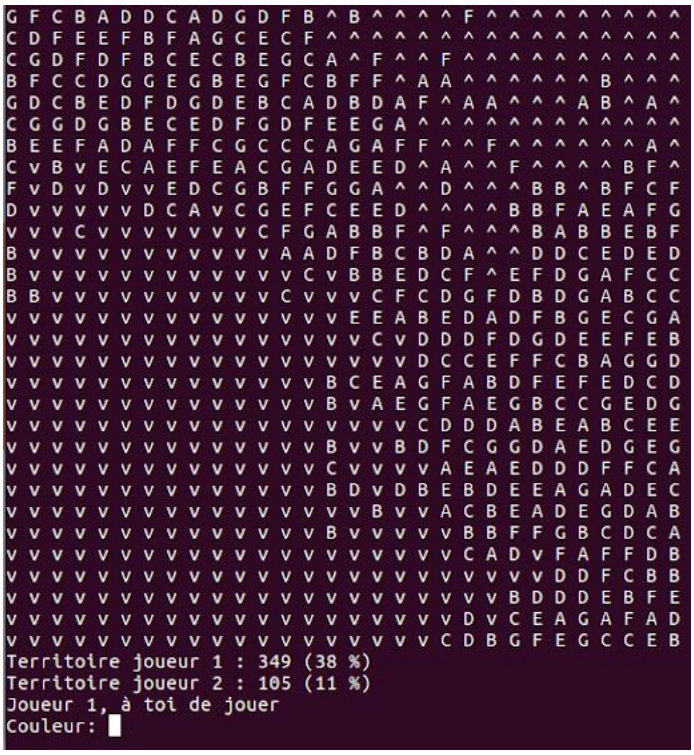

Figure 1 : Situation où le joueur hégémonique est bloqué

B. Question 12

On propose maintenant une amélioration de l'IA gloutonne : maintenant, elle anticipe le coup qui lui permet de gagner le plus de cases en 2 tours (fonction smart_glouton_color). On compte pour chacune des combinaisons de couleurs le nombre de cases qui serait dans le nouveau territoire (fonction count_territory). Cette IA ajoute en moyenne 10% de cases en plus que l'IA glouton (fichier : smart_glouton.c)

VII. Synthèse

Nous avons implémenté dans le jeu des sept couleurs plusieurs types de joueurs possible pouvant s'affronter : on a l'humain ou des intelligences artificielles. Dans l'ordre croissant de difficulté, il y a l'IA aléatoire de niveau 1 et son amélioration. Ensuite, il y a l'IA hégémonique qui optimise le périmètre du territoire et la gloutonne qui optimise le nombre de cases ajoutées à chaque tour. Enfin, il y a la gloutonne intelligente qui anticipe le coup d'après pour ajouter le plus de cases en deux coups.