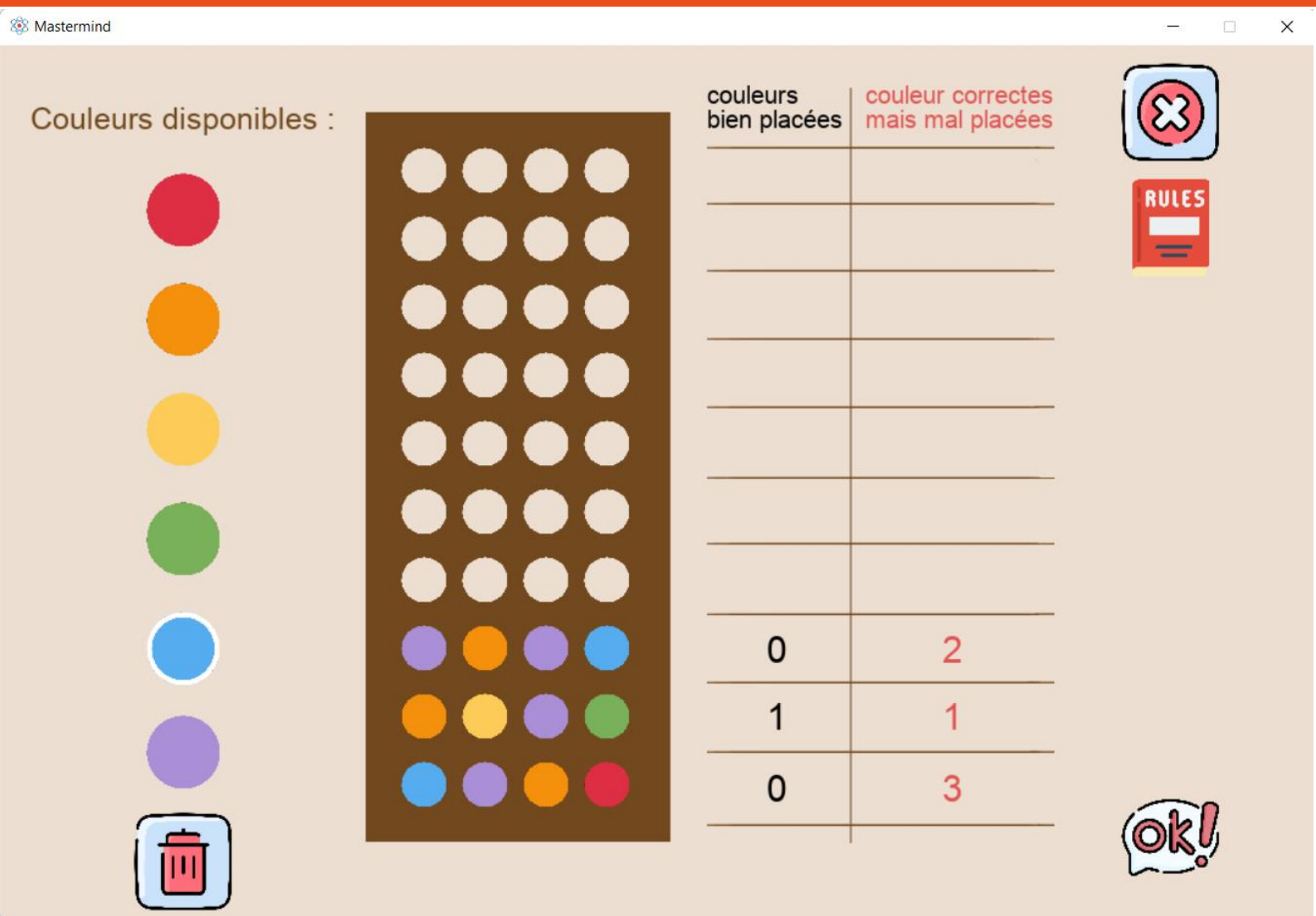
LES TROPHÉES NSI

Édition 2022

DOSSIER DE CANDIDATURE
PRÉSENTATION DU PROJET



Mastermind à 6 couleurs

> PRÉSENTATION GÉNÉRALE :

Dans le cadre de notre cours de NSI, nous devions réaliser un projet libre en python, afin de mettre en pratique ce que nous avons appris. Comme nous voulions nous amuser lors de la réalisation de ce projet, nous avions pensé à réaliser un jeu, avec interface graphique pour rendre l'expérience utilisateur plus agréable et aussi pour surmonter le défi que représentait cette interface graphique en elle-même, vu que nous n'en avions jamais réalisé auparavant. Pour cela, nous avons cherché un jeu que l'on appréciait tous les deux, quelque chose de ludique, pouvant être joué à n'importe quel âge : le Mastermind était donc un très bon choix, du fait de sa complexité à réaliser (notamment les indices que doit donner le programme) et qu'il soit un jeu connu quasiment par tous. De plus, nous tenions à réaliser tout le code en anglais (sauf les commentaires) pour nous immerger encore plus dans le monde professionnel, les programmes étant tous en anglais.

> ORGANISATION DU TRAVAIL :

Ce projet a été réalisé par deux élèves :

- Emma FAYE :
  Chargée de faire l'interface graphique (les interactions du joueur avec la grille pour rentrer les combinaisons de couleurs et les fonctions liées, les boutons…), a dessiné des éléments graphiques (la grille, les boutons, le tableau pour les indices…).

- Nathan CONTI ALUNNO :
  Chargé de faire toute la partie où le programme vérifie la combinaison rentrée par le joueur, donne des indices en fonction de cela et vérifie également si le joueur a gagné ou perdu (lorsque les 10 essais ont été épuisés).

- On a fait ensemble l'emboitement des différentes fonctions, c'est-à-dire l'articulation globale du projet.

Pour ce qui est de l'organisation, nous avons travaillé pendant les vacances de Noël, tous les jours et nous étions en appel sur Discord. Nous nous servions de cette application pour échanger du code, des dossiers, images ou liens menant à des vidéos/sites, et le partage d'écran était très pratique lorsque nous devions montrer quelque chose.

LES ÉTAPES DU PROJET :

Voici une partie du journal de séance que nous avons rédigé lors de la réalisation du projet :

1ère séance en classe : Définir les fonctions dont on avait besoin (nous avions déjà trouvé le thème du projet)

2ème séance en classe : Recherche des documentations et explications sur le module Tkinter, le plus dur était de comprendre comment utiliser ce module.

22/12 : Nous nous sommes rendu compte que Tkinter avait de gros soucis avec la gestion des images, des frames et canvas, donc nous avons repris notre projet à 0 en utilisant cette fois-ci le module Pygame. Comme pour Tkinter, nous avons commencé par essayer de comprendre comment appréhender ce module et toutes les fonctionnalités dont on avait besoin.

23/12 : Création du premier menu « Start » et des boutons « Play » et « Quit ». Cette partie a été assez longue car tout était nouveau pour nous et nous avions du mal à réellement comprendre par nous-même sans utiliser Internet.

24/12 : Création de la classe Button et Text, apprentissage de l'utilisation des classes et de Pygame. On cherchait également comment pouvoir revenir entre les différents menus du jeu (start, rules, game…).

25/12 : Continuation des recherches, menu start terminé, menu rules commencé. Nous avions l'impression de ne pas avancer rapidement à cause de notre inexpérience.

26/11 : Nous devions trouver comment faire pour naviguer entre les différents menus. On ne pouvait pas encore interagir avec les emplacements de la grille car les boutons « empty » n'existaient pas encore et le tableau n'était pas complet. Ensuite on a créé ces boutons « empty » avec 40 variables, pour créer les 40 boutons, ce qui rendait le code assez lourd. Par la suite, on a modifié cette grosse partie en quelques lignes de code, ce qui a été difficile.

Code initial

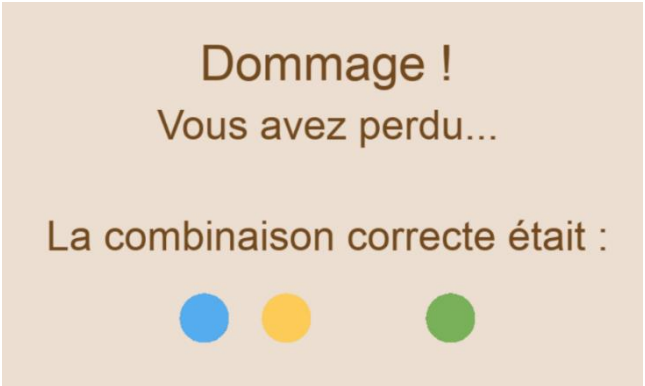
```
# Création des boutons vides
button_line1_1 = Button(347, 606, size_grid_button("pictures/empty.png"))
button_line1_2 = Button(397, 606, size_grid_button("pictures/empty.png"))
button_line1_3 = Button(447, 606, size_grid_button("pictures/empty.png"))
button_line1_4 = Button(497, 606, size_grid_button("pictures/empty.png"))
button_line2_1 = Button(347, 550, size_grid_button("pictures/empty.png"))
button_line2_2 = Button(397, 550, size_grid_button("pictures/empty.png"))
button_line2_3 = Button(447, 550, size_grid_button("pictures/empty.png"))
button_line2_4 = Button(497, 550, size_grid_button("pictures/empty.png"))
button_line3_1 = Button(347, 494, size_grid_button("pictures/empty.png"))
button_line3_2 = Button(397, 494, size_grid_button("pictures/empty.png"))
button_line3_3 = Button(447, 494, size_grid_button("pictures/empty.png"))
button_line3_4 = Button(497, 494, size_grid_button("pictures/empty.png"))
button_line4_1 = Button(347, 438, size_grid_button("pictures/empty.png"))
button_line4_2 = Button(397, 438, size_grid_button("pictures/empty.png"))
button_line4_3 = Button(447, 438, size_grid_button("pictures/empty.png"))
button_line4_4 = Button(497, 438, size_grid_button("pictures/empty.png"))
button_line5_1 = Button(347, 382, size_grid_button("pictures/empty.png"))
button_line5_2 = Button(397, 382, size_grid_button("pictures/empty.png"))
button_line5_3 = Button(447, 382, size_grid_button("pictures/empty.png"))
button_line5_4 = Button(497, 382, size_grid_button("pictures/empty.png"))
button_line6_1 = Button(347, 326, size_grid_button("pictures/empty.png"))
button_line6_2 = Button(397, 326, size_grid_button("pictures/empty.png"))
button_line6_3 = Button(447, 326, size_grid_button("pictures/empty.png"))
button_line6_4 = Button(497, 326, size_grid_button("pictures/empty.png"))
button_line7_1 = Button(347, 269, size_grid_button("pictures/empty.png"))
button_line7_2 = Button(397, 269, size_grid_button("pictures/empty.png"))
button_line7_3 = Button(447, 269, size_grid_button("pictures/empty.png"))
button_line7_4 = Button(497, 269, size_grid_button("pictures/empty.png"))
button_line8_1 = Button(347, 213, size_grid_button("pictures/empty.png"))
button_line8_2 = Button(397, 213, size_grid_button("pictures/empty.png"))
button_line8_3 = Button(447, 213, size_grid_button("pictures/empty.png"))
button_line8_4 = Button(497, 213, size_grid_button("pictures/empty.png"))
button_line9_1 = Button(347, 156, size_grid_button("pictures/empty.png"))
button_line9_2 = Button(397, 156, size_grid_button("pictures/empty.png"))
button_line9_3 = Button(447, 156, size_grid_button("pictures/empty.png"))
button_line9_4 = Button(497, 156, size_grid_button("pictures/empty.png"))
button_line10_1 = Button(347, 100, size_grid_button("pictures/empty.png"))
button_line10_2 = Button(397, 100, size_grid_button("pictures/empty.png"))
button_line10_3 = Button(447, 100, size_grid_button("pictures/empty.png"))
button_line10_4 = Button(497, 100, size_grid_button("pictures/empty.png"))

empty_buttons = [
button_line1_1, button_line1_2, button_line1_3, button_line1_4,
button_line2_1, button_line2_2, button_line2_3, button_line2_4,
button_line3_1, button_line3_2, button_line3_3, button_line3_4,
button_line4_1, button_line4_2, button_line4_3, button_line4_4,
button_line5_1, button_line5_2, button_line5_3, button_line5_4,
```

27/12 : Commencement fonctions Win et Lost et des fonctions pour mettre les couleurs dans la grille. On y a passé beaucoup de temps, et après avoir fixé les erreurs, ces parties-là fonctionnaient quasiment, mais nous ne pouvions pas tester sans avoir codé la partie de la vérification de la combinaison du joueur.

29/12 : Continuation des fonctions évoquées précédemment.

30 et 31/12 : Création de la fonction Comparison_combinations pour comparer la combinaison du joueur et de l'ordinateur.

03 et 04 /01 : Un autre bug à résoudre, en effet, nous avions rajouté la possibilité d'avoir deux fois la même couleur dans la combinaison alors qu'au départ nous avions tout codé dans l'optique de n'avoir qu'une seule fois chaque couleur. Résolution de bugs avec les fonctions Win, Lost et Comparison_combinations.



05/01 : Création de la fonction Show_rules pour afficher les règles durant la partie. Nous avions également découvert un autre problème concernant l'affichage des indices, car ces derniers n'étaient pas corrects, à cause de Comparison_combinations.

07/01 : Derniers ajustements du code, comme mettre tous les commentaires et retravailler la présentation pour qu'elle soit plus soignée et organisée, le projet était fini.

> FONCTIONNEMENT ET OPÉRATIONNALITÉ :

• Notre projet est terminé. Il est en effet opérationnel et répond à nos attentes. Bien évidemment il y a aurait plein de manières de l'améliorer mais ce qu'on voulait absolument faire a été fait.

• Afin de vérifier l'absence de bugs et de les traiter le plus rapidement possible, nous testions chaque fonction ou morceau de fonction indépendamment du reste du code dès sa création. Si tous les tests effectués, dont le but était de tester notre algorithme sur tous les différents cas globaux qu'il pouvait rencontrer, étaient concluants, nous passions à l'algorithme suivant.

Par exemple, lorsque nous traitions les fonctions donnant des indices au joueur, nous affichions dans la console avec un « print » la combinaison secrète afin que l'on sache exactement si le code fonctionnait bien. De plus nous avons effectué des tests réguliers pour vérifier que tout fonctionnait comme on le voulait lorsque l'on a relié les différentes fonctions entre elles.

Enfin, une fois le projet fini, nous l'avons fait tester à nos amis et à notre famille, pour avoir le maximum de retours (et être sûr de l'absence d'un quelconque problème non détecté).

• Au cours de notre projet, nous avons été confrontés à différents problèmes.

Ainsi, par exemple, lorsque l'on appuyait sur certains boutons et qu'on maintenait le curseur appuyé, un autre bouton situé au même endroit sur la page qui devait s'afficher juste après était cliqué automatiquement. Ainsi, on était redirigé sans qu'on le veuille dans une autre section de notre jeu. Lorsque nous avons compris la cause du problème, nous avons tantôt déplacé nos boutons (par exemple sur la fenêtre pour quitter le Mastermind en pleine partie, puisque sinon si on maintenait le curseur appuyé après avoir cliqué sur quitter la partie, elle se relançait automatiquement), tantôt rajouté un laps de temps de 0.5 secondes environ au programme pour que l'on ne soit pas redirigé instantanément.

On a également eu un problème avec la fonction *comparison_combinations*. En effet il s'est avéré difficile de déterminer le nombre de couleurs correctes dans la combinaison du joueur, à partir du moment où l'on a intégré la possibilité qu'il y ait jusqu'à deux fois la même couleur dans la combinaison du programme. En effet, il fallait traiter pour chaque élément s'il était présent zéro, une ou deux fois dans la combinaison de l'ordinateur, et zéro, une ou deux fois dans celle du joueur, sans pour autant compter en double certaines couleurs (ex : si une couleur était présente deux fois dans la combinaison de l'ordinateur mais une fois dans celle du joueur, il ne fallait que cela ne compte que pour une seule bonne couleur). Nous avons donc dû traiter différents cas. Nous comparions les couleurs du joueur à celles de l'ordinateur dans des listes où chaque couleur n'avait qu'une seule occurrence, et nous donnions le nombre de couleurs correctes en fonction du nombre d'occurrences d'une couleur donnée dans la liste du joueur, et dans celle de l'ordinateur.

On a aussi rencontré des difficultés pour gérer le moment où le joueur clique sur une couleur, puis sur un emplacement de la grille pour le remplir. En effet, d'une part, il y a quarante cases différentes dans la grille, et cela apparaissait très compliqué de pouvoir les sélectionner et considérer individuellement, à moins de créer quarante variables différentes. Heureusement, nous avons réussi à obtenir un résultat beaucoup plus optimisé, puisque chaque case s'est vue attribuée de manière automatisée (avec des « boucles for ») un numéro. De plus, nous avons utilisé une variable globale « Colour » correspondant à la couleur sélectionnée.

Grâce à ces deux éléments, nous avions connaissance de la case à remplir, et de la couleur à y mettre : la difficulté était alors surmontée.

> OUVERTURE :

Voici des fonctions qui permettraient d'améliorer notre projet :

Time – Fonction qui permet de chronométrer le joueur et indique le temps que ce dernier a mis pour trouver la réponse dans le menu Win, et Lost s'il a perdu.

Pseudo – Fonction qui permet d'entrer un pseudonyme en début de partie, et permettra de sauvegarder les performances en fonction de chaque joueur (nombre de parties jouées, temps record, nombre d'essais record, nombre de victoires et défaites, taux de réussite…). Un bouton permet d'accéder à ce scoreboard. Si le joueur a déjà joué au jeu, il peut directement sélectionner son pseudonyme qui a été sauvegardé.

Help – Fonction qui permet de révéler l'emplacement d'une couleur (celui qu'il veut) dans la combinaison de l'ordinateur si le joueur n'a toujours pas trouvé la réponse au bout du 7^{e} essai. Cette aide enlèvera deux essais. Si le joueur a eu recours au bouton help, ce sera marqué dans les menus « Win » et « Lost », et aussi dans le scoreboard.

Mode – Fonction qui permet de faire varier la difficulté du jeu.

- Facile : 4 couleurs à deviner
- Moyen : 5 couleurs à deviner, le joueur a 10 minutes pour trouver la réponse
- Difficile : 6 couleurs à deviner, le joueur a 6 minutes pour trouver la réponse et la couleur Grise a été rajoutée
- Personnalisé : tout est personnalisable (nombre de couleurs à deviner : entre 4 et 6, le temps : limite ou non, si une couleur est rajoutée ou non)

On aurait également voulu intégrer des musiques, effets sonores, et un mode multijoueur permettant de jouer en réseau à plusieurs.

• Pour faire connaître notre jeu, on pourrait en faire des gameplays dans des vidéos YouTube, ou même dans des lives Twitch ! On pourrait également faire des affiches (voire des flyers qu'on distribuerait à des gens dans la rue ou dans des boîtes à lettre, pour leur montrer un petit exemple de ce qu'on peut réaliser au bout d'un semestre de cours de NSI). Enfin, on a également pensé à faire des petites affiches qu'on collerait dans notre lycée, et sur lesquelles il y aurait un QR code permettant de télécharger notre jeu : si le jeu plaît, le système du bouche à oreille pourrait alors se révéler plus efficace que des vidéos publicitaires ! On pourrait aussi distribuer ces QR codes lors de la présentation des différentes spécialités pour les élèves de seconde, afin de leur donner un aperçu de ce que l'ont fait.

• Si c'était à refaire on ne changerait pas grand-chose. Il est vrai que nous aurions alors directement débuté notre projet en utilisant pygame, et non tkinter, ce qui nous a fait perdre du temps. Sinon, on peut estimer bien avoir gérer notre temps. On ne savait pas comment créer notre projet, et donc beaucoup de notre temps a été consacrer à se documenter, mais il s'agit bien évidemment d'une étape essentielle. Peut-être aurions-nous pu plus visualiser l'ensemble de notre projet au début, afin de ne pas avoir à abandonner des idées en cours de route, ces dernières se révélant trop longues à mettre en place. Néanmoins, ces changements d'organisation ne nous ont pas du tout gênés, et nous sommes très fiers de notre projet.