Jeu de la vie

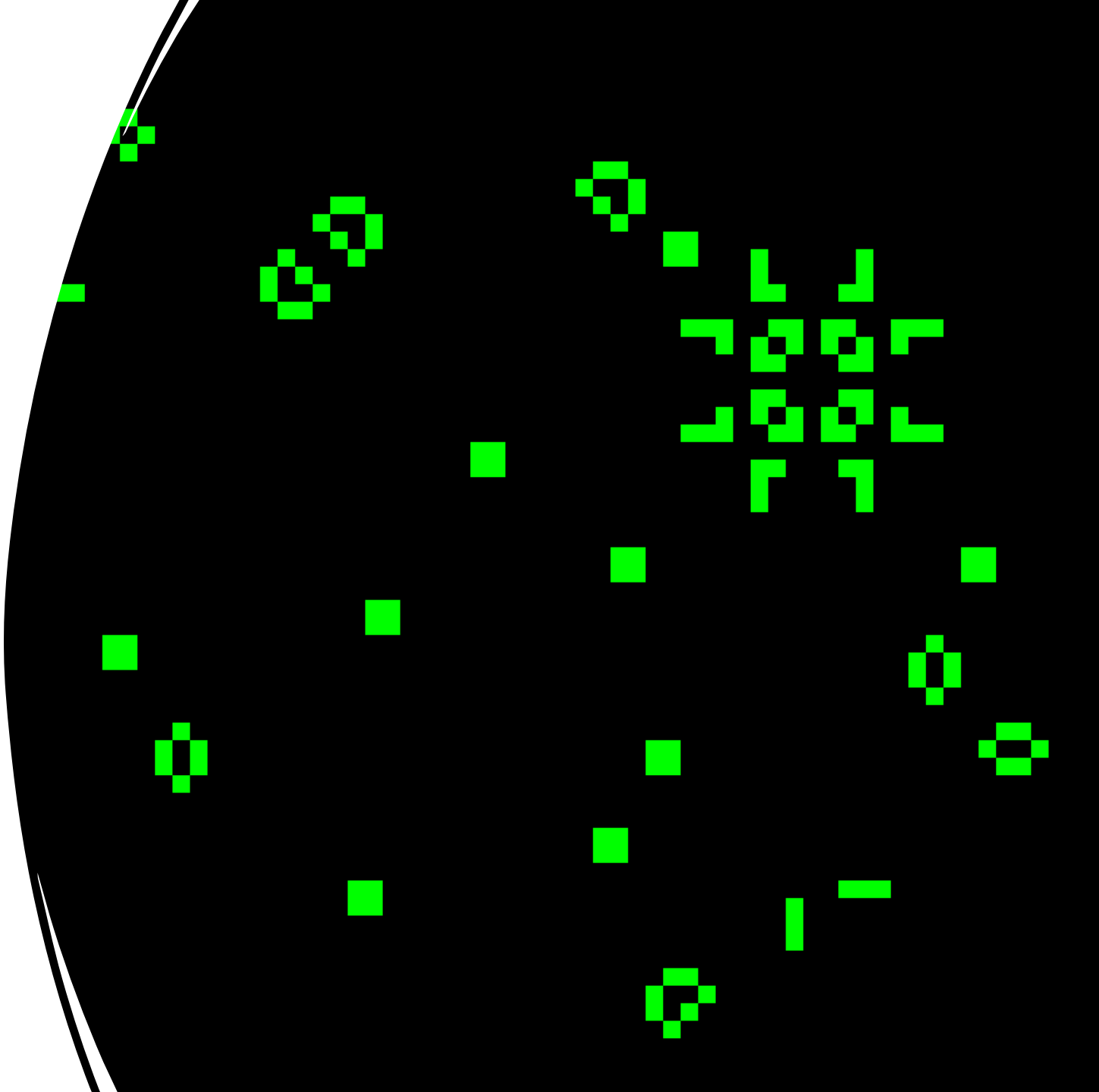
Fait par Adam et Urbain

Intro

I. Son histoire et ses règles

II. Les différente parties de notre code:
    1. Création de la grille
    2. Captages des alentours
    3. Nouvelle grille
    4. Version 1
    5. Version 2
    6. Version Lenia
    7. Version Final
    8. Version traduite

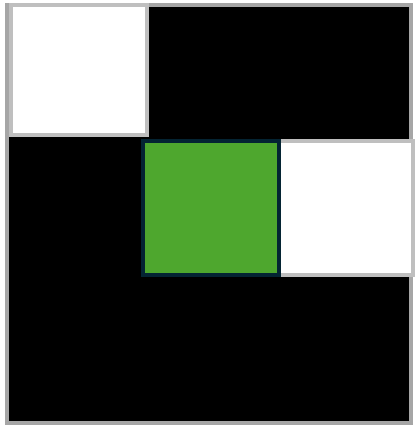
III. Logiciels utilisés



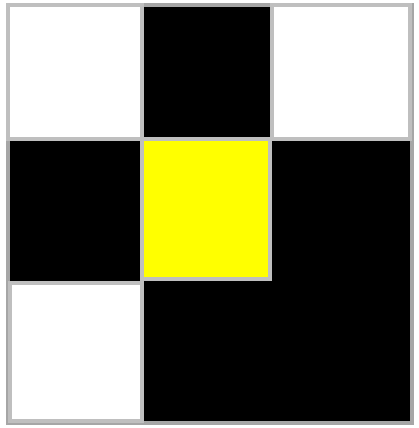
Son histoire et ses règles

- John Horton Conway
- Automate cellulaire
- 1970

Survie
Si 2 ou 3 cellules aux alentours



Naissance
Si 3 cellules aux alentours



Création de la grille

```
HEIGH = 5
WIDTH = 5
lagrille = [[randint(0, 1) for i in range(WIDTH)] for i in range(HEIGH)]
```

Captage des alentours

(x-1,y-1)	(x,y-1)	(x+1,y-1)
(x-1,y)	(x,y)	(x+1,y)
(x-1,y+1)	(x,y+1)	(x+1,y+1)

```
# diagonale haut-gauche
if grille[(y - 1) % largeur][(x + 1) % hauteur] == 1:
    nb += 1

# haut
if grille[y][(x + 1) % hauteur] == 1:
    nb += 1

# Diagonale haut-droite
if grille[(y + 1) % largeur][(x + 1) % hauteur] == 1:
    nb += 1

# gauche
if grille[(y - 1) % largeur][x] == 1:
    nb += 1

# droite
if grille[(y + 1) % largeur][x] == 1:
    nb += 1

# Diagonale bas-gauche
if grille[(y - 1) % largeur][(x - 1) % hauteur] == 1:
    nb += 1

# bas
if grille[y][(x - 1) % hauteur] == 1:
    nb += 1

# diagonale bas-droite
if grille[(y + 1) % largeur][(x - 1) % hauteur] == 1:
    nb += 1

return nb
```

Nouvelle grille

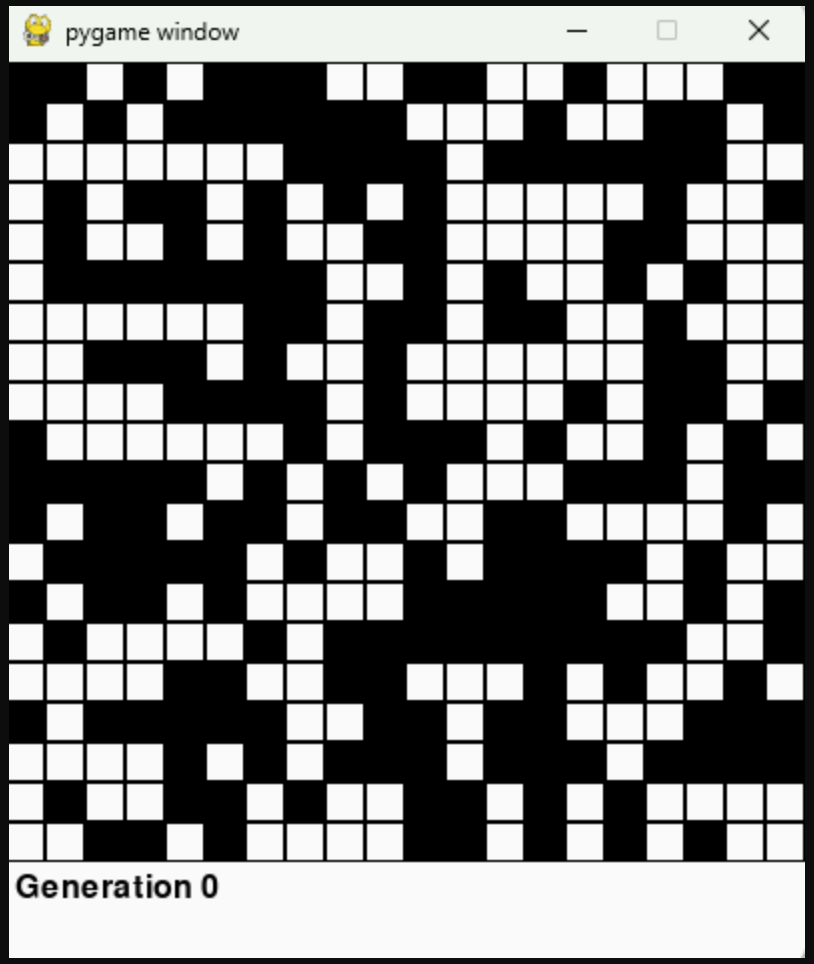
```python
def new_grid(grille):
    # Calcul des dimentions de la grille
    largeur = len(grille[0])
    hauteur = len(grille)

    nouvelle_grille = [[0 for i in range(largeur)] for i in range(hauteur)]

    # Parcourir les cellules
    for y in range(len(grille)):
        for x in range(len(grille[0])):
            alentours = voisins(x, y, grille)
            # Règle 1 - Mort d'isolement
            if (
                grille[y][x] == 1 and alentours < 2
            ):  # Si la cellule est vivante et qu'elle a un nombre de voisins inférieur à deux
                nouvelle_grille[y][x] = 0  # alors elle meurt

            # Règle 2 - Toute cellule avec 2 ou 3 voisins survit.
            if grille[y][x] == 1 and (
                alentours in [2, 3]
            ):  # Si une cellule est vivante et qu'elle a deux ou trois voisins
                nouvelle_grille[y][x] = 1  # alors elle reste en vie

            # Règle 3 - Mort par surpopulation
            if (
                grille[y][x] == 1 and alentours > 3
            ):  # si une cellule est vivante et qu'elle a plus de trois voisins
                nouvelle_grille[y][x] = 0  # alors elle meurt

            # Règle 4 - Naissance
            if (
                grille[y][x] == 0 and alentours == 3
            ):  # si une cellule est morte et qu'elle a trois voisins
                nouvelle_grille[y][x] = 1  # alors elle nait (son état est à vivant)

    return nouvelle_grille
```

Premier Version

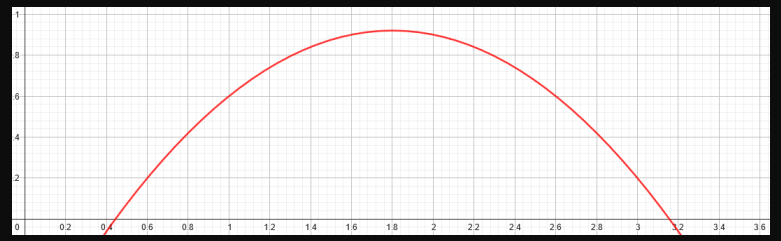
```
[[1, 0, 0, 0, 1], [0, 1, 0, 1, 0], [0, 0, 1, 1, 0], [0, 1, 1, 1, 1], [0, 1, 1, 1, 1]]
[[0, 0, 0, 0, 0], [1, 1, 0, 1, 0], [1, 0, 0, 0, 0], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
[[0, 0, 0, 0, 0], [1, 1, 0, 0, 1], [1, 1, 0, 0, 1], [0, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
[[1, 0, 0, 0, 0], [0, 1, 0, 0, 1], [0, 1, 0, 0, 1], [1, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
[[1, 0, 0, 0, 0], [0, 1, 0, 0, 1], [0, 1, 0, 0, 1], [1, 0, 0, 0, 0], [0, 0, 0, 0, 0]]
```

Deuxieme Version

```
Quelle dimentions : 20
Quelle taille de pixels : 20
```

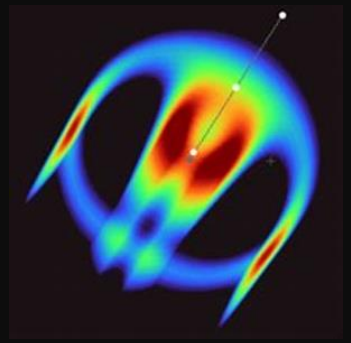



```python
from func import *
import pygame as pgm
from random import *

dimentions, taille_pixels = int(input("Quelle dimentions : ")), int(
    input("Quelle taille de pixels : ")
)
grille = [[randint(0, 1) for i in range(dimentions)] for i in range(dimentions)]

pgm.init()  # Initialiser la fenetre

screen = pgm.display.set_mode(
    (dimentions * taille_pixels, dimentions * taille_pixels + 50)
)  # Afficher la fenetre

compteur = 0  # Initalisation du compteur
font = pgm.font.Font(None, 24)
text = font.render("Generation 0", 1, (0, 0, 0))

run = True
while run:
    for event in pgm.event.get():
        # Fermeture de la fenetre
        if event.type == pgm.QUIT:
            run = False
        # Changer de grille
        if event.type == pgm.KEYDOWN and event.key == pgm.K_RIGHT:
            grille = new_grid(grille)
            compteur += 1  # Incrementation du compteur
            text = font.render(
                f"Generation {compteur}", 1, (0, 0, 0)
            )  # Actualisation du compteur

    screen.fill((250, 250, 250))  # Changer la couleur de fond

    # Dessin de la grille
    for y in range(len(grille)):
        for x in range(len(grille[y])):
            if grille[y][x] == 1:
                pgm.draw.rect(
                    screen,
                    (0, 0, 0),
                    [
                        x * taille_pixels,
                        y * taille_pixels,
                        taille_pixels,
                        taille_pixels,
                    ],
                )
            else:
                pgm.draw.rect(
                    screen,
                    (0, 0, 0),
                    [
                        x * taille_pixels,
                        y * taille_pixels,
                        taille_pixels,
                        taille_pixels,
                    ],
                    1,
                )
    screen.blit(text, (5, dimentions * taille_pixels + 5))

    pgm.display.flip()  # Actualisation de la fenetre
```



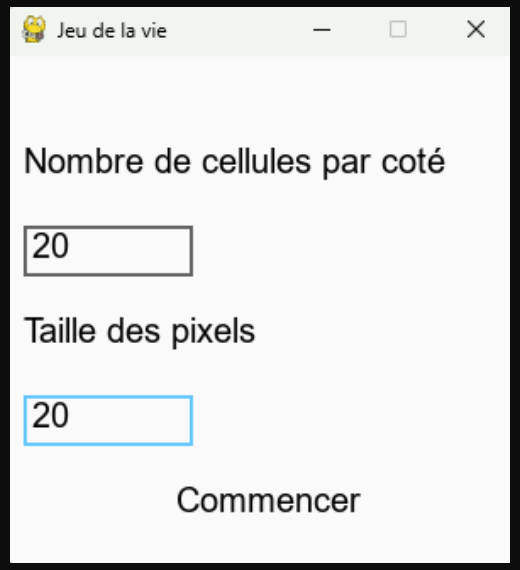
Version LENIA

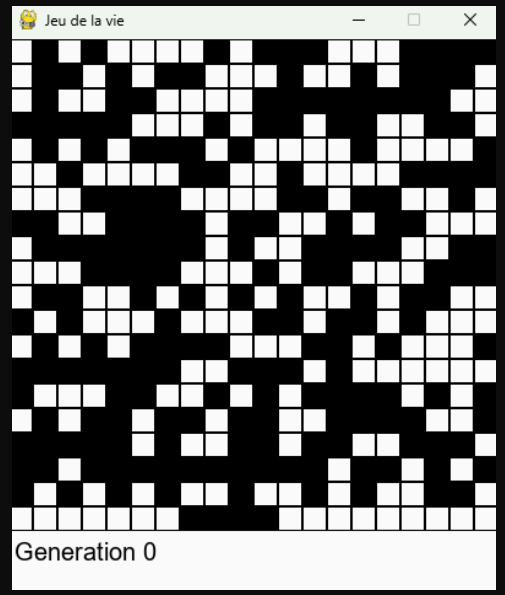
	Jeu de la vie	LENIA
Etat d'une cellule	0 ou 1 (mort ou vivant)	Nombre continu entre 0 et 1. Le nombre est en général représenté par l'intensité d'une couleur.
Voisinage	8 cellules adjacentes	*Zone*(s) de convolution en anneau autour de la cellule.
Espace	Discret	Continu
Règles de vie ou de mort	Nombre de cellules dans le voisinage	*Fonction de croissance* de la cellule en fonction de la somme des états couverts par le voisinage. Donne un taux de croissance, qui permet de calculer la variation de l'état de ma cellule en fonction du temps.
Temps	Discret	Continu

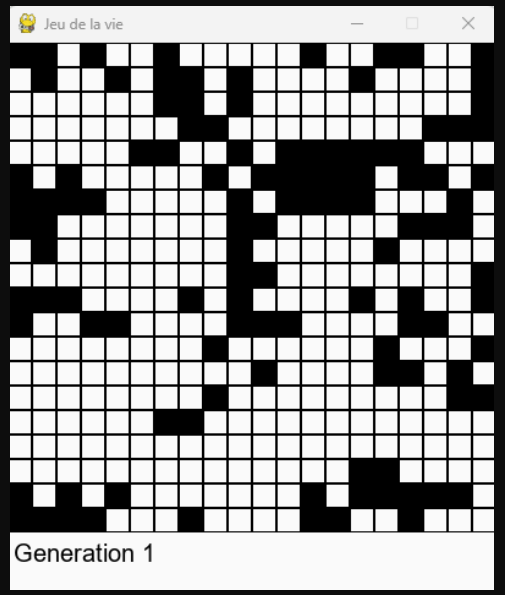


```python
from funcLENIA import *
import pygame as pgm
from random import *

dimentions, taille_pixels = int(input("Quelle dimentions : ")), int(
    input("Quelle taille de pixels : ")
)
grille = [[random() for i in range(dimentions)] for i in range(dimentions)]

pgm.init()  # Initialiser la fenetre

screen = pgm.display.set_mode(
    (dimentions * taille_pixels, dimentions * taille_pixels + 50)
)  # Afficher la fenetre

compteur = 0  # Initalisation du compteur
font = pgm.font.Font(None, 24)
texte_compteur = font.render("Generation 0", 1, (0, 0, 0))

run = True
while run:
    for event in pgm.event.get():
        # Fermeture de la fenetre
        if event.type == pgm.QUIT:
            run = False
        # Changer de grille
        if event.type == pgm.KEYDOWN and event.key == pgm.K_RIGHT:
            grille = new_grid(grille)
            compteur += 1  # Incrementation du compteur
            texte_compteur = font.render(
                f"Generation {compteur}", 1, (0, 0, 0)
            )  # Actualisation du compteur

    screen.fill((250, 250, 250))  # Changer la couleur de fond
```

```python
# Dessin de la grille
for y in range(len(grille)):
    for x in range(len(grille[y])):
        pgm.draw.rect(
            screen,
            (0, int(grille[y][x] * 255), int(grille[y][x] * 255)),
            [
                x * taille_pixels,
                y * taille_pixels,
                taille_pixels,
                taille_pixels,
            ],
        )

screen.blit(texte_compteur, (5, dimentions * taille_pixels + 5))

pgm.display.flip()  # Actualisation de la fenetre
```







Version Final

```python
# Importation des librairies
from func import *
import pygame as pgm
from random import randint


pgm.init()  # Initialiser la fenetre
clock = pgm.time.Clock()

# Afficher la fenetre
screen = pgm.display.set_mode((300, 300))
pgm.display.set_caption("Jeu de la vie")

compteur = 0  # Initalisation du compteur
# Création des textes
arialFont = pgm.font.SysFont("arial", 20)
titre = arialFont.render("Le jeu de la vie de Conway", 1, (0, 0, 0))
play = arialFont.render("Jouer", 1, (0, 0, 0))
dim_txt = arialFont.render("Nombre de cellules par coté", 1, (0, 0, 0))
taille_txt = arialFont.render("Taille des pixels", 1, (0, 0, 0))
start = arialFont.render("Commencer", 1, (0, 0, 0))
gen = arialFont.render("Generation 0", 1, (0, 0, 0))
# Booleen pour l'affichage des fenetres
parametre = False
jouer = False
# Zones de click
play_rect = pgm.Rect(120, 150, 43, 20)
start_rect = pgm.Rect(100, 250, 92, 20)
dim_rect = pgm.Rect(10, 100, 100, 30)
taille_rect = pgm.Rect(10, 200, 100, 30)
# Zones de texte des zones clickables
dimentions = ""
taille_pixels = ""
# Booleen pour la selection des zones de texte
dim_active = False
taille_active = False
```

```python
un = True
ile run:
    for event in pgm.event.get():
        # Fermeture de la fenetre
        if event.type == pgm.QUIT:
            run = False

        if event.type == pgm.MOUSEBUTTONDOWN:
            # Activer la fenetre des parametres
            if play_rect.collidepoint(event.pos):
                parametre = True

            # Demarer la simulation
            if start_rect.collidepoint(event.pos):
                dimentions = int(dimentions)
                taille_pixels = int(taille_pixels)
                jouer = True
                grille = [
                    [randint(0, 1) for i in range(dimentions)]
                    for i in range(dimentions)
                ]
            # Zones de textes
            if dim_rect.collidepoint(event.pos):
                dim_active = True
            else:
                dim_active = False
            if taille_rect.collidepoint(event.pos):
                taille_active = True
            else:
                taille_active = False

        if event.type == pgm.KEYDOWN:
            if dim_active:
                # Ecrire dans les zones de textes
                if event.key == pgm.K_BACKSPACE:
                    dimentions = dimentions[:-1]  # Faire fonctionner la fleche retours
                else:
                    dimentions += event.unicode

            if taille_active:
                # Ecrire dans les zones de textes
                if event.key == pgm.K_BACKSPACE:
                    taille_pixels = taille_pixels[
                        :-1
                    ]  # Faire fonctionner la fleche retours
                else:
                    taille_pixels += event.unicode

            # Changer de grille
            if event.type == pgm.KEYDOWN and event.key == pgm.K_RIGHT:
                grille = new_grid(grille)
                compteur += 1  # Incrementation du compteur
                gen = arialFont.render(
                    f"Generation {compteur}", 1, (0, 0, 0)
                )  # Actualisation du compteur
```

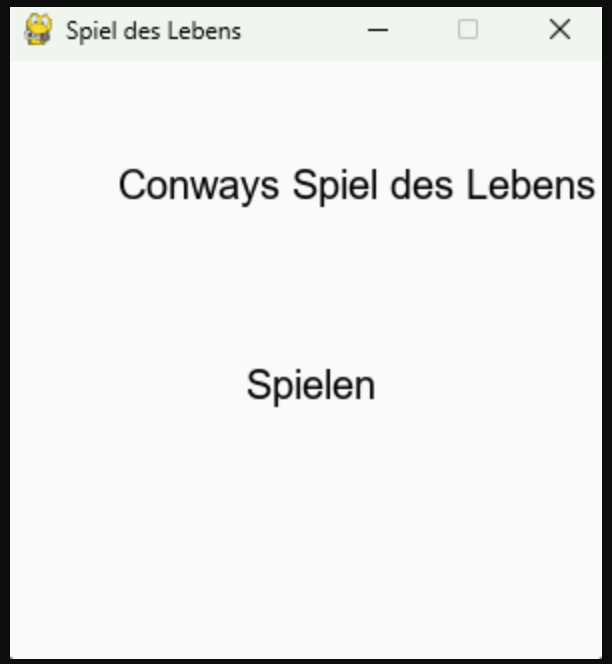
```python
screen.fill((250, 250, 250))  # Changer la couleur de fond

if parametre and not jouer:
    # Dessin des textes
    screen.blit(dim_txt, (10, 50))
    screen.blit(taille_txt, (10, 150))
    screen.blit(start, (100, 250))

    # Changer la couleur du cadre
    if dim_active:
        dim_color = (100, 200, 255)
    else:
        dim_color = (100, 100, 100)
    if taille_active:
        taille_color = (100, 200, 255)
    else:
        taille_color = (100, 100, 100)

    # Dessin des zones de textes
    pgm.draw.rect(screen, dim_color, dim_rect, 2)
    pgm.draw.rect(screen, taille_color, taille_rect, 2)

    # Afficher le texte capté
    dim_txt_surface = arialFont.render(dimentions, 1, (0, 0, 0))
    screen.blit(dim_txt_surface, (dim_rect.x + 5, dim_rect.y))
    taille_txt_surface = arialFont.render(taille_pixels, 1, (0, 0, 0))
    screen.blit(taille_txt_surface, (taille_rect.x + 5, taille_rect.y))

    # Elargir le rectangle au besoin
    dim_rect.w = max(100, dim_txt_surface.get_width() + 10)
    taille_rect.w = max(100, taille_txt_surface.get_width() + 10)
```

```python
# Commencer la simulation
elif jouer:
    screen = pgm.display.set_mode(
        (dimentions * taille_pixels, dimentions * taille_pixels + 50)
    )  # Afficher la fenetre
    screen.fill((250, 250, 250))  # Changer la couleur de fond
    # Parcourir la grille
    for y in range(len(grille)):
        for x in range(len(grille[y])):
            if grille[y][x] == 1:
                pgm.draw.rect(
                    screen,
                    (0, 0, 0),
                    [
                        x * taille_pixels,
                        y * taille_pixels,
                        taille_pixels,
                        taille_pixels,
                    ],
                )
            else:
                pgm.draw.rect(
                    screen,
                    (0, 0, 0),
                    [
                        x * taille_pixels,
                        y * taille_pixels,
                        taille_pixels,
                        taille_pixels,
                    ],
                    1,
                )
    screen.blit(
        gen, (5, dimentions * taille_pixels + 5)
    )  # Affichage de la zone du numéro de la generation
# Ecran titre
else:
    screen.blit(titre, (50, 50))
    screen.blit(play, (120, 150))

pgm.display.flip()  # Actualisation de la fenetre
clock.tick(60)  # 60 images par secondes
```

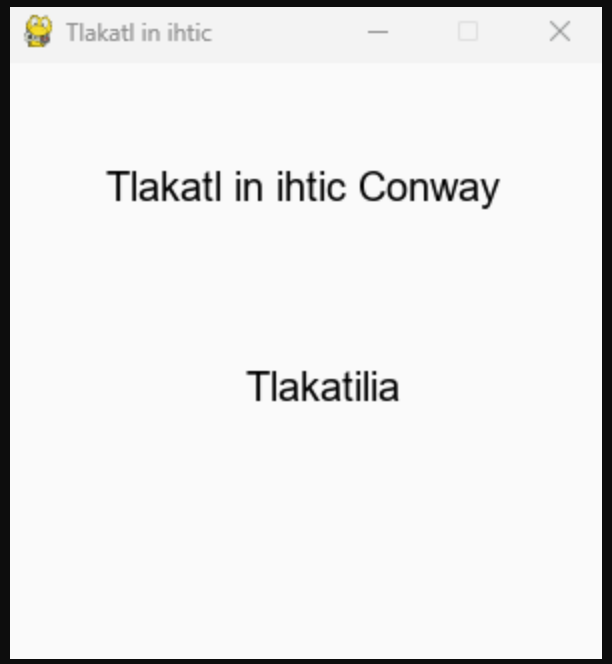
Version Traduite

Plus de 10 langue disponible

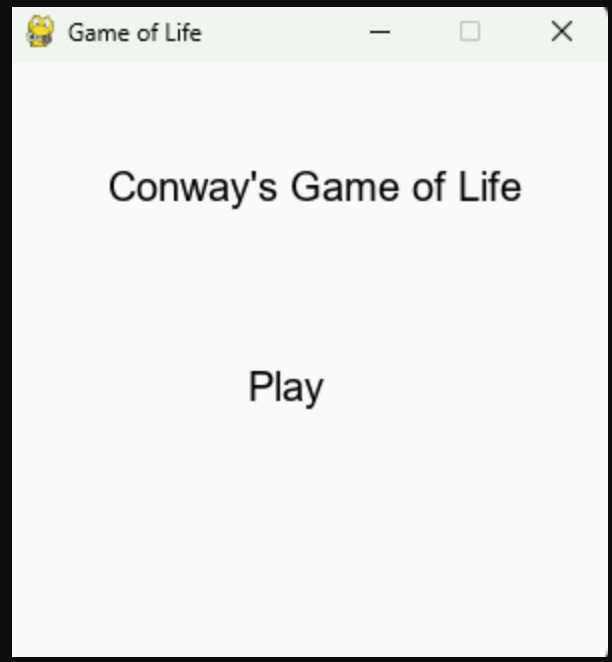
Allemand



Azteque

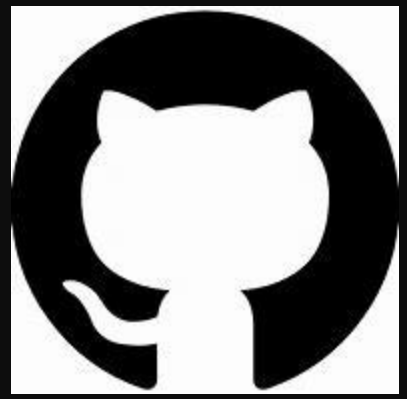


Anglais



Logiciels utilisés

GitHub



Git

Visual Studio Code

Applaudissez

Arrêtez

Reprenez