

Projet Hitori

Binôme: AHAMMAD Sadib, ABI HANNA DAHER Alexy

Groupe: TDB - TP4

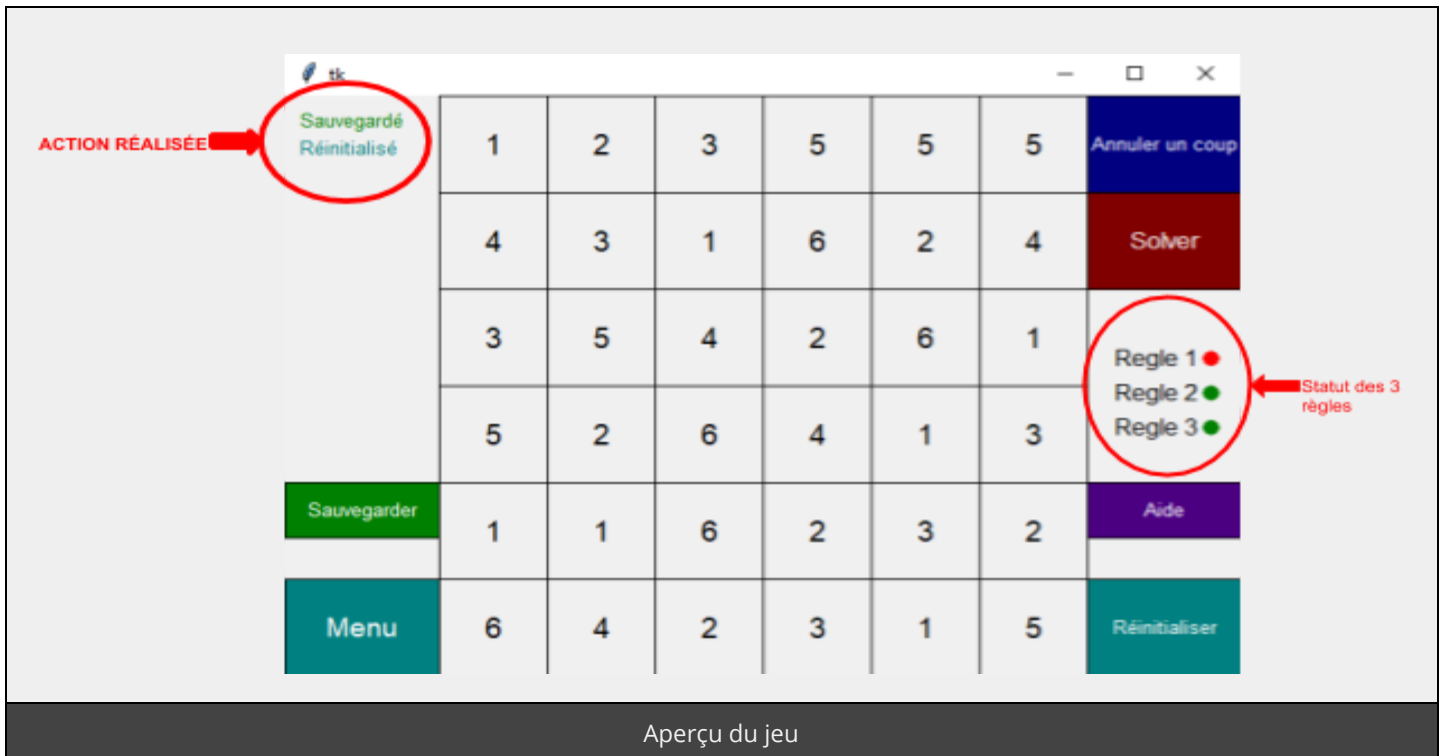
Guide du jeu

Au lancement du fichier 'hitori.py' on a le menu principal. Ce menu principal comporte les items de différentes tailles de grille et un item qui permet de charger votre propre fichier.

Après avoir choisi une grille, on a une nouvelle interface avec une grille chargée et plusieurs boutons qui ont tous une fonction différente.

Sommaire

★ Les parties obligatoires du projet.....	1
○ Les quatre tâches.....	1-3
■ Les fonctions importantes.....	2
■ Tableau de comparaison des deux solveurs.....	3
★ Les améliorations.....	3-4



Les parties obligatoire du projet

→ Tâche 1 :

- ◆ **lire_grille(nom_fichier)** : renvoyant une liste de listes décrivant les valeurs des cellules de la grille.
- ◆ **affiche_grille(grille)** : permet d'afficher la grille dans le terminal.

→ Tâche 2 :

Nous avons trois fonctions qui consistent les règles du jeu :

- ◆ **Règle 1 : sans_conflit(grille, noircies)** : renvoyant True si aucune des cellules visibles de la grille ne contient le même nombre qu'une autre cellule visible située sur la même ligne ou la même colonne, et sinon elle renvoie la liste des cases en conflit.
- ◆ **Règle 2 : sans_voisines_noircies(grille, noircies)** : renvoyant True si aucune cellule noircie n'est voisine d'une autre cellule noircie, et False sinon.
- ◆ **Règle 3 : connexe(grille, noircies)** : renvoyant True si la règle du jeu numéro 3 est respectée, autrement dit si les cellules visibles de la grille forment une seule zone, et False sinon.

→ Tâche 3 :

Interface graphique. Nous avons deux interface graphique qui sont : **un menu principal** et **le jeu**.

Les fonctions importantes:

```
def menu():
    """
    Fonction affiche la fenêtre Menu et renvoie le nom du fichier.
    :return: le nom du fichier.
    """

def retour_menu_button(return_menu, menu_button, ev):
    """
    Détecte si le bouton menu a été cliqué et
    renvoie un booléen.
    """

def retour_menu_message(return_menu, fenetre_x, fenetre_y, option):
    """
    Affiche le message les options pour retourner au
    menu principal. (Option 1)
    Renvoie un dictionnaire de coordonnées pour les
    boutons Oui et Non. (Option 2)
    """

def retour_menu_action(return_menu, backmenu, retour_option, ev):
    """
    Détection du menu retour (Oui ou Non) et
    renvoie un couple booléen.
    """

def affiche_tableau(coordonnees, grille):
    """
    Fonction affiche le tableau et les valeurs dans
    la fenêtre grâce aux coordonnées
    fournis.
    """

def clique_cases_noircies(return_menu, fin_jeu, tab_coord, noircies, ev):
    """
    Détecte la case qui a été cliqué et s'ajoute la
    case à la liste des noircies sinon
    s'enlève si la cases existe déjà dans la noircies.
    """

def coloriage_case(noircies, tab_coord, couleur, grille):
    """
    Fonction qui colorie les cases qui sont dans la
    liste noircies.
    """

def annule_coup(noircies, ev, annuler_button):
    """
    Fonction enlève le dernier couple de la liste
    noircies.
    """

def reinitialiser(nom_fichier, noircies, ev, resetup_button, resetup_msg):
    """
    Fonction enlève toutes les cases noircies ainsi dans le fichier 'log.txt' où les cases noircies ont
    été sauvegardé.
    """
```

→ Tâche 4 :

Recherche de solutions. Nous avons le solveur graphique et il est orienté par les conflits.

```
def resoudre(grille, noircies, tab_coord, case__=0):
    """
    Fonction résout la grille et renvoie une liste des cases noircies.
    """
```

Fichier : functions.py

Tableau comparant les différents temps mis par les deux solveurs (régulier et amélioré) sur les cinq grilles:

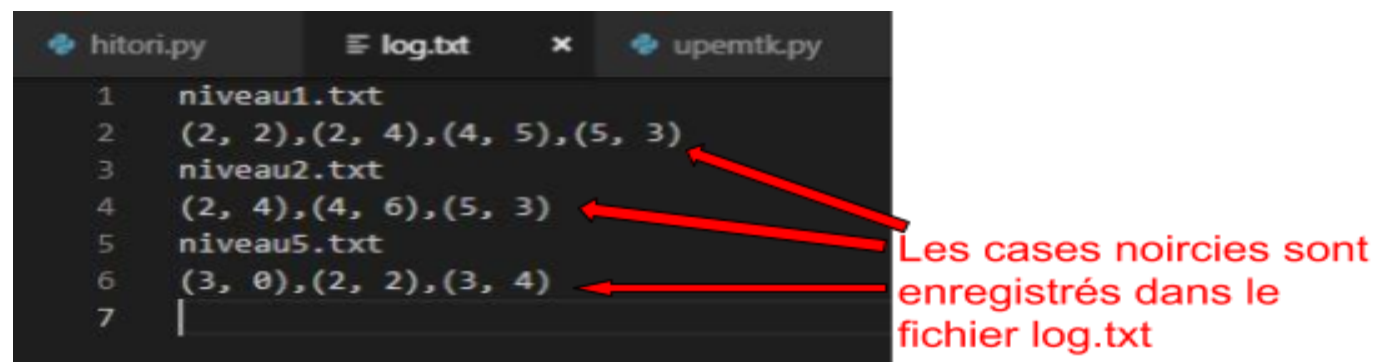
Grilles\Solveur	Régulier	Amélioré
Niveau 1	0.1023	0.0342
Niveau 2	0.3255	0.1998
Niveau 3	0.4258	0.0406
Niveau 4	0.3042	0.1581
Niveau 5	0.1142	0.0573

On remarque que le solveur amélioré prend significativement moins de temps que le solveur régulier.

Les améliorations

I. Sauvegarde d'une partie en cours.

```
def save(nom_fichier, noircies, ev, save_button, save_msg):
    """
    Fonction sauvegarde les cases noircies dans le fichier 'log.txt'.
    """
```

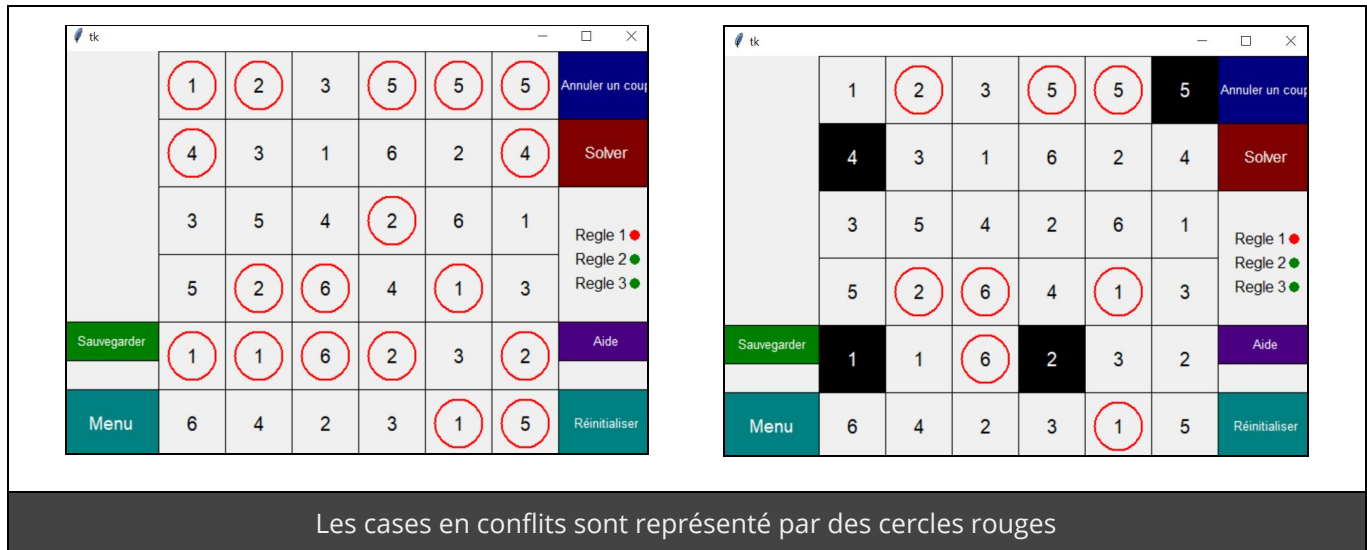


```
1  niveau1.txt
2  (2, 2),(2, 4),(4, 5),(5, 3)
3  niveau2.txt
4  (2, 4),(4, 6),(5, 3)
5  niveau5.txt
6  (3, 0),(2, 2),(3, 4)
7  |
```

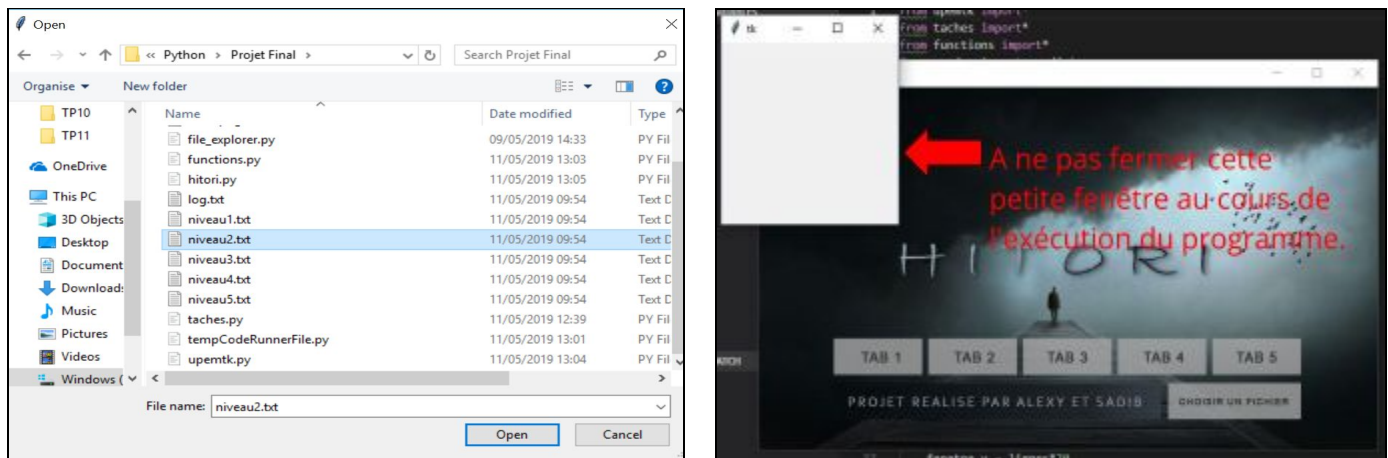
Les cases noircies sont enregistrés dans le fichier log.txt

Fichier : log.txt

II. Fonction aide, pour afficher les cases en conflits.



III. Choisir un fichier de son choix.



Changement dans le fichier upemtk.py à la ligne 77: `tk.Tk()` est remplacé par `tk.Toplevel()` pour permettre d'ouvrir plusieurs fenêtre simultanément mais qui ouvre aussi une petite fenêtre vide qu'on peut pas se débarrasser.

76		<code># root Tk object</code>	76		<code># root Tk object</code>
77		<code>self.root = tk.Tk()</code>	77		<code>self.root = tk.Toplevel()</code>

Fichier : upemtk.py