DOCUMENTATION DU CODE

Table des matières

1. Présentation du jeu

Hexologic est un jeu de puzzle reposant sur la logique et l'opération d'addition. Chaque niveau possède une grille. Cette grille est constituée d'un ensemble d'hexagones regroupé sous forme de lignes liées entre elles avec trois directions possibles et possédants à leurs extrémités un triangle marqué qui indique la direction de la ligne. Le joueur doit combiner 1 à 3 points à l'intérieur des hexagones vides de sorte que leur somme corresponde au nombre donné dans le triangle marqué.

La principale difficulté du jeu réside dans le fait qu'un hexagone peut appartenir à plusieurs lignes. Dans ce cas, il faudra que le nombre de points affichés par cet hexagone soit adapté aux besoins de toutes les lignes qui le partagent.

1. Architecture du code

Le code a été organisé de la manière suivante avec des importations entre les différents fichiers :

- **Fichier app.py** : qui permet de lancer le jeu
- **Dossier fonctionUtilitaire** : qui contient toutes les fonctions nécessaires dans différents fichiers
  - **Convention.py**
  - **Canevas.py**
  - **Interface.py**
  - **Interactions.py**
  - **Generateur_niveaux.py**
  - **Solveur.py**
- **Dossier Images** : contient les backgrounds du jeu
- **Dossier son** : contient le son du jeu

2. Structures de Données utilisée

Pour représenter l'état du jeu nous utilisons comme structure de données une matrice M contenant des lignes L d'hexagone. Chaque ligne est une liste qui contient elle-même des listes d'hexagone Hexa = [id, 0, 0] : id de l'hexagone créé, nombre de points, couleur. Après création de la matrice on a :

```
L  [[1, 0, 0], [2, 0, 0], [3, 0, 0], [4, 0, 0], [5, 0, 0], [6, 0, 0], [7, 0, 0], [8, 0, 0]]
L  [[9, 0, 0], [10, 0, 0], [11, 0, 0], [12, 0, 0], [13, 0, 0], [14, 0, 0], [15, 0, 0], [16, 0, 0]]
L  [[17, 0, 0], [18, 0, 0], [19, 0, 0], [20, 0, 0], [21, 0, 0], [22, 0, 0], [23, 0, 0], [24, 0, 0]]
L  [[25, 0, 0], [26, 0, 0], [27, 0, 0], [28, 0, 0], [29, 0, 0], [30, 0, 0], [31, 0, 0], [32, 0, 0]]
L  [[33, 0, 0], [34, 0, 0], [35, 0, 0], [36, 0, 0], [37, 0, 0], [38, 0, 0], [39, 0, 0], [40, 0, 0]]
```

Cette matrice est mise à jour après chaque clic et permet de représenter l'Etat du jeu. Les autres fonctions à lancer au clic s'exécute en parcourant la matrice M.

Mis à part cela, nous utilisons aussi une matrice M_NIVEAU pour représenter la grille de chaque niveau. M_NIVEAU contient les listes L de chaque ligne de la grille Les lignes de la grille sont représentées par des listes L contenant :

- L'état L [0] : validée 1 ou non 0
- La direction : L [1] avec :
  - 0 : l'horizontale -------
  - 1 : une diagonale en slash /
  - 2 : une diagonale en anti slash \
- La pondération : L [2]
- Les id des hexagones de la ligne

Exemple : M_NIVEAU = [[0, 0, 2, 21, 22], [0, 1, 4, 21,28], [0, 2, 4, 22, 30]]

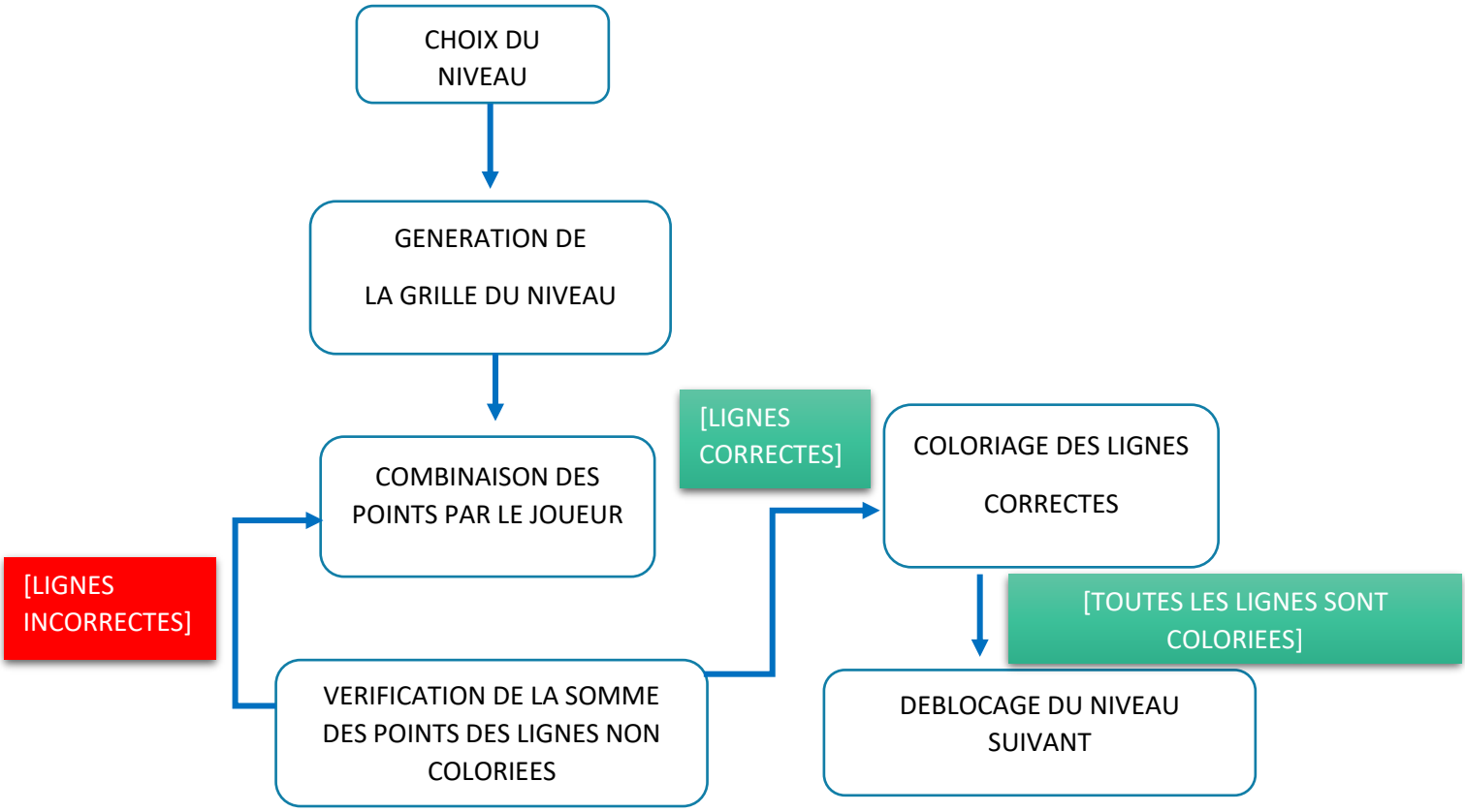
3. Description Partie Joueur

L'utilisateur peut interagir avec le jeu et accéder aux différentes fonctionnalités par le biais des boutons :

- Jouer : qui permet d'afficher la grille du niveau et démarrer le jeu
- Niveau Précédent : qui lui permet de revenir à un niveau antérieur
- Niveau suivant : qui lui permet d'aller au niveau suivant. Mais accessible uniquement si la grille du niveau en cours est totalement validée
- Son / Couper : qui lui permet de lancer ou de couper la musique du jeu

Le joueur peut également générer des points dans les hexagones au clic pour valider la grille.

Voici l'enchaînement logique du jeu :



Le joueur choisit un niveau et le programme génère et affiche la grille correspondant à ce niveau. A chaque clic du joueur dans un hexagone, le programme met à jour les structures de données qui représentent l'état du jeu : la matrice M des hexagones et la matrice M_NIVEAU de la grille du niveau. Cette mise à jour concerne le nombre de points de l'hexagone (0, 1 ou 2), sa couleur (bleu clair ou foncé) et l'état (Validée 1 ou non 0) de la ligne dans laquelle se trouve cet hexagone. Une vérification est également lancée à chaque clic pour voir si la ligne de l'hexagone est validée et si toute la grille est validée. Lorsque toute la grille est validée, on débloque le niveau suivant.

4. Description Partie Code

La partie Code est subdivisée en différents fichiers ayant chacun un rôle bien défini.

4.1. Fichier app.py

Ce fichier importe le contenu de tous les autres fichiers et permet de lancer le jeu grâce à la fonction jouer. Il contient les :

- Bouton Jouer : pour lancer le jeu
- Boutons Niveau Précédent et Niveau Suivant pour la navigation entre les différents niveaux du jeu

- Boutons lancer et couper son : qui permettent de jouer la musique ou de l'arrêter

4.2. Fichier Convention.py

Ce fichier contient les conventions pour la rédaction d'un code homogène par les différents membres du groupe. Il contient également quelques spécifications sur les structures de données les plus importantes et les plus récurrentes.

```
CONSTANTES :
-> Toutes les constantes sont écrites en majuscule
> Les constantes sont définies au tout début du fichier après les importations

PROGRAMME PRINCIPAL :
-> Il contient la création des canevas
-> Il est défini à a fin des fichiers

FONCTIONS :
-> Le nom de chaque fonction commence par une lettre majuscule
-> On utilise les underscores (tiret du 8) quand le nom est composé


COULEUR : = ["#318CE7", "#689EF9", "#0C306B"]
0 -> #318CE7 : COULEUR du canevas
1 -> Blue clair #689EF9 : l'hexagone est affiché mais contient 0 point
2 -> Blue foncé #0C306B : l'hexagone est affiché et contient des points

HEXAGONES :
-> Chaque hexagone est représenté par une liste : Hexa = [id, 0, 0]
-> La liste contient l'Id de l'hexagone Hexa [0]
                Le nombre de point qu'il contient Hexa [1]
                Et sa COULEUR Hexa [2]
-> On ne peut générer plus de 3 points dans un hexagone

MATRICE M D'HEXAGONES :
-> contient des lignes représentées par L
```

```
-> Chaque L contient des hexagones L = [[id1, 0, 0], [id2, 0, 0]]
-> on a M = [[[id1, 0, 0], [id2, 0, 0]]
          [[id3, 0, 0], [id4, 0, 0]]]
LIGNE GRILLE
Les lignes de la grille sont représentées par des listes L contenant :
-> L'état L [0] : validée 1 ou non 0
-> La direction : L [1]
    Avec : *** 0 : l'horizontale -------
            ** 1 : une diagonale en slash /
            ** 2 : une diagonale en anti slash \
-> la pondération : L [2]
-> les id des hexagones de la ligne

GRILLE DE NIVEAU :
La grille de chaque niveau est représentée par une matrice M_NIVEAU
Contenant les listes L de chaque ligne de la grille
```

4.3. Fichier Canevas.py

Ce fichier contient toutes les variables globales communes à tous les autres fichiers pour éviter les problèmes liés aux importations circulaires. Il est donc importé dans les autres fichiers en question. C'est ici qu'on crée le Canevas et on n'importe que le module Tkinter.

Les variables globales communes sont :

```
X0 = 150     # Initialisation du centre du premier hexagone du canevas
Y0 = 100
C = 50              # coté d'un hexagone
COULEUR = ["#318CE7", "#689EF9", "#0C306B"]
CONTOUR = "white"
H = (C/2) *(3) **(1/2) # calcul hauteur d'un hexagone
N_HEXA = 8              # Nbre d'hexagones sur une ligne
N_LIGNES = 5                    # Nbre de ligne
M = []              # Matrice des hexagones
L = []              # Liste des hexagones d'une ligne de la matrice
M_NIVEAU = [] # représente la grille de chaque niveau
COL = 0    # colonne et ligne de l'emplacement d'un hexagone dans la matrice M
LINE = 0
```

Ce fichier contient aussi la fonction Find_Hexa qui retourne la position d'un hexagone dans la matrice M des hexagones à partir de l'id passé en paramètre. Cette fonction est utilisée dans le fichier interactions.py et Solveur.py pour la mise à jour et la vérification des sommes enfin de palier au problème de complexité des nombreux parcours de matrice.

Exemple : LINE, COL = Find_Hexa (30) -> LINE, COL = 3, 4 soit M [3][4]

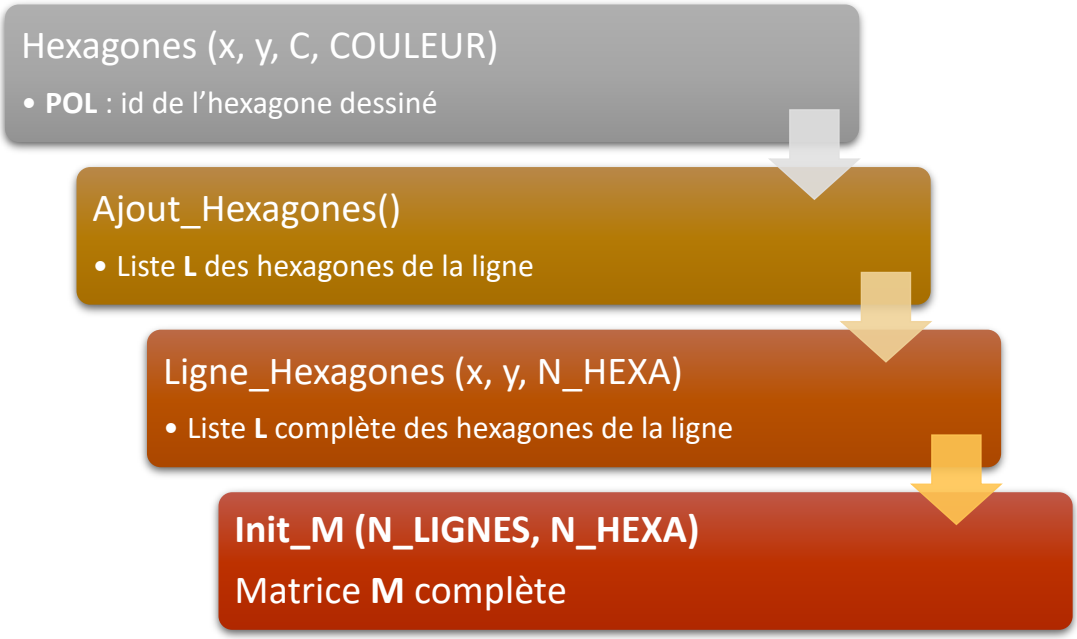
4.4. Fichier Interface.py

Ce fichier contient toutes les fonctions relatives à la création et le dessin des hexagones, ainsi que l'initialisation de l'une des structures de donnée principales du jeu à savoir la matrice M des hexagones.

On y a défini une seule variable globale POL qui sert à récupérer l'id lors du dessin d'un hexagone, les autres variables globales utilisées ayant déjà été initialisées dans le fichier Canevas.py.

Fonctions	Rôle	Entrées	Sorties
Init_M (N_LIGNES, N_HEXA)	Permet d'initialiser la matrice des hexagones en bouclant sur la fonction Ligne_Hexagones	- **N_LIGNES** : Nombre de ligne - **N_HEXA** : Nombre d'hexagones de chaque ligne	Matrice **M** complète
Ligne_Hexagones (x, y, N_HEXA)	Permet de créer des lignes d'hexagones en bouclant sur la fonction Ajout_Hexagone	- **x, y** : Centre du premier hexagone de la ligne - **N_HEXA** : Nbre d'hexagone sur la ligne	Liste **L** complète des hexagones de la ligne
Ajout_Hexagones ()	Permet d'ajouter un hexagone à une ligne	Aucune	Liste **L** des hexagones de la ligne
Hexagones (x, y, C, COULEUR)	Permet de dessiner un hexagone	- **x, y** : Coordonnées du centre - **C** : côté de l'hexagone - **COULEUR**	**POL** : id de l'hexagone dessiné

Organigramme :



4.5. Interactions.py

Ce fichier contient toutes les fonctions qui gèrent les interactions avec l'interface. Notamment le clic, la mise à jour de l'état du jeu à travers les matrices M et M_NIVEAU, le changement de couleur, le dessin des points et la vérification de la validité de la grille au parcours de la matrice à jour.

On utilise ici les variables globales suivantes :

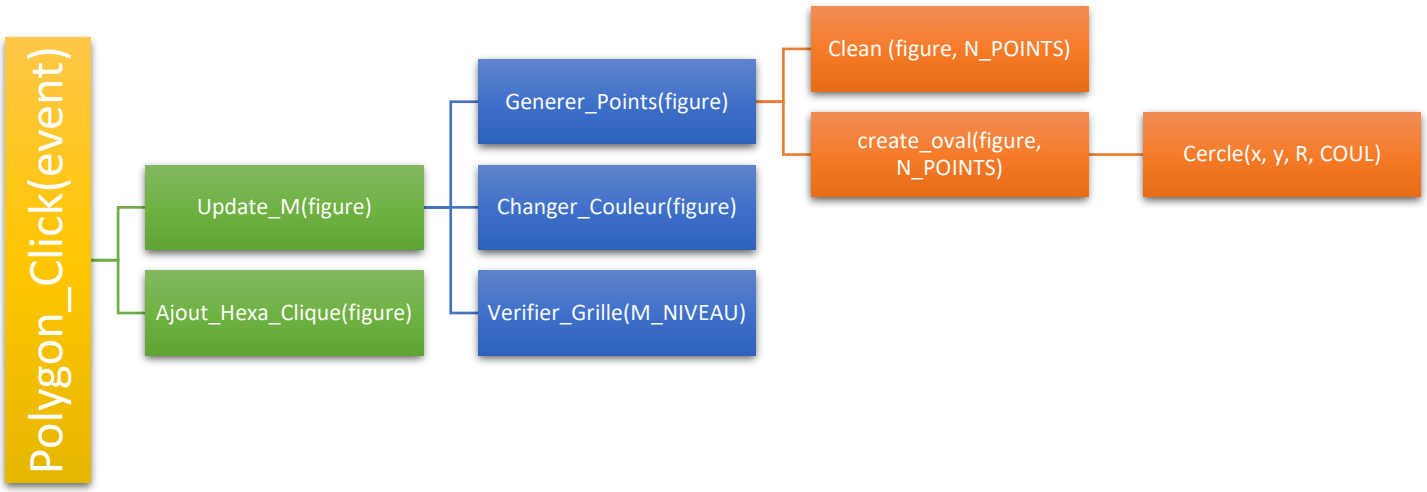
```
R = 10          # rayon des cercles dessinés dans les hexagones
COUL = "white » # couleur des cercles dessinés dans les hexagones
L_CLIQUE = [] # liste des hexagones où on a déjà cliqué
                # il contient l'id de l'hexagone et des points qui se trouv
ent dedans L_CLIQUE = [[figure, points]]
```

Fonctions	Rôle	Entrées	Sorties
Polygon_Click (event)	Lance la mise à jour de la matrice, le changement de couleur, la génération des points au clic et la vérification de la grille	Au clic (event)	Aucune

Ajout_Hexa_Clique (figure)	Parcours L_CLIQUE et Ajoute l'hexagone dans la liste des cliqués s'il n'y est pas encore	**Figure** : Id de l'hexagone où on a cliqué	**L_CLIQUE :** Liste des hexagones où on a déjà cliqué à jour
Update (figure)	Mets à jour la matrice des hexagones (nbre de pts et couleur) au clic	**Figure** : Id de l'hexagone où on a cliqué	**M :** Matrice des hexagones mis à jour
Changer_Couleur (figure)	Permet de changer la couleur des hexagones à partir d'un parcours de la matrice M	**Figure** : Id de l'hexagone où on a cliqué	Aucune
Valider(figure)	Permet de changer le contour d'un hexagone lorsque sa ligne est validée	**Figure** : Id de l'hexagone où on a cliqué	Aucune
Generer_Points (figure)	Génère les points dans les hexagones. Il appelle la fonction clean puis à la fonction create_oval	**Figure** : Id de l'hexagone où on a cliqué	**L_CLIQUE**
Clean (figure, N_POINTS)	Efface le contenu d'un hexagone (les points générés) en parcourant L_CLIQUE avant de dessiner	**- figure** : Id de l'hexagone où on a cliqué **- N_POINTS :** nombre de points que contient l'hexagone	**L_CLIQUE**
Create_oval (figure, N_POINTS)	Dessine et positionne les points suivant le nombre de points à dessiner dans un hexagone. Il met aussi à jour L_CLIQUE en y ajoutant les id des points dessinés dans chaque hexagone	**- figure** : Id de l'hexagone où on a cliqué **- N_POINTS :** nombre de points à dessiner	**L_CLIQUE :** liste des hexagones cliqués mis à jour avec les id des points

Cercle (x, y, R, COUL)	Dessine un seul point à partir des coordonnées du centre et du rayon	- **x, y** : Coordonnées du centre - **R** : rayon du cercle - **COUL :** couleur	**C :** id du cercle / point dessiné

Organigramme :



4.6. Fichier Generateur_niveaux.py

Ce fichier contient toutes les fonctions relatives au générateur de niveau statiques (création et dessin de la grille). On y importe les fichiers Canevas.py, Interface.py et Solveur.py. Il contient les variables globales suivantes :

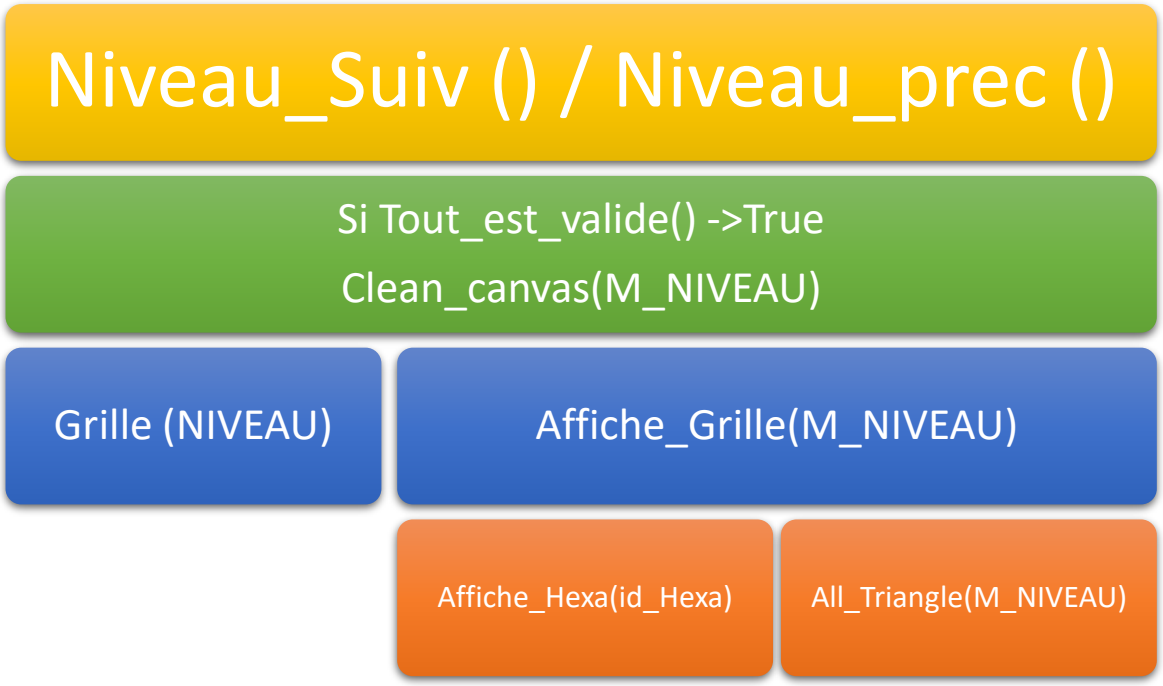
- NUM_NIV : qui représente le numéro du niveau, on s'en sert pour passer au niveau suivant ou précédent.
- NIVEAU : une liste contenant tous les niveaux statiques du jeu
- LIMITE = len (NIVEAU) : nombre total de niveaux

Description de l'algorithme : Lorsque le joueur fini de remplir la grille et que toutes les ligne sont vertes, il a possibilité de solliciter le niveau suivant ou le

précédent. Au clic sur l'un de ces boutons la fonction **Niveau_Suiv** ou **Niveau_Prec** est appelée. Ces fonctions appellent la fonction **Tout_est_valide** du solveur pour vérifier si toute la grille est valide. Si c'est le cas, le numéro du niveau est incrémenté ou décrémenté puis le Canvas est nettoyé par la fonction **Clean_Canvas**. Lorsque le joueur clique sur jouer, la fonction jouer appelle les fonctions **Grille** (pour générer la grille du niveau) et **affiche_grille** (pour afficher la grille en question). Et le jeu recommence. Notons que la fonction Grille

Fonctions	Rôle	Entrées	Sorties
Grille (NIVEAU)	Récupère et initialise la grille d'un niveau	**NIVEAU** : Numéro du niveau	**M_NIVEAU** : la matrice représentant le niveau
Niveau_Suiv ()	Permet de passer au niveau suivant lorsque toute la grille est validée	**Aucune**	**NUM_NIV** : Numéro du prochain niveau à jouer
Niveau_Prec ()	Permet de revenir au niveau précédent	**Aucune**	**NUM_NIV** : Numéro du prochain niveau à jouer
Clean_Canvas (M_NIVEAU)	Permet d'effacer les éléments du Canvas pour un nouveau niveau	**M_NIVEAU** : matrice représentant le niveau	**M** : Matrice des hexagones
Affiche_Hexa(id_Hexa)	Affiche un hexagone à partir de son id	**id_Hexa** : id de l'hexagone à afficher	**M** : Matrice des hexagones à jour
Affiche_Grille(M_NIVEAU)	Affiche la grille d'un niveau donné	**M_NIVEAU** : matrice représentant le niveau	**M** : Matrice des hexagones à jour
All_Triangle(M_NIVEAU)	Dessine tous les triangles nécessaires de la grille	**M_NIVEAU** : matrice représentant le niveau	**Aucun**
Triangle (direction, pondération, last_hexa)	Dessine un triangle suivant la direction de la ligne, sa pondération et	- **direction** : de la ligne - sa **pondération**	**Aucun**

	les coordonnées de son dernier hexagone	- **last_hexa** : id du dernier hexagone de la ligne	



4.7. Fichier Niveaux_dynamiques.py

Ce fichier contient toutes les fonctions relatives au générateur de niveaux dynamiques (création et dessin de la grille). On y importe les fichiers Canevas.py, Interface.py et Solveur.py. Il contient la variable globale L_NIVEAU représentant un niveau.

Description de l'algorithme : Ici, chaque niveau est représenté par :

- la liste L_NIVEAU contenant le numéro du niveau et les paramètres de construction de la matrice M qui lui est associée L_NIVEAU = [Num, M_col, M_lines]

- la matrice M_NIVEAU qui représente la grille du niveau

On construit dans un premier temps la matrice M, puis la matrice M_NIVEAU grâce aux fonctions Init_Matrice_M(L_NIVEAU) et Init_M_Niveau(L_NIVEAU). On complète ensuite la construction de M_NIVEAU en ajoutant les hexagones grâce à la fonction Init_Gril_Lines(nb_Hexa, k) qui détermine les id des hexagones et les

ajoute suivant la direction de la ligne. On prévoit une fonction pour déterminer la pondération de chaque ligne en testant des combinaisons de points jusqu'à trouver des combinaisons gagnantes. Ainsi le niveau est solvable et peut être retenu.

Fonctions	Rôle	Entrées	Sorties
Init_Matrice_M(L_NIVEAU)	Détermine les paramètres de construction de la matrice M d'hexagone rattaché à un niveau puis l'initialise M = Init_M (M_col, M_lines)	**L_NIVEAU** : liste d'un nveau	**L_NIVEAU** complet **M** initialisé
Init_M_Niveau(L_NIVEAU)	Permet d'initialiser la grille représentative d'un niveau	**L_NIVEAU** : liste représentative du niveau	**M_NIVEAU** initialisé sans les hexagones pour l'instant
Init_Gril_Lines (nb_Hexa, k)	Détermine les hexagones d'une ligne de la grille	**Nb_Hexa :** nombre d'hexagones que la ligne doit contenir **K :** la direction de la ligne	**M_NIVEAU** complet
CreateLH (nb_Hexa, id_Depart)	Permet de créer une ligne Horizontale	**Nb_Hexa :** nombre d'hexagones que la ligne doit contenir **id_Depart :** l'hexagone d'où commence la ligne	**L :** La ligne construite
CreateLS (nb_Hexa, id_Depart)	Permet de créer une ligne en slash	**Nb_Hexa :** nombre d'hexagones que la ligne doit contenir **Id_Depart :** l'hexagone d'où commence la ligne	**L :** La ligne construite
CreateLAS (nb_Hexa, id_Depart)	Permet de créer une ligne en anti Slash	**Nb_Hexa :** nombre d'hexagones que la ligne doit contenir **id_Depart :** l'hexagone d'où commence la ligne	**L :** La ligne construite

4.8. Fichier Solveur.py

Ce fichier sert à contenir toutes les fonctions relatives au solveur. On y importe les fichiers Canevas.py et Interactions.py. La seule variable globale utilisée ici est la variable M_NIVEAU qui représente la grille de chaque niveau.

Description de l'algorithme : A chaque clic, on vérifie la grille grâce à la fonction **Verifier_Grille** qui fait elle-même appel à la fonction **Verifier_Ligne** pour vérifier chaque ligne de la grille. Vérifier ligne calcul la somme totale de toute la ligne et la compare avec la pondération de départ de la ligne. Lorsque la correspondance est bonne, on valide la ligne en passant son état de 0 à 1 et inversement de 1 à 0 lorsque la correspondance n'est pas vérifiée. La fonction **Tout_est_valide** est appelée pour confirmer la validité de toute la grille en vérifiant que toutes les lignes ont été validées (ont été passé à l'état 1).

Fonctions	Rôle	Entrées	Sorties
Verifier_Grille(M_NIVEAU)	Vérifie si toute la grille est validée	**M_NIVEAU** : la matrice représentant le niveau	**Aucune**
Verifier_Ligne(L)	Vérifie la correspondance entre la pondération d'une ligne et le nombre de points qu'elle contient, puis appelle les fonctions de validation	**L** : la liste représentant une ligne de la grille	**L** à jour
Tout_est_valide(M_NIVEAU)	Permet de vérifier si toute la grille est validée	**M_NIVEAU** : la matrice représentant le niveau	**True** ou **False** selon que toute la grille est validée ou non