15/04/2022

Cassiopée GHIZELLAOUI

Paul ZANOLIN

Table des matières

[Introduction 3](#_Toc103532009)

[Présentation fonctionnelle 3](#_Toc103532010)

[Règles du Takuzu 3](#_Toc103532011)

[Laisser un joueur résoudre une grille 3](#_Toc103532012)

[Résoudre automatiquement une grille 3](#_Toc103532013)

[Générer une grille 4](#_Toc103532014)

[Fonctionnalités additionnelles 4](#_Toc103532015)

[Présentation technique 5](#_Toc103532016)

[Description des principaux algorithmes réalisés 5](#_Toc103532017)

[Interface utilisateur 5](#_Toc103532018)

[Gestion des fichiers 6](#_Toc103532019)

[Gestion des livres supprimés 6](#_Toc103532020)

[Calcul des similarité cosinus 6](#_Toc103532021)

[Structure de données 7](#_Toc103532022)

[Difficultés 8](#_Toc103532023)

[Présentation des résultats 9](#_Toc103532024)

[Menu principal 9](#_Toc103532025)

[Jouer 9](#_Toc103532026)

[Partie 1 9](#_Toc103532027)

[Partie 2 9](#_Toc103532028)

[Partie 3 10](#_Toc103532029)

[Gestion des livres 10](#_Toc103532030)

[Menu utilisateur 11](#_Toc103532031)

[Choix de l’interface 11](#_Toc103532032)

[Concernant les tests 11](#_Toc103532033)

[Conclusion 12](#_Toc103532034)

# Introduction

Nous avons dû réaliser un jeu nommé « Takuzu » en utilisant notre dernier langage tout récemment appris : le C. Un Takuzu qu’est- ce que c’est ? C’est un jeu de réflexion qui consiste à remplir une grille avec les chiffres 0 et 1 en suivant certaines règles dont on parlera plus tard.

L’objectif principal était de nous familiariser avec l’utilisation du langage C en l’appliquant sur un projet concret. On a notamment pu approfondir trois principales notions. Le passage par référence pour passer une grille à une fonction. Les structures pour stocker les grilles. Et pour finir, bien évidemment, les fonctions pour nous permettre d’avoir un code organisé.

# Présentation fonctionnelle

L’objectif final de ce projet était de créer un jeu nommé Takuzu, nous devions proposer au minima deux niveaux différents, l’un avec une grille 4x4 et l’autre avec une grille 8x8 pour lesquelles le joueur aurait la possibilité de les remplir. Pour cela nous avons procédé en trois grandes étapes que nous allons développer ci-dessous après une brève présentation des règles :

## Règles du Takuzu

* Entre deux 0 il ne peut y avoir uniquement un 1
* Entre deux 1 il ne peut y avoir uniquement un 0
* Il doit avoir le même nombre de 1 et de 0 sur une ligne ou une colonne
* Une série de deux 0 est entourée par un 1 à ses extrémités
* Une série de deux 1 est entourée par un 0 à ses extrémités
* Il ne peut avoir deux lignes ou deux colonnes identiques

## Laisser un joueur résoudre une grille

Le joueur doit pouvoir résoudre une grille de Takuzu. Cette dernière est constituée uniquement de 1 et de 0 mais lorsqu’elle est présentée au joueur seulement quelques cases sont visibles. Pour cela ont va générer un masque qui servira à savoir quelle case l’ordinateur doit montrer ou non. Revenons à l’utilisateur, il doit pouvoir choisir une case et dire s’il désire la remplir par un 0 ou par un 1. Soit le coup est valide soit il est invalide. Dans le dernier cas, le programme propose à l’utilisateur un indice, pour une grille, afin de la compléter. Il y a deux bons coups à différencier : un coup valide respecte les règles mais n’est pas celui attendu par le programme, a l’inverse un coup correct est la bonne réponse et celle attendue. De plus l’utilisateur possède 3 vies et lorsqu’un mauvais coup est effectué il en perd une.

## Résoudre automatiquement une grille

Dans cette partie on veut que l’ordinateur résolve lui-même la grille en prenant la place de l’utilisateur. Il faut aussi que chaque étape de la résolution soit affichée à l’écran.

## Générer une grille

## Fonctionnalités additionnelles

On a pu rajouter une fonctionnalité au niveau des niveaux justement. On a comme obligation de mettre une grille 4x4 et une 8x8. Nous avons pris la liberté de rajouter une grille 16x16.

Une image contenant texte, équipement électronique, clavier

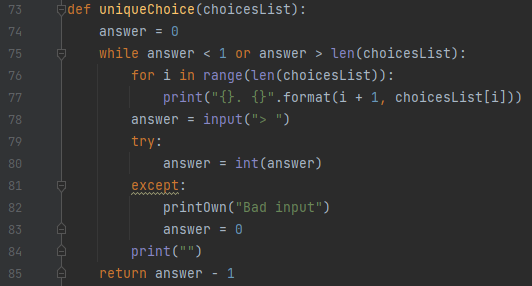
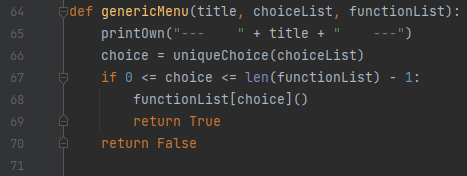
Description générée automatiquement

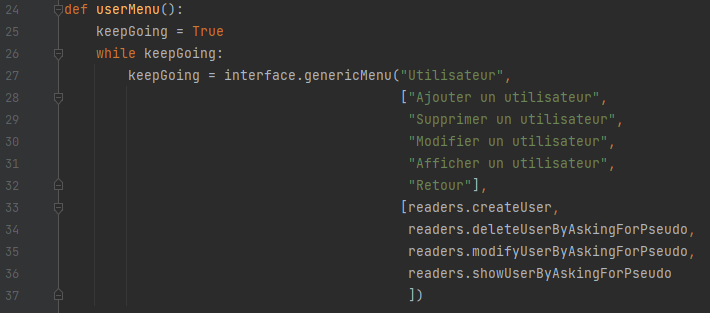
# Présentation technique

Passons maintenant à une présentation plus technique du projet. Nous allons nous focaliser sur quelques-uns des algorithmes de notre code, qui, pour nous, reflètent bien la complexité et les difficultés principales de ce projet. Nous montrerons ensuite la structure que nous avons choisi pour nos données, et enfin nous parlerons des difficultés que nous avons rencontré.

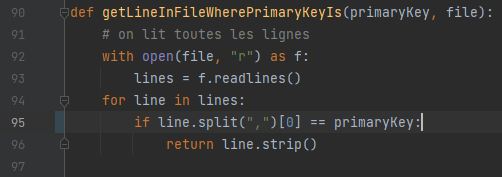
## Description des principaux algorithmes réalisés

### Interface utilisateur

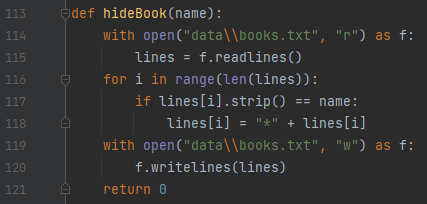
Pour commencer, quelques fonctions permettant de faire facilement faire des choix a l’utilisateur. On peut notamment citer interface.uniqueChoice() qui permet respectivement de faire faire un choix unique parmi une liste de choix possible à l’utilisateur. Cette fonction nous a été utile dans les menus, le choix de livre, de pseudo, et est vraiment une base fondamentale de notre interface. Son fonctionnement est le suivant : une boucle while qui s’exécute tant que la réponse de l’utilisateur n’est pas dans la plage désirée et une structure try qui tente de convertir la réponse de l’utilisateur en Integer. Dès que la réponse est dans la plage valide, on arrête la boucle et on renvoie la réponse a laquelle on soustrait 1, car par convention on affiche les réponses possibles de 1 à n, et non de 0 à n – 1.

Nous avons aussi créé une fonction pour faire des menus facilement. Cette fonction prend un titre, une liste de choix ainsi qu’une liste de fonctions correspondantes à effectuer. De là, on utilise la fonction précédente uniqueChoice, et on effectue l’action associée à ce choix. Si aucune action n’est associée, on sort en retournant False, ce qui permet de faire un choix Retour par exemple. On peut voir ci-après un exemple d’utilisation de la fonction genericMenu.

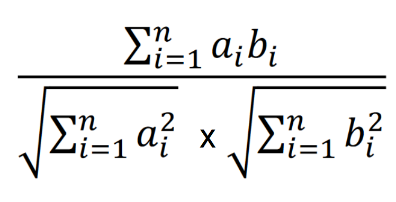
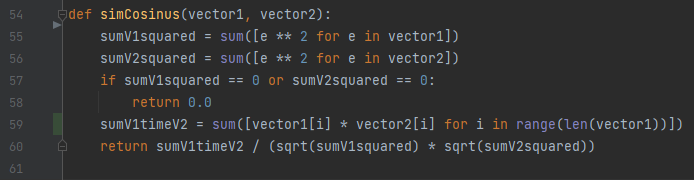
### Gestion des fichiers

Concernant la gestion des fichiers, on peut citer un autre de nos algorithmes, getLineInFileWherePrimaryKeyIs. Derrière ce nom barbare se cache une fonctionnalité pourtant très simple : on donne à cette fonction une chaine de caractère et le chemin d’un fichier, et elle retourne la première ligne de ce fichier qui commence par cette chaine de caractère. Pour cela, on lit d’abord la totalité du fichier, puis pour chaque ligne, on sépare aux virgules (cf. : structure de données) et on regarde si le premier élément correspond à celui qu’on chercher. Si oui, on renvoie la ligne en question en retirant le retour à la ligne finaux.

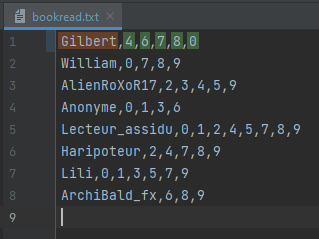
### Gestion des livres supprimés

Les livres supprimés étaient un véritable défi à gérer pour nous, car supprimer un fichier d’une ligne revenais à devoir décaler tous les numéros qui venaient ensuite. Nous avons donc opté pour une autre méthode : ajouter un astérisque devant le nom des livres qu’on souhaite supprimer. Cela permet une gestion beaucoup plus simple des livres car leurs index sont constants, et ainsi on peut avoir une donnée fixe pour les caractériser. C’est ce que va faire la fonction hideBook, en lisant chacune des lignes du fichier book.txt, tester si la valeur de chaque ligne, épurée de ses retours à la ligne de fin, équivaut au nom du livre qu’on souhaite supprimer, et si oui, on ajoute une \* en début de ligne pour cette ligne et finalement on réécrit tout dans le fichier book.txt.

### Calcul des similarité cosinus

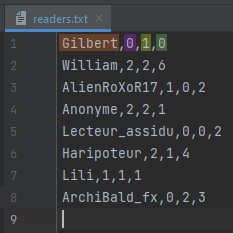
Cette fonction nous a donné du fil à retordre, mais plus a cause de l’aspect mathématique de la chose. Apres quelques tests, nous avons compris comment l’implémenter. J’en parle ici car la fonction en elle-même n’est pas forcément compliquée à comprendre, mais le processus par lequel nous sommes passés pour la faire fonctionner reflète bien la manière totale dont nous avons avancé sur le projet. On a ici utilisé sum() avec des listes en compréhension, car c’est une structure très simple à écrire et il est facile de faire le lien avec ∑ en mathématique. Cependant, après quelques tests, le cas de la division par zéro nous est apparue, et c’est là que nous avons dû ajouter les lignes 57 et 58. Ce processus de test et de correction représente bien la manière dont nous procédions pour réaliser ce projet.

## Structure de données

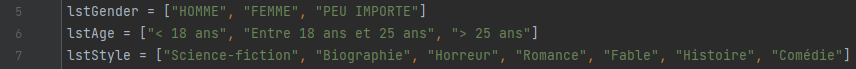
Pour notre structure de donnée, nous avons décidé de faire simple : les données de différents utilisateurs sont stockées sur des lignes différentes. Chaque ligne est composée d’une clé primaire, en début de ligne, qui est une donnée unique, qui ne sera pas en début d’une autre ligne. Ensuite, on insère toutes les données dont on connait le nombre, comme le genre, la tranche d’âge, qui sont des données qui n’ont qu’une seule valeur. Enfin, en fin de ligne, on insère les données qui ont plusieurs éléments potentiels, comme la liste des livres lus, ou les styles préférés. Cela n’a pas été le cas pour nous, mais s’il y avait dû avoir plusieurs données dont la quantité était variable, on aurait dû en amont noter la quantité de chaque afin de comprendre quelle donnée correspond à quoi.

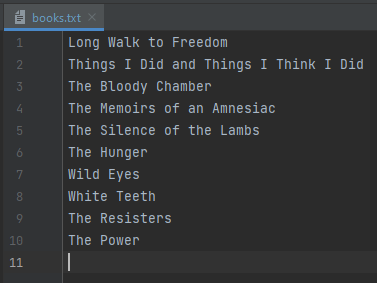
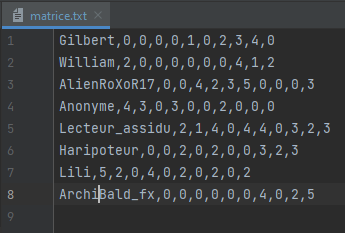
Cela nous donne la structure suivante :

Pour bookread, en orange, le nom d’utilisateur. En vert, les index des livres lus, séparé par une virgule.



Pour **readers,** en orange le nom d’utilisateur, en rose l’index du genre, en jaune l’index de la tranche d’âge, et en vert les styles préférés du lecteur.

Ici, la liste contenant les tranches d’âge, les genres ainsi que les styles de lecture sont définis dans le code de readers.py, comme suit :

La matrice de notation est simplement un fichier avec en début de ligne le nom d’utilisateur, et le reste de la ligne est constitué de la liste des notes de chaque livre séparée par des virgules (zéro quand non noté)

Enfin, la liste des livres est un simple fichier contenant un titre par ligne.

## Difficultés

Nous avons rencontré quelques difficultés lors de la réalisation de ce projet. Pour commencer, bien que la structure ait l’air simple à comprendre lors de notre explication, il a fallu une importante réflexion pour l’établir, notamment concernant la suppression des livres. De plus, lors de la manipulation de fichiers, nous n’étions pas à l’abri de détruire totalement le contenue d’un fichier, et cela nous est d’ailleurs arrivé plusieurs fois. Nous avons donc très vite réalisé l’importance de faire des sauvegardes régulières de nos jeux de données. Certains algorithmes ont aussi été difficiles à mettre en place, notamment une fonction nommée getXthElementIndexByIncreasingOrder qui était un véritable cauchemar, tant à nommer qu’à implémenter. Enfin, réfléchir à son code en essayant d’atomiser chaque fonction au maximum n’était pas une chose facile en premier lieu, mais nous a finalement permis d’écrire du code qu’on avait parfois la joie de réutiliser, et qui, j’en suis sûr, nous sera utile dans bien d’autres projets.

# Présentation des résultats

## Menu principal

Une image contenant texte

Description générée automatiquementIci le menu principal nous permet de faire un choix entre quatre possibilités.

## Jouer

### Partie 1

Une image contenant texte

Description générée automatiquementSi l’utilisateur choisi le choix 1 c’est-à-dire jouer, on lui propose alors de choisir la taille de la grille qu’il désire.

Une image contenant texte

Description générée automatiquementOn voit ici sur les deux captures, que l’utilisateur peut soit cliquer sur l’indice du choix, c’est-à-dire 1) 2) ou 3) Ou bien il a la possibilité de directement saisir la taille qu’il souhaite, c’est-à-dire 4, 8 ou 16.

### Partie 2

Une image contenant texte, équipement électronique, clavier

Description générée automatiquement

Une fois la taille sélectionnée on nous avons trois nouveaux choix qui s’offrent à nous. Ici l’utilisateur a sélectionné l’option « générer un masque »

### Partie 3

Une image contenant texte

Description générée automatiquement

Cela permet ensuite à l’utilisateur de jouer.

Il sélectionne d’abord la ligne puis la colonne dans laquelle il souhaite rentrer une valeur et pour finir il rentre donc la valeur désirée. On peut remarquer que le programme l’informe de la règle non respectée, lui donne ensuite son nombre de vies restantes et donne en plus un indice. Nous avons pu effectuer nos tests à l’aide des grilles qui nous ont été fournis

## Résoudre automatiquement une grille

## **Générer une grille**

# Conclusion

Ce projet nous as permis de coder un projet de A à Z et donc d’utiliser à une importante fréquence le langage C. On a pu se familiariser avec ce dernier et approfondir nos compétences. Le fait d’être en groupe nous as aussi permis d’un peu plus équilibrer nos niveaux qui étaient très différents. La communication au sein de notre groupe était très homogène en passant notamment par discord et GitHub on a pu rester constamment en contact et au même niveau sur le projet.