



Cégep de Saint-Hyacinthe
Département d'informatique

Programmation à base d'objets et d'événements (420-GEA-HY)

TP4 – Jeu d'échec - déplacement de pièces

Préparé par
Martin Lalancette
bureau D 2330A

DESCRIPTION

But :	Conception d'un programme en C# de type formulaire comportant l'utilisation des notions vues en classe.
Objectifs :	<ul style="list-style-type: none">• Générer un diagramme de classe• Pratiquer les notions antérieures• Notions d'héritage• Respecter les normes de programmation
Durée :	9 à 12 h
Pondération :	10 pts
Remise :	À la fin de la semaine 15.
Contenu général :	<ul style="list-style-type: none">• Remettre vos documents d'analyse et votre solution contenant le projet dans un document .ZIP via LÉA.
Notes	<ul style="list-style-type: none">• 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 <u>être disponible sur demande</u>.

Jeu d'échec - déplacement de pièces (projet formulaire) – 100 %

Vous devez concevoir une application **partielle** de type formulaire nommée « **Jeu d'échec** ». L'objectif de ce TP est de programmer les déplacements des pièces (Roi, Reine, Cavalier, fou, tour et le pion) en modélisant les classes selon les notions d'héritage vues durant les cours. Petite anecdote historique ([Kasparov vs Deep Blue](#)):... En mai 1997, Deep Blue calculait alors entre 100 et 300 millions de coups par seconde, il pouvait calculer de 12 demi-coups de profondeur en moyenne. Les grands maîtres d'échecs Miguel Illescas, John Fedorowicz, Nick De Firmian et Joel Benjamin aidèrent à sa conception, notamment à sa bibliothèque d'ouverture... **mais, n'embarquons pas là-dedans!!!** Dans ce TP, il suffit de programmer les déplacements uniquement sans tenir compte des autres pièces déposées sur l'échiquier.

Voici un aperçu de ce jeu:

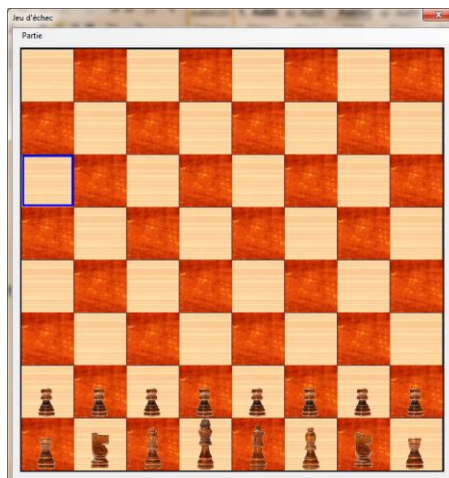

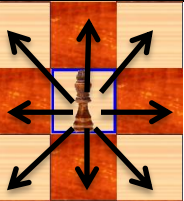

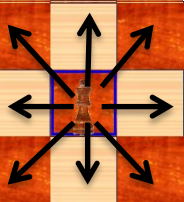

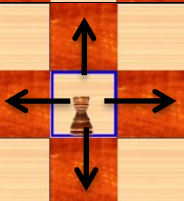

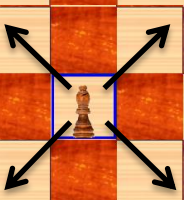

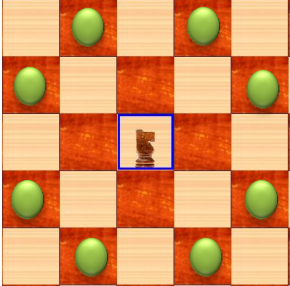




Figure 1 - Disposition des pièces au départ

Règlements - déplacement des pièces :

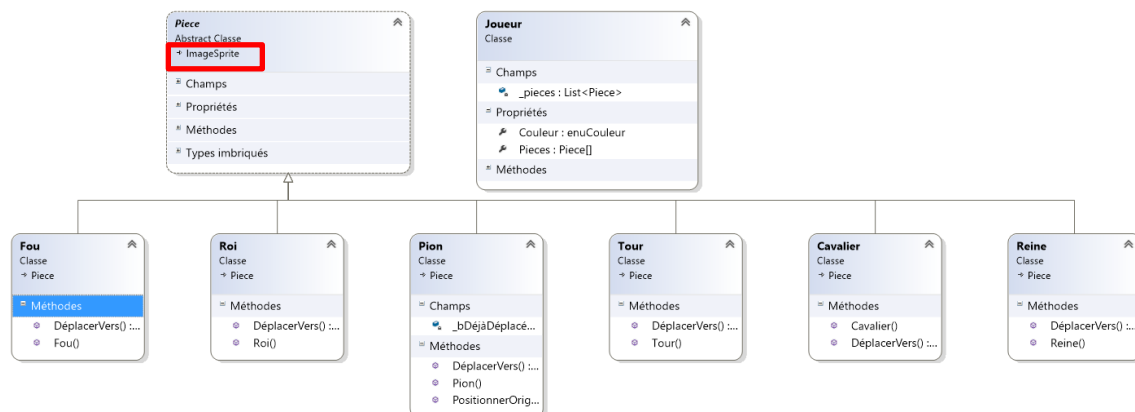
Voici un résumé des règlements associés au déplacement des pièces de ce jeu :

 ROI		<p><u>D'une case</u> dans toutes les directions. Le roi possède également un déplacement spécifique appelé <u>le roque</u> que nous n'avons pas à développer ici.</p>
 REINE		<p>La reine se déplace d'un <u>nombre quelconque de cases</u> dans toutes les directions.</p>
 TOUR		<p>D'un nombre quelconque de cases sur les lignes ou rangées.</p>
 FOU		<p>D'un <u>nombre quelconque</u> de cases sur les <u>diagonales</u>.</p>

 CAVALIER		Le déplacement du cavalier se fait en « L ». C'est la seule pièce qui peut passer par-dessus les autres.
 PION		<p>Le pion se déplace toujours vers l'avant. Depuis sa case initiale, il peut avancer d'une ou deux cases, ensuite il ne peut avancer que d'une case.</p> <p>Le pion attaque ou prend en diagonale. Pas à programmer pour l'instant.</p>

Architecture des classes :

Voici l'architecture de base à programmer afin de gérer les pièces de ce jeu. Le graphique suivant contient des champs, des propriétés et des méthodes de base. L'ajout d'autres éléments peut être nécessaire durant la conception.



La classe Piece dérive de l'objet ImageSprite (VisualArrays) qui va nous donner accès aux membres d'un Sprite, nous permettant ainsi de déplacer une pièce et d'y ajouter nos informations.

Stratégies de développement :

1. Dans le fichier TP4.zip, récupérer la solution nommée « TP4.sln » avec un projet de type formulaire nommé « JeuEchec ». Coder votre algorithme avec MS Visual Studio en C#.
 - a. Toutes les images nécessaires à ce jeu sont déjà présentes dans le projet. Donc pas d'importation à faire. Localisation de ces images :
 - i. **imlCases** : contient deux images représentant les cases de l'échiquier.

- ii. **Fichier ressource** : Contient l'ensemble des images représentant chaque pièce. Pour les accéder par programmation, voici un exemple :
`global::JeuEchec.Properties.Resources.blanc_roi;`
- 2. Programmer l'affichage de la grille en damier.
- 3. Ajouter et programmer les classes mentionnées plus haut.
- 4. Dans chaque méthode **DéplacerVers** de chaque objet (Fou, Roi, Pion, Reine, Cavalier, Tour), programmer le déplacement en fonction des règlements du jeu. La méthode pour le déplacement d'un Sprite est **MoveTo()**.
- 5. Instancier un **objet Joueur** représentant les **noirs**.
 - a. Ajouter les pièces (dans une liste de type Piece) et les déposer (définir les positions) sur l'échiquier comme la figure 1:
 - i. **8** pions, **2** tours, **2** fous, **2** cavaliers, **1** roi, **1** reine
 - b. Les informations utiles lors de la création (Constructeur) d'une pièce sont de fournir la **couleur** (BLANC, NOIR), la **position de départ** sur l'échiquier (Ligne et Colonne). Associer l'**image** (via le fichier ressource) grâce à la couleur passée dans le constructeur de la pièce. D'autres informations doivent être initialisées comme : Size, Animated, AllowDrag, Duration...
 - c. Par programmation, ajouter ces pièces à la grille en utilisant `viaEchiquier.Sprites.AddRange(...)`. Il faut lui passer un tableau, utiliser `ToArray()`.
- 6. Pour permettre l'utilisation de la souris pour le déplacement d'une pièce dans la grille, c'est l'événement **SpriteDragAndDropOccured** qu'il faut utiliser.
- 7. D'autres informations pourront être fournies durant les cours...

BONUS DE 20% (2 pts sur 10) supplémentaire

Vous avez fini et il vous reste du temps? Vous pouvez essayer d'ajouter les fonctionnalités suivantes :

- 1. Effectuer un test unitaire automatisé sur une méthode (à vous de choisir),
- 2. Ajouter les pièces blanches,
- 3. Éviter de passer par-dessus les pièces (sauf le cavalier qui peut le faire),
- 4. Manger une pièce adverse en les faisant disparaître de l'échiquier. Cas spécial du pion qui attaque en diagonale aussi,
- 5. Alternner entre joueur (BLANC et NOIR) pour le coup à jouer. À partir d'ici, une vraie partie peut être possible entre deux utilisateurs sur le même ordinateur.
- 6. Implanter le protocole TCP/IP pour communiquer avec deux ordinateurs (CommunicationLib).

Barème d'évaluation

Respect des normes de programmation.....	/1.0
Affichage de la grille en damier.....	/0.5
Structurer les classes selon l'analyse.....	/2.0
Utilisation de l'héritage - DéplacerVers.....	/1.0
Création d'un objet Joueur.....	/1.0
Création des pièces de façon dynamique	/1.0
Déposer les pièces - position de départ.....	/1.5
Respect du déplacement des pièces	/2.0
Note totale*	/10.0

BONUS **/2.0**

* 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 #3 : Toute évaluation sommative remise **après la date d'échéance fixée** se voit **attribuer la note zéro**.