Le jeu des MOTS GLISSES

Préambule

Ce travail est à réaliser en binôme du même groupe

Il est à rendre au plus tard à la fin de votre dernière session, semaine qui précède les vacances de Noël

Vous déposerez votre solution Visual Studio sur DVO au format .zip sous vos noms

Attention à ne pas déposer une partie de votre solution, vérifiez que vous avez intégré tous les fichiers nécessaires.

Un programme ne compilant pas ou ne s'exécutant pas entraîne une note de 0/20.

L'ensemble des codes sera analysé par un système anti-plagiat. Un plagiat entraîne une note de 0/20 au module (pour les 2 protagonistes).

Le sujet proposé a pour objectif de mettre en application tous les concepts vus en TD, c'est pourquoi il est impératif de suivre les énoncés tels qu'ils ont été écrits.

La dernière séance donnera lieu à une revue de code. Un étudiant doit être en mesure de décrire n'importe quelle partie de code (y compris une portion écrite par son binôme).

Recommandations générales

Ne pas oublier votre projet Test Unitaire sur au moins le test de 5 fonctions.
Ne pas oublier les commentaires ///
Ne pas oublier d'écrire les variables et fonctions avec des noms lisibles
Ne pas oublier l'indentation.

Un ensemble de méthodes sont imposées pour faciliter une correction précise

Vous pouvez rajouter d'autres méthodes si les besoins de votre code l'imposent évidemment.

Présentation du problème

Le jeu se présente comme un plateau de lettres choisies au hasard parmi un ensemble de lettres de l'alphabet.

Deux joueurs sont en compétition, chacun ayant tour à tour un délai à respecter pour trouver des mots sur ce plateau de la manière suivante :

La recherche des mots se fait systématiquement depuis la base de la matrice (présentée en rouge sur l'exemple fourni)

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e	n	j	n
s	d	h	a	n	o	g	r
r	r	c	a	i	s	c	p
c	b	e	m	a	i	u	s

A partir de la base, vous pouvez aller à gauche, à droite ou monter à la verticale pour composer votre mot. Les mots composés en diagonales sont à traiter une fois le jeu opérationnel.

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e	**n**	j	n
s	d	h	a	n	**o**	g	r
r	r	c	a	i	**s**	c	p
c	b	e	**m**	**a**	**i**	u	s

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e	n	j	n
s	d	h	a	**n**	**o**	g	r
r	r	c	a	i	**s**	c	p
c	b	e	**m**	**a**	**i**	u	s

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e	**n**	j	n
s	d	h	a	n	**o**	g	r
r	r	c	**a**	**i**	**s**	c	p
c	b	e	**m**	a	i	u	s

Le joueur propose un mot, (en l'occurrence maison ici) et votre jeu en fonction de la stratégie adoptée dans votre code privilégiera l'une ou l'autre des directions. Le mot doit être d'au moins 2 lettres. Si le mot n'est pas compatible avec le plateau, il faudra retourner un message d'erreur et reproposer au joueur de saisir un nouveau mot. Par ailleurs le joueur n'a pas le droit de proposer un mot qu'il a déjà trouvé.

Une fois le mot trouvé sur la grille, il faut s'assurer que celui-ci appartienne bien au dictionnaire français fourni. (MotsFrançais.txt)

Une fois toutes ces contraintes respectées, les lettres de chacune des colonnes impactées par le mot vont glisser vers le bas afin de combler toutes les cases utilisées par le mot (à la manière du Puissance 4)

Supposons la stratégie opérée suivante pour le mot « maison »

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e	n	j	n
s	d	h	a	n	o	g	r
r	r	c	a	i	s	c	p
c	b	e	m	a	i	u	s

a	b	c	d	e	f	g	h
e	j	s	t	e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	u	e		j	n
s	d	h	a	n		g	r
r	r	c				c	p
c	b	e		a	i	u	s

Colonne par colonne, les lettres des 3 colonnes impactées vont glisser vers la base du plateau

a	b	c		e	f	g	h
e	j	s		e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	s	n	m	m
i	i	c	d	e		j	n
s	d	h	t	n		g	r
r	r	c	u			c	p
c	b	e	a	a	i	u	s

a	b	c			f	g	h
e	j	s		e	u	k	o
p	t	c	d	e	a	r	v
u	e	z	t	e	n	m	m
i	i	c	d	s		j	n
s	d	h	t	e		g	r
r	r	c	u	n		c	p
c	b	e	a	a	i	u	s

a	b	c				g	h
e	j	s		e		k	o
p	t	c	d	e		r	v
u	e	z	t	e	f	m	m
i	i	c	d	s	u	j	n
s	d	h	t	e	a	g	r
r	r	c	u	n	n	c	p
c	b	e	a	a	i	u	s

Ainsi le plateau se présente de la manière suivante pour le mot suivant :

a	b	c				g	h
e	j	s		e		k	o
p	t	c	d	e		r	v
u	e	z	t	e	f	m	m
i	i	c	d	s	u	j	n
s	d	h	t	e	a	g	r
r	r	c	u	n	n	c	p
c	b	e	a	a	i	u	s

Le jeu est terminé quand le temps alloué au jeu est terminé ou bien lorsque qu'il n'y a plus de lettres sur le plateau.

Vous déterminez enfin les scores de chacun des joueurs selon les mots qu'ils ont trouvés sachant que le score dépendra de la longueur des mots et du poids de chacune des lettres. Le poids est défini dans le fichier Lettres.csv. Privilégiez les mots longs !

Vous expliquerez votre formule.

Fichiers fournis

Lettres.txt

Ce fichier est composé de 3 colonnes :

La lettre, le nombre maximal de fois qu'elle peut apparaître sur le plateau et son poids

A,10,1 ainsi la lettre A peut apparaitre jusque 10 fois sur le plateau et possède un poids de 1

Vous pouvez modifier les valeurs proposées.

Pour générer le plateau de manière aléatoire, il faudra tenir compte de ces contraintes.

Utiliser la classe File pour la lecture de ce fichier

MotsFrancais.txt

Un dictionnaire français est à votre disposition. Chaque ligne du dictionnaire correspond aux mots commençant par une lettre de l'alphabet. Les mots ne sont pas ordonnés. Vous avez donc 26 lignes dans ce dictionnaire.

L'utilisation du Tri fusion ou Quick Sort est à privilégier

La recherche d'un mot dans le dictionnaire doit se faire par le truchement de la recherche dichotomique

Utiliser les classes : StreamReader et StreamWriter pour l'usage du dictionnaire.

Vos algorithmes seront basés sur la programmation objet et pour cela vous créerez au moins 4 classes : Joueur, Plateau, Dictionnaire et la classe Jeu. D'autres classes peuvent être utilisées

Classe Joueur (à réaliser dans un fichier Joueur.cs)

Un joueur est caractérisé par son nom, par les mots trouvés et les scores par plateau au cours de la partie

La création d'un joueur n'est possible que si celui-ci a un nom au départ du jeu. Le score et les mots trouvés sont respectivement égal à 0 et null au départ du jeu.

Vous créerez les propriétés en fonction des besoins de votre programme

Les méthodes suivantes sont imposées : (les signatures peuvent être ajustées en fonction de votre code)

`public void Add_Mot (string mot) ajoute le mot dans la liste des mots déjà trouvés par le joueur au cours de la partie`

`public string toString()qui retourne une chaîne de caractères qui décrit un joueur.`

`public void Add_Score(int val) qui ajoute une valeur au score`

`public bool Contient (string mot) teste si le mot est déjà dans la liste des mots déjà trouvés par le joueur au cours de la partie`

Classe Dictionnaire (à réaliser dans un fichier Dictionnaire.cs)

Les méthodes sont imposées : (les signatures peuvent être ajustées en fonction de votre code)

`public string toString()qui retourne une chaîne de caractères qui décrit le dictionnaire à savoir ici le nombre de mots par lettre et la langue`

`public bool RechDichoRecursif(string mot) qui teste si le mot appartient bien au dictionnaire. Vous utiliserez cette méthode pour rechercher un mot dans le dictionnaire !`

`public void Tri_XXX() pour trier le dictionnaire (quick_sort ou tri-fusion)`

Classe Plateau (à réaliser dans un fichier Plateau.cs)

Une instance de plateau est définie par une matrice

Deux cas doivent être pris en considération :

- Générée automatiquement et aléatoirement à partir du fichier Lettres.txt
- Qui s'initialise avec une instance de plateau définie à partir d'un fichier déjà existant. (exemple Test1.csv)

Vous créerez les propriétés en fonction des besoins de votre programme

Les méthodes suivantes sont imposées : (les signatures peuvent être ajustées en fonction de votre code)

`public string toString()qui retourne une chaîne de caractères qui décrit le plateau.`

`public void ToFile(string nomfile) qui sauvegarde l'instance du plateau dans un fichier en respectant la structure précisée ci-dessus`

`public void ToRead(string nomfile) qui instancie un plateau à partir d'un fichier`

`public object Recherche_Mot(string mot)` qui teste si le mot passé en paramètre est un mot éligible sur le plateau. Le type de retour proposé est le plus générique, il vous faut l'adapter en fonction de vos besoins. Une recherche récursive sera grandement appréciée.

Recherche en verticalité (vers le haut)

Recherche en horizontalité (gauche et droite)

Recherche en diagonale (gauche et droite)

`public void Maj_Plateau(object objet)` qui permet de mettre à jour la matrice en fonction du mot au préalable trouvé

Ne pas oublier que l'utilisation de

Random r = new Random()

ne peut se faire qu'une seule fois. Ensuite r.Next(..) peut se faire autant de fois que nécessaire

Classe Jeu (à réaliser dans un fichier Jeu .cs)

La classe Jeu possède entre autres les attributs suivants :

- Une structure de Dictionnaire
- Un plateau courant
- Les joueurs

Le programme principal Main va donc à tour de rôle donner la main à un joueur puis à un autre pendant un laps de temps pour la partie à définir (2mn par exemple). Ceci doit être configurable avec une valeur par défaut

Chaque joueur a X secondes maximum à définir au début du jeu par tour.

A la fin du temps imparti, le joueur suivant prend la main.

Le jeu est terminé quand le temps de la partie est terminé ou qu'il n'y a plus de lettres sur le plateau

Voir les structure et classe DateTime et TimeSpan

Compléments Interfaçage

Une interface console est obligatoire, ayez le soin de prévenir les mauvaises saisies de l'utilisateur pour éviter autant que possible les crashs

Création des joueurs et des critères de temps des tours et de la partie

Boucler sur le menu suivant :

Jouer à partir d'un fichier
Jouer à partir d'un plateau généré aléatoirement
Sortir

afin de pouvoir jouer successivement à partir de plusieurs fichiers csv représentant des plateaux différents. Ceux-ci vous seront proposés lors de la revue de code.

Une interface graphique est optionnelle est fera l'objet d'un bonus

Documents à rendre au dernier TD

Vous déposez sur DVO lors de votre dernière séance en début de séance

- Votre solution zippée

Vous déposez sur DVO lors de votre dernière séance en fin de séance

- Le diagramme de classe UML
- Et un document explicatif sur une page maximum

Vous pourrez lors de cette séance, compléter les commentaires ou votre projet Tests Unitaires auquel cas, vous ferez un nouveau dépôt