SR01_A20

Rapport de Devoir 3 : Programmation Python

Responsable : LAKHLEF Hicham

Responsable TD Python : JABER Ghada

TD Groupe 7, Vendredi 14h15 – 16h15

Etudiant: HO Xuan Vinh

Table des matières

1. Les principales fonctions

```
class App():
    def __init__(self, canvas_size=600):
        self.canvas_size    = canvas_size
        self.width          = self.canvas_size+105
        self.height         = self.canvas_size
        #where is the buttons
        self.init_layout()
        #what will these button do
  self.init_handlers()
      #defaut values
       self.init_defaults()
```

Tout d'abord, je vais créer la classe principale de mon programme. Je vais définir la valeur par défaut de la taille de la fenêtre est 600, la valeur de la largeur est plus grande pour l'espace des boutons.

Ensuite, je vais définir la position des boutons, les fonctions associées à ces boutons et certaines valeurs par défaut.

```
def init_layout(self):
       #create main window, in fact there is just one
    self.root = Tk()
    self.root.title("Jeu de la vie")
    self.root.geometry(f'{self.width}x{self.height}')

    self.canvas = Canvas(self.root, width=self.canvas_size, height=self.canvas_size, background='white')
    self.canvas.grid(row=0,column=0)

    self.frame_control = Frame(self.root)
    self.frame_control.grid(row=0, column=1)

    buttons         = ('Initialiser', self.initialize), ('Lancer',self.launch), ('Arreter',self.stop), ('Quitter', self.quit)
    self.buttons    = [ Button(self.frame_control,text=name,command=f) for (name,f) in buttons]

    scales          = ('Taille dela grille', 'n_cell'), ('%de Vie', 'p_alive'), ('Vitesse','speed')
    self.scales     = { key : Scale(self.frame_control, from_=2, to=100, orient='horizontal', label=label) for label,key in scales}

    widgets         = self.buttons[:-1] + list(self.scales.values()) + [self.buttons[-1]]

    for i, widget in enumerate(widgets):
        widget.grid(row=i, column=0, sticky='nesw')
```

J'ai créé cette fonction pour configurer mes boutons, ces fonctions et actions lorsque j'ai cliqué sur ces fonctions et d'autres éléments.

Dans ma fenêtre, il n'y a qu'une ligne et deux colonnes. La première ligne (ligne = 0, colonne = 0) est pour la partie principale, où je vais afficher ma matrice. La colonne des secondes (ligne = 0, colonne = 1) est pour tous les boutons.

Avec les boutons, je vais créer une liste avec un élément est un nom de bouton et une fonction de bouton (ou je peux l'appeler une action). Avec la partie Scales, j'ai la taille de matrice (n), le% vivant et la vitesse. Enfin, j'ai mis 3 premiers boutons, partie Scales et un bouton de sortie. Sticky = nesw pour notre bouton est au milieu.

```
def init_handlers(self):
    self.root.protocol("WM_DELETE_WINDOW", self.quit)


def init_defaults(self):
    self.scales['n_cell'].set(30)
    self.scales['p_alive'].set(20)
    self.scales['speed'].set(1)
```

Si l'utilisateur fait quelque chose comme cliqué pour quitter le bouton, ferme la fenêtre ou ferme le programme (par Ctrl + C, etc.), l'application sera fermée. Ensuite, il y a toutes les valeurs par défaut.

```
def initialize(self):
    self.stop()
    self.n_cell = int(self.scales['n_cell'].get())
    self.cell_size      = self.canvas_size/self.n_cell
    self.p_alive    = self.scales['p_alive'].get()/100

    print("Initialize grid.")
    self.state = [[random()<self.p_alive for _ in range(self.n_cell)] for _ in range(se
lf.n_cell)]
    self.draw_state()
```

Fonction pour le bouton "**Initialiser**". Je vais arrêter le processus, puis créer une nouvelle matrice selon les valeurs de la partie Scales, puis la dessiner par la fonction **draw_state**.

```
def launch(self):
    def periodic_update():
        print("Start updating forever.")
        while True:
            if self.update_loop_stop:
                print("No longer updating.")
                return
```

```
            self.update()
               #max is 5seconds, 0.05 for not chaotic
            delta_t = 0.05+self.scales['speed'].get()/20.0
            print(f"Updated. Next update in {delta_t}")
               #go sleep
            time.sleep(delta_t)

    if not hasattr(self, 'state'):
        print("Please initialize first.")
        return

    if self.update_loop_stop:
        self.update_loop_stop = False
        self.thread = threading.Thread(target=periodic_update)
        self.thread.start()
    else:
        print("Already running")
```

Fonction pour le bouton "**Lancer**". Le processus fonctionnera pour toujours, j'utilise While True. Je vérifierai également si nous sommes en état d'arrêt ou non, si nous sommes en état d'arrêt, j'écrirai une notification. J'ai utilisé fonction **Update** pour créer la prochaine génération, la **vitesse** dépend de la partie Scales.

Je vais vérifier si l'utilisateur clique sur Initialiser avant ou non.

Si nous ne sommes pas dans l'état d'arrêt, cela signifie que nous cliquons dans Lancer lorsque le processus est en cours d'exécution, alors j'écris une notification. Si nous sommes dans l'état d'arrêt, cela signifie que nous sommes dans un processus ou que nous sommes sur le point de commencer un nouveau processus. Pour que je continue et crée un nouveau thread avec la cible, c'est ma fonction "**periodic_update**". Je pense que c'est similaire à la fonction **fork()** en C.

En fait, nous pouvons utiliser la fonction **self.canvas.after()** mais je pense que ma solution est plutôt ok.

```
def stop(self):
    self.update_loop_stop = True
```

Je fais simplement de notre état un "état d'arrêt".

```
def update(self):
    flip_indices = []

    for i,j in product(range(self.n_cell), range(self.n_cell)):
        alive = self.state[i][j]
        neighbors = (i-1, j-1), (i-1,j), (j-1, j+1), (i,j-1), (i,j+1), (i+1,j-
1), (i+1,j), (i+1,j+1)
        n_alive_neighbors = sum([self.state[i][j] if 0<=i<self.n_cell and 0<=j<self.n_c
ell else 0 for i,j in neighbors])
```

```
            under_population    = alive and n_alive_neighbors < 2
            over_population     = alive and n_alive_neighbors > 3
            reproduction        = not alive and n_alive_neighbors == 3

            if under_population or over_population or reproduction:
                flip_indices.append((i,j))

        for i,j in flip_indices:
            self.state[i][j] = not self.state[i][j]

        self.draw_state(flip_indices)
```

C'est la fonction pour créer une nouvelle génération. Je vais également calculer le nombre de voisins vivants de toute la position. On a `(i-1, j-1)`, `(i-1,j)`, `(j-1, j+1)`, `(i,j-1)`, `(i,j+1)`, `(i+1,j-1)`, `(i+1,j)`, `(i+1,j+1)`.

Avec à ma manière, je dois retourner ma matrice avant de la dessiner.

• Si elle a exactement 2 ou 3 voisins vivants, elle survit à la génération suivante.

• Si elle a au moins 4 cellules voisines vivantes, elle meurt d'étouffement à la génération suivante. **Over_population.**

• Si elle a au plus une cellule voisine vivante, elle meurt d'isolement à la génération suivante. **Under_population.**

• Si elle a exactement 3 voisins vivants, une cellule naîtra à la génération suivante. **Reproduction.**

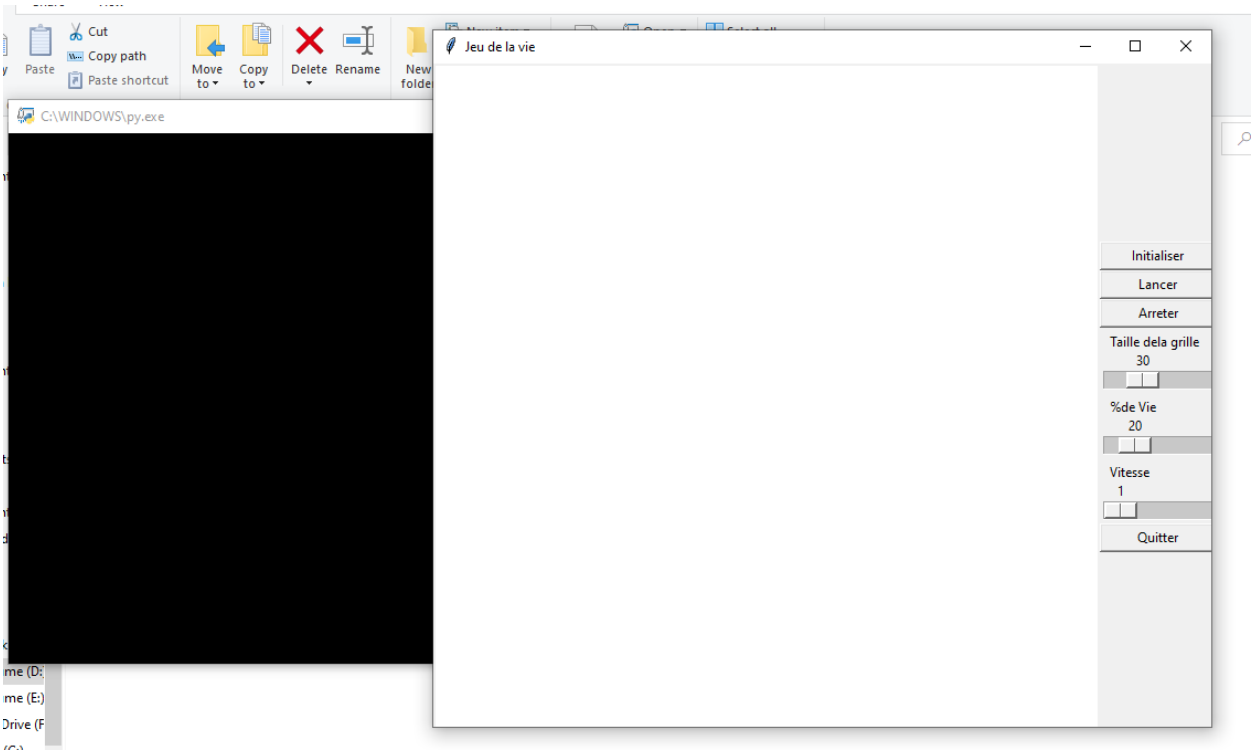
Après avoir défini une génération, l'étape suivante ce que j'utilise la fonction **draw_state()** pour imprimer à l'écran du résultat.

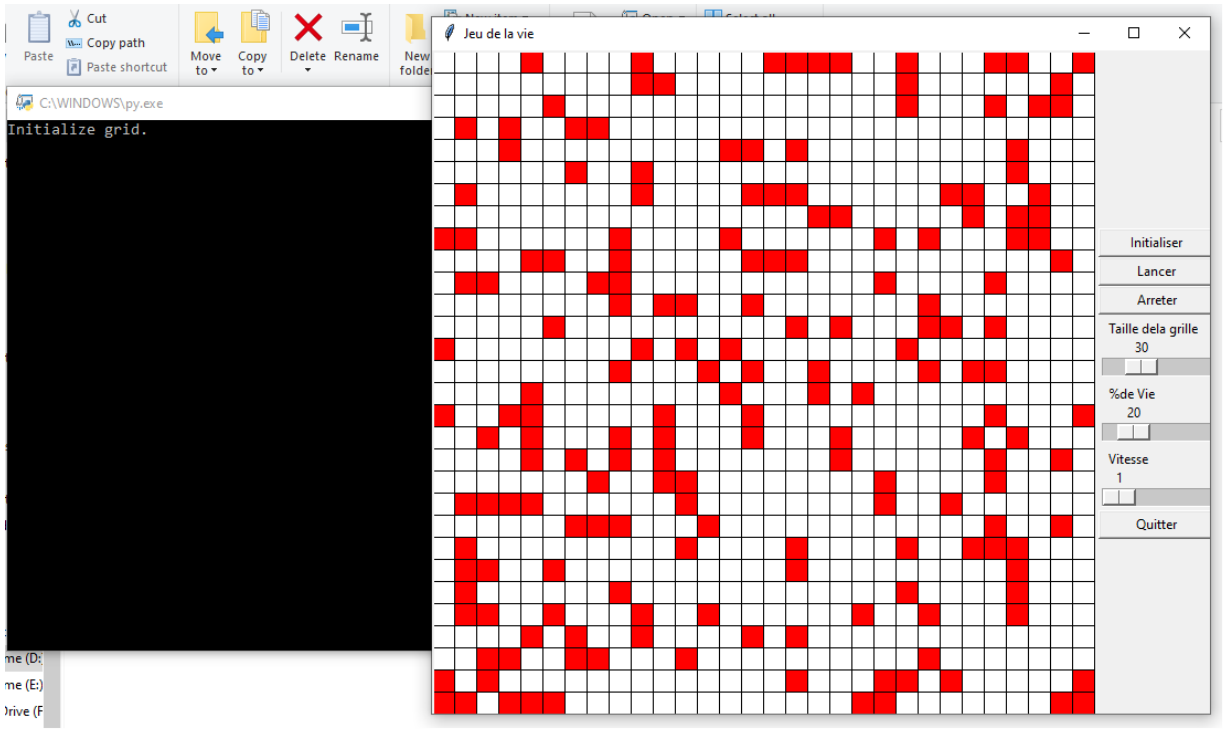
```
def draw_state(self, changes=None):
    if changes == None:
        changes = product(range(self.n_cell), range(self.n_cell))
    for i,j in changes:
        rectangle = i*self.cell_size, j*self.cell_size, (i+1)*self.cell_size, (j+1)*sel
f.cell_size
        color = "red" if self.state[i][j] else "white"
        self.canvas.create_rectangle(*rectangle, fill=color)


def quit(self):
    self.update_loop_stop = True
    self.root.destroy()
```

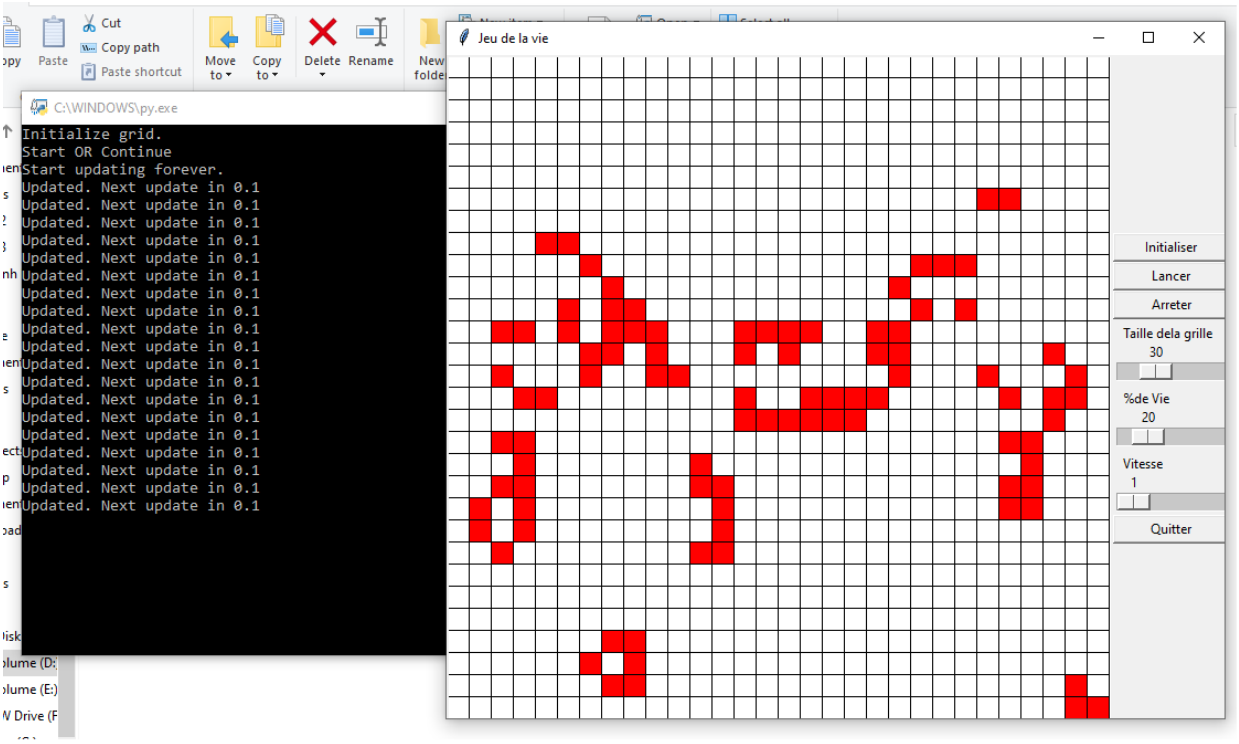
C'est une fonction pour dessiner la matrice et la fonction pour quitter l'application. Similaire à la fonction **update**, j'ai utilisé des boucles pour définir la couleur et l'imprimer à l'écran à l'aide de la fonction **self.canvas.create_rectangle (* rectangle, fill = color).**

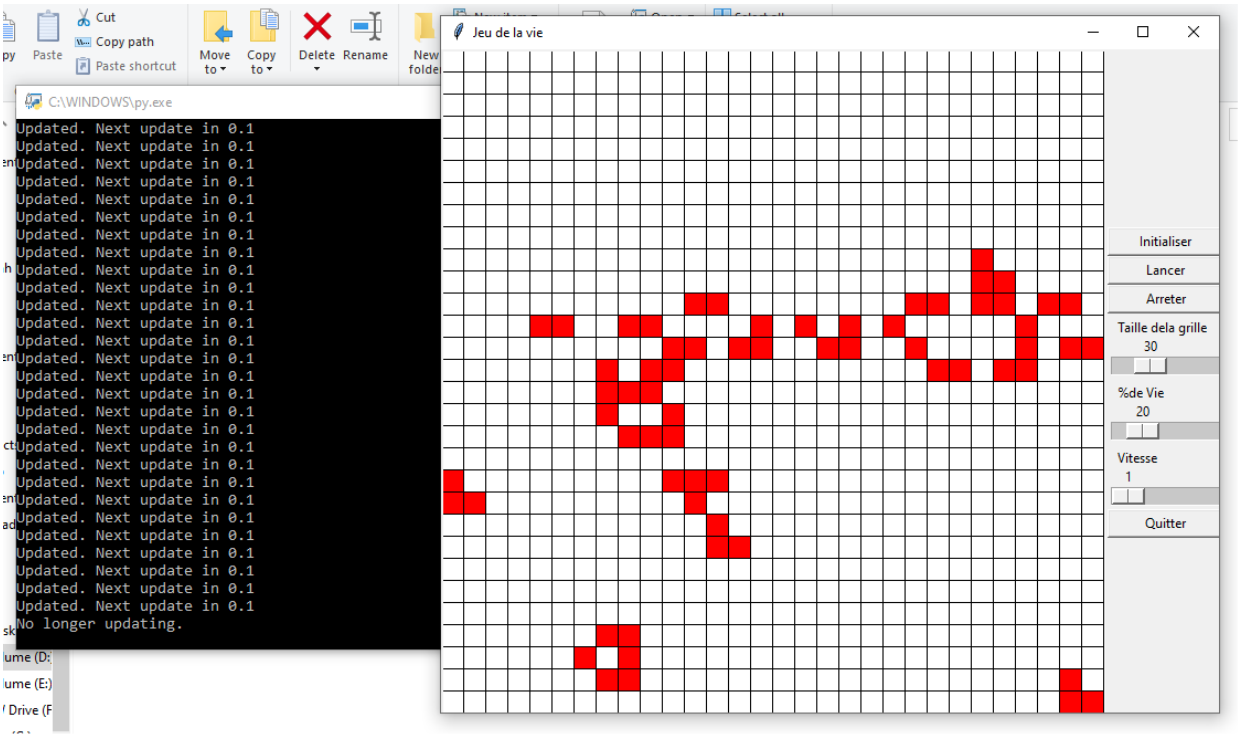
La méthode **quit** utilisera la méthode **destroy** pour arrêter le programme.

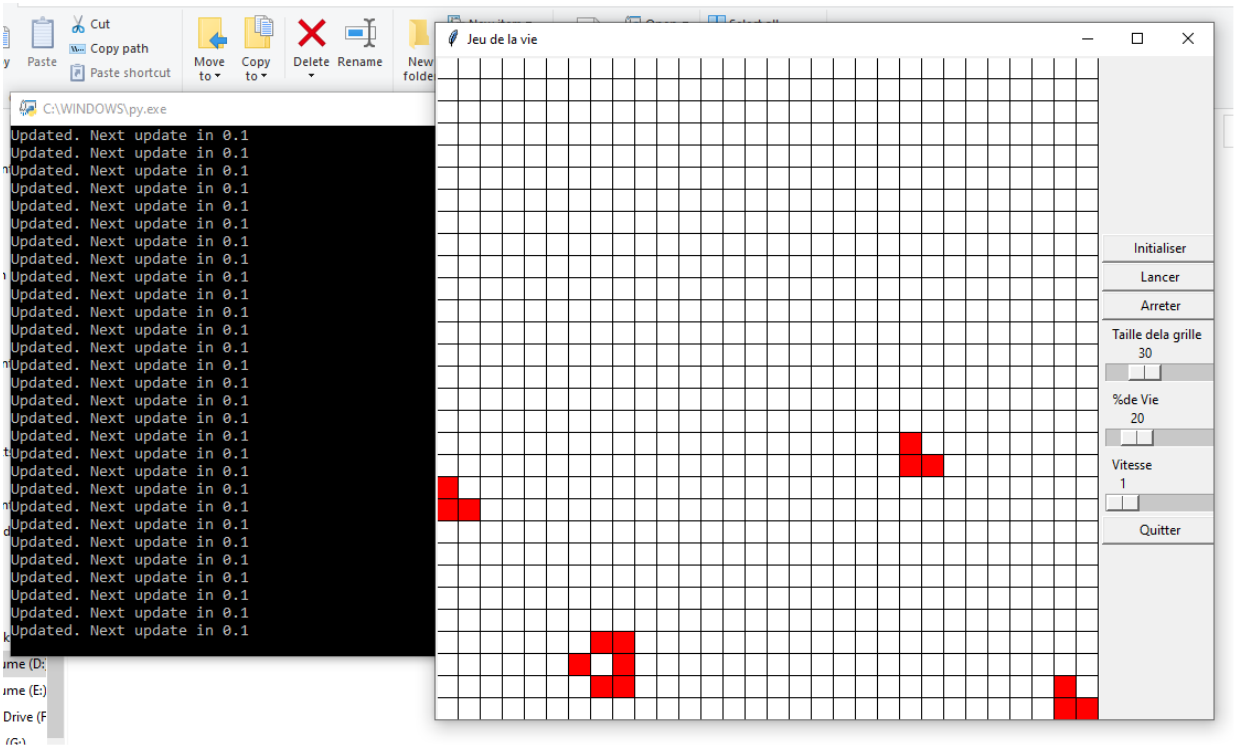
2. Quelques résultats

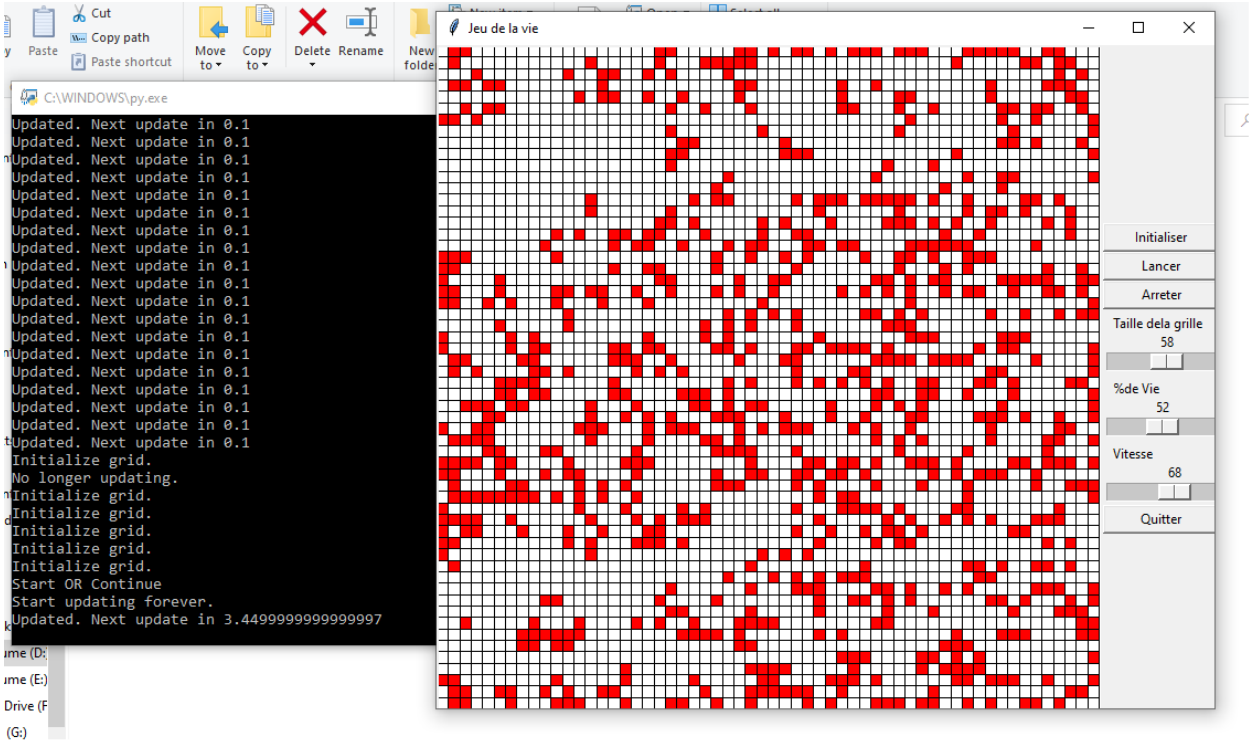


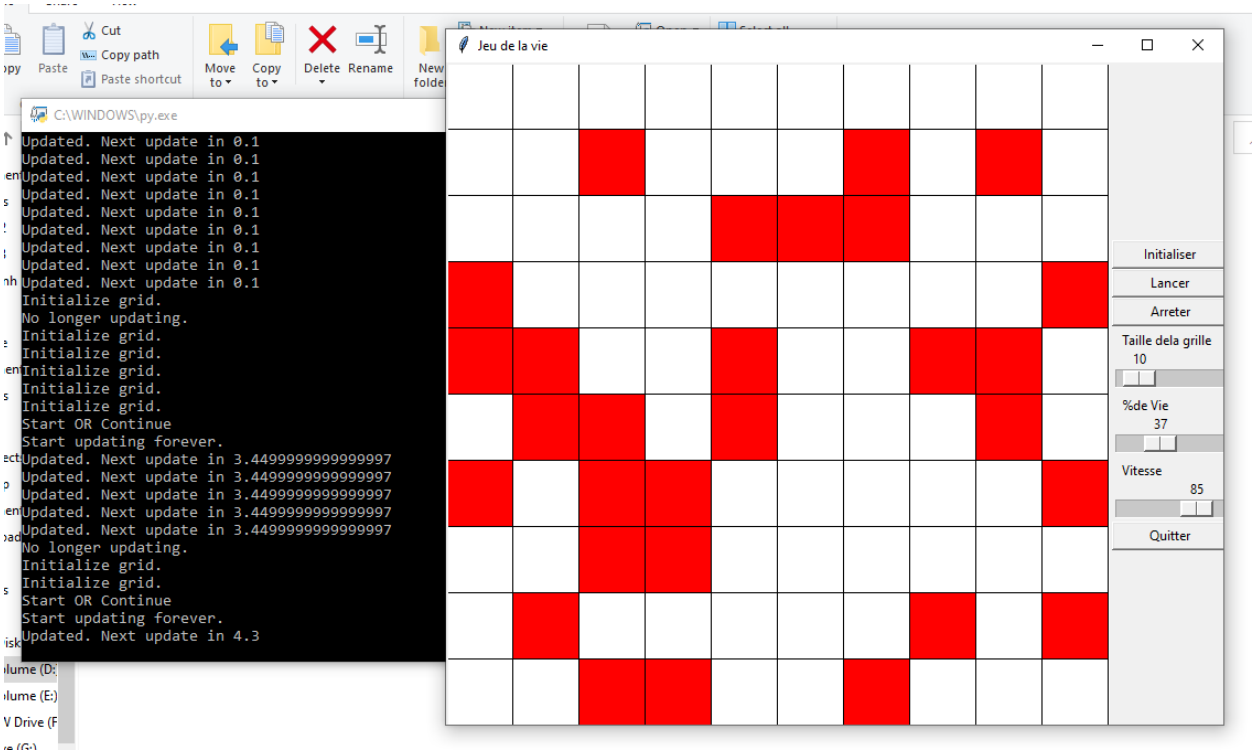

Cut
Copy path
Paste shortcut
Paste
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Initialize grid.
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
30
%de Vie
20
Vitesse
1
Quitter

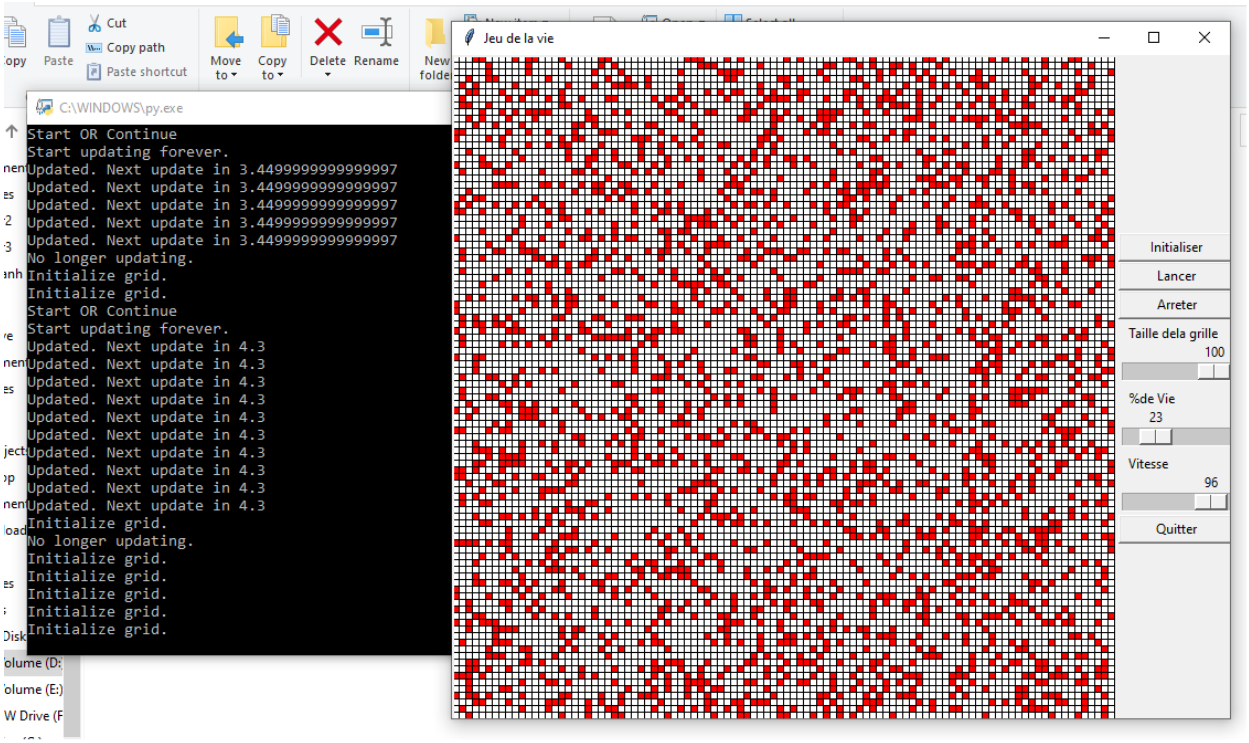

Cut
Copy path
Paste shortcut
Paste
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Initialize grid.
Start OR Continue
Start updating forever.
Updated. Next update in 0.1
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
30
%de Vie
20
Vitesse
1
Quitter


Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Updated. Next update in 0.1
No longer updating.
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
30
%de Vie
20
Vitesse
1
Quitter


Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Updated. Next update in 0.1
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
30
%de Vie
20
Vitesse
1
Quitter


Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Updated. Next update in 0.1
Initialize grid.
No longer updating.
Start OR Continue
Start updating forever.
Updated. Next update in 3.4499999999999997
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
58
%de Vie
52
Vitesse
68
Quitter


Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Updated. Next update in 0.1
Initialize grid.
No longer updating.
Start OR Continue
Start updating forever.
Updated. Next update in 3.4499999999999997
Updated. Next update in 4.3
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
10
%de Vie
37
Vitesse
85
Quitter


Cut
Copy path
Paste shortcut
Move to
Copy to
Delete
Rename
C:\WINDOWS\py.exe
Start OR Continue
Start updating forever.
Updated. Next update in 3.4499999999999997
Updated. Next update in 3.4499999999999997
Updated. Next update in 3.4499999999999997
Updated. Next update in 3.4499999999999997
Updated. Next update in 3.4499999999999997
No longer updating.
Initialize grid.
Initialize grid.
Start OR Continue
Start updating forever.
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Updated. Next update in 4.3
Initialize grid.
No longer updating.
Initialize grid.
Initialize grid.
Initialize grid.
Initialize grid.
Initialize grid.
Jeu de la vie
Initialiser
Lancer
Arreter
Taille dela grille
100
%de Vie
23
Vitesse
96
Quitter