



Le memory ou jeu de paire

I- Présentation

Définition wikipédia :

https://fr.wikipedia.org/wiki/Jeu_de_paires

Le jeu se compose de paires de cartes portant des illustrations identiques. L'ensemble des cartes est mélangé, puis étalé face contre table. À son tour, chaque joueur retourne deux cartes de son choix. S'il découvre deux cartes identiques, il les ramasse et les conserve, ce qui lui permet de rejouer. Si les cartes ne sont pas identiques, il les retourne faces cachées à leur emplacement de départ.

Le jeu se termine quand toutes les paires de cartes ont été découvertes et ramassées. Le gagnant est le joueur qui possède le plus de paires.

II- L'interface graphique

On vous fournit une interface graphique à travers la classe **GUImemory** que vous pouvez importer depuis le module fourni. Si le répertoire **GUI_memory** est dans le même répertoire que votre fichier : :

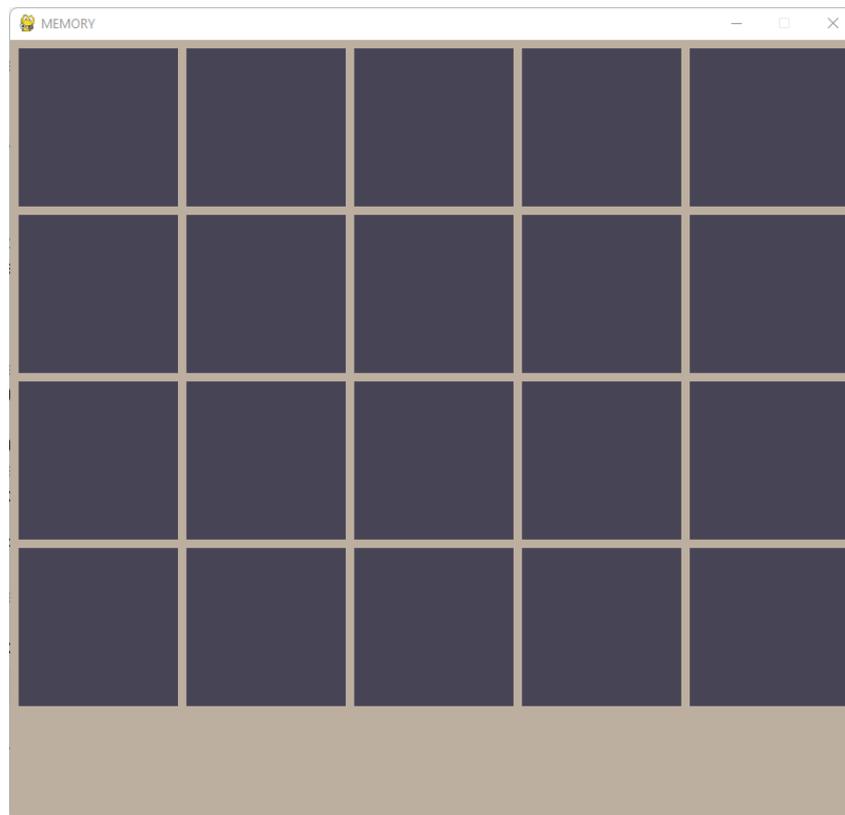
```
from GUI_memory.guiMemory import GUImemory
```

Constructeur de la classe :

- Attribut public : **aucun**.
- Le constructeur prend en paramètre x, y respectivement les nombres de case en abscisse et en ordonnée. Il permet de créer un objet graphique vierge.

Par exemple pour une grille de 5 par 4 :

`GUI = GUImemory(5, 4)`



- Les méthodes :
 - **refresh(g, txt)** : Cette méthode rafraichie l'affichage du jeu conformément à la grille passée en argument.
 - **g** est une liste de nb listes (ou n'importe quel type indexable contenant des éléments indexables). Par exemple, dans une grille de 5*4 :
 - `g[0][0]` est la case en haut à gauche ;
 - `g[3][4]` est la case en bas à droite.
 - 0 -> case vide
 - 1 -> image 1
 - 2 -> image 2
 - ...
 - 20 -> image 20 !! attention c'est la dernière.

Demander l'image 21 lève une exception. Les images sont issues du fichier :

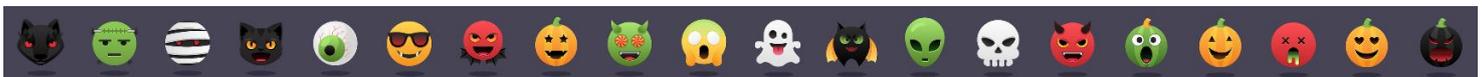


Image libre : <https://pixabay.com/fr/vectors/halloween-emoji-%c3%a9motic%c3%b4ne-marrant-4576764/>

- **txt** est un texte affiché à destination du joueur, sous la grille. Ce paramètre est optionnel.

Par exemple :

```
grille = [[1 + i + j for i in range(5)] for j in range(4)]  
GUI.refresh(grille, '20 images différentes')
```



- **gagne(txt)** : Cette méthode permet d'afficher un message de victoire qui indique le score final plein écran noir sur fond vert.

Par exemple :

```
GUI.gagne('en 18 coups !!')
```



- **waitClick()** : Cette méthode attend l'action d'un joueur. Elle gère trois types d'actions :
 - **demande fermeture de la fenêtre** : fermeture propre de la fenêtre pygame et fin du programme python.
 - **click sur la fenêtre** : retourne un tuple contenant les numéros (x, y) de la case choisie.
 - **appui sur des touches spéciales** :
 - fleche RIGHT : retourne '_R';
 - fleche LEFT : retourne '_L';
 - fleche DOWN : retourne '_D';
 - fleche UP : retourne '_U';
 - touche BACKSPACE : retourne '_B';
 - touche RETURN : retourne '_E';
 - touche ESCAPE : retourne '_S';
 - **appui sur une autre touche du clavier** : retourne le caractère unicode correspondant.

Attention, une fois exécutée, on ne peut sortir de cette méthode que par l'une de ces quatre actions.

III- Information professeur

La méthode `refresh(grille, txt)` attend par défaut une structure grille de type liste de liste auquel elle accède par la syntaxe :

`grille[y][x]`

où `x` et `y` deux entiers valant 0 pour la case en haut à gauche de la grille.

Cette opération est réalisée par une primitive nommée `_getValue`.

```
def _getValue(self, grille, x, y):
    """
    Une primitive pour accéder a une valeur dans grille.

    patrameters
    -----
    grille : list
        Par défaut, elle est adaptée a une structure type liste de liste.
        A redéfinir si grille a un autre format
    x : int
    y : int
        >= 0. coordonnée de la case : 0, 0 en haut a gauche
    """
    value = grille[y][x]

    assert value <= 20, "Il n'y a que 20 figures disponibles dans cette GUI"
    return value
```

Si vous souhaitez adapter la GUI à un autre type de structure, vous pouvez, au choix :

- Modifier cette méthode directement dans le fichier `guiMemory.py`
- Redéfinir la méthode en créant une classe héritée. A faire par le professeur (pas au programme de NSI).

Par exemple si votre structure est un dictionnaire où les clés sont les lettres 'A', 'B', ... correspondant à chaque ligne représentée par une liste :

```
1 from GUI_memory.guiMemory import GUImemory
2
3 class GUImemory_2(GUImemory):
4     def _getValue(self, grille, x, y):
5         """
6         Une primitive pour accéder a une valeur dans grille.
7         grille de type dictionnaire.
8         Les clés sont des str de type 'A'; 'B', 'C', ...
9         Les valeurs sont des listes de int.
10        """
11        value = grille[chr(65 + y)][x]
12
13        assert value <= 20, "Il n'y a que 20 figures disponibles dans cette GUI"
14        return value
15
16 gui = GUImemory_2(2, 2)
17 g = {'A': [0, 1], 'B':[1, 0]}
18 gui.refresh(g, 'grille est un dictionnaire')
```

