



**CENTRO DE CIENCIAS BÁSICAS**  
**DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN**  
**AUTÓMATAS II**  
**7° "A"**

**Juego de la Vida**

**Dr. Francisco Javier Ornelas Zapata**

**Alumnos:**

**Almeida Ortega Andrea Melissa**  
**Espinoza Sánchez Joel Alejandro**  
**Flores Fernández Óscar Alonso**  
**Gómez Garza Dariana**  
**González Arenas Fernando Francisco**  
**Orocio García Hiram Efraín**

**Fecha de Entrega:** Aguascalientes, Ags., **26** de noviembre de 2021

En 1970, el matemático británico John Conway creó su **"Juego de la vida"**, un conjunto de reglas que imita el crecimiento caótico pero modelado de una colonia de organismos biológicos.

El "juego" tiene lugar en una cuadrícula bidimensional que consiste en células "vivas" y "muertas", y las reglas para pasar de generación en generación son simples:

- \* Sobrepoblación: si una célula viva está rodeada por más de tres células vivas, muere.
- \* Estasis: si una célula viva está rodeada por dos o tres células vivas, sobrevive.
- \* Subpoblación: si una célula viva está rodeada por menos de dos células vivas, muere.
- \* Reproducción: si una célula muerta está rodeada por exactamente tres células, se convierte en una célula viva.

Al hacer cumplir estas reglas en pasos secuenciales, pueden aparecer patrones hermosos e inesperados.

El Juego de la Vida se realizó con el siguiente código realizado en Python:

```
import pygame
import time, os
import numpy as np

nxC = 30
nyC = 30

os.environ["SDL_VIDEO_CENTERED"] = "1"

pygame.init()

pygame.display.set_caption("Juego de la vida ")

width, height = 700, 700

screen = pygame.display.set_mode((height, width))
```

```

# Color de fondo, casi negro
bg = 25, 25, 25

screen.fill(bg)

# Ancho y alto de cada celda
dimCW = width / nxC
dimCH = height / nyC

# Estructura de datos que contiene todos los estados de las diferentes celdas
# Estados de las celdas: Vivas = 1 - Muertas = 0
# Inicializo matriz con ceros
gameState = np.zeros((nxC, nyC))

# Control de la ejecución - En True se inicia pausado (Para poder ver la forma
inicial de los autómatas):
pauseExec = True

# Controla la finalización del juego:
endGame = False

# Acumulador de cantidad de iteraciones:
iteration = 0

while not endGame:

    newGameState = np.copy(gameState)

    screen.fill(bg)

    time.sleep(0.1)

    ev = pygame.event.get()

    # Contador de población:

```

```

population = 0

for event in ev:

    if event.type == pygame.QUIT:
        endGame = True
        break

    if event.type == pygame.KEYDOWN:

        # Si tocan escape finalizo el juego
        if event.key == pygame.K_ESCAPE:
            endGame = True
            break

        # Si tocan la tecla r limpio la grilla, reseteo población e iteración y pongo
        pausa
        if event.key == pygame.K_r:
            iteration = 0
            gameState = np.zeros((nxC, nyC))
            newGameState = np.zeros((nxC, nyC))
            pauseExec = True
        else:
            # Si tocan cualquier tecla no contemplada, pauso o reanudo el juego
            pauseExec = not pauseExec

        # Detección de click del mouse:
        mouseClick = pygame.mouse.get_pressed()

        # Obtención de posición del cursor en la pantalla:
        # Si se hace click con cualquier botón del mouse, se obtiene un valor en
        mouseClick mayor a cero
        if sum(mouseClick) > 0:

            # Click del medio pausa / reanuda el juego
            if mouseClick[1]:

```

```

    pauseExec = not pauseExec

else:

    # Obtengo las coordenadas del cursor del mouse en pixeles
    posX, posY, = pygame.mouse.get_pos()

    # Convierto de coordenadas en pixeles a celda clickeada en la grilla
    celX, celY = int(np.floor(posX / dimCW)), int(np.floor(posY / dimCH))

    # Click izquierdo y derecho permutan entre vida y muerte
    newGameState[celX, celY] = not gameState[celX, celY]

if not pauseExec:
    # Incremento el contador de generaciones
    iteration += 1

# Recorro cada una de las celdas generadas
for y in range(0, nxC):
    for x in range(0, nyC):

        if not pauseExec:

            # Cálculo del número de vecinos cercanos
            n_neigh = (
                gameState[(x - 1) % nxC, (y - 1) % nyC]
                + gameState[x % nxC, (y - 1) % nyC]
                + gameState[(x + 1) % nxC, (y - 1) % nyC]
                + gameState[(x - 1) % nxC, y % nyC]
                + gameState[(x + 1) % nxC, y % nyC]
                + gameState[(x - 1) % nxC, (y + 1) % nyC]
                + gameState[x % nxC, (y + 1) % nyC]
                + gameState[(x + 1) % nxC, (y + 1) % nyC]
            )

```

```

# Una célula muerta con exactamente 3 células vecinas vivas "nace"
# (es decir, al turno siguiente estará viva).
if gameState[x, y] == 0 and n_neigh == 3:
    newGameState[x, y] = 1

# Una célula viva con 2 o 3 células vecinas vivas sigue viva,
# en otro caso muere (por "soledad" o "superpoblación")
elif gameState[x, y] == 1 and (n_neigh < 2 or n_neigh > 3):
    newGameState[x, y] = 0

# Incremento el contador de población:
if gameState[x, y] == 1:
    population += 1

# Creación del polígono de cada celda a dibujar
poly = [
    (int(x * dimCW), int(y * dimCH)),
    (int((x + 1) * dimCW), int(y * dimCH)),
    (int((x + 1) * dimCW), int((y + 1) * dimCH)),
    (int(x * dimCW), int((y + 1) * dimCH)),
]

if newGameState[x, y] == 0:
    # Dibujado de la celda para cada par de x e y:
    # screen      -> Pantalla donde dibujar
    # (128, 128, 128) -> Color a utilizar para dibujar, en este caso un gris
    # poly        -> Puntos que definan al polígono que se está dibujando
    pygame.draw.polygon(screen, (128, 128, 128), poly, 1)
else:
    if pauseExec:
        # Con el juego pausado pinto de gris las celdas
        pygame.draw.polygon(screen, (128, 128, 128), poly, 0)
    else:
        # Con el juego ejecutándose pinto de blanco las celdas
        pygame.draw.polygon(screen, (255, 255, 255), poly, 0)

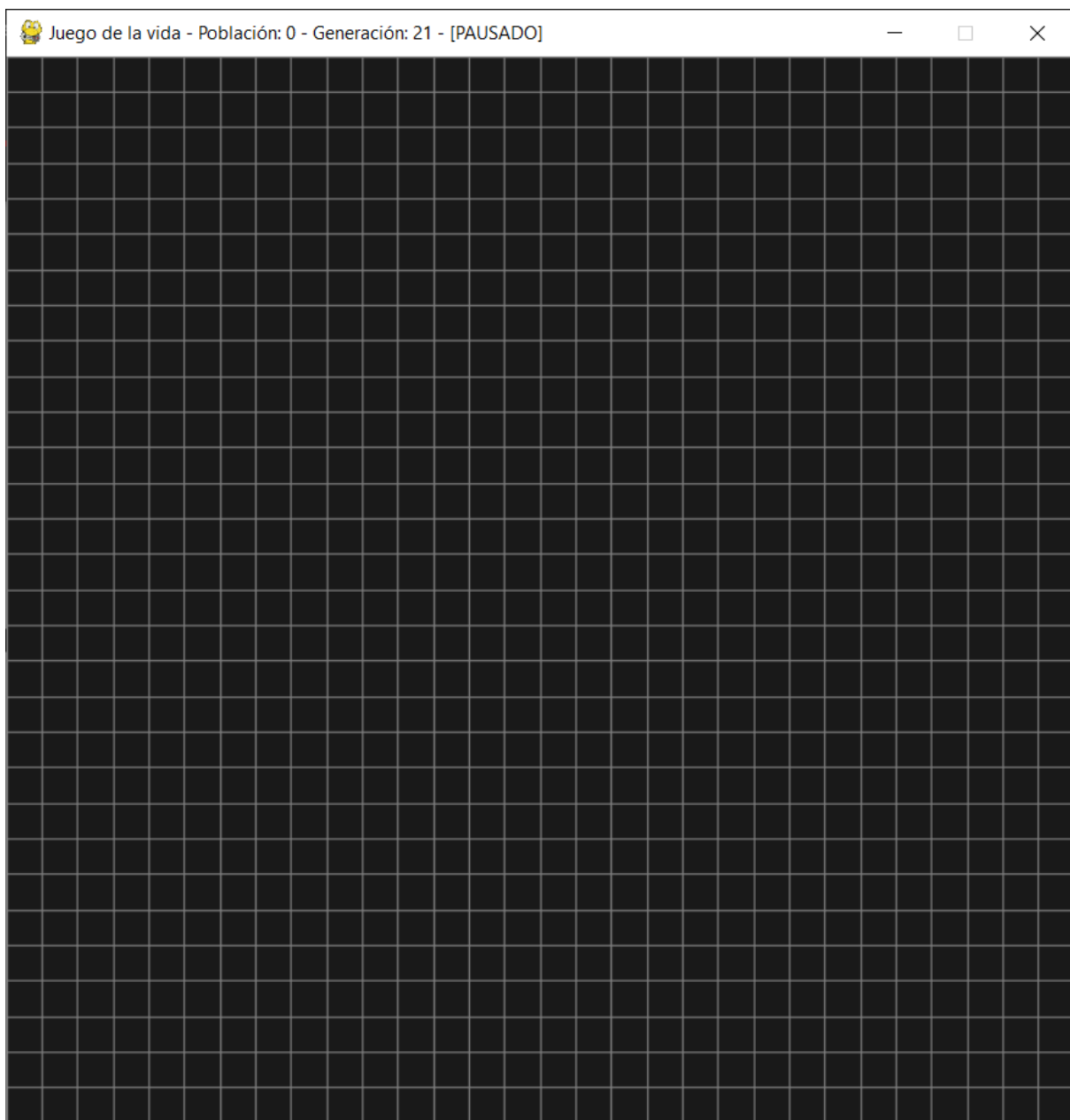
```

```
# Actualizo el título de la ventana
title = f"Juego de la vida - Población: {population} - Generación: {iteration}"
if pauseExec:
    title += " - [PAUSADO]"
pygame.display.set_caption(title)
print(title)

# Actualizo gameState
gameState = np.copy(newGameState)

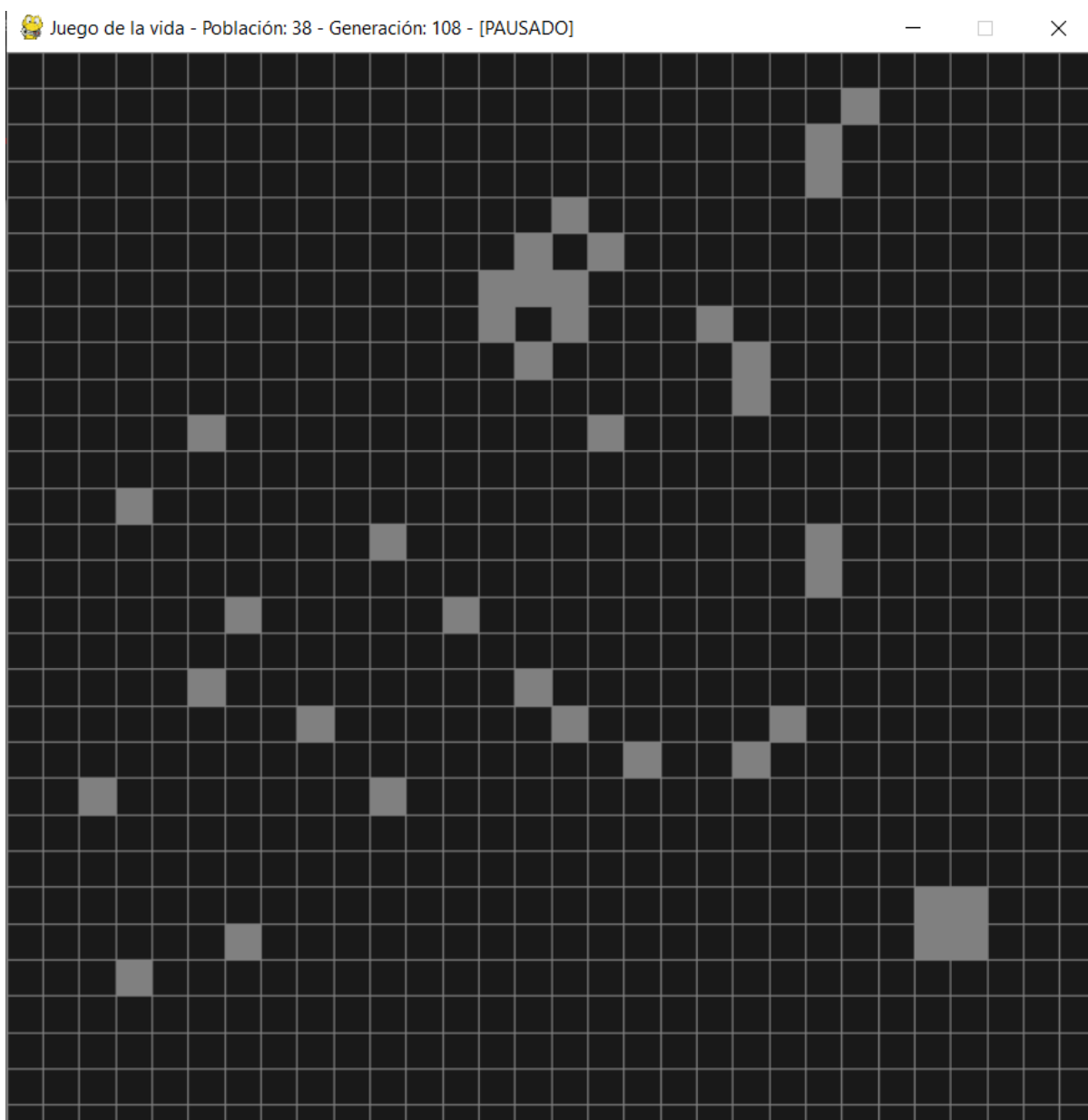
# Muestro y actualizo los fotogramas en cada iteración del bucle principal
pygame.display.flip()
```

## Inicio del programa






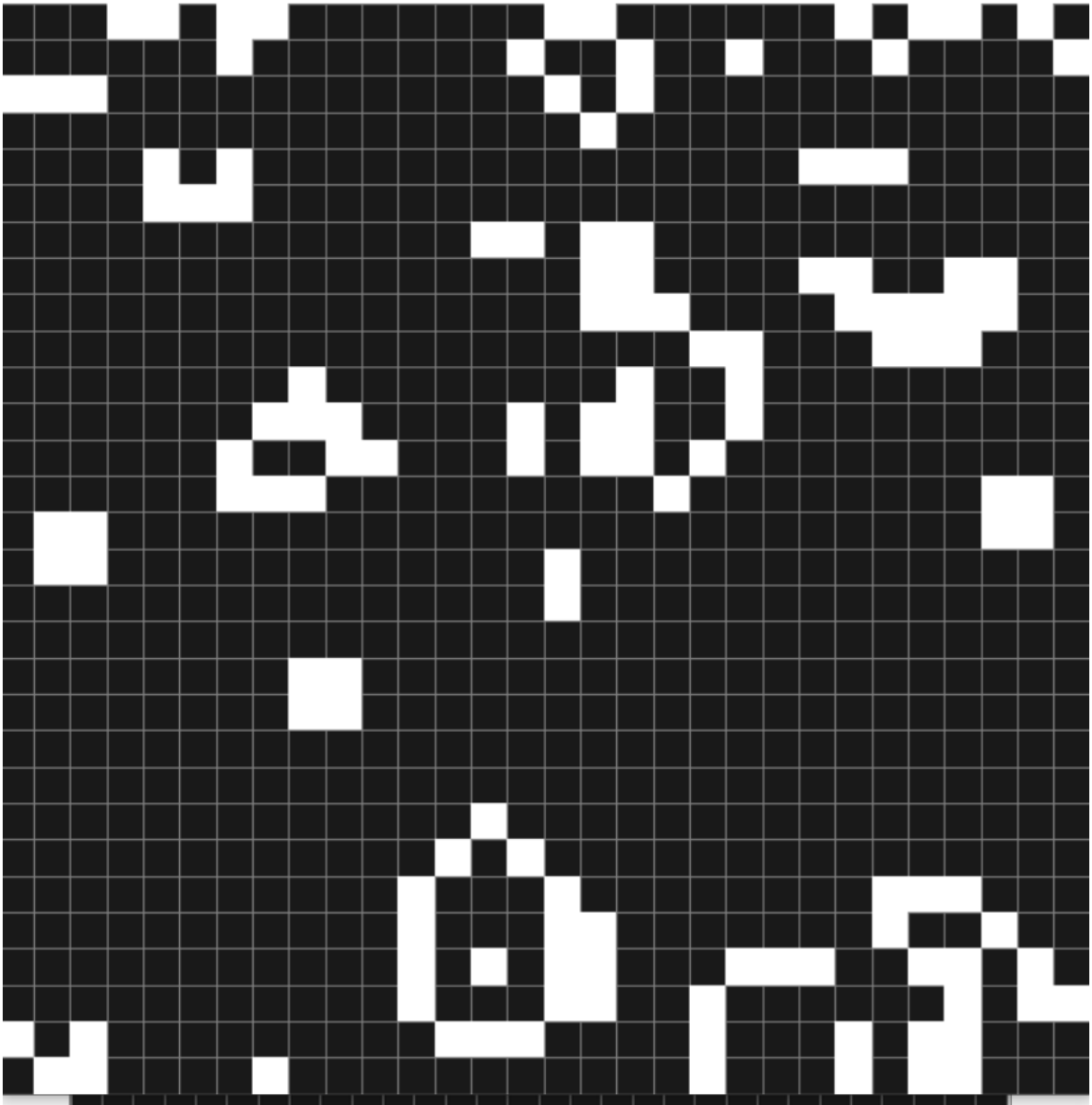
## Población creada por el usuario



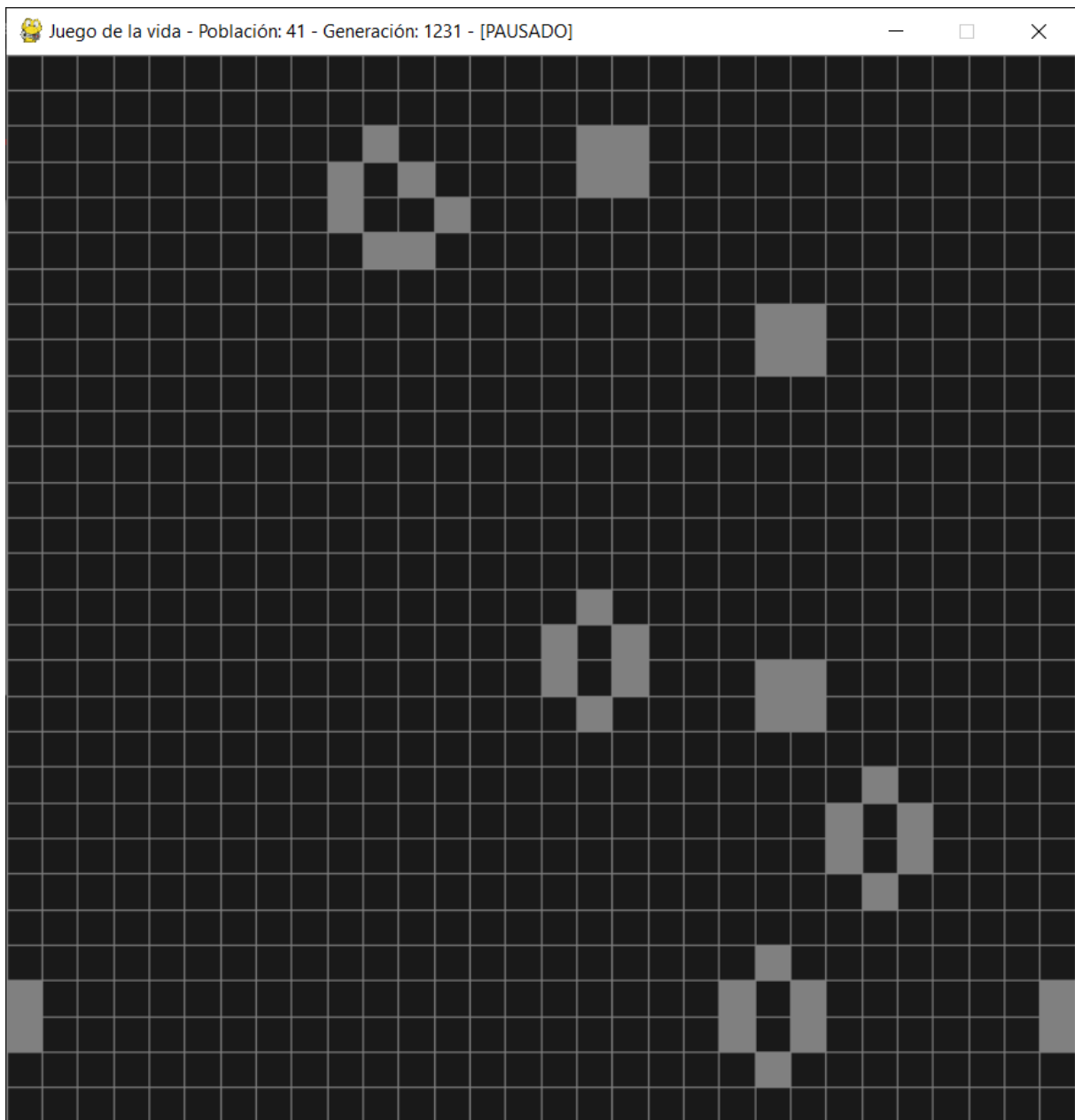
## Avance

 Juego de la vida - Población: 133 - Generación: 507

— □ ×



## Final



## Referencias

Programa el Juego de La Vida. . . en 10 MINUTOS! (2020, 19 abril). YouTube.  
<https://www.youtube.com/watch?v=qPtKv9fSHZY>