Le jeu du Boogle

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 commence avec un plateau carré composés de dés (par défaut 4X4 qui pourra être adapté en fonction de la demande du joueur). Chaque dé a 6 faces. Chaque dé possède une lettre différente sur chacune de ses faces. Les dés sont lancés sur le plateau, et seule leur face supérieure est visible. Vous traduirez le lancement de dé par un tirage aléatoire d'une face parmi les 6 d'un dé et ce pour tous les dés. Après cette opération, un compte à rebours de N minutes est lancé qui établit la durée de la partie. Le jeu s'arrête à l'issue de ce temps.

Chaque joueur joue l'un après l'autre pendant un laps de temps de 1 mn par défaut, configurable dans le jeu. Vous redéfinissez un nouveau plateau pour chacun des tours de chacun des joueurs.

Chaque joueur cherche des mots (**dans la langue choisie au préalable**) pouvant être formés à partir de lettres adjacentes du plateau. Par adjacence, il est sous-entendu horizontalement, verticalement ou en diagonale vers le haut ou le bas, la droite ou la gauche. Les mots doivent être de 2 lettres au minimum, peuvent être au singulier ou au pluriel, conjugués ou non, **mais ne doivent pas utiliser plusieurs fois le même dé pour le même mot. Par ailleurs, les mots trouvés lors d'un même tour doivent tous être différents.** Les joueurs saisissent tous les mots qu'ils ont trouvés au clavier.

Le calcul de points doit tenir compte des informations suivantes :

- Un score par joueur est mis à jour que si ce mot appartient au dictionnaire.
- Le score doit tenir compte du poids de chacune des lettres. Un z a un poids plus fort qu'un a, vous pouvez vous inspirer du Scrabble pour établir le ratio des lettres.
- Le score doit tenir compte de la taille du mot.

A l'issue du jeu vous créez le nuage de mots réalisés pour chacun des joueurs.

Démarches

Votre jeu configure tout d'abord :

- Le choix d'une langue et d'un jeu de lettres différents. En effet le poids des lettres est différent entre les langues.
- Le choix du plateau carré (4X4 par défaut)
- La définition des pseudos des joueurs (2 minimum)

Le fichier **Lettres.txt** indique pour chaque lettre sa valeur, son nombre et son poids. A partir de ce fichier, vous définissez les dés de votre plateau pour une partie.

Exemple dans le fichier **Lettres.txt** fourni la lettre W ne peut se produire qu'une seule fois dans le plateau et la pondération de cette lettre est de 10.
Si vous souhaitez modifier les valeurs de ce fichier ou ajouter d'autres critères, libre à vous.

Vous générez ensuite un plateau de dés par la création successives d'autant de dés que vous avez de cases sur votre plateau en respectant le nombre possible de chaque lettre du fichier Lettres.txt

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

Une fois le tableau prêt, le jeu peut commencer.

A chaque tour, les dés du plateau sont lancés de manière à avoir une face visible.

Le premier joueur devra saisir les mots qu'il trouve (un mot à la fois !).

A chaque saisie, votre programme **vérifie que ce mot respecte la contrainte de longueur (au moins 2 caractères de long), que le mot appartient bien au dictionnaire de mots connus et bien entendu qu'il est possible de former ce mot à partir des faces visibles du plateau**. Si tous ces tests sont valides alors le mot sera ajouté à la liste de mots trouvés par le joueur et le score du joueur sera crédité des points correspondants. C'est alors au tour du joueur suivant de jouer.

Voici un extrait du dialogue du jeu dans une interface minimale

```
C'est au tour de DURAND de jouer

O W S I
N V B E
K S E I
A J Y I

Saisissez un nouveau mot trouvé
JASE
 Le score de DURAND est de 2 grâce aux mots cités suivants
JASE
Saisissez un nouveau mot trouvé

C'est au tour de DUPOND de jouer

N N E E
B A I R
H I E A
M L S E

Saisissez un nouveau mot trouvé
RIEN
 Le score de DUPOND est de 2 grâce aux mots cités suivants
RIEN
Saisissez un nouveau mot trouvé
LIEE
 Le score de DUPOND est de 4 grâce aux mots cités suivants
RIEN LIEE
Saisissez un nouveau mot trouvé
BANIE
Saisissez un nouveau mot trouvé
AIR
 Le score de DUPOND est de 5 grâce aux mots cités suivants
RIEN LIEE AIR
Saisissez un nouveau mot trouvé
RASE
 Le score de DUPOND est de 7 grâce aux mots cités suivants
RIEN LIEE AIR RASE
Saisissez un nouveau mot trouvé
NEE
 Le score de DUPOND est de 8 grâce aux mots cités suivants
RIEN LIEE AIR RASE NEE
C'est au tour de DURAND de jouer

W R A E
Z S N J
A S R A
R G O L

Saisissez un nouveau mot trouvé
```

Avec le tirage au sort des faces du dé, Dupond a joué, sans inspiration, un seul mot a été trouvé.

Durand a un tirage plus favorable et trouve 5 mots validés (BANIE n'est pas validé car n'appartient pas au dictionnaire dont un extrait est présenté ci-dessous).

AFFECTIONNEE AFFECTIVITE APPRENDRONS ANTIVOLS AVALERIEZ APPUYAIENT AFFECTIEZ ANTINUCLEAIRE ACCAPARAIENT ABOMINABLES ADMINISTRERAS ABOLIRAIS AFFRONTERAIENT ACCABLEZ ACCUMULERAIS AIGRIRA AVALISEE ACCABLENT ACCOMMODERAIS AVALISERENT APPROPRIERENT AUTOBIOGRAPHIE ADIEUX AMENAGEANT AUDIBLE ALLEZ ASSASSINERA ACCULEREZ AEROGARES AGREGAT ASSIMILIEZ ATTIGER AFFIRMERENT ATTRISTERONT ADJURATION ATOMIQUE ALARMONS ARGILEUSES AGREGERONS ASSIMILERAIENT AUTOCOLLANTE ANNOTATIONS AGGLUTINERENT AMADOUAIT ACQUIESCE ALLONGIEZ ACQUERRIEZ AIREZ ABORDERAIENT AY APPROVISIONNE AFFAIBLISSENT ARC ASSAILLIRAIENT AGGRAVEZ AMEUTERAIS AORTE ADDITION ASTUCIEUSE APPROPRIEREZ ABDICATION APPLAUDIREZ ACTUALISEES ACERBE AMINCIRONS AVENTURERAI APPRECIERAIENT AGREMENTERAS AISES AMENAGERENT ACCEDENT ACTIONNEES AMPERES ANEANTIREZ AFFLIGEE AGENOUILLERIONS AIGUISERAIS APPESANTIES AMELIORERAS ACHETENT ABRUTISSANTES AGONISERAIENT AGREAIT ADMINISTRATRICE AFFIRMEE ACCOSTAIS

Le fichier de mots **MOTS_FRANCAIS.txt** devra au préalable avoir été traité et restructuré de manière à éviter d'explorer tout le fichier pour valider chaque mot trouvé. Vos stratégies de structuration et de recherche doivent être étudiées de telle manière à avoir un coût minimal dans le pire des cas **Proposez plusieurs solutions (3 minimum) et validez votre choix final avec des critères rationnels.**

Pour cela vous pouvez utiliser la classe StopWatch depuis System.Diagnostics

Vos algorithmes seront basés sur la programmation objet et pour cela vous créerez au moins 4 classes : Joueur, De, Plateau, Dictionnaire et la classe Program, (vous pourrez la renommer Jeu) modélisera le jeu.

Il vous est demandé de modéliser ce cahier des charges sous forme de diagramme de classes UML en tenant compte des classes imposées.

Exercice 1 : Classe Joueur (à réaliser dans un fichier Joueur.cs)

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

La création d'un joueur n'est possible que si celui-ci a un nom.

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 bool Contain(string mot) qui teste si le mot passé appartient déjà aux mots trouvés par le joueur pendant la partie`

`public void Add_Mot (string mot) ajoute le mot dans la liste des mots déjà trouvés par le joueur au cours de la partie en modifiant le nombre d'occurrences si nécessaire`

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

Exercice 2 : Classe Dé (à réaliser dans un fichier De.cs)

Un dé est caractérisé par un ensemble de lettres et une lettre tirée au hasard parmi ces 6 lettres pour représenter la face visible du plateau

La création d'un dé doit vérifier que toutes les faces ont bien une lettre attribuée à partir du contenu du fichier Lettres.txt.

Vous créerez les constructeurs et 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 Lance( Random r) : cette méthode permet de tirer au hasard une lettre parmi les 6. Cette lettre devient la lettre visible.`

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

Exercice 3 : Classe Dictionnaire (à réaliser dans un fichier Dictionnaire.cs)

Vous devez partir impérativement des fichiers fournis Mots_PossiblesFR.txt et Mots_PossiblesEN.txt qui représentent un sous-ensemble des mots possibles mélangés dans la langue française ou anglaise.

A partir de ces fichiers, vous associez un ensemble de mots structuré ainsi qu'une langue pour chaque instance de dictionnaire.

Les méthodes suivantes sont imposées :

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

`public bool RechDichoRecursif(string mot, … ) qui teste que le mot appartient bien au tableau de mots de ce dictionnaire, cette méthode est obligatoirement récursive comme son nom l'indique`

Exercice 4 : Classe Plateau (à réaliser dans un fichier Plateau.cs)

Une instance de plateau est définie par les 16 dés (tableau de dés par défaut) et leur valeur supérieure

Vous créerez les propriétés en fonction des besoins de votre programme

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 un plateau.`

`public bool Test_Plateau(string mot) qui teste si le mot passé en paramètre est un mot éligible c'est-à-dire qu'il respecte les contraintes d'adjacence décrites ci-dessus et que ce mot appartient au dictionnaire.`

`Réfléchissez à un algorithme récursif pour résoudre cette méthode.`

Exercice 5 : Classe Jeu (à réaliser dans un fichier Jeu .cs)

Le programme principal Main va donc à tour de rôle donner la main à un joueur puis à un autre (dans le cas par défaut 2 joueurs) pour un laps de temps à définir (6mn par exemple)

Chaque joueur a une minute (voir la classe DateTime et TimeSpan) pour trouver les mots sur une nouvelle instance de plateau.

Le joueur saisit un mot après l'autre. Après chaque saisie, vous testez si ce mot est éligible.

A la fin du temps imparti, l'autre joueur joue.

Au bout d'un laps de temps prévu au départ du jeu (6mn par exemple), vous affichez les scores des joueurs et indiquez le gagnant.

Exercice 6 : Nuage de mots avec un outil d'IA génératif.

Exemple de nuage de mots

Créer à la fin de votre partie un nuage de mots par joueur. Pour cela, vous devrez explorer des bibliothèques C# graphiques (de type Drawing ou autre).

Vous êtes autorisés à utiliser un outil d'IA générative pour cette occasion.

- Analyser le rendu si votre prompt est en français ou en anglais
- Transcrire les prompts successifs que vous avez utilisés pour améliorer au fil de l'eau le code généré.
- Soyez critique sur votre démarche et les différents allers-retours que vous avez utilisés avec l'outil

Vous ne manquerez pas de comprendre complètement le code fourni et d'avoir un regard critique sur le code généré pour enfin l'améliorer par vous-même.

BONUS

Le joueur est une 'IA' : Vous devez donc explorer toutes les positions du plateau à la recherche des mots qui sont dans le dictionnaire.

Il est important de limiter la recherche au maximum afin de s'assurer que la tâche sera achevée dans un temps raisonnable.

L'une des stratégies les plus importantes consiste à déterminer lorsque la recherche s'engage sur une voie morte pour l'abandonner au plus vite. Par exemple, si la recherche a construit un chemin menant au préfixe 'zx', vous pouvez utiliser votre dictionnaire pour déterminer qu'il n'y a pas de mot qui commence par ce préfixe. Et donc, vous pouvez arrêter cette recherche pour continuer sur une voie meilleure. Si vous n'appliquez pas cette optimisation, votre ordinateur passera un temps non négligeable à vérifier la construction de mots qui n'existent pas tel que 'zxgub'.

Livrables

Ce travail est à réaliser en binôme du même groupe dans le cas d'un groupe d'étudiants impairs un seul trio est autorisé par classe et le BONUS devient alors obligatoire.

Votre solution C# est à déposer au début de votre dernière séance de TD, et uniquement celle-ci. Toute remise tardive sera sanctionnée d'un malus. Vous déposez dans la zone prévue à cet effet votre solution zippée à vos 2 noms. Seul un des 2 du groupe dépose la solution.

Cette dernière séance fera l'objet d'une revue de code pendant laquelle vous pourrez ajouter les commentaires, finaliser votre rapport, et votre projet Tests Unitaires. La solution complétée devra alors être déposée dans un 2ème dépôt prévu à cet effet toujours sous la forme d'une archive .zip.

Cette dernière solution doit comporter :

- un diagramme de classes UML qui doit être compatible avec votre solution
- un rapport qui explique
    - vos diverses tentatives pour améliorer le temps de recherche d'un mot dans le dictionnaire (au moins 2)
    - votre exploration et votre esprit critique des résultats fournis par ChatGPT sur le nuage de mots
- votre solution complétée par des commentaires /// et votre Projet Tests Unitaires

Vous déposerez votre solution Visual Studio sur DVL au format .zip sous vos noms séparés par un _

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 à savoir ceux qui donnent le code et ceux qui le reçoivent**)

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.

Lors de la revue de code, vous devez être en mesure de décrire n'importe quelle partie de code (y compris une portion écrite par votre binôme). Vous préciserez lors de cette revue la part de chacun dans la réalisation de ce projet en pourcentage.

Je précise que la note du problème ne sera définitive qu'à l'issue de l'examen. En effet, tout écart trop important entre la note du problème et celle de l'examen donnera lieu à une explication en face à face.