Cégep de Saint-Hyacinthe
Département d'informatique

Programmation orientée objet (420-2DP-HY)

TP2 – Branches, bibliothèque et WPF

Préparé par
Martin Lalancette
bureau B-2230

DESCRIPTION

But :	Conception d'une bibliothèque, d'une application multiformulaires de type WPF + *Data Binding*.
Objectifs :	• Créer une bibliothèque • Créer une application de type formulaire (WPF) • Gestion d'une solution multiprojets • Applications multiformulaires (modal) • Apprentissage de nouvelles composantes visuelles (TextBox, Button, CheckBox, ComboBox, Label, etc.) • **Isoler la logique du jeu de l'affichage (Encapsulation)** • Notions de « Binding » et « DataContext » • Respecter les normes de programmation
Durée :	6 h
Pondération :	10 pts
Remise :	À la fin de la semaine 9.
Contenu général :	• Remettre vos documents d'analyse et votre solution contenant le projet dans un document .ZIP via LÉA.
Notes	• Conserver une copie de sécurité. Il est de **votre responsabilité** de conserver une copie de sécurité dans l'éventualité où la lecture des données serait impossible. Cette copie doit **être disponible sur demande**.

Spécifications du travail

Ce travail pratique est à faire en **équipe de deux personnes obligatoires**. Il est la suite du travail effectué pour concevoir le TP1. Il touche deux nouvelles technologies soit les **bibliothèques de classes** ainsi que des **projets de type WPF**.

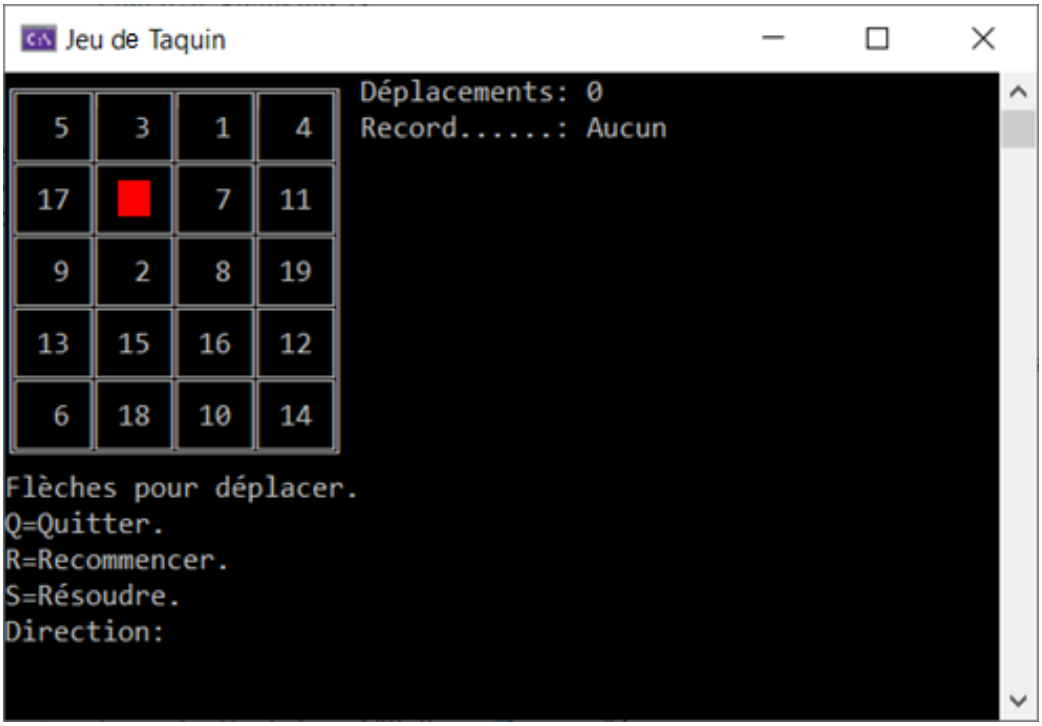
Vous devez modifier la solution qui contiendra trois projets : **Bibliothèque de classes** (.NET Core), **Application console** (.NET Core) et **Application WPF** (.NET Core). Pour réaliser une partie de ce TP, vous aurez besoin de partir du TP1 et le restructurer.

Partie #1 : Nouvelle branche GIT et bibliothèque de classes (.NET Core) – 20 %

Cette partie consiste à créer une nouvelle branche (GIT) et une bibliothèque de classes afin d'y déposer le fichier **JeuTaquin.cs**.

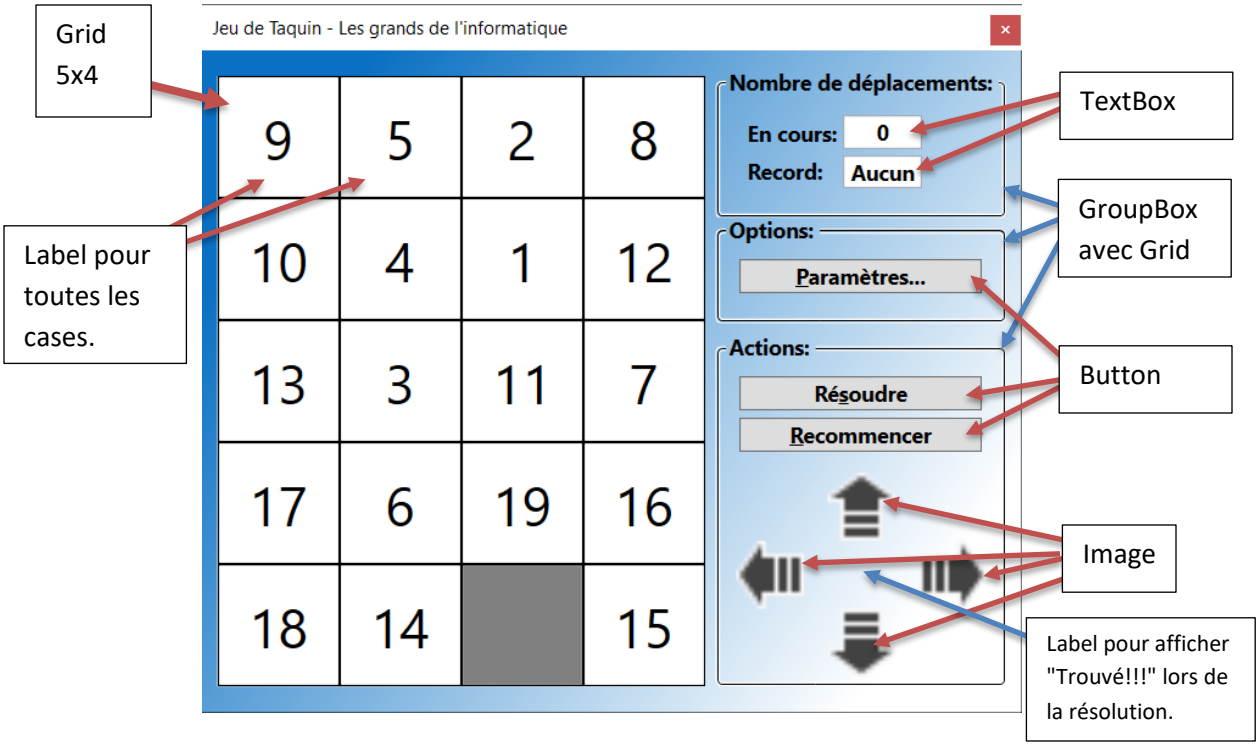
Consignes :

- Étiqueter la branche *master* (du TP1) sous le nom de **Version 1.0**.
- À partir de cette étiquette, créer une nouvelle branche nommée **Version2**. Pousser cette nouvelle branche pour que votre coéquipier puisse la récupérer de son côté.
- À la solution actuelle, ajouter un projet de type **Bibliothèque de classes (.NET Core)** nommé **JeuxLib**. Déplacer le fichier **JeuTaquin.cs** dans cette nouvelle bibliothèque.
- Modifier le projet **JeuTaquinConsole** afin d'inclure cette nouvelle bibliothèque et s'assurer que le jeu fonctionne adéquatement.



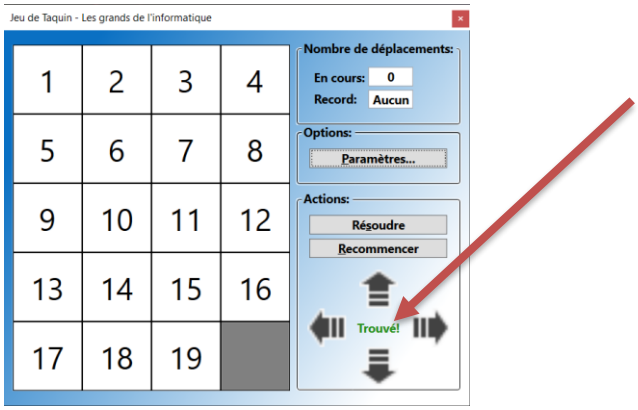
Partie #2 : Projet WPF Application (.NET Core) – Jeu de Taquin de type formulaire – 40 %

Cette partie consiste à concevoir une interface visuelle de type formulaire du jeu de Taquin. La logique de ce jeu ne devrait pas être refaite ici, car elle est centralisée dans la bibliothèque précédemment (Partie #2) créée. Voici l'interface à produire et les consignes :



Lien vidéo : https://web.microsoftstream.com/video/0fef4ccf-813c-4fcc-bc24-0c47534dc9b5

Lorsque la grille est résolue:



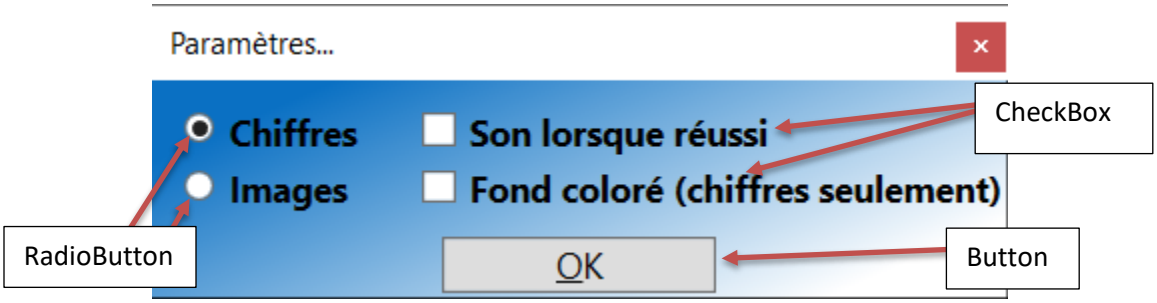
Consignes :

- À la présente solution, ajouter un nouveau projet de type **WPF App (.NET Core)** nommé **JeuTaquinWPF**.
- Structurer selon MVVM.
- Les textes sont 14px de grandeur de police et en gras.
- Ajouter une référence à la bibliothèque **JeuxLib.**
- Déclarer un objet **JeuTaquin** (dans les champs) dans votre formulaire et l'utiliser (Faire appel à ses membres).
- Les flèches doivent déplacer les chiffres vers la case vide. Utiliser les images de flèches fournies (Microsoft - Docs, 2022) avec cet énoncé.

- Pour quitter, le joueur peut cliquer sur le X.
- Au démarrage de l'application, la grille doit être mélangée. Indice : événement Load ou bien dans le constructeur.
- Assurez-vous que le jeu fonctionnement sensiblement de la même façon que dans la console.

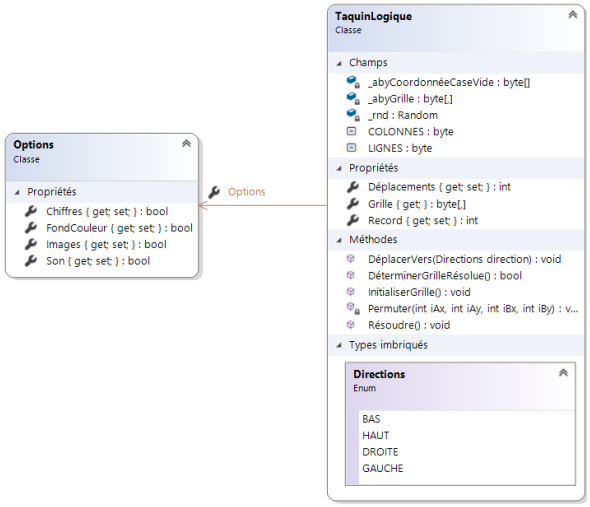
Partie #3 : Gestion du multiformulaire – Les options – 40 %

Cette partie consiste à ajouter un formulaire supplémentaire au projet **JeuTaquinWPF** afin de supporter des options. Voici l'interface à concevoir et à afficher lorsque l'utilisateur clique sur le bouton « **Paramètres…** » :



Consignes :

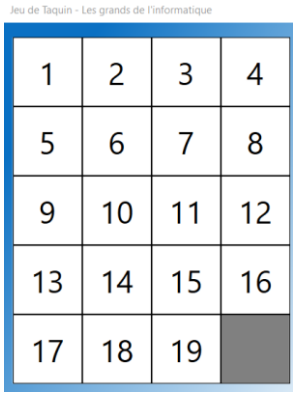
- Ajouter un formulaire nommé **frmParamètres** et y déposer les composantes visuelles. Les textes sont 14px de grandeur de police et en gras, sauf pour les CheckBox qui sont en 12px.
- Dans la bibliothèque **JeuxLib**, avec la classe **JeuTaquin**, créer une nouvelle classe nommée **Options** comme suit :



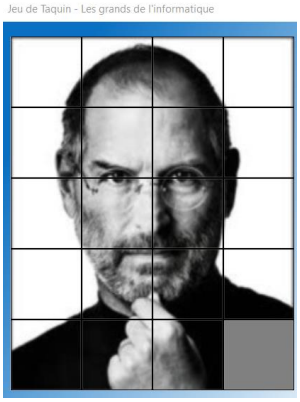
Ajouter une propriété **Options** (de type Options) à la classe **JeuTaquin**.

- Les propriétés de la classe **Options** devront être associées au formulaire via le *DataContext* et pour chacun des champs visuels via *Binding*.
- Les options **Chiffres** et **Images**: Permet de voir des chiffres ou bien des morceaux d'images (les images sont jointes à l'énoncé) de Steve Jobs (Isaacson, 2011). Exemple:

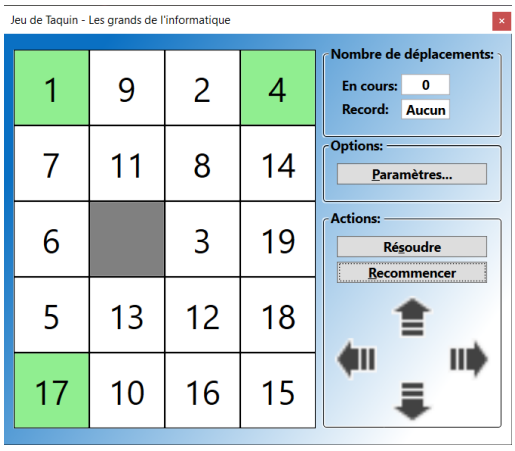
Mode **Chiffres**



Mode **Images**



- L'option **Fond coloré (chiffres seulement)** : Lorsque cette option est activée, la couleur de l'arrière-plan devient verte **lorsque le chiffre se trouve à la bonne position** dans la grille. Exemple:



- L'option **Son lorsque réussi** : Lorsque cette option est activée, le son **success.mp3** est joué lorsque la grille est résolue. Le son doit être implémenté dans le formulaire MainWindow.xaml. L'objet à utiliser est **MediaPlayer** avec les méthodes **Open** et **Play**. Vous devez créer un dossier **Sons** et y déposer **le fichier .MP3**. **Vous devez toujours copier les fichiers de son avec l'exécutable pour que ça fonctionne**.

Barème d'évaluation

Partie #1 – Branche *master* étiquetée, création de la branche **Version 2** + préparation..... /1.0
Partie #1 – Création et incorporation de la bibliothèque, jeu console fonctionnel..... /1.0
Partie #2 – Création du projet de type WPF (+ composantes visuelles) + bibliothèque..... /2.0
Partie #2 – Programmation de la logique d'affichage en utilisant la classe JeuTaquin ..... /2.0
Partie #3 – Ajout du formulaire **frmParamètres** (+ composantes visuelles) ..... /1.0
Partie #3 – Création de la classe **Options** et association avec le formulaire ..... /1.0
Partie #3 – Programmation de la logique des options..... /2.0
Note totale* ..... /10.0

* Tout travail plagié en partie ou en totalité se verra attribuer une note totale de 0 %.

PDEA #1: Lors d'activités d'évaluation sommative en classe ou hors classe (documentation, rapport de laboratoire, rapport de stage, examen), une **pénalité maximale de 10 %** peut être retranchée de la note finale de ladite évaluation (le barème étant de **0,5%/erreur incluant les fautes répétitives**). Pour les commentaires dans les programmes informatiques, le barème est de **0,5% par faute**. Pour les interfaces utilisateur, le barème est de **1% par faute**.

PDEA #4 : Toute évaluation sommative remise après la date d'échéance fixée se voit attribuer la note zéro pour les étudiants **de 2e à 6e session**. Afin de faciliter l'accueil et l'intégration des nouveaux étudiants de **1re session**, cette règle s'appliquera de la façon suivante : 30 % de pénalité pour une 1re offense (à condition que la remise soit faite dans les 24 heures suivant la date de remise officielle), 100% de pénalité pour les offenses subséquentes.

Références

Isaacson, W. (2011). *Steve Jobs.* LATTES.

Microsoft - Docs. (2022, 01 18). *Bibliothèque d'images Visual Studio*. Récupéré sur Visual Studo: https://docs.microsoft.com/fr-fr/visualstudio/designers/the-visual-studio-image-library?view=vs-2022