420-P45-BB
Travail pratique # 1
Professeur : Mathieu Grenon

The Sherlock Scrolls :

Londres

Objectifs du travail

- Concevoir une interface graphique adaptée
- Utiliser les contrôles de base, leurs propriétés, méthodes et événements
- Créer des menus simples

Dates de remise

Le lundi 18 février 2019 à 13h : Analyse

- Page titre complète
- Description de la tâche à effectuer (une demi-page bien remplie)
- Imprime-écran et tableau descriptif pour chacun des écrans de jeu (tel que présenté dans le document d'analyse)

Le lundi 18 mars 2019 à minuit : remise finale (sur le R:\ ou Léa)

- Une auto-évaluation de votre programme (ce qui fonctionne très bien, moins bien, pas du tout et pourquoi), en plus d'indiquer les éléments bonis que vous avez fait, s'il y a lieu.
- Les jeux d'essais utilisés pour tester : liste des jeux d'essais et explication des résultats
- Projet complet remis sur Léa

Consignes

La compagnie **Resthela**, qui se spécialise dans le développement de jeux vidéo, vous charge de recréer un jeu informatique qui sera populaire auprès des jeunes. Vous êtes le dernier espoir de cette compagnie qui a perdu toute crédibilité ces dernières années après avoir sorti tour à tour les fiascos « Fallout : 76 façons de gaspiller mon argent », et « Doom 2019 : version SFW ». Il faudra intéresser les jeunes en combinant ce qui est le plus « hot » et « chill » pour eux ces temps-ci (selon le département de marketing de Resthela), soit **les voleurs sympathiques mais mystérieux** et **les jeux de société**. Pour sauver du temps, vous décidez donc de reprendre un jeu existant qui a fait ses preuves : Scotland Yard.

Règles du jeu

Voir le document PDF des règles originales sur le R:\

Modifications des règles originales:

- Il n'y a qu'un seul joueur, qui contrôle 3 détectives [1].
- Il n'y a plus de carte « Coup double » pour le voleur… seules les cartes « Black Ticket ».
- Les positions initiales sont déterminées aléatoirement par le programme.

Spécifications :

On vous fourni deux parties de code à utiliser dans votre programme :

1- L'écran de jeu minimal, avec boutons positionnés adéquatement sur le plateau de jeu. De plus, vous avez le graphe programmé des liens entre chacune des cases, selon le moyen de transport.

2- L'intelligence artificielle du voleur en DLL (`ScotAI.dll`), à intégrer à votre travail. Vous n'avez donc pas à programmer cette intelligence artificielle, mais libre à vous de le faire si vous le désirez.

Note : Voir l'annexe pour voir comment intégrer la librairie (DLL) à votre projet et autres précisions sur celle-ci.

[1] Les méthodes fournies permettent d'avoir un nombre variable de détectives, mais on vous demande d'en programmer 3 ici.

Interface graphique

Vous devrez concevoir une interface graphique ergonomique qui répondra *minimalement* aux exigences suivantes :

Un menu dans la fenêtre de jeu:

1. Le menu Fichier contient trois entrées :
    - Nouvelle partie (CTRL-N)
    - Options (Un second écran indépendant) (CTRL-O)
    - ----------
    - Quitter (CTRL-Q)
2. Le menu « ? » contient deux entrées :
    - Aide, qui affiche les règles de jeu (vous pouvez afficher le PDF directement)
    - À propos, qui affiche votre nom et la date de remise du travail

Fenêtre Options

Le menu « Options » fait apparaître une boîte de dialogue d'options, qui permet de modifier le comportement du jeu. On peut grâce aux options :

1. Spécifier le nom des 3 joueurs.
2. Changer la couleur de chacun des joueurs.

Note : Modifier le nom ou la couleur des détectives ne redémarre pas une nouvelle partie, mais les éléments sont mis à jour dans l'interface de jeu immédiatement.

Spécifications

Vous devrez concevoir une interface graphique ergonomique de votre moi intérieur qui répondra aux exigences demandées par le jeu et l'énoncé :

- Un menu qui offre les fonctionnalités attendues d'une application Windows standard.
- Une représentation graphique du tableau de jeu. Vous pouvez modifier la représentation graphique du jeu afin de respecter les règles ergonomiques apprises dans le cours **(ex : Ne pas surcharger l'écran d'information!).**
- Tout élément jugé pertinent pour assurer le suivi du jeu par les joueurs

Attention

L'important n'est pas d'être très fidèle visuellement au jeu original, mais bien d'adapter (et améliorer) le mieux possible le jeu à l'ordinateur et aux contraintes informatique (qui ne sont pas les mêmes qu'un jeu de table).

Soyez critique envers votre propre travail : Après chaque décision prise, demandez-vous s'il est possible de faire mieux : moins de cliques, interface épurée ou organisée autrement, automatisation d'opérations de jeu etc.

Et n'oubliez pas de bien lire les instructions, ce sont vos exigences du client. En cas d'ambiguïté ou d'incompréhension d'une règle, vérifiez avec le professeur. Vous pourrez être pénalisé si une règle de jeu n'est pas suivie.

Autres précisions :

Votre programme ***doit*** respecter les normes de programmation de Microsoft.
Vous devez utiliser Microsoft Visual Studio 2017 pour écrire le code (C#).
Vous devez réaliser ce travail seul.

Un maximum de 10% pourra être soustrait pour le respect du français dans l'interface et les documents d'accompagnement.

Un maximum de 10% (indépendant du français) pourra être soustrait pour le respect des normes de syntaxe en C#, incluant la présence de commentaires sur les méthodes et classes.

Critères d'évaluation	Points
Analyse	**/4**
Document d'accompagnement du programme Jeux d'essais Auto-évaluation	**/3**
Interface Esthétique Facilité d'utilisation (Qualité du français)	**/8**
Fonctionnement[2] Exactitude des résultats Validation des entrées Respect des consignes	**/15**
Programmation Qualité de la codification Utilisation des paradigmes orienté-objet Séparation vue et modèle	**/5**
Total	**/35**

Boni STEAM :

La personne ayant l'interface la plus intéressante, fonctionnelle et professionnelle (selon moi) se méritera un jeu Steam parmi plusieurs (mauvais) choix.

[2] Le fonctionnement est évalué négativement et de manière significative : par exemple, un programme qui n'a que 50% des fonctionnalités présentes aura une note BIEN inférieure à 50% pour le critère fonctionnement.

Annexe- Utiliser une référence (DLL) à son projet

1- Lier la référence à son projet

a) Copier le fichier DLL à la racine de votre projet

b) Dans Visual Studio : Projet → Ajouter une référence → Onglet « Parcourir » : Sélectionner le fichier ScotAI.dll

2- Utiliser les méthodes et propriétés publiques de la DLL

a) Dans la classe désirée, ajouter l'importation dans le haut de la classe :

```
using ScotAI;
```

b) Appeler les méthodes/propriétés en passant par la référence :

```
/// <summary>
/// Méthode qui détermine le prochain mouvement du voleur en fonction de l'état
      actuel du jeu
/// </summary>
/// <param name="repetable">Si les mouvements du voleur sont peuvent être aléatoires (donc
répétables) ou non</param>
/// <param name="caseVoleur">La case où se trouve le voleur</param>
/// <param name="nbTaxi">Quantité de déplacements de taxis du voleur</param>
/// <param name="nbBus">Quantité de déplacements de bus du voleur</param>
/// <param name="nbMetro">Quantité de déplacements de métros du voleur</param>
/// <param name="nbBlackTicket">Quantité de déplacements de black tickets du
            voleur</param>
/// <param name="transportChoisi">Paramètre "out" (doit être passé null) -->
            Contiendra le type de "Transport" choisi par le
      voleur au retour</param>
/// <param name="utiliseBlackTicket">Paramètre "out" (doit être passé null) -->
            Indiquera si le joueur utilise un black
      ticket</param>
/// <param name="casesDetectives">Passer un a un chaque position des
            joueurs</param>
/// <returns>La position où le voleur choisi de se déplacer.</returns>
public static int ProchaineCaseVoleur(bool repetable, int caseVoleur, int nbTaxi,
            int nbBus, int nbMetro, int nbBlackTicket,
            out Transports? transportChoisi, out bool? utiliseBlackTicket,
            params int[] casesDetectives)
```

Exemple : Appel de la méthode ProchaineCaseVoleur :

```
Transports? transportChoisi = null;
bool? utiliseFrime = null;

int noCase = ScotAI.Case.ProchaineCaseVoleur(true, 132, 10, 8, 5,
   Voleur.NbFrime, out transportChoisi, out utiliseFrime, 44, 122, 47);
```

Note :

Utilisez l'enum `Transports` défini dans la DLL.