le cnam

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL ASSOCIE DE STRASBOURG

Module : Paradigmes de programmation

Séance 1 : Programmation Orientée Objet en C#, application via la modélisation d'un jeu vidéo de morpion.

Intervenant : Edouard MANGEL

A l'issue de ce cours théorique, il est temps de passer à la pratique. Vous réaliserez ainsi le programme suivant en C# dans le logiciel avec lequel vous êtes à l'aise, en autonomie.

Vous écrirez **un** programme composé de plusieurs classes qui seront en rapport avec les différents paragraphes suivants. Vous serez évalués sur la qualité du code, l'application des bonnes pratiques évoquées lors des cours théoriques et l'absence d'erreurs dans vos programmes.

Les questions bonus sont à réaliser <u>uniquement</u> quand toutes les autres questions ont été développées.

Les sources sont à mettre sur un repo git **public** avant la veille de la prochaine séance à 23h59. Je vous mettrai à disposition un fichier partagé où donner l'adresse de votre repo git.

L'adresse donnée doit permettre d'exécuter la commande « git clone » sans action supplémentaire (autre qu'éventuellement l'authentification).

Objectif : programmer un jeu de morpion complet et évolutif :

Traditionnellement, un morpion se joue à deux joueurs sur une grille de 3 x3. Le plateau de jeu peut être modélisé par un tableau d'entiers à deux dimensions. En C#, un tableau en deux dimensions d'entiers peut s'écrire :

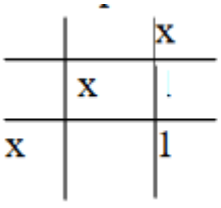
int nom_tableau[NB_LIGNES][NB_COLONNES];

Par convention, le premier indice entre crochet désigne la ligne (entre 0 et NB-LIGNES-1) et le deuxième indice désigne la colonne (entre 0 et NB-COLONNES-1).

Dans notre cas, on pourra donc définir une grille de morpion de la manière suivante :

int grille[3][3];

Dans notre exemple,



0 0 1
0 1 2
1 0 2

grille[0][2] = 1, grille[2][2]=2, grille[1][0]=0 ;

On convient que si un élément (une case) vaut 0 alors personne n'a joué cette case, si une case vaut 1alors le joueur 1 a déposé un jeton sur cette case. Si une case vaut 2 alors le joueur 2 a déposé un jeton sur cette case.

I La classe Grille Morpion

Ecrivez une classe « grille de morpion » ayant les méthodes suivantes :

•Case vide : retourne true ou false si la case donnée en paramètre est libre ou non.

•Déposer un jeton. Cette méthode permet de déposer un jeton d'un joueur donné sur la case spécifiée en paramètre.

•Ligne complète. retourne true si la ligne donnée en paramètre est rempli par des jetons du joueur dont l'identifiant est donné en paramètre.

•Colonne complète. retourne true si la colonne donnée en paramètre est rempli par des jetons du joueur dont l'identifiant est donné en paramètre.

•Diagonale complète. retourne true si la diagonale donnée en paramètre est rempli par des jetons du joueur dont l'identifiant est donné en paramètre.

• Victoire joueur. retourne true si l'utilisateur dont l'identifiant est donné en paramètre a rempli une condition de victoire.

•Affichage de la grille

II La classe Jeu

Créez une classe Jeu qui permettra de jouer une partie à deux joueurs successivement sur le même clavier.

Chaque tour de jeu se déroule de la façon suivante :

1. Le joueur dont c'est le tour est invité à saisir la case sur laquelle il veut poser un pion,
2. On vérifie que la valeur saisie est bien un entier compris entre 1 et le nombre de lignes, sinon on redemande de saisir.
3. On vérifie que la case sélectionnée est vide et dans les bornes du tableau,
4. Si c'est le cas, on place le pion pour le joueur et on affiche la grille,

5. On vérifie si le joueur a gagné, si oui, la partie est terminée et on félicite le vainqueur.
6. On vérifie si la grille est pleine, si oui on annonce le match nul et on propose de jouer à nouveau.

III Le puissance 4

Vous avez maintenant un morpion fonctionnel, vous allez maintenant développer un puissance 4. Pour cela, vous allez ajouter une classe GrillePuissance4 dont la largeur sera de 7, et la hauteur de 4, soit 4 lignes et 7 colonnes.

Règle du puissance 4 : chaque joueur joue alternativement (d'abord le premier, le deuxième, ...). Le joueur choisit une case non jouée (sans jeton) pour déposer un de ces jetons. Un joueur gagne la partie lorsque qu'il a réussi à remplir une ligne, une colonne, ou une diagonale de ces jetons. Contrairement au morpion, on choisit uniquement la colonne dans laquelle on joue. On parcourt ensuite la colonne, le pion est placé dans la case la plus basse qui soit vide au moment de jouer.

Contrairement au morpion, le joueur ne gagne pas quand il remplit une ligne, une colonne ou une diagonale, mais quand il aligne 4 pions de sa couleur en ligne en colonne ou en diagonale.

Vous développerez donc les mêmes méthodes que pour la classe GrilleMorpion, à savoir celles qui permettent d'afficher la grille, de placer un pion, de vérifier qu'un joueur a gagné.

Vous modifierez votre classe Jeu pour qu'elle puisse prendre en paramètre indifféremment une grille de morpion ou de puissance 4, et diriger une partie de l'un ou de l'autre.

IV Bonus :

Que faudrait-il faire pour modifier la classe jeu pour qu'elle puisse prendre en paramètre n'importe quelle Grille, et qu'elle puisse administrer une partie selon les règles de n'importe quel jeu prenant une grille rectangulaire composée de cases carrées ?

Sans la coder, proposez une solution modulaire qui soit facilement adaptable en argumentant sommairement et en agrémentant éventuellement d'un ou plusieurs diagrammes UML qui vous sembleraient adaptés.