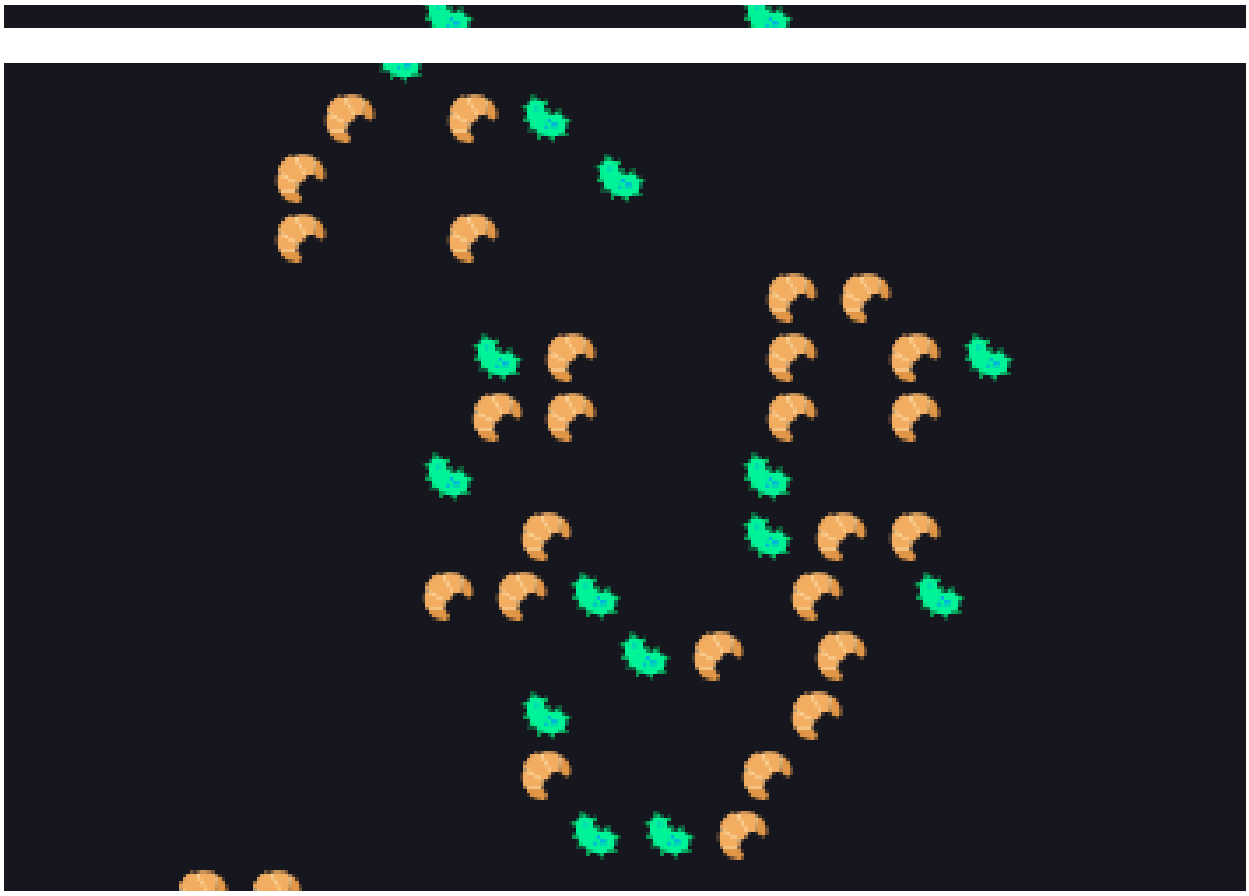


Juego de la vida de conway

15/01/2024

—

Javier Serna Huertas

“Juego de la vida”

El Juego de la vida es un autómata celular diseñado por el matemático británico John Horton Conway en 1970. Es un juego de cero jugadores, en el que su evolución es determinada por un estado inicial, sin requerir intervención adicional. Se considera un sistema Turing completo que puede simular cualquier otra Máquina de Turing.

Reglas

1. Cualquier célula viva con menos de dos "vecinos vivos" muere (SUBPOBLACIÓN)
2. Cualquier célula con 2 o 3 "vecinos vivos" vive para la siguiente generación (SIMBIOSIS)
3. Cualquier célula con más de 3 "vecinos vivos" muere (SUPERPOBLACIÓN)
4. Cualquier célula muerta con 3 "vecinos vivos" revive (REPRODUCCIÓN)

EL código “”

1. Importación de módulos:
    - copy: Se utiliza para realizar copias profundas de objetos, especialmente para duplicar el estado del tablero en cada generación.
    - os: Proporciona una forma de interactuar con el sistema operativo, en este caso, se usa para limpiar la pantalla de la consola.
    - time: Se utiliza para introducir un breve retraso entre generaciones, dando la sensación de animación.
    - platform: Se utiliza para determinar el sistema operativo y, así, realizar una limpieza de pantalla compatible.

2. Función inicializador_tablero(filas, columnas):

juego de la vida - pygameV1.py

```
7  def inicializador_tablero(filas, columnas):
8      # Inicializar el tablero con una célula viva en el centro
9      tablero = [[0] * columnas for _ in range(filas)]
10     centro_fila, centro_columna = filas // 2, columnas // 2
11     tablero[centro_fila][centro_columna] = 1
12     return tablero
```

Crea un tablero cuadrado de tamaño filas x columnas e inicializa una célula viva en el centro del tablero. Devuelve el tablero.

3. Función imprimir_tablero(tablero)

juego de la vida - pygameV1.py

```
14 def imprimir_tablero(tablero):
15     for fila in tablero:
16         print(' '.join(['🦠' if celda else ' ' for celda in fila]))
```

Imprime en la consola el estado actual del tablero, representando las células vivas con un emoji 🦠 y las células muertas con un espacio.

4. contar_vecinos(tablero, fila, columna)

juego de la vida - pygameV1.py

```python
def contar_vecinos(tablero, fila, columna):
    filas, columnas = len(tablero), len(tablero[0])
    vecinos = [
        (fila + i, columna + j)
        for i in range(-1, 2)
        for j in range(-1, 2)
        if i != 0 or j != 0
    ]
    contar = 0
    for f, c in vecinos:
        if 0 <= f < filas and 0 <= c < columnas and tablero[f][c] == 1:
            contar += 1
    return contar
```

Cuenta el número de células vivas alrededor de una célula en la posición (fila, columna) en el tablero. Utiliza una vecindad de 8 células (horizontal, vertical y diagonal).

5. evolucionar(tablero, generacion_actual)

juego de la vida - pygameV1.py

```python
def evolucionar(tablero, generacion_actual):
    nuevo_tablero = copy.deepcopy(tablero)
    filas, columnas = len(tablero), len(tablero[0])

    for fila in range(filas):
        for columna in range(columnas):
            vecinos_vivos = contar_vecinos(tablero, fila, columna)

            if tablero[fila][columna] == 1:  # Célula viva
                if vecinos_vivos < 2 or vecinos_vivos > 3:
                    nuevo_tablero[fila][columna] = 0  # Muere por subpoblación o superpoblación
            else:  # Célula muerta
                if vecinos_vivos == 3:
                    nuevo_tablero[fila][columna] = 1  # Revive por reproducción

    return nuevo_tablero
```

Calcula la siguiente generación del juego de la vida de Conway aplicando las reglas del juego a cada célula del tablero. Devuelve el nuevo tablero.

6. Main

juego de la vida - pygameV1.py

```python
if __name__ == "__main__":
    filas, columnas = 25, 50
    tablero = inicializador_tablero(filas, columnas)

    generaciones = 0
    while True:
        os.system('cls' if platform.system() == 'Windows' else 'clear')
        imprimir_tablero(tablero)
        tablero = evolucionar(tablero, generacion_actual=generaciones)
        time.sleep(0.5)

```

Es la parte principal del script que se ejecuta cuando se ejecuta el archivo directamente (no cuando se importa como un módulo). Aquí se establece el tamaño inicial del tablero, se crea el tablero inicial, y se ejecuta un bucle infinito que muestra y actualiza el tablero con ciertos intervalos de tiempo.

Resultados

V1 “Muerte en soledad”

Es una versión del juego de la vida la cual solo generará una única bacteria la cual morirá debido a que es incapaz de reproducirse o de hacer simbiosis.

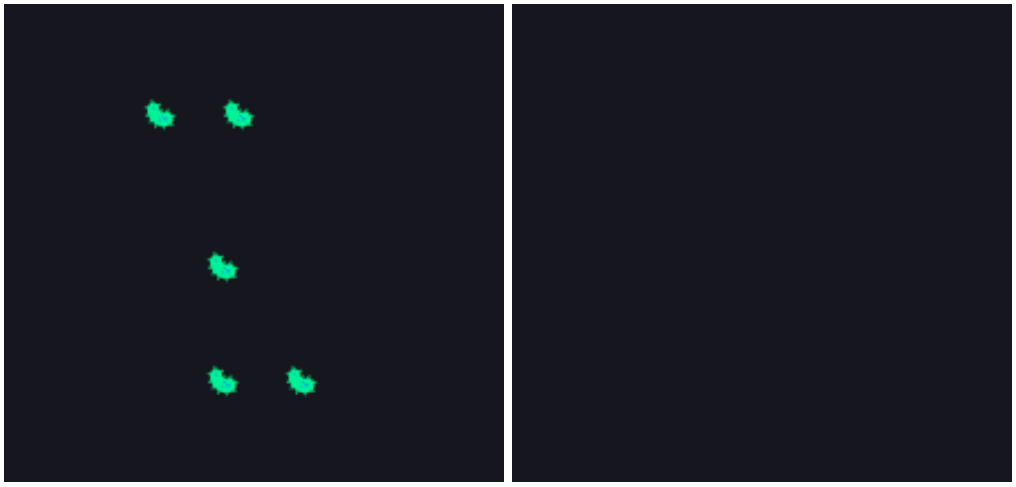
En la primera imagen se puede ver cómo se genera una bacteria de mediante el la función inicializador_tablero.

En la segunda imagen la bacteria muere debido a que no cumple ningún requisito para poder reproducirse o persistir.

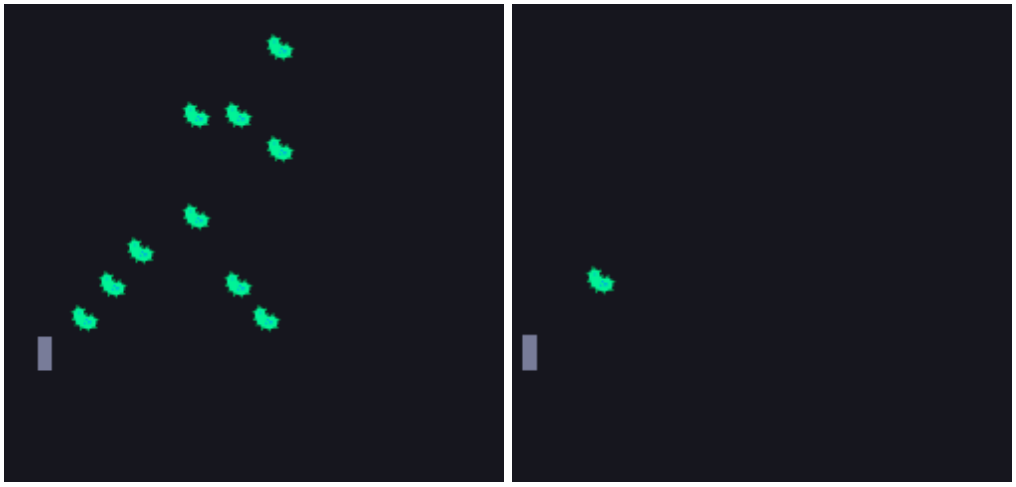
[1] Véase una bacteria sola. (Un claro ejemplo de la vida de un programador de visual basic)

V2 “Tal vez con amigos”

Esta segunda versión cambia el número de generación de bacterias entre 5 y 10 y podemos observar las primera interacciones complejas entre ellas.



En esta casuística podemos ver cómo se generan 5 bacterias pero debido a que ninguna cumple ninguna de las casuística necesarias para la supervivencia mueren.



En esta generación se puede ver como una de estas células consigue hacer simbiosis pero no consigue multiplicarse.

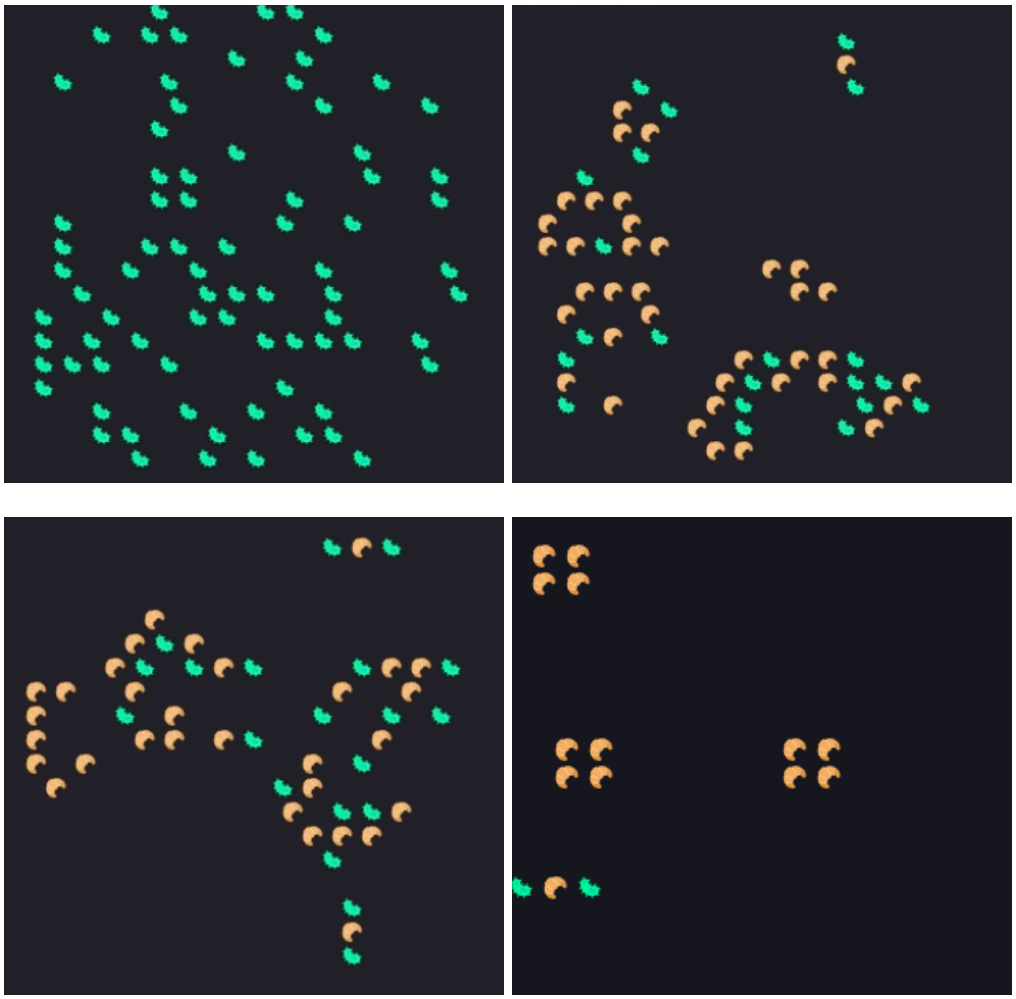
Al generar tampocos bacterias es muy difícil que haya interacciones muy complejas y estas interacciones son inversamente proporcionales al tamaño del campo.

V3 “La vida sin frenos”

En esta versión no está limitada la cantidad de bacterias generadas y el campo es mucho mayor.

Además de ha añadido una 5 regla “La evolución” ya que la vida tiende a evolucionar y no siempre como nos gustaría.

Esta regla consiste en que si una bacteria a sobrevivido más de 10 generaciones esta evolucionara a un 🥐[2]



1. imagen: Se puede contemplar la primera generación de bacterias
2. imagen: Generación 10 muchas bacterias han muerto pero otras ha vivido lo suficiente como para evolucionar.
3. imagen: Generación 30 las bacterias han seguido su curso muriendo y sobreviviendo
4. imagen: Generación 500 han llegado a un equilibrio, formas estables.

[2] Si, a así es como nacen los franceses

La evolución

A continuación explicaré los cambios realizados para generar la 5 regla.

1. inicializador_tablero(filas, columnas, probabilidad_vida=0.2): En lugar de inicializar el tablero con una célula viva en el centro, este inicializador utiliza una probabilidad (probabilidad_vida) para asignar aleatoriamente el estado inicial de cada célula (viva, muerta o especial). Se puede ajustar la probabilidad para cambiar la densidad de células vivas en el tablero.

juego de la vida - pygameV3.py

```
9  def inicializador_tablero(filas, columnas, probabilidad_vida=0.2):
10     return [[(1, 0) if random.random() < probabilidad_vida else (0, 0) for _ in range(columnas)] for _ in range(filas)]
```

2. imprimir_tablero(tablero, generacion_actual): Además de imprimir el estado del tablero, ahora muestra la generación actual. Las células especiales ahora se representan con el emoji 🥐. También se ha agregado un tercer estado posible para las células (2), que cambian a este estado después de 10 generaciones de vida.

juego de la vida - pygameV3.py

```
12 def imprimir_tablero(tablero, generacion_actual):
13     print(f"Generación: {generacion_actual}")
14     for fila in tablero:
15         print(' '.join(['🦕' if celda[0] == 1 else ' ' if celda[0] == 0 else '🥐' if celda[0] == 2 else ' ' for celda in fila]))
```

3. evolucionar(tablero, generacion_actual): Se han introducido cambios en las reglas del juego. Ahora, después de 10 generaciones de vida, las células vivas cambian a un estado especial (representado por 🥐). Además, las células muertas ahora pueden revivir si tienen exactamente 3 vecinos vivos, y las células vivas ahora pueden morir después de un tiempo específico (en lugar de depender solo del número de vecinos).

juego de la vida - pygameV3.py

```python
def evolucionar(tablero, generacion_actual):
    nuevo_tablero = copy.deepcopy(tablero)
    filas, columnas = len(tablero), len(tablero[0])

    for fila in range(filas):
        for columna in range(columnas):
            estado, generaciones = tablero[fila][columna]

            vecinos_vivos = contar_vecinos(tablero, fila, columna)

            if estado == 1 or estado == 2:  # Célula viva
                if vecinos_vivos < 2 or vecinos_vivos > 3:
                    nuevo_tablero[fila][columna] = (0, 0)  # Muere por subpoblación o superpoblación
                # elif generacion_actual - generaciones >= 10:
                elif generaciones >= 10:
                    nuevo_tablero[fila][columna] = (2, generacion_actual)  # Cambia a '🥐' después de 10 generaciones de vida
                else:
                    nuevo_tablero[fila][columna] = (1, generaciones + 1)  # Incrementa el contador de generaciones
            else:  # Célula muerta
                if vecinos_vivos == 3:
                    nuevo_tablero[fila][columna] = (1, generacion_actual)  # Revive por reproducción

    return nuevo_tablero
```