L'ÉNIGME DU CAVALIER

2024-2025

2024-2025
ÉCOLE TECHNIQUE DES MÉTIERS DE LAUSANNE/GYMNASE D'YVERDON-LES-BAINS
MSIG

Remerciements

Je tiens tout d'abord à exprimer mes remerciements envers mes répondants, Madame Aurélie Curchod de l'ETML et Monsieur Sébastien Comtesse du Gymnase d'Yverdon-les-Bains, pour leur suivi constant et leur implication dans mon travail. Je remercie également Monsieur Jonathan Melly, mon maître de classe, pour son soutien dans la réalisation de mon travail pratique (code). Enfin, un grand merci à Kilian Stonckinger, camarade de classe et très bon ami, pour son aide précieuse dans la structuration de mon code.

Résumé du Travail de maturité spécialisé (TMsp)

Nom, Prénom : Mascarenhas, Joao

Classe : MSIG 2024-2025

Titre : L'énigme du cavalier

Répondant GYYV[1] : Sébastien Comtesse

Répondante ETML[2] : Aurélie Curchod

Résumé du travail de maturité :

Mon travail de maturité, réalisé sur une période de six mois, porte sur l'énigme du cavalier proposé par Leonhard Euler sous une forme mathématique. J'ai développé un jeu console basé sur cette énigme, en utilisant le langage C# sur Visual Studio 2022.

Afin d'expliquer l'ensemble du processus de création de mon code, j'ai rédigé un rapport détaillant chaque étape.

Ce rapport commence par une introduction sur Leonhard Euler, le problème du cavalier et mes motivations personnelles. Il se poursuit par une présentation du développement de mon jeu console, ainsi qu'une réflexion sur la problématique posée. Ensuite, diverses solutions sont présentées, suivies de la conclusion du rapport.

X

Joao Mascarenhas

[1] Gymnase d'Yverdon-les-Bains
[2] Ecole technique des métiers de Lausanne

Table des matières

1 Introduction

Commenté [SC1]: Bien

1.1 Leonhard Euler et l'énigme du cavalier

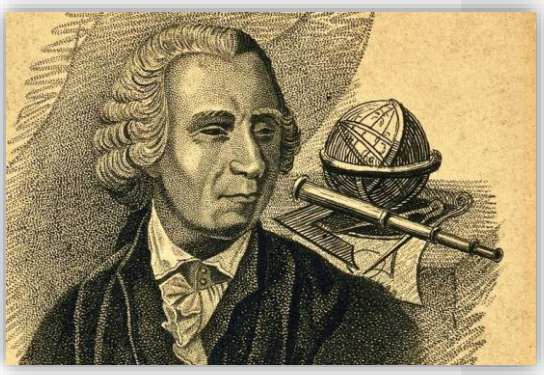


Figure 1: Leonhard Euler

Leonhard Euler était un mathématicien et physicien. Il est reconnu comme l'un des plus prestigieux génies des mathématiques de tous les temps, ayant contribué à l'évolution de nombreux domaines mathématiques tel que l'algèbre, la géométrie, la théorie des graphes ou même la mécanique. Mais aussi, d'autres domaines tel que la physique et l'astronomie. Il a créé des concepts comme la somme infinie, la notation exponentielle (e^x), ainsi que la lettre f pour désigner une fonction. Enfin, L'énigme du cavalier, ou problème du tour du cavalier, a été formulée par Christian Goldbach, mais c'est Leonhard Euler qui a proposé une approche mathématique pour la résoudre. Euler a analysé le problème en utilisant la théorie des graphes, en considérant les cases de l'échiquier comme des sommets et les déplacements du cavalier comme des arêtes. Elle a été évoquée sous forme d'un jeu de société, inspiré des échecs et principalement du pion du cavalier.

1.2 Règles et concept de l'énigme

Le problème du cavalier est une énigme de logique et de mathématiques concernant les mouvements d'un cavalier sur une table d'échec. Le but du joueur est de parcourir toutes les cases d'un échiquier de huit carrés sur huit en ne passant qu'une seule fois par chaque case. Le joueur devra donc réfléchir à chaque déplacement du cavalier car il existe différentes solutions pour résoudre cette énigme. Il existe deux variantes principales de cette énigme.

Figure 2: Déplacements cavalier

La variante fermée : Dans cette variante, le cavalier doit retourner sur la même case d'où il a débuté.

La variante ouverte : Dans cette variante, le cavalier doit passer sur chaque case. Mais, contrairement à la variante fermée, le cavalier ne retournera pas sur la case de départ.

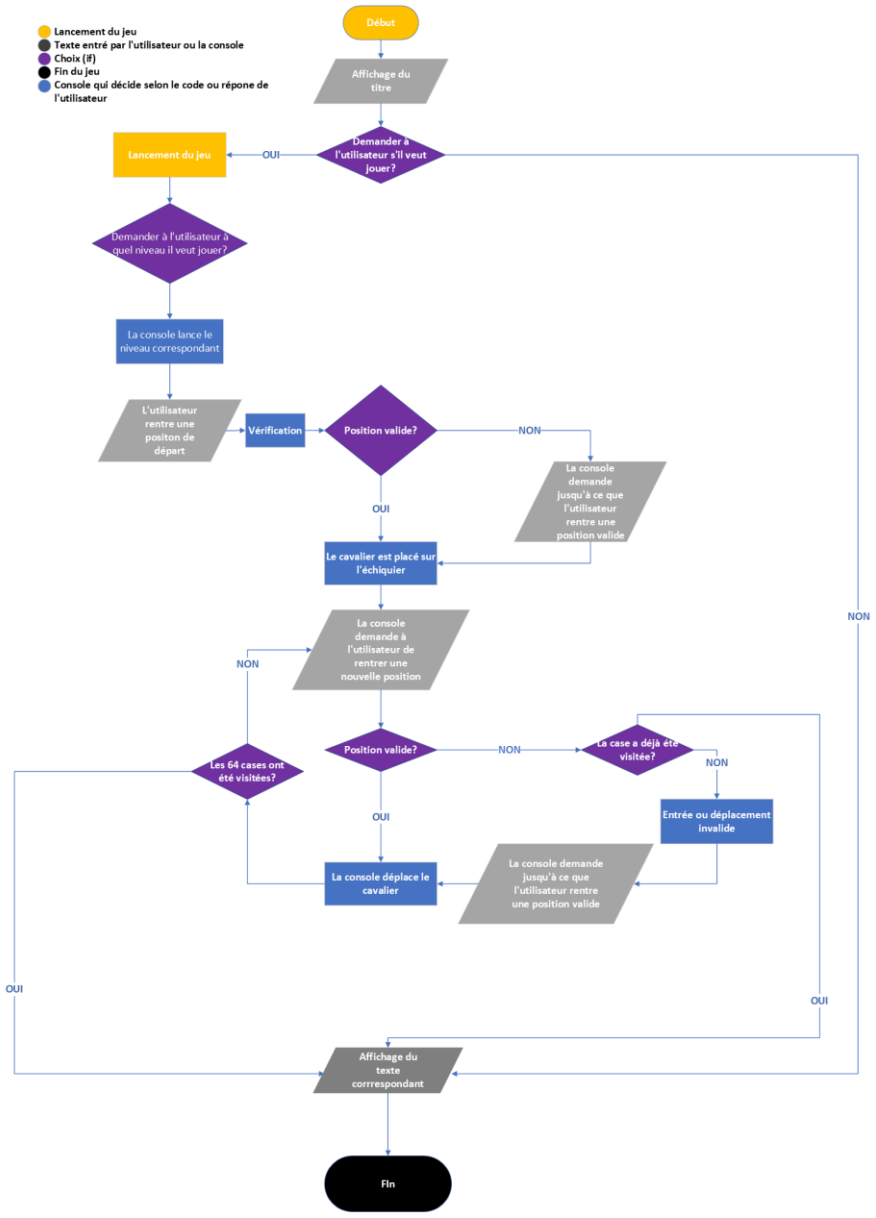
1.3 Mes motivations

Depuis mon plus âge, je pratique les échecs, un jeu auquel je me suis initié en jouant avec mon grand frère. Par conséquent, je maitrise les règles et les différentes stratégies de ce jeu. Cependant, à force d'y jouer, mon attachement envers ce jeu a progressivement diminué. Néanmoins, le défi proposé par

l'énigme a ravivé mon désir de relever un défi plus complexe. Pour ce projet, j'ai choisi de créer ma propre version de ce problème, en y ajoutant de nouvelles variantes, et en utilisant le langage de programmation C# (C SHARP). Ce choix ne traduit pas un intérêt pour la technique, mais aussi d'une volonté de mieux comprendre les attentes de chaque utilisateur, ce qui me permet d'anticiper plus facilement les défis proposés par ce minutieux projet. De ce fait, je peux me mettre dans la peau de l'utilisateur et ainsi mieux interpréter le sujet.

2 Développement

2.1 Diagramme de flux de l'énigme du cavalier



2.2 Problématique

Comment susciter l'intérêt d'un jeune, âgé entre 12 et 25 ans, pour un jeu basé sur l'énigme du cavalier, de manière qu'il ait envie de se divertir tout en relevant le défi proposé ?

2.2.1 Choix de la problématique

L'énigme du cavalier est une activité fascinante pour le cerveau, mais elle peut aussi être perçue comme complexe, compliquée, exigeante, nécessitant de la concentration, au point d'être parfois jugée ennuyante. En tant que personne jeune, je suis extrêmement bien placé pour comprendre que ce genre de jeu intellectuel peut ne pas plaire aux générations actuelles et à venir. On est plus attiré par des jeux interactifs ou plus graphiques. Cette réflexion soulève plusieurs questions telles que : Comment susciter l'envie de jouer chez un jeune, quels moyens utiliser pour y parvenir, comment le rendre plus ludique et plus intéressant et quel est le public cible ?

2.3 Langage de programmation

Pour ce projet complexe, j'ai décidé d'utiliser le langage de programmation orienté objet développé par Microsoft, en 2000, C#. C Sharp est un langage polyvalent qui permet de développer des applications desktop, web, mobiles ou même des jeux sur Unity. Il est inspiré des langages de programmation Java et C++.

2.4 Structure du code

Dans les pages à suivre, mon but sera d'expliquer toute la structure de mon code de manière simple et compréhensible. Je traverserai donc toutes les étapes à l'élaboration de mon énigme du cavalier. Pour mieux comprendre mes explications, voici une image de mon jeu console. On peut donc apercevoir un tableau avec des coordonnées, la position du cavalier (C) et ses déplacements disponibles en bleu et la case déjà utilisé en rouge.

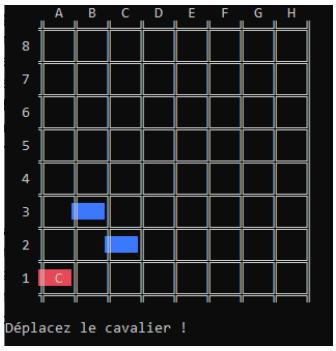


Figure 3: Aperçu jeu console

2.4.1 Concept de Programmation

Afin de structurer mon code de manière claire et efficace, j'ai utilisé plusieurs outils qui me permettront d'optimiser mon code. Ces outils visent à réduire les répétitions, améliorer la lisibilité et maximiser la réutilisation du code, ce qui contribue à un développement plus rapide et à un code plus polyvalent. L'objectif avec ces outils est donc de rendre le programme plus facile à lire.

2.4.1.1 La Classe

La classe est un élément clé de la programmation orientée objet (POO). Elle permet de regrouper des attributs et des méthodes. Les attributs définissent les caractéristiques d'un objet, tandis que les méthodes représentent les actions que cet objet peut effectuer ou les actions qu'on peut lui appliquer. Une classe facilite la réutilisation du code, permettant ainsi de ne pas répéter plusieurs fois les mêmes

Commenté [SC2]: Ajouter un exemple concret qui provient de votre code

instructions. En outre, une classe agit comme un raccourci qui simplifie et structure le code de manière plus efficace.

Par exemple, l'exemple suivant montre une classe appelée « Car ». Pour cette classe, les attributs incluent des éléments comme la capacité du réservoir ou la vitesse maximale de la voiture (objet). Les méthodes, quant à elles, représentent des actions telles que faire le plein de carburant, obtenir la vitesse actuelle ou tourner à gauche. Tous ces éléments sont organisés au sein d'une classe, ce qui permet de les réutiliser facilement.

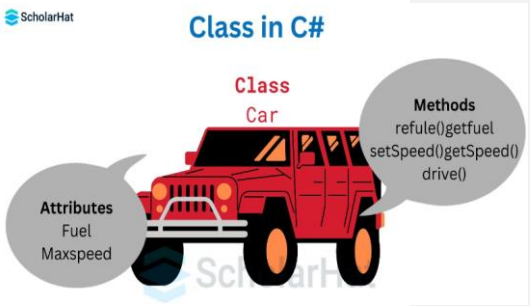


Figure 4: Une classe POO

2.4.1.2 *Le constructeur*

Un constructeur est une méthode spéciale d'une classe, appelée automatiquement lors de la création d'un objet. Il sert à initialiser les attributs de l'objet et garantit que celui-ci est dans un état valide dès son appel.

```
2 références
public Cavalier(int taille = 8)
{
    Taille = taille;
    Plateau = new char[Taille, Taille];
    VisitedPositions = new List<string>();
    DeplacementsCavalierX = new int[] { -2, -1, 1, 2, 2, 1, -1, -2 };
    DeplacementsCavalierY = new int[] { 1, 2, 2, 1, -1, -2, -2, -1 };
    InitialiserPlateau();
}
```

Figure 5: Les constructeurs de mon code

2.4.1.3 *Les attributs*

Un attribut est une variable définie à l'intérieur d'une classe. Il sert à stocker des informations propres à chaque objet créé à partir de cette classe. Les attributs permettent à un objet de mémoriser son état et d'être manipulé grâce à des méthodes.

```
public int LastX;
public int LastY;

7 références
public char[,] Plateau { get; private set; }
7 références
public List<string> VisitedPositions { get; private set; }
15 références
public int Taille { get; private set; }
3 références
public int[] DeplacementsCavalierX { get; private set; }
3 références
public int[] DeplacementsCavalierY { get; private set; }
```

Figure 6: Les attributs

2.4.1.4 *Les méthodes*

En programmation orientée objet (POO), une méthode est une fonction définie à l'intérieur d'une classe. Elle représente une action qu'un objet peut ou doit effectuer. Une méthode est réutilisable et rend le code plus lisible et mieux structuré. Pour la rappeler, il faut écrire 'nomdelaméthode(lesparamètres) ;'

```
2 références
public void MarquerPositionVisitee(int x, int y)
{

    string position = $"{(char)(y + 'A')}{Taille - x}";
    if (!VisitedPositions.Contains(position))
    {
        VisitedPositions.Add(position);
    }
}
```

Figure 7: Les méthodes

2.4.1.5 Les tableaux et les listes

Un tableau est une manière pratique de stocker une collection d'éléments du même type sous un se ul nom, où chaque élément est identifié par une position unique, ou index. Cela vous permet d'accéd er à chaque élément individuellement en connaissant son index, ce qui facilite la gestion de grandes quantités de données.

```
7 références
public char[,] Plateau { get; private set; }
```

Figure 8: Le tableau

En C#, une liste est une structure de données dynamique, qui permet de stocker plusieurs éléments du même type (int, char, etc) durant le code.

```
7 références
public List<string> VisitedPositions { get; private set; }
```

Figure 9: La liste des cases visitées

2.4.2 *Titre*

Mon code s'initialise avec mon titre « LE CAVALIER » écrit en art ASCII[3], l'art ASCII utilise des caractères du code ASCII pour créer des représentations visuelles ou des dessins, en disposant chaque caractère comme une sorte de pixel pour former des images ou du texte stylisé. Dans un premier temps cela permet d'informer l'utilisateur que le jeu console est sur le point de démarrer et que le programme a bien été lancé. Dans un deuxième temps, l'utilisation de cet art apporte un aspect

```
static void Main()
{
    // Affichage du titre
    string cavaliertitre = @"
 _     _____   ____    _    _     ___  _____ ____
| |   | ____| / ___|  / \  | |   |_ _|| ____|  _ \
| |   |  _|   | |     / _ \ | |    | | |  _| | |_) |
| |___| |___  | |___ / ___ \| |___ | | | |___|  _ <
|_____|_____|  \____/_/   \_\_____|___||_____|_| \_\
";
    Console.WriteLine(cavaliertitre);
    Thread.Sleep(1500);
```

plus soigné et visuellement attrayant, ce qui rends le jeu console plus captivant pour un jeune joueur ou une jeune joueuse.

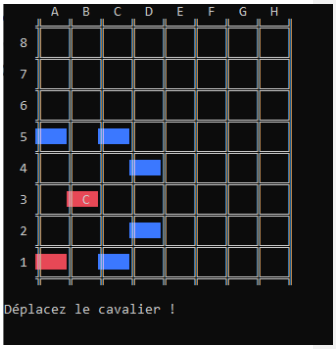
Tout d'abord, je crée mon titre et je le stocke dans une variable appelée « cavaliertitre ». Ensuite, j'utilise 'Console.WriteLine(cavaliertitre);' pour afficher le contenu de cette variable à l'écran. De cette façon, je peux réutiliser le titre plusieurs dans mon code efficacement et rapidement. Et pour finir, je

[3] American Standard Code for Information Interchange

dis à la console d'attendre 1500 millisecondes tout en affichant le titre avant de démarrer le corps du programme grâce à 'Thread.Sleep(1500) ;'.

2.4.3 L'échiquier

Tout d'abord, le plateau de jeu est présenté dans ce code sous forme d'un échiquier, mais suivant les déplacements du pion du cavalier. C'est une grille de taille 8x8, avec des cases définies par des lettres pour les colonnes (A à H) et chiffres pour les lignes (1 à 8). La valeur 8 est stockée dans une variable constante nommée « taille ». Ainsi, chaque fois que le programme utilise taille, il comprendra qu'elle correspond à 8.



Afin de le rendre plus esthétique, j'ai utilisé la table des caractères Windows pour ajouter des caractères introuvables sur le clavier. De cette manière, j'utilise de plus grosses lignes pour donner la forme d'un échiquier réel.

Après chaque mouvement du cavalier, l'échiquier précédent est effacé, libérant ainsi l'espace pour le nouveau plateau, où la position précédente du cavalier est affichée en rouge et les cases accessibles sont colorées en bleu.

Figure 10: Résultat échiquier avec toutes les fonctionnalités

2.4.3.1 *Difficultés liées à l'échiquier*

La première difficulté rencontrée a été de créer un tableau avec des cases bien structurées pour représenter l'échiquier. Pour cela, j'ai utilisé des boucles « for » qui parcourent chaque ligne, en veillant à ce que le nombre total de lignes soit égal à 8. Cette approche permet de remplir le tableau de manière dynamique et de garantir que toutes les cases du plateau soient correctement initialisées avant le début du jeu.

```
for (int i = 0; i < Taille; i++)
{
    Console.Write("    ");
    for (int j = 0; j < Taille; j++)
        Console.Write("╬═══");
    Console.WriteLine("╬");

    Console.Write("  " + (Taille - i) + " ");
```

2.4.3.2 *Difficulté liée aux coordonnées*

Durant l'émolument de mon code, j'ai pu rencontrer une difficulté. Le problème était de pouvoir entrer des coordonnées valides telles que sur un échiquier physique (A1, B3). En débutant ce projet, les coordonnées étaient uniquement indiquées avec des chiffres pour les lignes et les colonnes (par exemple, ligne : 1, colonne : 3).

Face à ce problème, qui perturbe l'expérience utilisateur, j'ai décidé d'afficher les lettres des colonnes pour l'utilisateur. Lorsqu'il saisit une lettre, elle est convertie en chiffre pour la console, puis renvoyée sous forme de lettre. Cela est possible grâce à la conversion des lettres : A devient 0, B devient 1, et ainsi de suite, puisque le premier

```
// Convertir la notation de type A1 en indices de tableau
int x = taille - (position[1] - '0');
int y = position[0] - 'A';
```

indice d'un tableau commence à 0.

2.4.4 *Déplacement cavalier*

Commenté [SC3]: Bien

Le cavalier est une pièce des échecs avec un mouvement particulier en forme de L. Cela signifie qu'il se déplace de deux cases en horizontale et 1 case perpendiculairement en vertical, ou inversement. Les mouvements du cavalier sont définis dans deux tableaux différents, 'deplacementsCavalierx' qui définit les déplacements sur l'axe x du tableau avec tous les mouvements possibles (+2, +2, -2, -2, +1, +1, -1, -1), puis les mouvements verticaux inverses aux horizontaux sur l'axe y (+1, -1, +1, -1, +2, -2, +2, -2).

```
static int[] deplacementsCavalierX = new int[] { 2, 2, -2, -2, 1, 1, -1, -1 };
static int[] deplacementsCavalierY = new int[] { 1, -1, 1, -1, 2, -2, 2, -2 };
static List<string> visitedPositions = new List<string>();
```

Au début du code, une liste est utilisée pour enregistrer les coordonnées saisies par l'utilisateur. Chaque coordonnée est mémorisée et ne pourra plus être utilisée. Lorsque l'utilisateur rentre un mouvement, la console vérifie si le mouvement est valide grâce aux mouvements annoncé dans les axes x et y. Et ensuite, elle vérifie si la coordonnée n'a pas déjà été visitée en la recherchant dans la liste des positions visitées. Si le déplacement est valide et que la case n'a pas été visitée, le cavalier est déplacé sur la nouvelle case choisie.

Si le mouvement n'est pas compatible avec l'axe x et y, la console affiche déplacement invalide. Et si l'utilisateur retourne sur une case déjà présente dans la liste 'visitedPositions' la console affichera que l'utilisateur a perdu et que, par conséquent, il n'a pas résolu l'énigme du cavalier.

2.4.4.1 *Difficultés liées au cavalier*

Commenté [SC4]: Très bien

Lors de l'élaboration de mon projet, la gestion des déplacements du cavalier a représenté une difficulté. J'ai rencontré plusieurs difficultés que je vais détailler ci-dessous.

La première difficulté était de créer les mouvements du cavalier. Pour remédier à cela, comme mentionnée plus haut, j'ai créé des tableaux contenant les déplacements sur l'axe x et y. Le principe est simple : si le cavalier se déplace de 2 cases sur l'axe X, il doit simultanément se déplacer de 1 case sur l'axe Y, et inversement.

La deuxième difficulté était d'implémenter correctement le déplacement du cavalier sur l'échiquier. Une fois la nouvelle coordonnée saisie par l'utilisateur, le programme commence par vérifier si le mouvement est valide. Si c'est le cas, la console met à jour l'affichage en effaçant l'ancien plateau, en supprimant le symbole 'C'de la position précédente, puis en coloriant cette case en rouge pour indiquer qu'elle a déjà été visitée. Ensuite, le cavalier est placé sur sa nouvelle position, et le plateau est réaffiché avec les mises à jour.

La troisième difficulté était de faire en sorte que le cavalier ne sorte pas du tableau de jeu. En effet, sans vérification du programme l'utilisateur pourrait rentrer une coordonnée hors de l'échiquier. Pour régler cela, avant de valider le déplacement du cavalier, le programme vérifie si la position entrée reste dans les limites, entre les indices 0 et 7 pour chacun des axes. Si la position proposée dépasse les limites de jeu, la console affichera que le déplacement n'est pas valide et le demandera de rentrer une valeur valide

```
static string ObtenirPositionValidedif()
{
    while (true)
    {
        string position = Console.ReadLine().ToUpper();

        if (position.Length == 2 &&
            position[0] >= 'A' && position[0] <= 'H' &&
            position[1] >= '1' && position[1] <= '8')
        {
            return position;
        }
        else
        {
            Console.WriteLine("Position invalide. Entrez une position valide (ex: A1, B3, etc.).");
        }
    }
}
```

2.4.5 Cases disponibles

Afin d'améliorer la lisibilité des déplacements possibles pour l'utilisateur, j'ai décidé d'afficher toutes les options de mouvement depuis la case où se trouve le cavalier. Pour cela, le programme parcourt les déplacements définis dans les tableaux 'deplacementsCavalierX' et 'deplacementsCavalierY', puis vérifie si la case correspondante est disponible. Si un déplacement est valide et que la case n'a pas encore été visitée, son fond devient bleu afin d'indiquer à l'utilisateur qu'il peut s'y déplacer, l'outil utilisé est le 'if' qui est une condition pour que le programme continue. Cela aide l'utilisateur à se déplacer plus facilement et rapidement, rendant ainsi l'énigme du cavalier plus engageante pour les jeunes grâce à une meilleure perception des déplacements possibles.

Commenté [SC5]: Bien

```
else
{
    // Colorier les cases possibles en bleu
    bool mouvementValide = false;
    for (int k = 0; k < 8; k++)
    {
        int nouvelleposX = cavalierX + deplacementsCavalierX[k];
        int nouvelleposY = cavalierY + deplacementsCavalierY[k];

        // Vérifier si la case est un déplacement valide
        if (i == nouvelleposX && j == nouvelleposY && !visitedPositions.Contains($"{(char)(j + 'A')}{8 - i}"))
        {
            mouvementValide = true;
            break;
        }
    }

    if (mouvementValide)
    {
        Console.BackgroundColor = ConsoleColor.Blue;
        Console.Write(" " + plateau[i, j] + " ");
        Console.ResetColor();
    }
```

2.4.5.1 Difficulté liée aux cases disponibles

La principale difficulté rencontrée, était que la console affiche les cases bleues pour les cases ayant déjà été visitées. Afin d'éviter ce problème, j'ai modifié le code pour que la console ignore les cases dans 'visitedPositions'. Ainsi, les cases en bleu ne s'affichent que sur les cases vides où un déplacement est possible.

2.4.6 Cases indisponibles

Les cases indisponibles dans le programme sont celles que le cavalier a déjà visité. Conformément aux règles de l'énigme du cavalier, ces cases ne peuvent plus être empruntées une seconde fois, ce qui les rend inaccessibles. Sur un échiquier physique, les joueurs doivent marquer leur passage grâce des outils ou visuellement, ce n'est pas automatique. Afin d'améliorer l'expérience de l'utilisateur et de faciliter la gestion de ces cases, j'ai décidé de les colorier en rouge à chaque fois que le cavalier passe dessus.

Si la case est dans la liste des positions visitées, la console colorie la case en rouge et le rempli d'un espace vide.

```
for (int j = 0; j < taille; j++)
{
    // Colorier les cases visitées en rouge
    if (visitedPositions.Contains($"{(char)(j + 'A')}{8 - i}"))
    {
        Console.BackgroundColor = ConsoleColor.Red;
        Console.Write(" " + plateau[i, j] + " ");
        Console.ResetColor();
    }
```

Cette approche rend l'énigme plus fluide et aide les joueurs à mieux suivre les déplacements sur l'échiquier.

2.4.6.1 Difficultés liées aux cases indisponibles

La difficulté rencontrée a été de s'assurer que les cases visitées soient correctement exclues des déplacements valides. Le programme ne doit pas afficher ni permettre de déplacer le cavalier sur une case qui a déjà été visitée. Pour ce faire, une vérification supplémentaire a été ajoutée dans la logique des déplacements.

```
if (i == nouvelleposX && j == nouvelleposY && !visitedPositions.Contains($"{(char)(j + 'A')}{8 - i}"))
{
    mouvementValide = true;
    break;
}
```

Avant de colorier une case en bleu, le programme teste si la case est déjà présente dans la liste 'visitedPositions'. Si c'est le cas, la case n'est pas considérée comme disponible pour le déplacement et ne sera pas affichée en bleu, donc elle restera rouge.

2.5 Niveaux

Afin d'attirer un public plus large de jeunes et d'améliorer mon jeu sur console, j'ai conçu quatre niveaux de difficulté progressive, c'est à l'utilisateur de décider à quel niveau il désire jouer. Le premier niveau, le plus simple, est conçu pour guider l'utilisateur vers la réussite avec un temps illimité, des cases disponibles affichées en bleu et des cases visitées en rouge. Dans le deuxième niveau, les indications bleues disparaissent, ne laissant visibles que les cases visitées en rouge. Enfin, le troisième niveau, reprend les mêmes règles que le deuxième, mais avec une limite de temps de 360

secondes, ce dernier représente le défi ultime. En intégrant des éléments de compétition et de progression, on encourage les jeunes à se dépasser tout en s'amusant, renforçant ainsi leur engagement et leur plaisir à relever le défi.

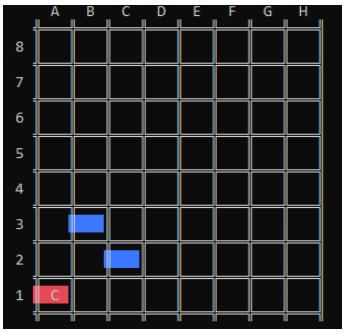


Figure 11: Niveau 1

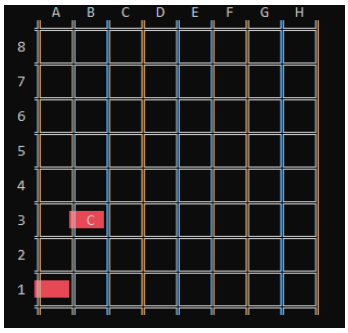


Figure 12: Niveau 2

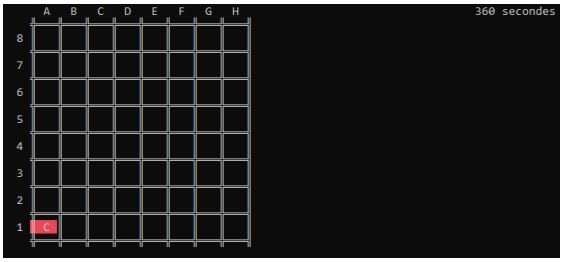


Figure 13: Niveau 3

2.6 Améliorations Possibles

Pour élargir l'attrait du jeu du cavalier et capter l'attention d'un plus vaste public de jeunes, plusieurs améliorations peuvent être apportées au code et à l'expérience du jeu. Ces ajustements visent à rendre le jeu plus interactif, motivant et accessible tout en renforçant l'engagement des joueurs. Voici quelques suggestions. Je vous propose trois autres moyens d'attirer un maximum de jeunes.

2.6.1 Interface Graphique

Les jeunes d'aujourd'hui sont davantage attirés par des jeux interactifs. En tant que jeune moi-même, je peux confirmer que j'ai une préférence pour les jeux avec une interface graphique. De plus, plusieurs études viennent appuyer cette tendance.

Par exemple, Nielsen, une entreprise spécialisée dans l'analyse de données et les études de marché, a mené une étude sur ce sujet. Selon cette étude : « 65 % des joueurs âgés de 18 à 34 ans privilégient les jeux avec une interface de haute qualité et une expérience immersive. Ils estiment qu'une interface graphique bien conçue est essentielle pour leur plaisir de jeu, et une expérience visuelle stimulante et interactive joue un rôle clé dans leur choix de jeu. »[4]

Un autre exemple, situé à droite de ce paragraphe, démontre qu'aux Etats-Unis la génération Z, donc les personnes nées entre 1997 et 2012, sont l'échantillon de la population qui joue le plus au jeux vidéo, suivi de la génération Millénial avec trois pourcents de moins. Et plus on recule à travers les générations plus le pourcentage descends, pour terminer à 1% pour la génération Silence.

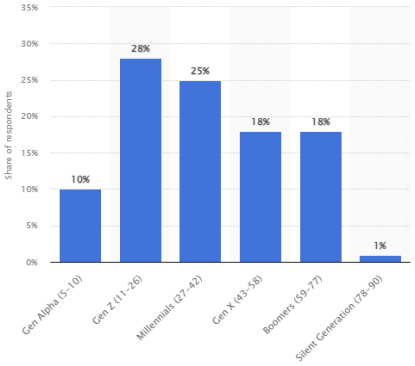


Figure 14: Répartition des joueurs (vidéos) aux Etats-Unis en 2023, par génération.

De ce fait, on peut en déduire que les personnes âgées entre 11 et 42 ans jouent plus à des jeux avec une interface graphique et de la haute qualité.

En conclusion, ajouter une interface graphique à mon énigme du cavalier aurait pu attirer plus de jeunes joueurs et intéressé d'avantages de personnes.

2.6.2 Mode Multijoueur

Dans le but d'attirer un large public de jeunes joueurs, j'ai eu une autre idée qui consistait à introduire un mode multijoueur. Cette stratégie s'inspire de mon expérience personnelle, car j'ai énormément joué à ce type de jeux durant mon adolescence, tels que Fortnite, Grand Theft Auto et Minecraft. Ces jeux sont mondialement reconnus pour leurs modes multijoueur, qui ont joué un rôle clé dans leur succès.

À l'heure actuelle, les jeunes préfèrent, davantage, les jeux multijoueurs aux jeux solos. D'après une étude menée par SELL, ils privilégient les expériences en groupe, tandis que les joueurs plus âgés ont une préférence pour les jeux en solo. « Si le jeu en solo reste une expérience appréciée, le jeu collaboratif gagne du terrain. 61 % des joueurs (+1 point vs 2022) s'adonnent au jeu multijoueur, que ce soit en ligne ou physiquement. Cette tendance est particulièrement marquée chez la jeune génération : 81 % des enfants jouent à plusieurs, contre 58 % des adultes. » (SELL, 2023).

[4] Texte tiré d'une étude paru en 2017

2.6.3 *Version mobile ou application*

Commenté [SC6]: Bien

De nos jours, le téléphone mobile est devenu l'une des plateformes de jeu les plus populaires et les plus utilisées. Grâce à sa portabilité, on peut jouer n'importe où, à tout moment, et avec qui l'on souhaite.

Les jeunes, en particulier, se tournent de plus en plus vers les applications mobiles, tentés par la diversité et la richesse de jeux disponibles. L'introduction d'un jeu au format graphique et mobile pourrait ainsi attirer un large public de jeunes, séduits par la possibilité d'accéder à une expérience de jeu immersive et accessible à tout moment, tout en bénéficiant d'une vaste gamme de choix.

En conclusion, avec l'évolution technologique et l'optimisation des smartphones, les jeux mobiles sont désormais au cœur des tendances de divertissement et cette idée peut rendre mon énigme du cavalier beaucoup plus intéressante pour les jeunes.

2.7 Solutions

Pour résoudre cette énigme, il existe deux types de solution possible. La première est la solution ouverte, le cavalier doit compléter toutes les cases de l'échiquier, mais peut terminer son parcours dans une autre que celle où il a démarré la session. La deuxième est la solution fermée, contrairement à l'autre solution, le cavalier doit exclusivement retourner sur la case dont il est parti.

Pour la solution fermée, sur un échiquier de 8x8, il existe plus de 13 millions de solutions différentes. Alors que pour la solution ouverte il existe environ 33 millions de solutions. En effet, il est plus facile de résoudre l'énigme en variante fermée.

Il existe également des algorithmes permettant de résoudre cette énigme de manière plus efficace pour les personnes n'ayant pas d'expériences, tels que l'algorithme de Warnsdorff, créé en 1823. Dans cet algorithme, le cavalier commence sur une case donnée et se déplace vers la case offrant le moins de possibilités de mouvements. Cette approche vise à minimiser les risques de se retrouver bloqué, car elle permet d'éviter les cases qui pourraient restreindre les mouvements futurs du cavalier. Ainsi, au fur et à mesure des déplacements, le jeu devient de plus en plus facile, car les options se réduisent de manière stratégique, rendant l'énigme plus accessible et facile à résoudre.

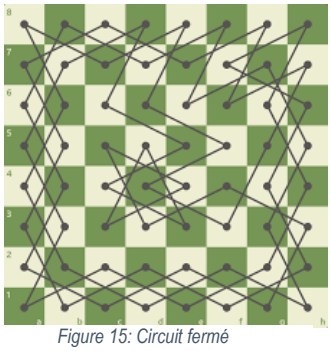


Figure 15: Circuit fermé

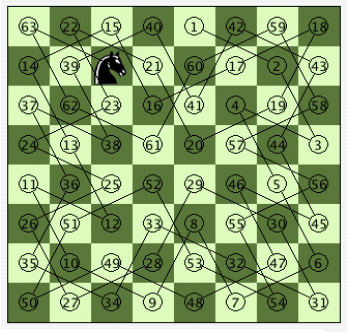


Figure 16: Circuit ouvert

3 Conclusion

En conclusion, l'énigme du cavalier, tout en étant un défi intéressant pour le cerveau, présente des enjeux de taille pour rendre son approche plus accessible et attrayante, surtout pour les jeunes générations. Leonhard Euler, à travers sa contribution à cette problématique, a non seulement apporté des concepts mathématiques fondamentaux, mais a aussi démontré l'importance de la logique et des mathématiques dans des domaines variés, allant de la mécanique à l'astronomie. Cependant, cette énigme, malgré sa richesse, peut paraître complexe ou même lassante à certains, notamment à un public jeune, habitué à des expériences plus visuelles et interactives.

À travers ce projet, mon objectif a été de reproduire cette énigme en y apportant une dimension ludique et moderne, adaptée aux attentes des joueurs d'aujourd'hui. L'utilisation du langage de programmation C# m'a permis de créer une version dynamique du jeu, tout en respectant les fondements mathématiques de l'énigme.

La problématique, qui est comment susciter l'intérêt d'un jeune, âgé entre 12 et 25 ans, pour un jeu basé sur l'énigme du cavalier, de manière qu'il ait envie de se divertir tout en relevant le défi proposé n'est pas facile. La réponse à cette question réside dans l'adaptation des mécanismes de jeu aux préférences actuelles des jeunes. L'idée de rendre les déplacements visibles, de colorier les cases disponibles et déjà visitées, d'ajouter trois niveaux différents afin de rendre l'énigme du cavalier plus complexe pour l'utilisateur et de guider l'utilisateur tout au long de son parcours sont des choix pour rendre cette expérience plus engageante, compétitive et interactive. Mais en offrant des solutions comme une interface graphique, un mode multijoueur ou encore une version mobile, on peut considérablement améliorer l'expérience de jeu et la rendre plus attrayante. Ces améliorations ne sont pas seulement des ajouts techniques, mais des moyens d'apporter plus de charme à un jeu qui, sans cela, pourrait passer inaperçu. Ainsi, la version du jeu que j'ai réalisée représente une base solide qui pourrait, avec quelques ajustements, séduire un plus large public.

Finalement, ce projet m'a permis de redécouvrir l'énigme du cavalier sous un nouveau jour, tout en m'aidant à mieux comprendre comment adapter des concepts intellectuels à un monde en constante évolution. La recherche de solutions accessibles, tout en préservant les défis mathématiques et logiques, est au cœur de ce projet et des idées qui peuvent, à l'avenir, rendre ce genre d'énigmes plus populaires et attrayantes pour les jeunes générations.

4 Bibliographie

24h, A. l. (2025, Janvier 14). *Mat avec deux Cavaliers.* Récupéré sur Apprendre les echecs 24h: https://www.apprendre-les-echecs-24h.com/progresser-aux-echecs/mat-avec-deux-cavaliers/

Beaulieu, G. d. (2025, Janvier 14). *Gymnase de Beaulieu*. Récupéré sur Travaux de Maturités: https://www.gymnasedebeaulieu.ch/travaux-maturites/

Chess. (2025, 02 26). *Termes échiquiens.* Récupéré sur Chess: https://www.chess.com/fr/terms/le-probleme-du-cavalier-echecs

d'Yverdon-les-Bains, G. (2025, 02 27). *Gymnase d'Yverdon.* Récupéré sur Facebook: https://www.facebook.com/gymnaseyverdonlesbains/?locale=fr_FR

Générateur d'art ASCII. (2025, 01 28). Récupéré sur Creative Fabrica: https://www.creativefabrica.com/fr/tools/ascii-art-generator/?text=LE+CAVALIER

Hippie, E. (2016, 03 09). *Le problème du cavalier.* Récupéré sur Youtube: https://www.youtube.com/watch?v=zSOu4z6mKrg

J.Clement. (2024, 05 01). *Distribution of video gamers in the United States in 2023.* Retrieved from Statista: https://www.statista.com/statistics/189582/age-of-us-video-game-players/

larousse. (2025, 01 30). *larousse synonymes*. Récupéré sur larousse: https://www.larousse.fr/dictionnaires/synonymes/mot/14245

naturelles, A. d. (2025, Janvier 14). *Le Goethe des mathématiques.* Récupéré sur Scnat Réseau: https://scnat.ch/fr/uuid/i/a308344e-6b35-527a-ab52-44a4d2acdf83-Le_Goethe_des_math%C3%A9matiques

Nielsen. (2017, 05 01). *U.S. Games 360 Report: 2017*. Récupéré sur Nielsen: https://www.nielsen.com/insights/2017/us-games-360-report-2017/

ScholarHAt. (2025, 01 04). *Objects And Classes I C#.* Récupéré sur ScholarHat: https://www.scholarhat.com/tutorial/csharp/objects-and-classes-in-csharp

SELL. (2023, 10 19). *Etude SELL - Médiamétrie "Les français et le jeu vidéo".* Récupéré sur AFJV: https://afjv.com/news/11292_etude-mediametrie-2023-francais-jeux-video.htm?

Senscritique. (2025, 01 30). *Les records du monde des jeux vidéo*. Récupéré sur Senscritique: https://www.senscritique.com/liste/les_records_du_monde_des_jeux_video/151420

W3schools. (2025, 03 06). *C#*. Récupéré sur W3schools: https://www.w3schools.com/cs/index.php

Wikipédia. (2025, 02 2025). *Ecole techniques des métiers de Lausanne.* Récupéré sur Wikipédia: https://fr.wikipedia.org/wiki/%C3%89cole_technique_des_m%C3%A9tiers_de_Lausanne

Wikipédia. (2025, Janvier 9). *Problème du cavalier*. Récupéré sur Wikipédia: https://fr.wikipedia.org/wiki/Probl%C3%A8me_du_cavalier#:~:text=Le%20probl%C3%A8me%20du%20cavalier%20(ou,chaque%20case%20sans%20y%20repasser.

5 Table des Illustrations

6 Annexe