Projet C final :

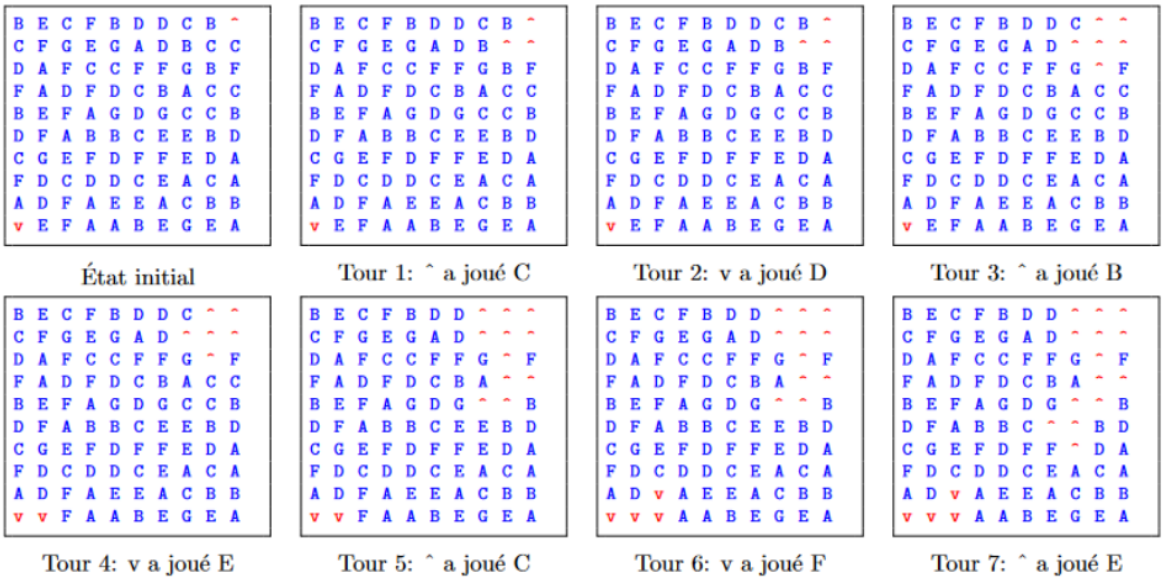
Les 7 merveilles du monde des 7 couleurs

Sujet inspiré du sujet de TP final de M.Quinson

L'objectif de ce projet est de coder successivement un petit jeu et une « intelligence artificielle » qui y joue. Deux stratégies pour cette « intelligence artificielle » seront proposées dans le projet. Sachez qu'à la fin, je vous ferais un rendu graphique du tournois entre vos différentes intelligences artificielles et que tant qu'elles correspondent aux contraintes définies plus loin, vous serez libre d'en proposer plusieurs.

Règles du jeu :

Voici un exemple de début de partie de ce jeu de stratégie :



Ici les joueurs v et ^ s'affrontent pour occuper le maximum de territoire dans le monde carré du jeu des 7 couleurs. Pour jouer, les joueurs vont tour après tour choisir une des sept couleurs (représentées par les lettres de A à G) et va conquérir la totalité du territoire de la couleur choisie qui est adjacent à son territoire. Le territoire de chaque joueur est d'une couleur unique qu'il n'est pas possible de choisir. La carte est initialement remplie de couleurs aléatoires et chaque joueur occupe un coin de la carte (respectivement en haut à droite et en bas à gauche). La partie s'arrête lorsqu'un joueur a recouvert plus de 50% de la carte ou que les deux joueurs ne peuvent plus gagner de territoire. Voici une démonstration du jeu original, qui date de 1991 : https://youtu.be/o1Lz4EIzY9I
Comme vous pouvez le constater, dans le jeu original, le territoire d'un joueur est représenté par la couleur choisie au tour précédent et les règles sont légèrement différentes. Si un joueur choisi une couleur qui n'est pas adjacente à son territoire, tant pis pour lui, il perd son tour.

Règles spécifiques au projet :

Votre projet devra être rendu au moyen d'un github (forshadowing).

Pour ce projet un fichier GameState.h, un fichier GameState.c et un Makefile simple vous seront fournis, ainsi qu'une arborescence qu'il faudra respecter. Le fichier .h fourni ne devra pas être modifié, et le fichier .c devra uniquement être complété. Le terrain de jeu est décrit par la structure

GameState, qui contient un tableau de Color (enumération) map que vous devrez allouer sur le tas (heap), ainsi que la taille du terrain.

Chaque fonction et variable globale que **vous définirez** devra être préfixée par GRnuméro_de_votre_groupe_ Un exemple : si vous êtes dans le groupe 5, le nom des fonctions que vous écrirez commencera par GR5_

Les fonctions dédiées à l'affichage devront être dans un fichier différent des fonctions dédiées au fonctionnement du jeu.

Les bonus ne seront pris en compte que si ***l'intégralité du projet de base fonctionne !!***

Le projet que vous enverrez doit être **compilable au moyen d'un makefile**.

(conseil :) Ecrire des fonctions de tests et compiler/tester au fur et à mesure vous permettra de détecter les éventuels problèmes

Vous devrez également fournir un rapport qui explique ce que vous faîtes dans le projet. Si cela ne fonctionne pas, ce sera votre filet de secours, pensez bien à détailler votre approche (pas ligne par ligne, mais l'idée générale).

La map sera un tableau 1D contenant les informations. Ce tableau 1D sera ensuite utilisé comme un tableau 2D à l'aide des fonctions get_map_value et set_map_value.

Travail à réaliser :

La base du jeu :

Question 1 :

Compléter les fonctions dans le fichier GameState.c. Voici leur rôle respectif :

create_empty_game_state : initialise le GameState passée par référence en paramètre en allouant sur le tas l'espace nécessaire pour un terrain de jeu carré de côté size (map).

set_map_value : fixe la couleur dans la position x, y.

fill_map : remplis la carte de valeurs aléatoires, puis positionne les joueurs aux positions :

Player1 : x = 0, y = size-1

Player2 : x = size-1, y = 0

La fonction main doit lire le premier argument passé à la ligne de commande qui définira la taille de la carte, générer la carte, puis lancer la partie

Question 2 :

Définir une fonction pour afficher l'état actuel de la partie (avec des lettres dans le terminal).

bonus : faire un affichage graphique dans le terminal (en couleurs)

bonus : faire un affichage graphique (seulement si tout le reste a été traité)

Question3 :

Définir une fonction qui met à jour le monde après le coup d'un joueur. Voici une stratégie qui fonctionne, mais peut être améliorée en bonus :

Parcourir linéairement le monde jusqu'à trouver une case à mettre à jour (qui est de la couleur indiquée et dont une case adjacente est déjà dans la zone du joueur) et la mettre dans la zone. Continuer à parcourir le monde jusqu'au bout. Si j'ai modifié au moins une case en parcourant le monde, le reparcourir. S'arrêter lorsqu'on fait un parcours complet sans changement.

Question 4 :

Écrire une fonction qui permet de déterminer si le jeu est fini et quel est le vainqueur

Question 5 :

Ecrire les fonctions nécessaires pour permettre à deux joueurs humains de s'affronter. Dans le rapport, exposez les limites éventuelles de votre implémentation.

Intelligence artificielle :

Avoir un jeu qui fonctionne pour s'affronter entre humains est une chose, mais c'est un peu nul quand on est tout seul comme votre correcteur. Voilà pourquoi vous devrez réaliser une ou plusieurs « intelligences artificielles » pour me permettre de jouer à ce jeu que vous venez de réaliser. Afin d'être considéré valide, un joueur artificiel doit fournir une fonction qui prend en paramètre uniquement (dans cet ordre) Un pointeur vers un élément de type Map et la couleur du joueur qu'il joue et retournera la couleur qu'il veut jouer

Voici celles qui sont obligatoires :

« Je fais n'importe quoi »

Question 6 :

Ecrire un joueur artificiel qui joue un coup aléatoire.

Question 7 :

Ecrire un joueur artificiel qui joue un coup aléatoire, parmi les choix qui améliorent son territoire.

« le glouton»

Question 8 : écrire un joueur artificiel qui joue le coup qui lui fait gagner le plus de territoire à ce coup ci

Question 9 : faire s'affronter le joueur de la question 6 avec le joueur glouton. Sur 500 parties, lequel

gagne le plus ?

« Hégémonique »

Question 10 :

Ecrire un joueur artificiel qui joue le coup qui maximise les frontières de son territoire

« Mixte »

Question 11 :

Ecrire un joueur artificiel qui maximise les frontières de son territoire, mais joue comme un glouton lorsqu'il ne lui est plus possible de faire progresser son territoire.

Question 12 :

Faire un tournois entre vos IAs sur plusieurs tailles de grille et les comparer (dans le rapport)

Question 13 (bonus) : adapter votre programme pour que quand on lance la ligne de commande ./Votre_programme taille_de_la_carte --ia nom_de_lia numero_du_joueur liste_etats le programme affiche et retourne le numéro de la couleur qui serait joué par l'ia concernée.

Le numéro du joueur sera la couleur du joueur

La liste des états sera dans le sens « normal » de lecture (de gauche à droite et de haut en bas).

Par exemple :

./7color 2 --ia random_player 1 1 3 5 2 renverra (par exemple) 4

La grille sera donc 1 3
5 2

Question 13 bis (bonus) : permettez à votre programme de jouer contre un autre groupe en utilisant sa ligne de commande

L'arène (bonus)

Proposez plusieurs intelligences artificielles que je ferais s'affronter entre elles dans un combat coloré afin de déterminer le vainqueur du tournois. Il existe de nombreuses stratégies, je vous conseille de regarder par exemple min-max, ou d'autre stratégie de maximisation des frontières. Attention, pour que le tournois ne dure pas une éternité, votre intelligence artificielle devra répondre un coup en moins de 500ms. Si ce n'est pas le cas, elle passera son tour.