Mettre **mod_vie.zip** sur le site

1 Présentation du jeu

Le jeu de la vie est un automate cellulaire imaginé par John Horton Conway en 1970. C'est un « jeu à zéro joueur », puisqu'il ne nécessite pas d'intervention. Il se déroule sur une grille à deux dimensions. Les cellules peuvent prendre deux états : *vivantes* ou *mortes*.
À chaque étape, l'évolution d'une cellule est entièrement déterminée par l'état de ses huit voisines de la façon suivante :

— une cellule morte possédant exactement trois voisines vivantes devient vivante (elle naît),

— une cellule vivante possédant deux ou trois voisines vivantes le reste, sinon elle meurt.

Quelques exemples illustrent ces règles simples :

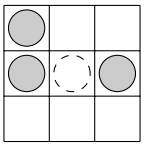

FIGURE 1 – Une cellule morte naît car elle a trois voisins

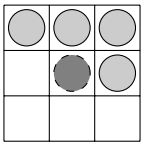

FIGURE 2 – Une cellule vivante meurt car elle a plus de trois voisins

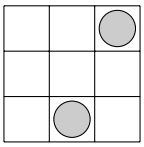

FIGURE 3 – Une cellule morte le reste car elle n'a que deux voisins.

2 Outil graphique

La bibliothèque *tkinter* est l'interface graphique standard de Python. Le code 1 crée une *fenêtre* graphique et y place un *canevas*. Un canevas est une surface de dessin.

```
1 fenetre = Tk()
2 fenetre.title("Jeu de la vie")
3 canevas = Canvas(fenetre, width = 100, height = 100)
4 canevas.pack()
5
6 # ligne àplacer en fin de programme
7 fenetre.mainloop()
```

Code 1 – Préparer la fenêtre graphique

Christophe Viroulaud

Activité 1 :

1. Créer un fichier jeu-de-la-vie.py et écrire le code 1.
2. Créer deux variables :

```
1 TAILLE = 8 #dimension d'une cellule,
2 CELLULES = 200 #nombre de cellules en largeur et en hauteur.
```

   Ces deux valeurs seront des constantes du jeu.
3. Modifier le code 1 pour créer un canevas aux bonnes dimensions.

3 Préparation du jeu

3.1 Créer la grille

Nous allons représenter les cellules par un tableau de booléens. Chaque ligne contiendra *CELLULES* cases et il y aura *CELLULES* lignes. Une cellule vivante sera codée *True* et une cellule morte, *False*.

Activité 2 : Construire, en compréhension, le tableau représentant la grille du jeu.

3.2 Initialiser la grille

Pour que le jeu démarre il faut rendre certaines cellules vivantes. Il est possible de les choisir manuellement ou d'en générer aléatoirement un certain nombre.

Activité 3 : Écrire la fonction **aleatoire(g : list, n : int) → None** qui génère au hasard n cellules vivantes dans la grille g.

3.3 Compter les voisins

À chaque cycle des cellules naissent ou meurent en fonction de leurs voisins. Pour chaque cellule, il faut donc regarder les huit cellules voisines et compter le nombre de ces cellules qui sont vivantes.

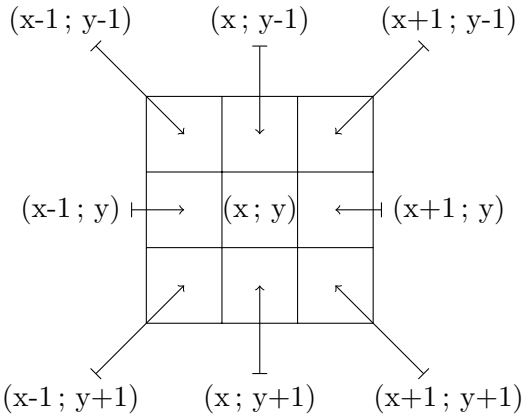

FIGURE 4 – Coordonnées de cellules voisines
L'axe des ordonnées est orienté vers le bas.

Activité 4 : Écrire la fonction **compter_voisins(g : list, cel : tuple) → int** qui compte et renvoie le nombre de cellules vivantes de la cellule de coordonnées *cel*. Il peut être nécessaire de :

— utiliser la variable locale *delta*,

Christophe Viroulaud

```
delta = ((-1,-1), (-1,0), (-1,1), (0,-1), (0,1), (1,-1), (1,0), (1,1))
```

— penser à gérer les « bords » de la grille.

4 Créer le jeu

4.1 Réaliser un cycle

Lors d'un cycle, le jeu regarde chaque cellule de la grille et la fait naître ou mourir en fonction de ses voisines.

Activité 5 : Compléter la fonction **cycle (f, c, g : list) → None** du code 2. Les paramètres *f, c, g* représentent respectivement la fenêtre tkinter, le canevas et la grille du jeu.

```
def cycle(f, c, g: list)->None:
    # création d'une nouvelle grille vide
    nouvelle =

    """
    À RÉALISER
    parcourir la grille g pour faire naître ou mourir les cellules.
    La nouvelle configuration est enregistrée dans la grille 'nouvelle'
    """

    # la grille est actualisée
    g = nouvelle

    # affichage de la grille
    c.delete("all")
    for i in range(CELLULES):
        for j in range(CELLULES):
            if g[i][j]:
                # création d'un cercle bleu
                c.create_oval(TAILLE*(j),
                              TAILLE*(i),
                              TAILLE*(j+1),
                              TAILLE*(i+1),
                              fill="blue")
    """
    nouveau cycle: la méthode after rappelle la fonction
    cycle toutes les 100ms
    """
    f.after(100, cycle, f, c, g)
```

Code 2 – La fonction principale du jeu

4.2 Jouer

Toutes les fonctions sont maintenant disponibles. Pour jouer, il faut :

Christophe Viroulaud

— créer une grille vide,
— créer une surface tkinter,
— remplir la grille aléatoirement,
— lancer le cycle.

4.3 Des structures prédéfinies

La bibliothèque *mod_vie* contient les structures :

— une_ligne,
— aleatoire,
— vaisseau,
— canon.

Activité 6 :

1. Télécharger le module sur le site https://cviroulaud.github.io et le décompresser dans le même dossier que le programme.
2. Importer toutes les fonctions du modules dans le programme.
3. Lancer une fois le programme avec ces nouveaux imports.
4. Dans la console, taper :

```
1 help(vaisseau)
```

   Dans Spyder, il est possible de faire apparaître cette aide (la *docstring)* en plaçant le curseur sur la fonction et en appuyant sur les touches *ctrl+i*.
5. Ouvrir le fichier *mod_vie.py* et observer la construction des docstrings.
6. Utiliser les différentes structures dans le programme.

Christophe Viroulaud