TP Anagrammes

Dans ce TP, vous découvrez d’autres contrôles graphiques, vous apprenez à chercher dans la documentation des informations sur le fonctionnement de Windows Forms et à composer une application plus riche en implémentant un jeu d’anagrammes.

A la fin des deux séances (un total de 4 heures de travail), vous devrez déposer **une archive compressée (zip, rar, tar etc…)** contenant ce compte-rendu rempli et l’ensemble du code source de votre application dans l’espace prévu sur Moodle.

ATTENTION : N’oublie pas de respecter les conventions de nommage des contrôles que tu ajoutes dans ton application (propriété « name ») et de rendre un code propre et commenté si nécessaire.

# Présentation de l'application à réaliser

On souhaite réaliser une application permettant de trouver un mot proposé dans le désordre pour lequel le joueur a droit à un nombre maximum de tentatives.

Voici un storyboard de l'interface utilisateur :

Ainsi que les tables fonctionnelles potentielles de cette application :

|  |  |  |  |
| --- | --- | --- | --- |
| **Objets** | **Propriétés** | **Représentations** | **Opérations** |
| Proposition | * chaîne de caractères | Un champ de texte (textbox) permettant de modifier le mot à proposer | saisir  tester |
| Partie | * numéro * Etat (gagnée/perdue/en cours) * nombre d’essais restants * total d’essais | Partie en cours : gauche de l’interface avec label représentant le mot à deviner en haut et plus bas, un groupe de contrôle montrant le nombre de propositions restantes et une liste montrant les propositions qui ont déjà été faîtes.  Partie précédente : partie droite de l’application, une partie est représenté par un élément d’une liste déroulante avec ses détails qui s’affiche en dessous si elle est sélectionnée | commencer |

Opérations pour l’objet **Proposition**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opérations** | **Commandes** | **Retours** | **Réponses** |
| Saisir | * (1) zone de saisie au clavier | * (2) Le texte saisi apparait dans le champ de texte Proposition et est automatiquement mis en majuscule | * (3) Le texte saisi apparait dans le champ Proposition et est automatiquement mis en majuscule |
| Tester | * (1) Click sur bouton "Tester" | * (2) Il y a un retour sur le bouton | * (3)Mise à jour des affichages selon état partie (infos partie, historique, saisie mot, mot à trouver) * (4 si fin de partie)boîte de dialogue perdu ou gagné. |

Opérations pour l’objet **partie**

|  |  |  |  |
| --- | --- | --- | --- |
| **Opérations** | **Commandes** | **Retours** | **Réponses** |
| Commencer | * (1) Click sur bouton "Rejouer" * (3) Confirmation annulation de partie en cours dans la boîte de dialogue | * (2) retour bouton puis boîte de dialogue qui s’ouvre * (4) La boîte de dialogue se referme | * (5) partie en cours perdue * (6) initialisation nouvelle partie * (7) mise à jour affichages (infos partie, historique, saisie mot, mot à trouver). |

# Utilisation de la documentation Visual Studio

Il t’est demandé tout d’abord de rechercher un certain nombre d'informations à l’aide de la documentation de Visual Studio, du C# et de Windows Forms (disponible en ligne). Tu peux tester le code dans l’application également, fais des essais !

1. Donne une description de la classe **Random** et la méthode **Next()**. Ecris le code permettant d'obtenir un nombre au hasard compris entre deux entiers positifs x et y, bornes comprises (attention à cette consigne)!.

**La classe random eprésente un générateur de nombres pseudo-aléatoires. Il s’agit d’un algorithme qui produit une séquence de nombres conformes à certains prérequis statistiques liés à l’aspect aléatoire.**

**Next() retourne un nombre aléatoire entier non négatif.**

**Int res = Next(x,y+1) ;**

1. Donne le nom et un exemple d'utilisation de la méthode ou la propriété permettant de :

* insérer une chaîne de caractères dans une une chaine de caractères à une position donnée

**.Insert**

**String a = "ab" ;**

**a.Insert(1,"Owo") ;**

* convertir une chaîne de caractères en majuscules (puis minuscules)

**.ToUpper() ;**

**.ToLower() ;**

* obtenir la longueur d'une chaîne de caractères et la taille d'un tableau

**String a = "aaa" ;**

**a.Length ;**

**Idem pour un tableau.**

1. Recherche d'informations sur les contrôles TextBox et ListBox:

* Quelle **propriété** du contrôle TextBox permet de convertir les caractères tapés dans la TextBox en majuscules (ou minuscules) ? Quelles sont les valeurs possibles ?

**textBox1.CharacterCasing = CharacterCasing.Upper**

**textBox1.CharacterCasing = CharacterCasing.Lower**

* Comment ajouter des lignes à un contrôle ListBox (mets un exemple) ?

**ListBox.add("UWU Je joue Yummi Jungle");**

* Comment vider une ListBox (mets un exemple) ?

**ListBox.Clear();**

# Programmation de l’application et de l’interaction

Quelques consignes :

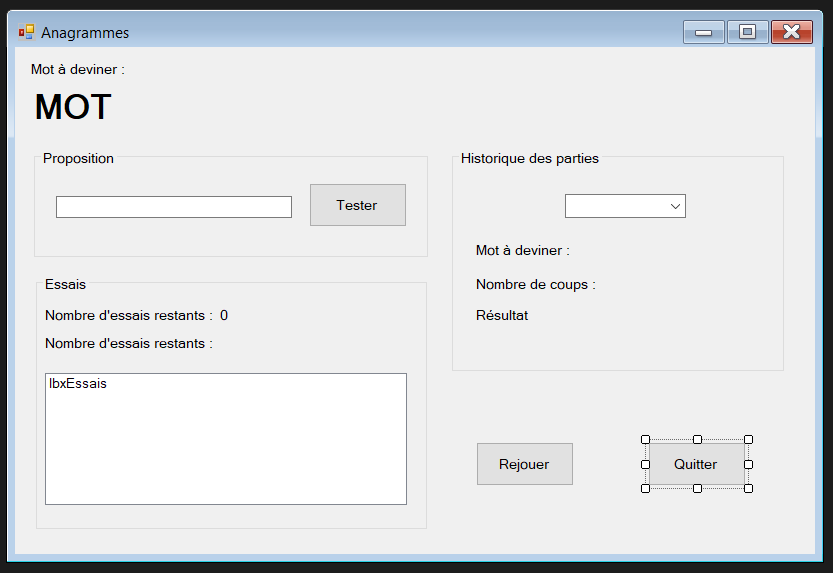
Les mots à trouver seront stockés en majuscules dans un tableau de chaînes de caractères :

private String[] tabMots

Chaque mot sera ensuite "mélangé" avant d'être affiché dans le Label adéquat**.** Les mots saisis par le joueur seront convertis en majuscules, immédiatement lors de la saisie, pour éviter les problèmes de casse. L'utilisateur aura droit à un nombre maximum d'essais.

La partie A et la partie B du travail concerne la partie gauche de l’application, le cœur du jeu, alors que la partie C concerne la partie droite, l’historique des parties.

Tu peux d’abord placer tous les contrôles dans l’interface à l’aide de la vue Designer avant de commencer à coder :



**Partie A**

Dans un premier temps il t’est demandé de définir et d’écrire dans ton code les 5 procédures et fonctions mentionnées plus loin, qui seront nécessaires pour ton application. Attention, ces procédures ne sont pas les procédures événementielles, mais elles seront appelées par les procédures événementielles.

Tu peux les écrire au même endroit où s’écrivent les procédures événementielles (dans le fichier .cs de ta Form), voici un exemple pour la procédure initialisation() :

Une image contenant texte

Description générée automatiquement

*J’en profite pour te dire que lorsque qu’une fenêtre (une Form) est chargée pour la première fois, celle-ci lance un événement « Load ». Une procédure événementielle qui réagit à cet événement est donc l’endroit parfait pour écrire du code qui doit être lancé une seule fois au démarrage de l’application. Dans la vue Designer, si tu double-cliques sur la Form, cela va te créer automatiquement une procédure événementielle qui réagit à cet événement :*

Une image contenant texte

Description générée automatiquement

Teste chacune des procédures/fonctions que tu écris avant de passer à la suivante pour être sûr (pour ceci, tu peux ajouter un bouton temporaire Test dont le seul but de sa procédure événementielle est d'appeler la procédure à tester ou alors tu peux les appeler dans la procédure événementielle qui réagit au « Load »).

Ne donne pas le code de ces fonctions dans ce compte-rendu, mais précise juste pour chacune, quelle procédure évènementielle provoquera l’exécution (l’appel) de cette dernière, directement ou indirectement (remarque : ces procédures événementielles seront écrites dans un deuxième temps).

*N’oublie pas de commenter ton code.*

1. Écrire la procédure private void initialisation() qui réalisera les initialisations à effectuer au lancement de l'application. Pour le moment, ceci se limite à l’initialisation du tableau de mots. Attention : cette procédure ne sera appelée qu'une seule fois et ne doit pas être mélangée avec la procédure nouvellePartie() qui elle sera appelée plusieurs fois (voir plus loin).

Bien réfléchir à la déclaration de vos variables : si elles seront utilisées dans plusieurs procédures, alors les déclarer globalement, en attributs de classe, sinon, les déclarer localement à la procédure concernée.

Allez, je suis sympa, je te montre à quoi ça pourrait ressembler pour cette première procédure (regarde également où est déclaré le tableau de mots :

Une image contenant texte

Description générée automatiquement

Maintenant, tu peux me dire par qui sera appelée cette procédure :

Appelée par : (je t’ai donné la réponse plus haut dans le sujet)

frmPrincipale

1. Écrire une fonction private String melanger(String chaine) qui, à partir d'un paramètre chaîne en entrée, renvoie une nouvelle chaîne résultant d'un mélange aléatoire de ses lettres. Pour réaliser ce mélange, l'algorithme suivant vous est proposé : prendre chaque caractère de la chaîne initiale et l'insérer dans une nouvelle chaîne, initialement vide, à une position aléatoire.

Appelée par :

nouvellePartie

1. Écrire une procédure private void nouvellePartie() qui réalisera les initialisations à effectuer à chaque nouvelle partie, c'est-à-dire, pour chaque nouveau mot à trouver. Pour le moment, ceci se limite à effectuer le tirage aléatoire d'un mot dans le tableau tabMots, le mélange de ses lettres (grâce à la fonction melanger) et son affichage dans l’interface utilisateur.

Appelée par :

Rejouer

Initialisation

1. Écrire la procédure motCorrect() qui réalisera les tâches nécessaires si le joueur a gagné. Pour le moment, ceci se limite à lui demander s'il veut rejouer.

Appelée par :

Tester

1. Écrire la procédure motIncorrect() qui réalisera les tâches nécessaires si le mot proposé par le joueur n'est pas le bon. S'il ne lui reste plus d'essais, alors il a perdu : on lui affiche le mot qui était à trouver et on lui demande s'il veut rejouer. S'il reste des essais, alors il ne se passe rien et le jeu continue (le joueur peut proposer un nouveau mot).

Appelée par :

Tester

**Partie B**

Ecrire maintenant les procédures événementielles de ton application afin d’appeler au bon moment ces 5 procédures/fonctions précédentes. Il se peut que tu aies besoin de compléter les procédures précédentes selon ta façon de faire.

**Partie C : Fin de partie, historique des parties et fin de jeu**

Complète les différentes procédures et fonctions afin de pouvoir afficher, dans la partie droite de l’application, pour chaque partie jouée, si le joueur a gagné (en combien de coups) ou perdu et quel était le mot à trouver. Il te faudra utiliser le contrôle ComboBox et son événement qui est déclenché lorsque qu’une nouvelle valeur est sélectionnée dans la liste. Tu auras peut-être également besoin de déclarer des tableaux supplémentaires pour garder en mémoire la liste des parties jouées et leurs propriétés (Une alternative serait même de créer une Classe « Partie » afin de représenter cet objet conceptuel et ses propriétés).

# Extensions et améliorations (facultatif mais apprécié)

Vous pourrez adapter et améliorer l’interface proposée pour la rendre plus attrayante (couleurs, disposition des contrôles, animations etc.) mais aussi y apporter plus de fonctionnalités (sans pour autant en réduire l’utilisabilité !!!). Par exemple, vous pourrez empêcher l’utilisateur de cliquer sur le bouton Tester s’il n’a pas saisi de mot, vider la Textbox après chaque essai, permettre d’initialiser le tableau de mots avec des mots saisis par l’utilisateur, permettre de changer le nombre de tentatives, proposer un score et un affichage plus avancés, etc.

