Projet algorithmique et POO – Mots-glissés *DOCUMENT EXPLICATIF*

Réalisation du projet entièrement en langage C# et en utilisant les principes de la programmation orientée objet.

Explications brèves du fonctionnement du programme & structures de données utilisées

Dictionnaire : Nous avons au départ décidé de stocker le dictionnaire dans un tableau de chaines de caractères, même si nous nous sommes rendu compte durant le dernier TD que cela aurait été plus simple en utilisant une sortedList. Une autre solution est d'ailleurs annexée dans le ZIP. Celle-ci contient le traitement du dictionnaire comme une sortedList et les constats faits sur son efficacité.

Par ailleurs, le dictionnaire est lu à partir du fichier « Mots_français.txt » à l'aide d'un StreamReader puis est stocké dans ce tableau de chaine de caractère.

- ➔ Utilisation du Tri Rapide, son fonctionnement est expliqué dans un commentaire /// dans la solution. Nous avons également mis la source utilisée pour comprendre le principe de pivot et de partitionnement.
- ➔ Une fois le tableau trié, on réalise une recherche dichotomique classique pour rechercher le mot

Joueur : la classe est faite telle que décrite dans l'énoncé du projet

Ajout d'une classe Lettre : Nous avons décidé d'ajouter cette classe pour simplifier l'identification des lettres sur le plateau ainsi que le poids et pour se souvenir des lettres qu'il faut faire disparaitre du tableau. Remarque : si nous avions utilisé une SortedList, peut-être qu'ajouter cette classe n'aurait pas été nécessaire.

Plateau : Définit comme dans l'énoncé

- ➔ Fonction ToFile : sauvegarde l'instance actuelle du plateau dans un fichier .txt (ou csv) à l'aide d'un StreamWriter
- ➔ Fonction ToRead : lit une instance de plateau de jeu à partir d'un fichier .txt (ou csv) à l'aide d'un StreamReader.
- ➔ Fonction Recherche_mot : Fonction récursive qui vérifie d'abord si la première lettre du mot entré est présente dans la base du plateau (dernière ligne de la matrice). Puis cette fonction traite tous les cas (diagonales, horizontal et vertical (vers le haut)) et fait un appel récursif à chaque fois. La fonction s'arrête quand le mot a été trouvé ou qu'une des lettres est manquante.
- ➔ Les indices des lettres trouvées sont sauvegardés et sont remplacés par un « » si et seulement si tous les mots ont été trouvés. Ceci se fait grâce à l'appel de la fonction StateIndicesCrush() qui n'est appelée que si le mot est trouvé entièrement sur le plateau sans erreur.

Jeu : Définie comme dans l'énoncé. Cette classe contient la fonction de jeu principale. Il s'agit d'une boucle de jeu classique à laquelle on a ajouté des instances des classes DateTime et TimeSpan pour gérer le temps dans le jeu. Les joueurs choisissent la durée d'une partie et du tour de chacun à partir de valeur prédéfinies.

Documentation & sources

Tri Rapide | classe DateTime | TimeSpan | Fichiers | StreamReader & StreamWriter

Manipulation de la console (interfaçage) |