

Trame projet - Démineur

Travail à faire

Le projet consiste en la **création, complète ou partielle, d'un jeu de démineur** dont voici un visuel :



Pour cela il va être nécessaire de **produire les codes Python** permettant de :

1. **construire la grille** du jeu ;
2. **ajouter aléatoirement des mines** dans la grille ;
3. **déterminer le nombre de mines à proximité d'une case** ;
4. **calculer le nombre de mines présentes autour d'une case donnée** ;
5. **permettre l'interaction du joueur à l'aide de clics de souris**, via la **mise en place d'une interface homme / machine**, et cela dans le respect des règles du jeu du démineur ;
6. **apporter les modifications nécessaires** pour que le jeu soit le plus fidèle possible à l'original.

Constitution de la grille

Voici un exemple de grille de jeu de démineur d'une taille de 3 lignes sur 5 colonnes décrite ici sous la forme d'un tableau de tableaux comportant des éléments de type chaîne de caractères (`str`) :

```
grille = [['m', 'o', 'o', 'm', 'o'],
          ['o', 'o', 'o', 'o', 'o'],
          ['m', 'o', 'm', 'm', 'o']]
```

- `'m'` : indique la présence d'une mine ;
- `'o'` : indique l'absence de mine.

Travail 1 : Création d'une grille sans mine

Mettre en place une fonction `creation_grille_vide` recevant en paramètres le nombre de lignes et le nombre de colonnes de la grille puis qui renvoie la grille ainsi construite sous la forme d'un tableau de tableaux.

Vérifier le bon fonctionnement de cette fonction.

Travail 2 : Ajout de mines dans la grille

Mettre en place une fonction `ajout_mines_dans_grille` recevant en paramètres la grille (tableau de tableaux) et le niveau de difficulté puis qui ajoute dans cette grille des mines disposées aléatoirement.

Le niveau de difficulté est une valeur comprise entre `0` (difficulté minimale) et `1` (difficulté maximale) représentative du nombre de mines présentes dans la grille.

Vérifier le bon fonctionnement de cette fonction.

? Travail 3 : Nombre de mines dans la grille

Mettre en place une fonction `nombre_mines_dans_grille` recevant en paramètre la grille (tableau de tableaux) et qui renvoie le nombre de mines présentes dans la grille.

Vérifier le bon fonctionnement de cette fonction.

? Travail 4 : Calcul du nombre de mines présentes autour d'une case donnée

Mettre en place une fonction `nb_mines_voisines` recevant en paramètres la grille (tableau de tableaux), le numéro d'une ligne et le numéro d'une colonne puis qui renvoie le nombre de mines présentes autour de la case spécifiée par la ligne et la colonne fournies en paramètres.

Vérifier le bon fonctionnement de cette fonction.

☰ Travail à faire

Dans cette partie, on va progressivement **mettre en place l'interface homme / machine du jeu** dont voici pour rappel le visuel :

? Travail 5 : Affichage de la grille de jeu

Compléter et tester le code ci-dessous permettant d'afficher la grille vide (sans mine).

```
1  import pygame
2
3  def creation_grille_vide(nb_lignes, nb_colonnes):
4      ... # A compléter avec le travail 1
5
6  def affichage_grille(grille):
7      for l in range(len(grille)):
8          for c in range(len(grille[0])):
9              pygame.draw.rect(fenetre, (255,255,255), (c * pixels, l * pixels, pixels-2, pixels-2))
10         pygame.display.update()
11
12  g = creation_grille_vide(3, 5)
13
14  pygame.init()
15  pixels = 40
16  fenetre = pygame.display.set_mode((len(g[0]) * pixels - 2, len(g) * pixels - 2))
17  pygame.display.set_caption("Démineur")
18
19  affichage_grille(g)
```

Comprendre et expliquer rigoureusement le rôle de chacune des lignes de ce code.

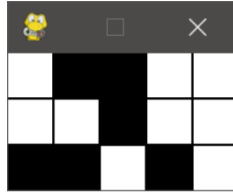
? Travail 6 : Affichage des mines présentes dans la grille

Mettre en place une fonction `affichage_grille_mines` recevant en paramètre la grille (tableau de tableaux) et qui affiche la présence des mines dans la grille selon le rendu visuel ci-dessous.

Soit le tableau de tableaux suivant :

```
[[ 'o', 'm', 'm', 'o', 'o'],  
 [ 'o', 'o', 'm', 'o', 'o'],  
 [ 'm', 'm', 'o', 'm', 'o']]
```

Le visuel associé est le suivant :



Vérifier le bon fonctionnement de cette fonction.

? Travail 7 : Interaction avec le joueur

Compléter, tester, analyser et **comprendre** le code Python ci-dessous permettant la gestion des événements engendrés par des clics de souris sur la grille.

⚠ Attention

Sans une **fine analyse du code** ci-dessous la suite du projet ne sera pas envisageable.

Comprendre et **expliquer** rigoureusement le rôle de chacune des lignes de ce code.

Pour cela, **se documenter sur le module** `pygame` et sur les fonctions associées.

```
1  import pygame  
2  
3  def creation_grille_vide(nb_lignes, nb_colonnes):  
4      ... # A compléter avec le travail 1  
5
```

```

6  def affichage_grille(grille):
7      ... # A compléter avec le travail 6
8
9  g = creation_grille_vide(3, 5)
10
11  pygame.init()
12  pixels = 40
13  fenetre = pygame.display.set_mode((len(g[0]) * pixels - 2, len(g) * pixels - 2))
14  pygame.display.set_caption("Démineur")
15
16  partie_en_cours = True # Variable drapeau de la partie en cours
17  while partie_en_cours: # Boucle principale
18
19      # Gestion des événements utilisateur
20      for event in pygame.event.get():
21
22          if event.type == pygame.QUIT: # Clic sur la croix pour fermer l'application
23              partie_en_cours = False
24
25          if event.type == pygame.MOUSEBUTTONDOWN: # Clic sur un bouton de la souris
26              # Calcul de la ligne et de la colonne associées à la case sur laquelle le joueur a cliqué
27              ligne = event.pos[1] // pixels
28              colonne = event.pos[0] // pixels
29
30              if event.button == 1:
31                  print(f"Vous avez cliqué gauche sur la case de coordonnées {ligne} , {colonne}")
32              elif event.button == 3:
33                  print(f"Vous avez cliqué droit sur la case de coordonnées {ligne} , {colonne}")
34
35      affichage_grille(g)

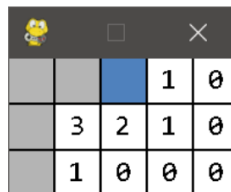
```

? Travail 9 : Affichage du nombre de mines voisines

? Travail 10 : Repérage des éventuelles mines

On souhaite pouvoir **repérer la présence d'éventuelles mines**.

Suite à un clic gauche de souris sur la case suspectée de cacher une mine, la présence de cette éventuelle mine doit être repérée par un rectangle bleu :



			1	0
	3	2	1	0
	1	0	0	0