Rapport Projet C : Réalisation du jeu Takuzu

Réalisé par **Gaël RAIMBAULT** et **Marc ROUGAGNOU**

EFREI, Groupe F

2021/2022

Sommaire :

Présentation du projet :

Le projet a pour but de réaliser le jeu du Takuzu. Il faut donc que le joueur/utilisateur puisse résoudre une grille de jeu, d'avoir la possibilité de choisir la taille de ses grilles, de plus la possibilité que l'ordinateur puisse lui aussi résoudre une grille, et enfin laisser la possibilité au joueur de créer une grille de jeu, le tout en langage de programmation C.

Fonctionnalité du projet :

Le *joueur* peut :

- Résoudre une grille de 4x4
- Résoudre une grille de 8x8
- Créer un masque à sa guise puis résoudre la grille

L'*ordinateur* peut :

- Résoudre une grille de 4x4
- Résoudre une grille de 8x8
- Générer une grille solution

Présentation technique du projet :

1 / Grille de dimension 4x4 :

Parmi les fonctions principales on a :

- `int placement_4x4(int T[4][4], int T_solut[4][4])`
  - → Fonctionnalité : qui permet au joueur de placer les 0/1 à la case souhaitée. Elle fait appel à la plupart des fonctions concernant le 4x4
  - → Paramètres : **T** est la grille de jeu | **T_solut** est la grille solution associée,
  - → Renvoie : 1 ou 2 ou 0 suivant si le coup est valide/correct ou non.
  - → Méthode : La fonction vient demander la case puis le chiffre voulu et vient le placer ou non par rapport à la fonction de coup_Valide et de coup_correct.

- `int move_correct_4x4(int T[4][4], int cas, int tak_solu[4][4])`
  - → Fonctionnalité : Vérifie si le coup est correct
  - → Paramètres : **T** est la grille de jeu | **cas** est la case choisie par le joueur sur ce tour |**T_solut** est la grille solution associée
  - → Renvoie : 0 quand le coup est correct 1 sinon
  - → Méthode : La fonction vient comparer la case dans la grille mask et celle de tak_solu.

- `int move_valid_4x4(int T[4][4], int cas)`
  - → Fonctionnalité : Vérifie la validité d'un coup
  - → Paramètres : **T** est la grille de jeu | **cas** est la case choisie par le joueur sur ce
  - → Renvoie : 0 quand le coup est correct 1 sinon
  - → Méthode : La fonction fait appel à des fonctions qui elles vérifient une validité de coup, et affiche pourquoi le coup est invalide si c'est le cas.

- `void print_tak_4x4(int T[4][4])`
  - → Fonctionnalité : Affiche une grille 4x4
  - → Paramètres : **T** est la grille souhaitée
  - → Renvoie : Rien
  - → Méthode : La fonction affiche une grille à l'aide de printf par rapport à ce qui est dans le tableau (si c'est un -1 ne l'affiche pas, 0 et 1 sont affichés)

- `void tak_play_4x4(int tak_jeu[4][4], int mask[4][4], int tak_solu[4][4])`
    - → Fonctionnalité : Génère une grille jouable en 4x4
    - → Paramètres : **tak_jeu** est la grille finale |**mask** qui est le mask associée | **tak_solu** qui est la grille solution associée
    - → Renvoie : Rien
    - → Méthode : La fonction vient comparer le mask et la grille de solution pour associer à la valeur 1, 0 ou -1.

- `void crt_mask_4x4(int mask[4][4])`
    - → Fonctionnalité : Permet à l'utilisateur de générer un mask
    - → Paramètres : **mask** est la grille qui est associée au mask
    - → Renvoie : Rien
    - → Méthode : La fonction va demander une case que le joueur va découvrir puis demande s'il veut continuer, le mask se met à jour tant le joueur le veut.

- `void auto_tak_4x4(int T[4][4], int T_solut[4][4]`
    - → Fonctionnalité : Permet à l'ordinateur de remplir automatiquement une grille de jeu
    - → Paramètres : **T** est la grille de jeu | **T_solut** est la grille de solution associés
    - → Renvoie : Rien
    - → Méthode : La fonction rempli la grille à l'aide de fonctions secondaire qui font les différentes vérifications.

- `void crt_solut_4x4(int tak[4][4])`
    - → Fonctionnalité : Permet à l'ordinateur de générer aléatoirement une grille solution
    - → Paramètres : **tak** une grille
    - → Renvoie : Rien
    - → Méthode : La fonction vient utiliser le code binaire d'une ligne pour vérifier certaines conditions puis vérifie l'entièreté de la grille.

2 / Grille dimension 8x8 : (Même fonctions que pour une grille 4x4 mais adapté à une 8x8)

Parmi les fonctions principales on a :

- `int placement_8x8(int T[8][8], int T_solut[8][8])`
    - → Fonctionnalité : qui permet au joueur de placer les 0/1 à la case souhaitée. Elle fait appel à la plupart des fonctions concernant le 8x8
    - → Paramètres : **T** est la grille de jeu | **T_solut** est la grille solution associée,
    - → Renvoie : 1 ou 2 ou 0 suivant si le coup est valide/correct ou non.
    - → Méthode : La fonction vient demander la case puis le chiffre voulu et vient le placer ou non par rapport à la fonction de coup_Valide et de coup_correct.

- `int move_correct_8x8(int T[8][8], int cas, int tak_solu[8][8])`
    - → Fonctionnalité : Vérifie si le coup est correct
    - → Paramètres : **T** est la grille de jeu | **cas** est la case choisie par le joueur sur ce tour |**T_solut** est la grille solution associée
    - → Renvoie : 0 quand le coup est correct 1 sinon
    - → Méthode : La fonction vient comparer la case dans la grille mask et celle de tak_solu.

- `int move_valid_8x8(int T[8][8], int cas)`
    - → Fonctionnalité : Vérifie la validité d'un coup
    - → Paramètres : **T** est la grille de jeu | **cas** est la case choisie par le joueur sur ce
    - → Renvoie : 0 quand le coup est correct 1 sinon
    - → Méthode : La fonction fait appel à des fonctions qui elles vérifient une validité de coup, et affiche pourquoi le coup est invalide si c'est le cas.

- `void print_tak_8x8(int T[8][8])`
    - → Fonctionnalité : Affiche une grille 8x8
    - → Paramètres : **T** est la grille souhaitée
    - → Renvoie : Rien

- → Méthode : La fonction affiche une grille à l'aide de printf par rapport à ce qui est dans le tableau (si c'est un -1 ne l'affiche pas, 0 et 1 sont affichés)

- `void tak_play_8x8 (int tak_jeu[8][8], int mask[8][8], int tak_solu[8][8])`
  - → Fonctionnalité : Génère une grille jouable en 8x8
  - → Paramètres : **tak_jeu** est la grille finale |**mask** qui est le mask associée | **tak_solu** qui est la grille solution associée
  - → Renvoie : Rien
  - → Méthode : La fonction vient comparer le mask et la grille de solution pour associer à la valeur 1, 0 ou -1.

- `void crt_mask_8x8(int mask[8][8])`
  - → Fonctionnalité : Permet à l'utilisateur de générer un mask
  - → Paramètres : **mask** est la grille qui est associée au mask
  - → Renvoie : Rien
  - → Méthode : : La fonction va demander une case que le joueur va découvrir puis demande s'il veut continuer, le mask se met à jour tant le joueur le veut.

- `void auto_tak_8x8(int T[8][8],int T_solut[8][8])`
  - → Fonctionnalité : Permet à l'ordinateur de remplir automatiquement une grille de jeu
  - → Paramètres : **T** est la grille de jeu | **T_solut** est la grille de solution associés
  - → Renvoie : Rien
  - → Méthode : La fonction rempli la grille à l'aide de fonctions secondaire qui font les différentes vérifications.

On a également des fonctions secondaires dont 6 pour la validité d'un coup, et 3 pour la génération de grille.

Nous avons choisi d'utiliser en quelque sorte la méthode de « diviser pour régner » qui consiste à séparer le travail avec le maximum de fonction qui répondront à des attentes faciles à réaliser pour les réunir dans une seule à la fin. De plus nous avons utilisé les tableaux statiques pour bien faire la différence entre une grille 4x4 et 8x8, lors de la programmation, de l'utilisation des fonctions et la compréhension du code.

Difficultés rencontrées :

Nous n'avons pas rencontré de grosses difficultés jusqu'à la partie II qui demandait de résoudre automatiquement (par l'ordinateur) une grille. En effet, le problème a été de se lancer car lorsqu'un humain joue il y a un facteur d'aléatoire qui peut être pris en compte or lorsque c'est un ordinateur il faut imiter l'humain en appliquant des logiques de jeu. D'autre part pour faire une grille solution en aléatoire cela nous a donné du fils à retordre, puisque certaines conditions n'étaient pas respectées sans que nous ne voyions le problème. Nous avons réussi à les résoudre en divisant le tout dans des fonctions intermédiaires.

Présentation en image de différents résultats :

Jouer au jeu sur une grille 4x4 :

```
Menu principale :
1 - Resoudre une grille
2 - Resoudre automatiquement une grille
3 - Generer une grille
4 - Creer son mask et jouer dessus
0 - Quitter
1

Menu Resoudre une grille :
Choisissez la taille du Takuzu :
1 - 4x4
2 - 8x8
1
    A   B   C   D
  -----------------
1 |   |   | 0 | 0 |
  -----------------
2 |   |   |   |   |
  -----------------
3 |   |   |   | 1 |
  -----------------
4 |   | 1 |   | 0 |
  -----------------
Saisir colonne puis ligne :
A1
Saisir 1 ou 0 :
1
    A   B   C   D
  -----------------
1 | 1 |   | 0 | 0 |
  -----------------
2 |   |   |   |   |
  -----------------
3 |   |   |   | 1 |
  -----------------
4 |   | 1 |   | 0 |
  -----------------
Coup correct !!Il vous reste 3 vies
Saisir colonne puis ligne :
```

On peut donc voir le menu d'accueil du jeu qui dans lequel on vient mettre le numéro de la fonctionnalité que l'on veut effectuer (dans l'exemple résoudre une grille soit 1), puis apparaît un autre menu pour choisir la taille de la grille (dans l'exemple une de 4x4 soit 1), après on a donc la grille de jeu qui s'affiche et il faut maintenant rentrer la case (colonne puis ligne) (lettre majuscule et chiffre, la saisie est sécurisée), si la case est validée alors on demande la valeur souhaitée (0 ou 1). Si le coup est correct alors un message le signale, mais il existe différents types de message comme :

```
Coup valide mais incorrect
```

Pour un coup valide mais incorrect

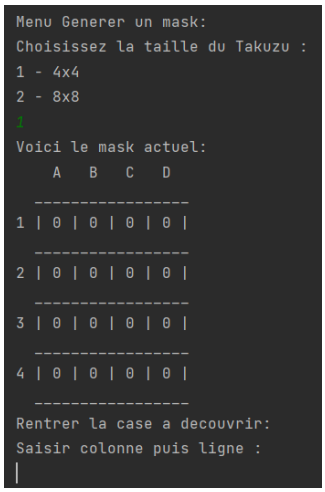
```
Il y a plus d'un certain chiffre qu'un autre, verifier la colonne.
```

S'il y a plus de 0 que de 1 sur la colonne et inversement

```
    A   B   C   D   E   F   G   H
  ---------------------------------
1 |   |   | 1 |   |   |   |   |   |
  ---------------------------------
2 |   |   |   |   |   |   |   |   |
  ---------------------------------
3 |   |   |   |   |   |   |   |   |
  ---------------------------------
4 | 0 |   |   |   |   |   |   |   |
  ---------------------------------
5 |   |   |   |   | 0 | 1 |   |   |
  ---------------------------------
6 |   |   | 0 |   |   |   | 0 |   |
  ---------------------------------
7 |   | 1 |   | 0 |   |   | 1 |   |
  ---------------------------------
8 |   | 1 |   |   |   |   |   |   |
  ---------------------------------
```

Exemple d'affichage d'une grille 8x8

L'utilisateur rentre un mask et joue ensuite :



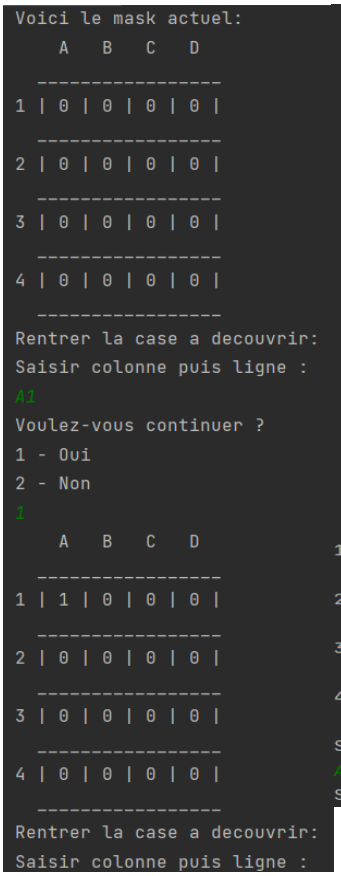
Une grille remplie de 0 va apparaître à l'utilisateur, c'est la grille du mask sans rien qui va s'afficher. Il va maintenant choisir la/les case(s) qu'il veut afficher lors du jeu. Il doit rentrer une case. Une fois fait une question apparaît et demande s'il veut continuer, 1 ou 2 pour oui ou non et on répète ce schéma.

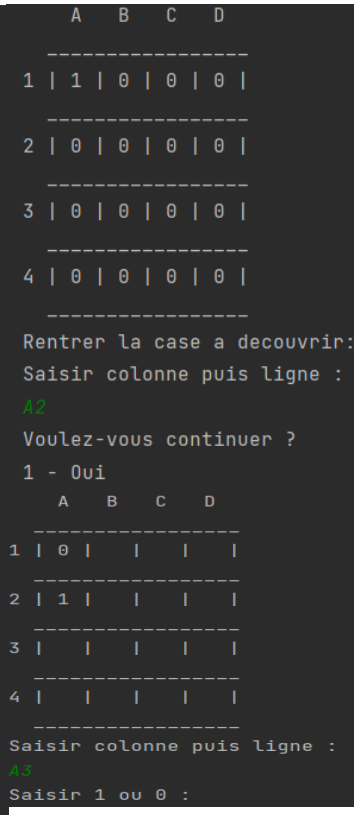
A gauche :

On peut voir que le choix a été fait de continuer en rentrant le chiffre 1

A droite :

On peut voir que le choix a été fait de s'arrêter (suite de la partie de gauche d'où le fait que A1 soit déjà découverte), en rentrant le chiffre 2, le jeu se lance donc avec ce mask.





Conclusion :

- Apprentissage technique : On a appris à mieux utiliser le langage C, la relation entre les différents fichiers, les tableaux à une, et deux dimensions et la réalisation d'une IA.

- Apprentissage organisation/ communication : Lors de ce projet on a communiqué par message et lorsqu'on se voyait en cours, ou encore par appel visioconférence dans lesquels on s'aidait si l'un des deux était bloqué. Puis nous mettions en commun pour que le programme de chacun soit à jour.

- Apprentissage gestion du temps : Lors de ce projet on a dû se dépêcher à certains moments, car en parallèle on avait d'autres projets, des révisions pour des partiels et également cours. Mais on a su s'organiser en fonction de l'emploi du temps de tous les deux en se donnant des dates limites où un certain travail devait être effectué de chaque côté.