Algorítimica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Numpy

Contenido

- Repaso sobre arreglos bi-dimensionales
- Matrices

Creación de matrices

```
In [31]: import numpy as np

In [32]: A = np.zeros([4,3])

In [33]: A
Out[33]:
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])

In [34]: B = np.random.rand(3,4)

In [35]: B
Out[35]:
array([[7.46810605e-01, 6.81163002e-02, 5.31811809e-01, 5.58997055e-01],
       [7.76227349e-01, 2.45910055e-01, 8.22781075e-01, 7.53557845e-02],
       [7.45266221e-01, 9.35181310e-04, 8.81995072e-01, 9.55490950e-01]])
```

ITAM

Creación de matrices

```
In [36]: C = np.array([[3,1,5,6], [1,2,3,4], [8, 9, 10, 12]])

In [37]: C
Out[37]:
array([[ 3,  1,  5,  6],
       [ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])
```

Creación de matrices

```
In [36]: C = np.array([[3,1,5,6], [1,2,3,4], [8, 9, 10, 12]])

In [37]: C
Out[37]:
array([[ 3,  1,  5,  6],
       [ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])
```

Leyendo información

```
In [45]: B
Out[45]:
array([[7.46810605e-01, 6.81163002e-02, 5.31811809e-01, 5.58997055e-01],
       [7.76227349e-01, 2.45910055e-01, 8.22781075e-01, 7.53557845e-02],
       [7.45266221e-01, 9.35181310e-04, 8.81995072e-01, 9.55490950e-01]])

In [46]: B[0,0]
Out[46]: 0.746810604543355

In [47]: B.shape[0]
Out[47]: 3

In [48]: B.shape[1]
Out[48]: 4

In [49]: B[B.shape[0]-1,B.shape[1]-1]
Out[49]: 0.955490950475630 4
```

Leyendo información

```
In [45]: B
Out[45]:
array([[7.46810605e-01, 6.81163002e-02, 5.31811809e-01, 5.58997055e-01],
       [7.76227349e-01, 2.45910055e-01, 8.22781075e-01, 7.53557845e-02],
       [7.45266221e-01, 9.35181310e-04, 8.81995072e-01, 9.55490950e-01]])
```

```
In [50]: B[1,3]
Out[50]: 0.07535578446509494

In [51]: B[2,3]
Out[51]: 0.9554909504756304

In [52]: B[2,4]
Traceback (most recent call last):

  File "<ipython-input-52-40c0fa2f7200>", line 1, in <module>
    B[2,4]

IndexError: index 4 is out of bounds for axis 1 with size 4
```

ITAM

Leyendo información con operador :

```
In [55]: C
Out[55]:
array([[ 3,  1,  5,  6],
       [ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])

In [56]: C[:,:]
Out[56]:
array([[ 3,  1,  5,  6],
       [ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])

In [57]: C[1:,:]
Out[57]:
array([[ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])

In [58]: C[:1,:]
Out[58]: array([[3, 1, 5, 6]])

In [59]: C[:,2:]
Out[59]:
array([[ 5,  6],
       [ 3,  4],
       [10, 12]])

In [60]: C[:,:2]
Out[60]:
array([[3, 1],
       [1, 2],
       [8, 9]])
```

```
In [65]: C[0,:]
Out[65]: array([3, 1, 5, 6])

In [66]: C[1,:]
Out[66]: array([1, 2, 3, 4])

In [67]: C[2,:]
Out[67]: array([ 8,  9, 10, 12])

In [68]: C[:,0]
    ...:
Out[68]: array([3, 1, 8])

In [69]: C[:,1]
Out[69]: array([1, 2, 9])

In [70]: C[:,2]
Out[70]: array([ 5,  3, 10])

In [71]: C[:,3]
Out[71]: array([ 6,  4, 12])
```

ITAM

Leyendo información con operador :

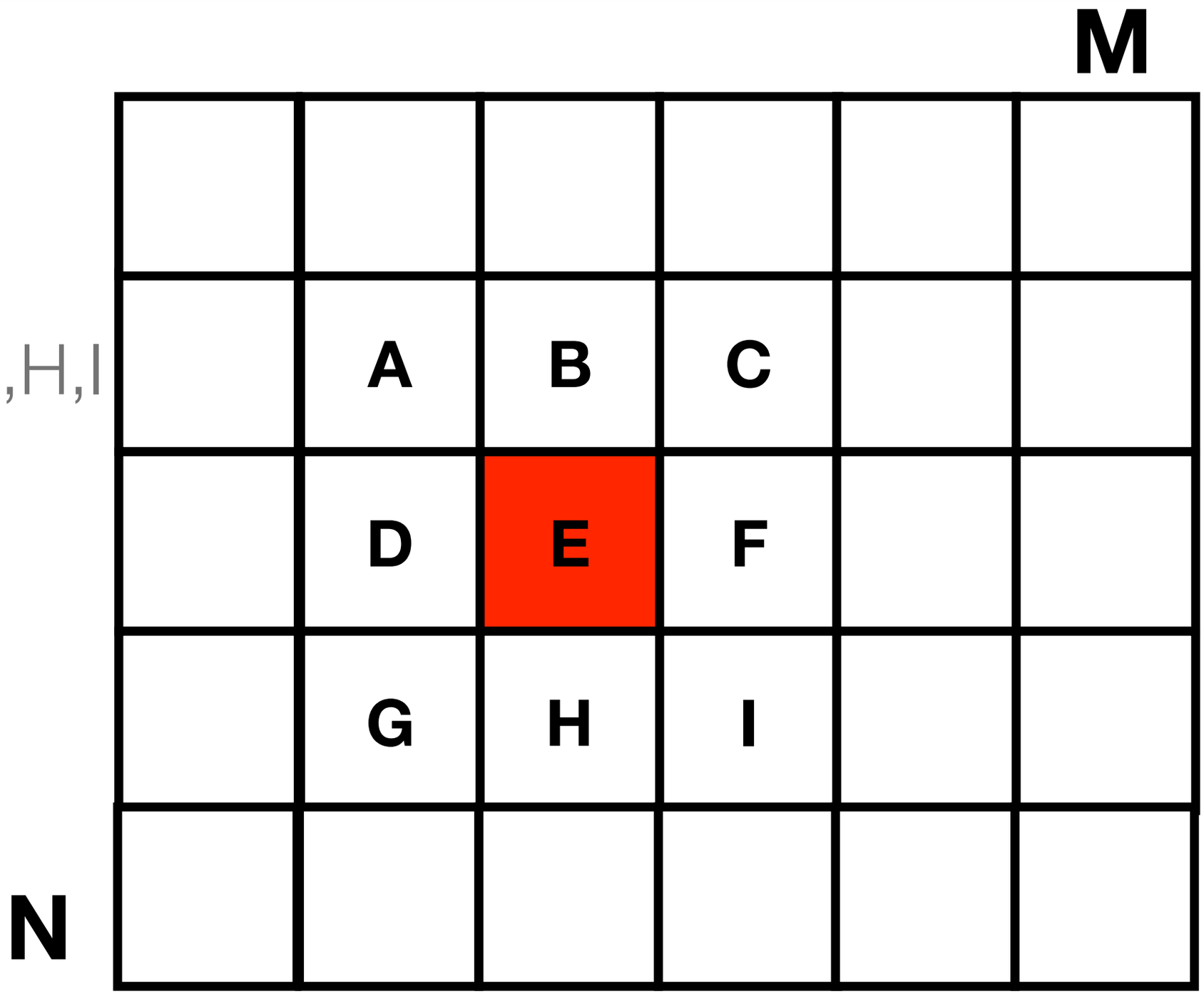
```
In [76]: C
Out[76]:
array([[ 3,  1,  5,  6],
       [ 1,  2,  3,  4],
       [ 8,  9, 10, 12]])

In [77]: C[0:2,:]
Out[77]:
array([[3, 1, 5, 6],
       [1, 2, 3, 4]])

In [78]: C[0:2,1:3]
Out[78]:
array([[1, 5],
       [2, 3]])

In [79]: C[0:2,1:2]
Out[79]:
array([[1],
       [2]])
```

Ejercicios para llevar - practicar para el examen

- Desarrollar funciones para calcular:
  - Máximo y mínimo
    - por renglon y por columna
  - Suma
    - por renglones y por columna

Concepto de vecindad

- Matriz N x M
- Vecinos de E
  - 8: A,B,C,D,F,G,H,I
  - 4: B, D, F, H



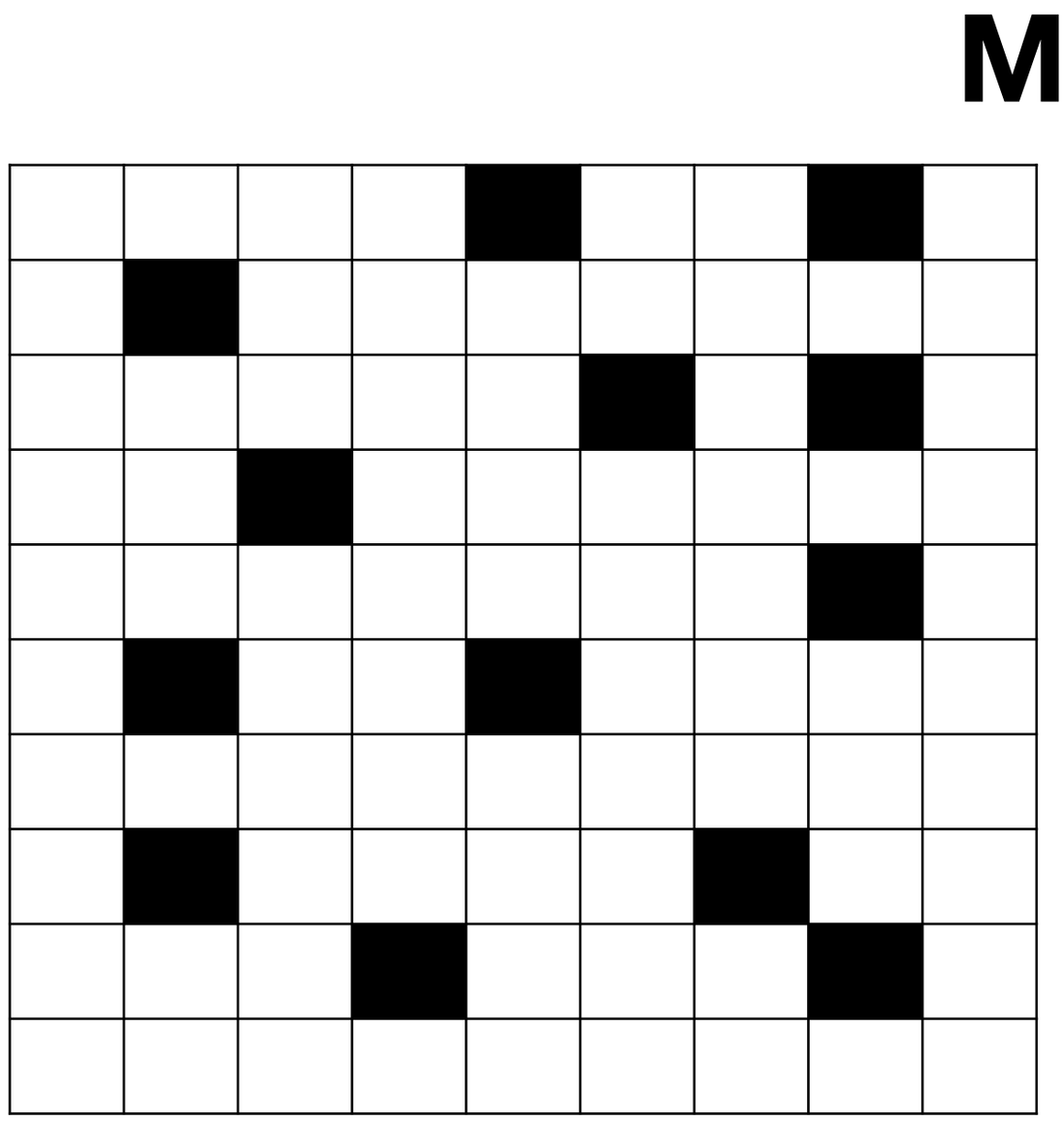
Ejercicio en clase

- Game of life
- Diseñado por el matemático británico John Horton Conway en 1970
- Estudiar conceptos de emergencia y autoorganización
- Observar surgimiento de patrones complejos a partir de de reglas muy sencillas y transiciones generacionales

ITAM

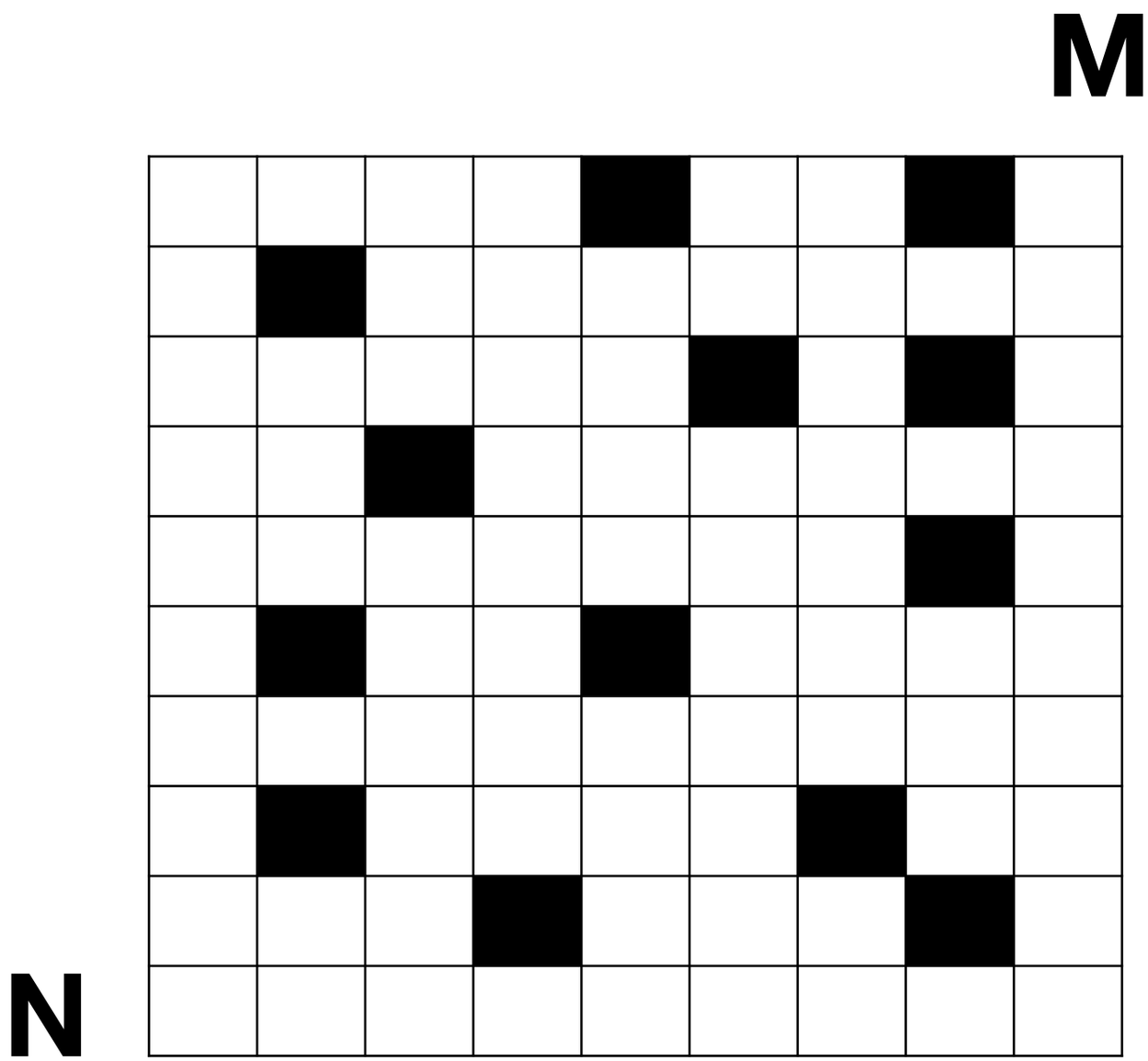
Game of life

- Un juego …. con cero jugadores
- Un **mundo** cuadricular (una matriz) de N x M
- Cada cuadro representa una **célula**: viva o muerta



Game of life

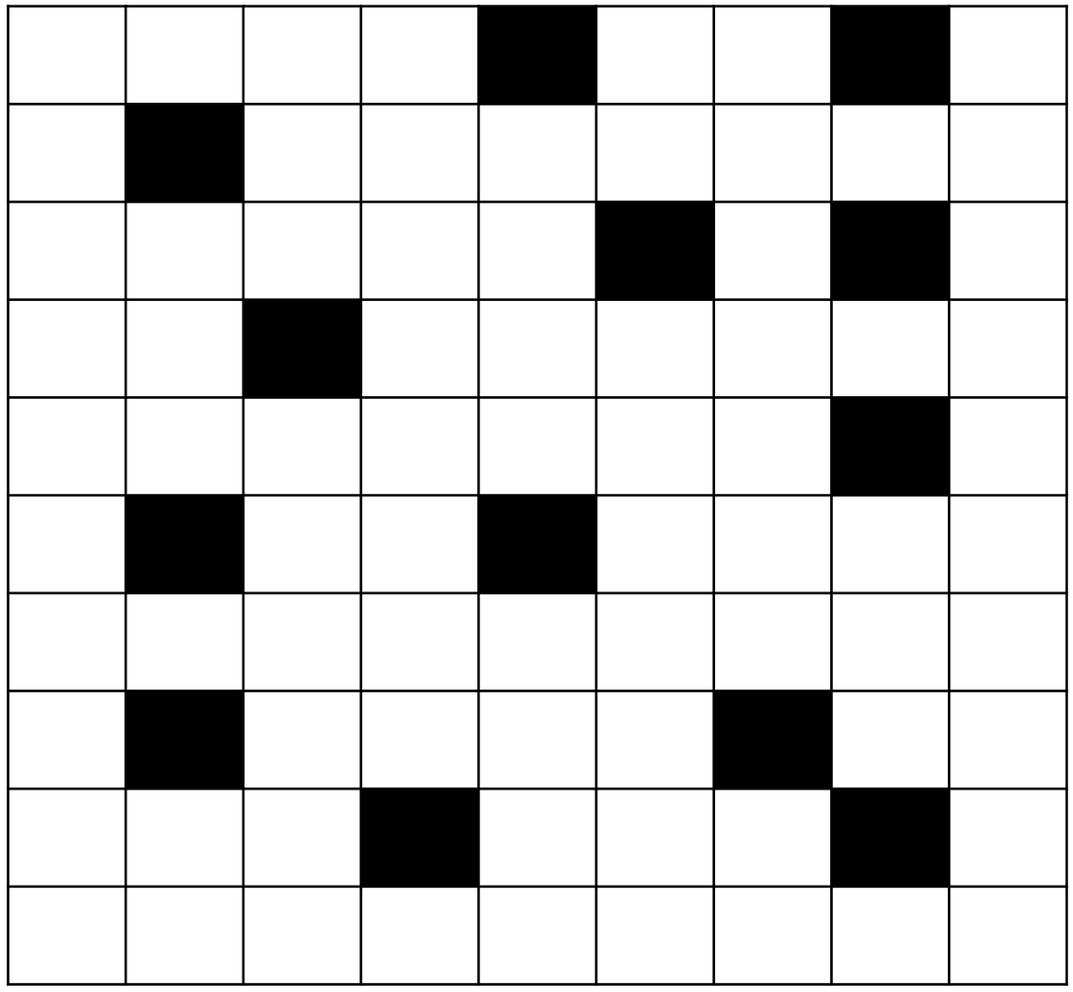
- El estado de una célula cambia de “generación” en “generación”
- El cambio está dado por ciertas reglas



Game of life

- Una célula muerta con exactamente 3 células vecinas vivas “re-nace" (es decir, al turno siguiente estará viva).
- Una célula viva con 2 o 3 células vecinas vivas sigue viva
- Una célula viva muere:
  - por “soledad” (menos de 2 vecinos vivos)
  - “sobrepoblación" (más de 3 vecinos vivos)

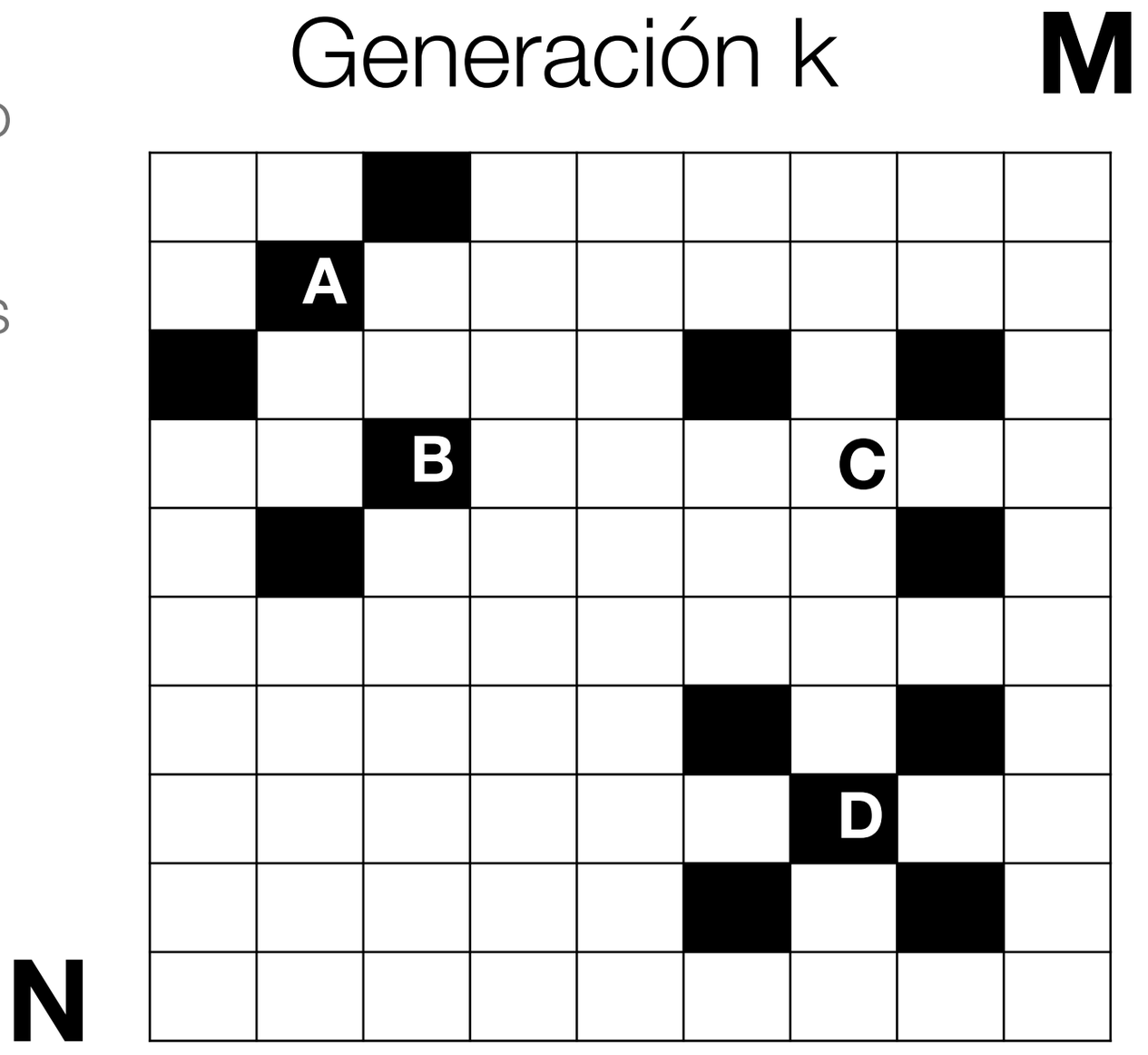
M



N

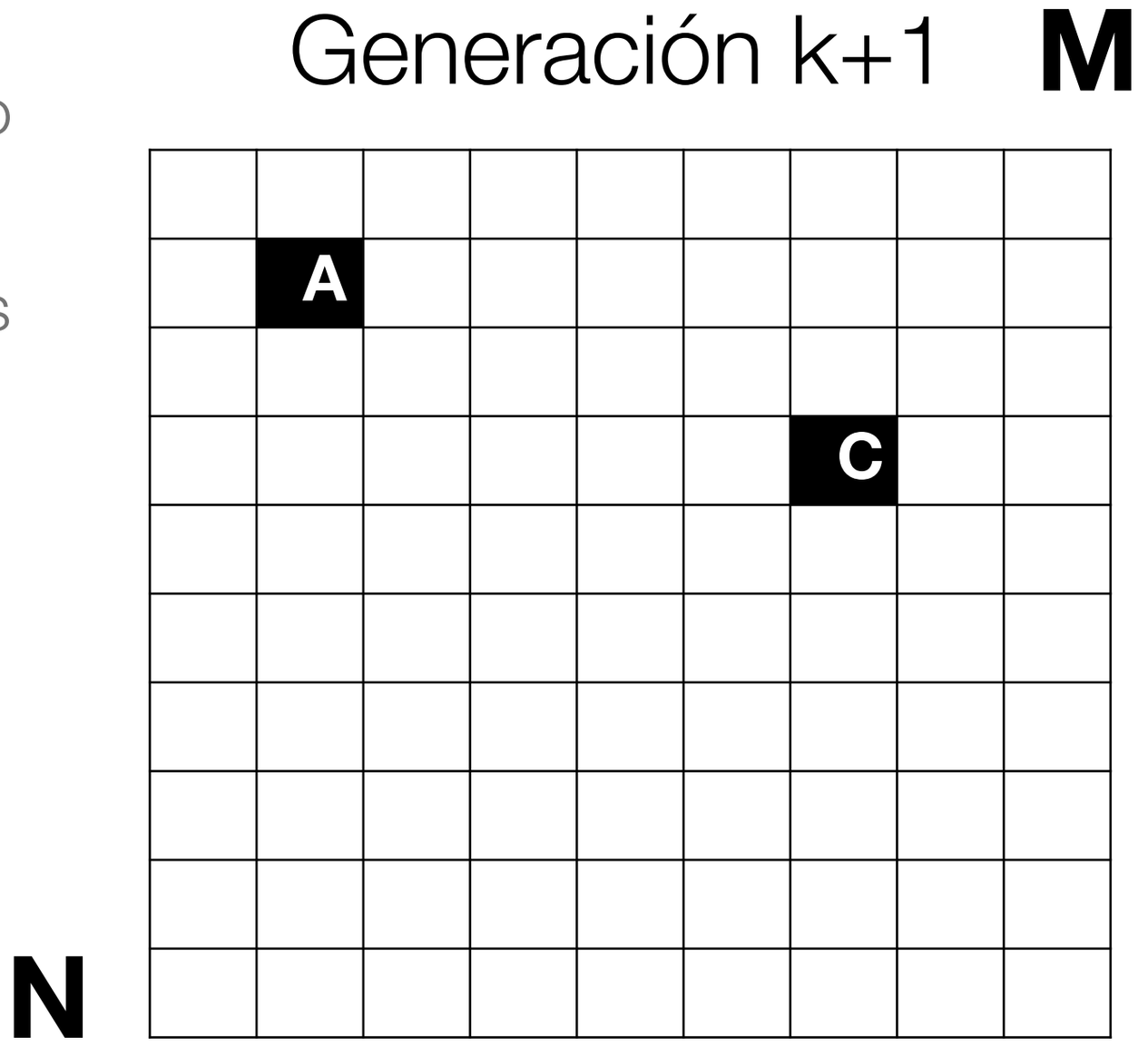
Game of life

- Ejemplo
- Una célula muerta con exactamente 3 células vecinas vivas “re-nace" (es decir, al turno siguiente estará viva). **C**
- Una célula viva con 2 o 3 células vecinas vivas sigue viva. **A**
- Una célula viva muere:
  - por “soledad” (menos de 2 vecinos vivos). **B**
  - “sobrepoblación" (más de 3 vecinos vivos). **D**

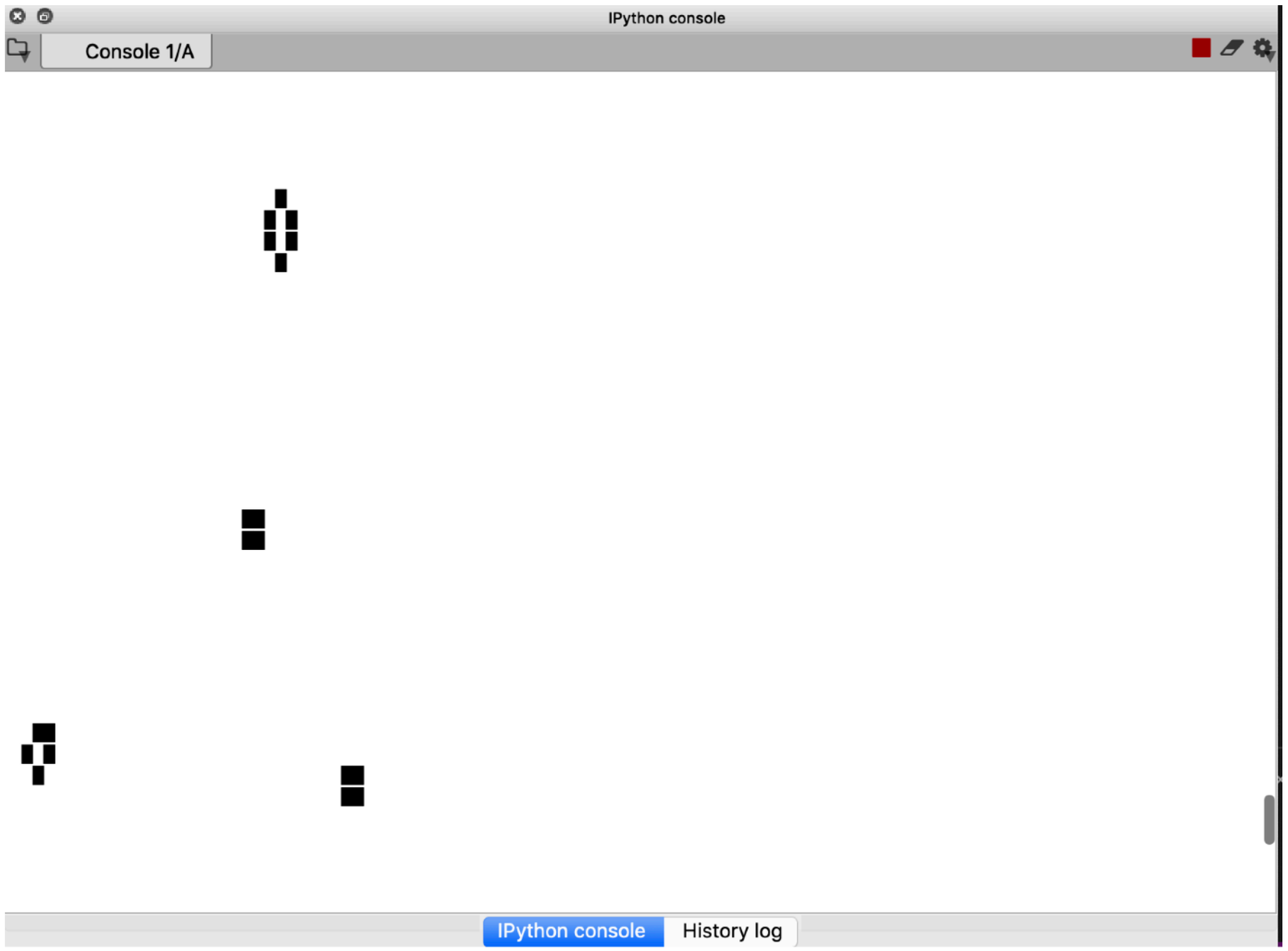


Game of life

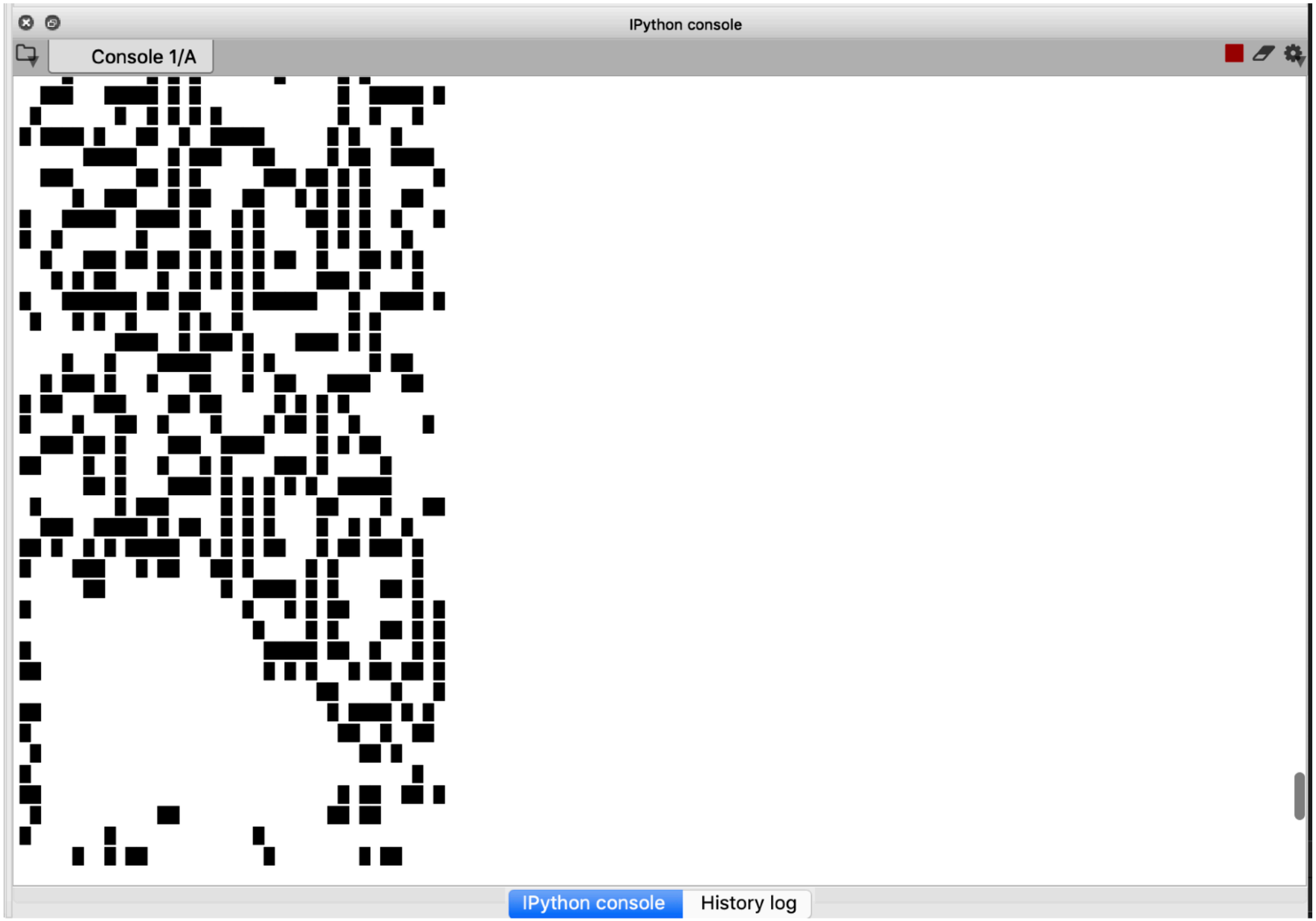
- Ejemplo
- Una célula muerta con exactamente 3 células vecinas vivas “re-nace" (es decir, al turno siguiente estará viva). **C**
- Una célula viva con 2 o 3 células vecinas vivas sigue viva. **A**
- Una célula viva muere:
  - por “soledad” (menos de 2 vecinos vivos). **B**
  - “sobrepoblación" (más de 3 vecinos vivos). **D**



How does success look like?



How does success look like?



Diseño de la solución

- Ingredientes (estructura de datos)
  - Numpy array bi-dimensional
- Procedimientos (funciones)
  - Creación del “mundo cuadricular” N x M
    - Incluye algunas células vivas
  - Visualizar mundo cuadricular
  - Actualización del mundo, i.e. aplicar reglas

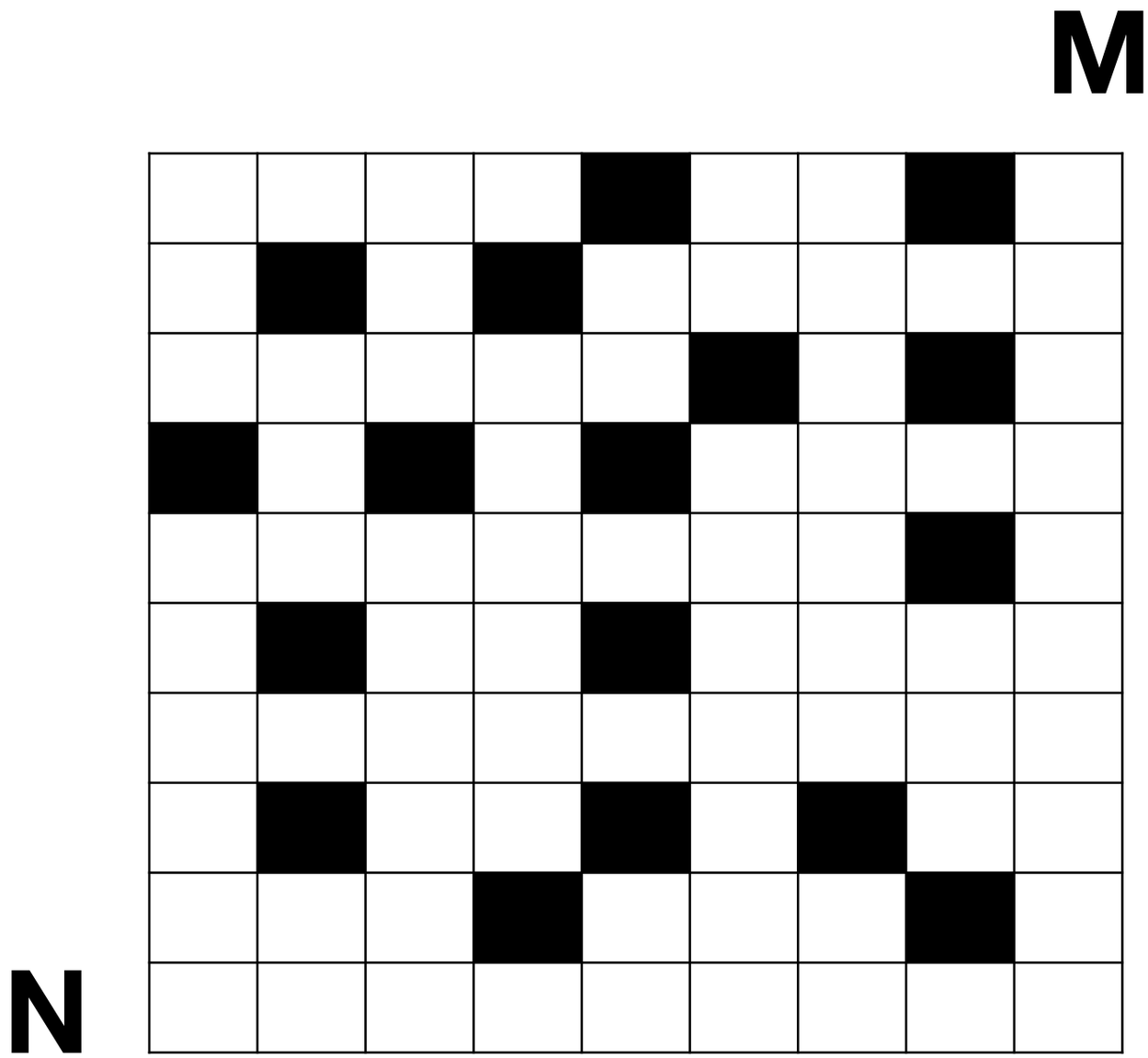
ITAM

Diseño de la solución

- Creación del “mundo cuadricular” N x M

Cero

Uno



Diseño de la solución

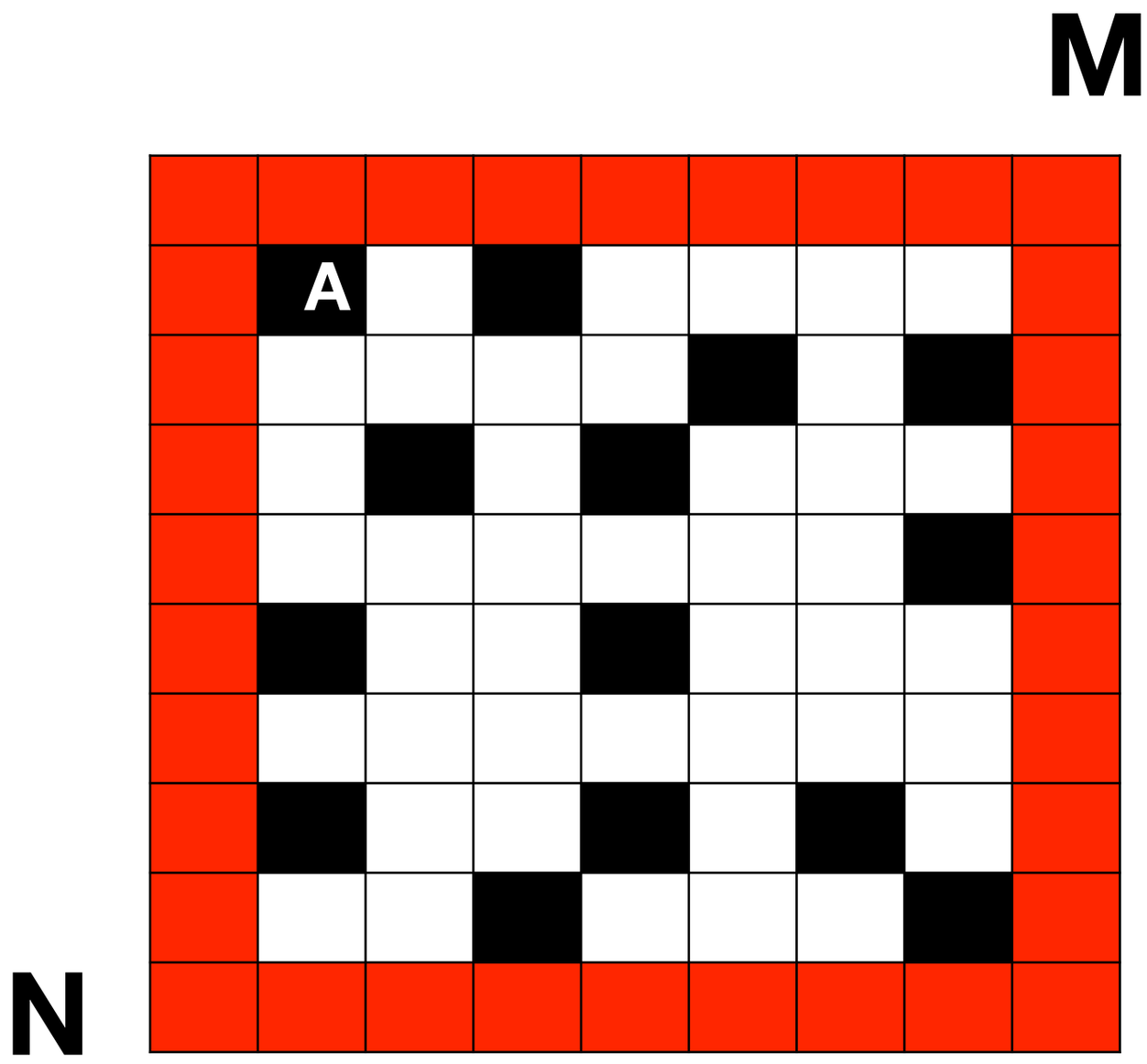
- Visualizar mundo cuadricular

if Uno print -> ■

else print -> □

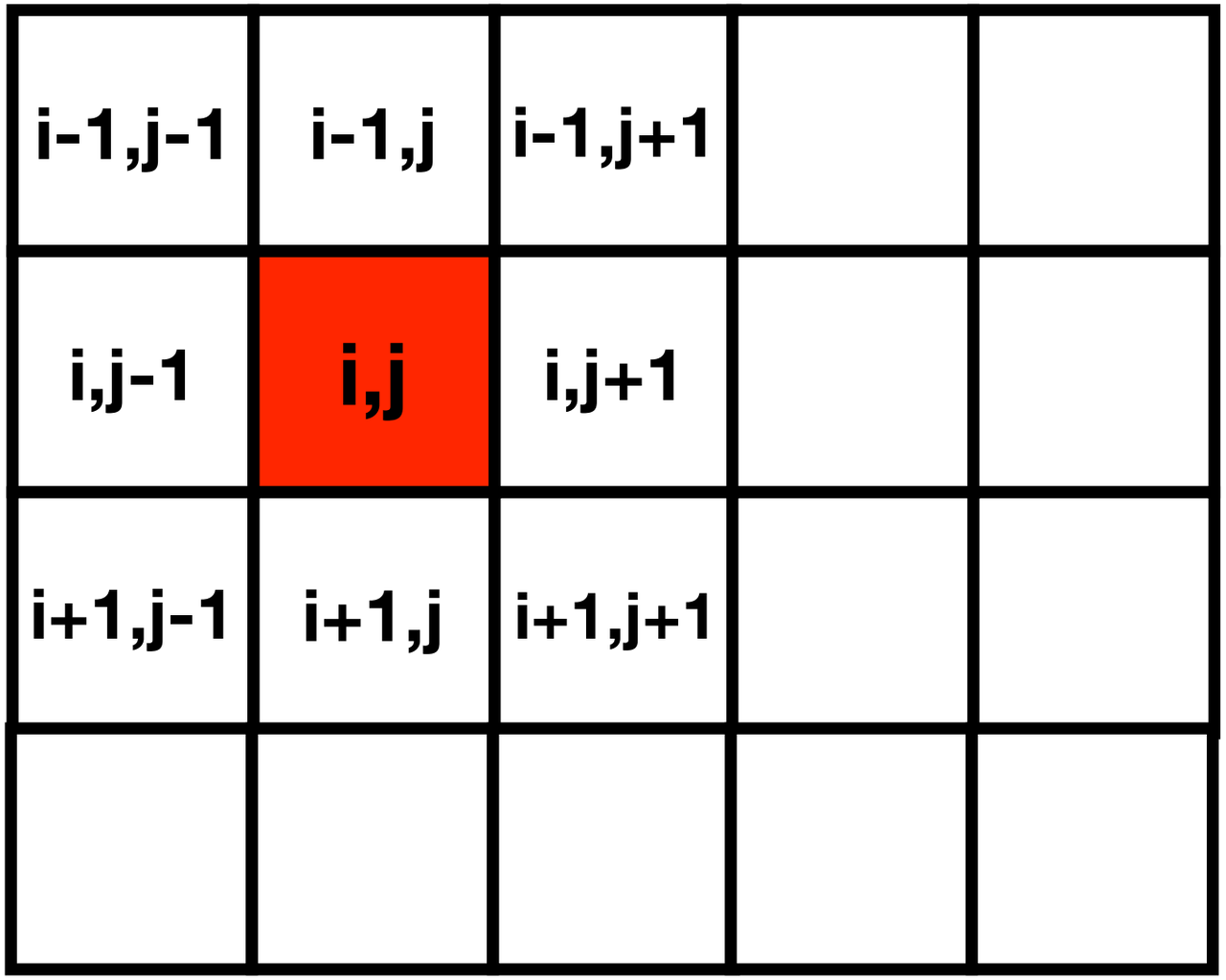
Diseño de la solución

- Actualización del mundo
  - Por simplicidad, recorrer celdas (células) dentro del recuadro rojo
- Aplicar reglas



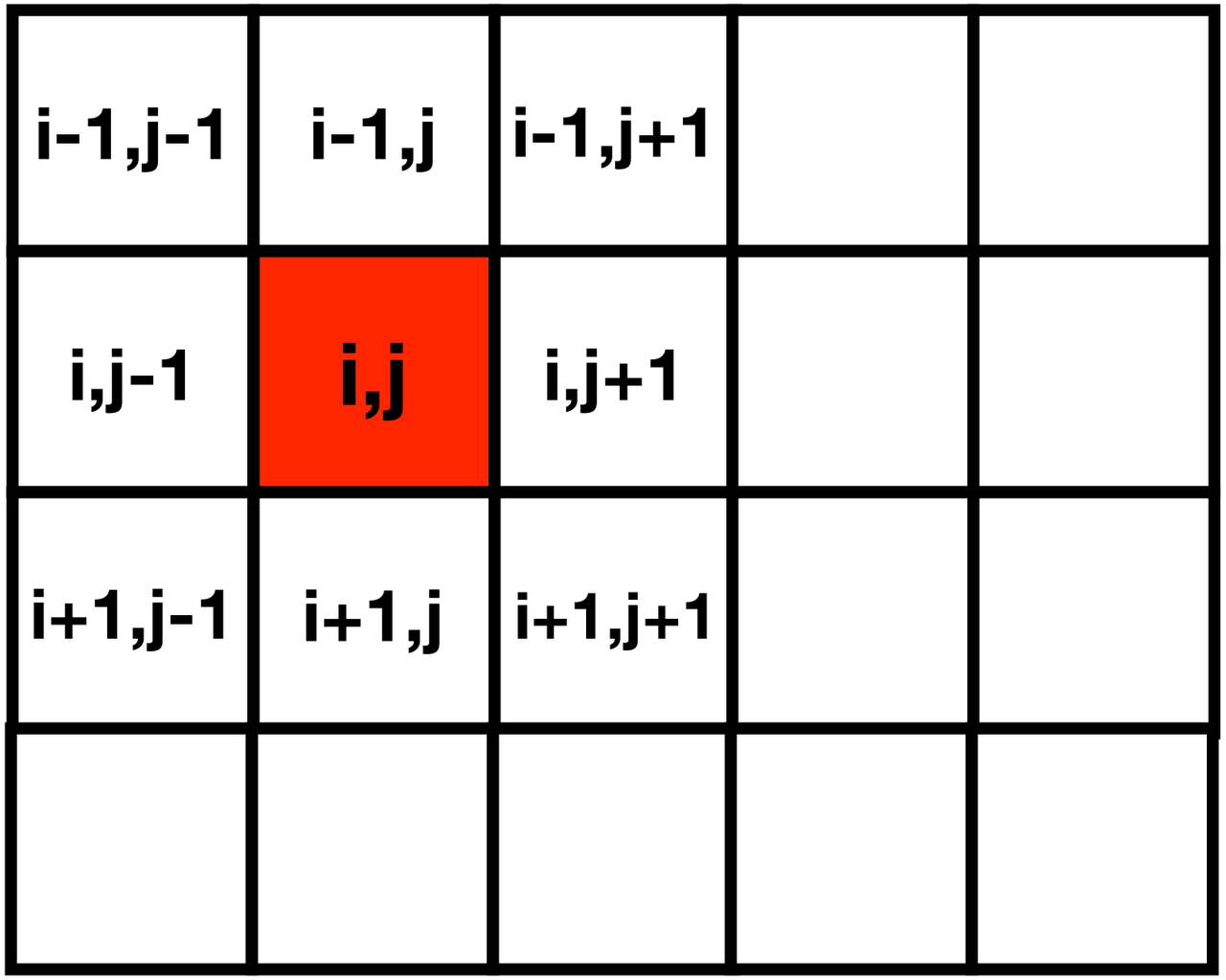
Diseño de la solución

- Aplicar reglas
  - Calcular valores en la vecindad



Qué pasa si se cambian algunas reglas?

- Una célula muerta con exactamente **3** células vecinas vivas “re-nace" (es decir, al turno siguiente estará viva).
- Una célula viva con **2 o 3** células vecinas vivas sigue viva
- Una célula viva muere:
  - por “soledad” (menos de **2** vecinos vivos)
  - “sobrepoblación" (más de **3** vecinos vivos)



Cambiar la forma de visualizar células

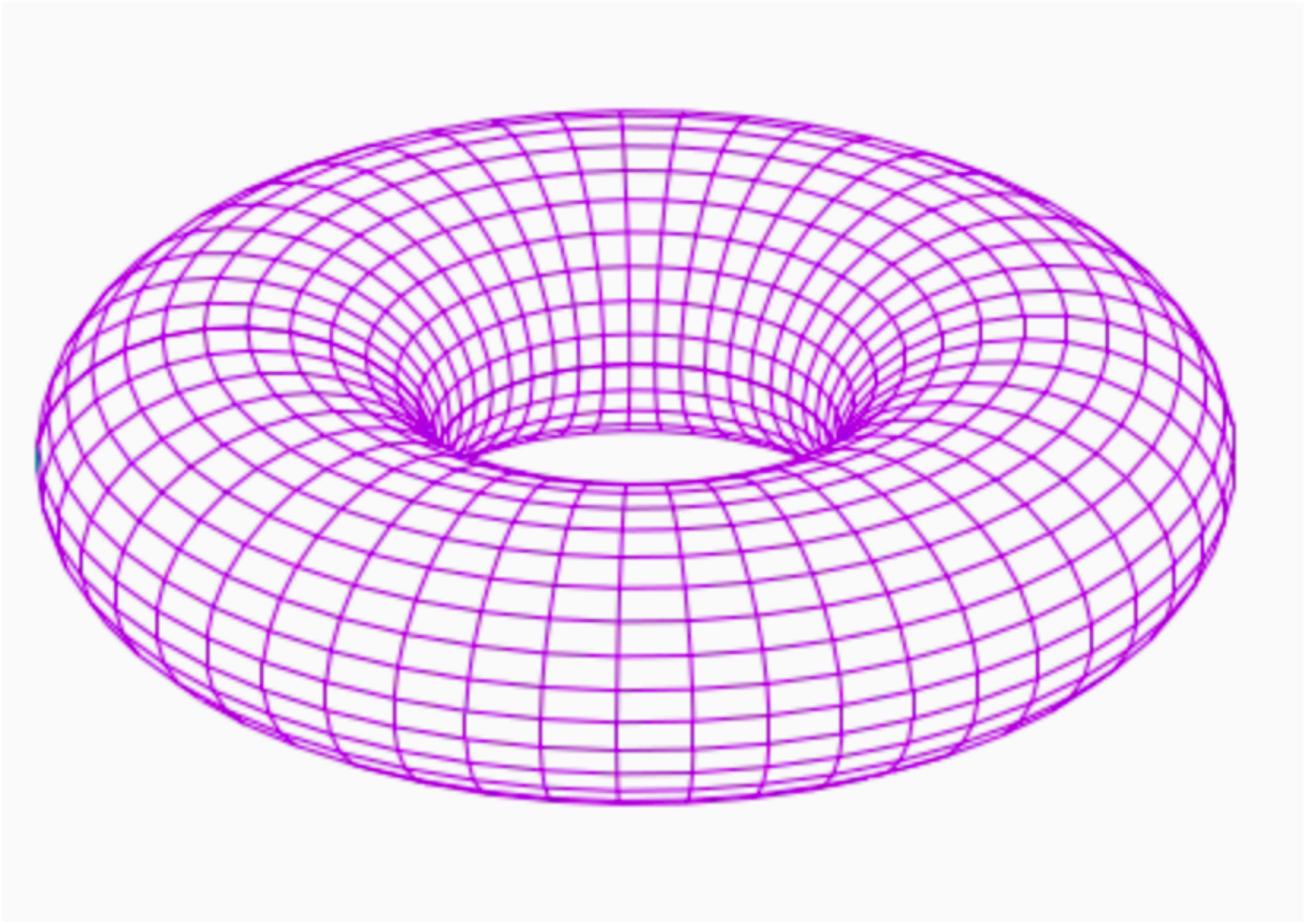
- fila.append(“o")
- fila.append("\u2665")

Punto extra

- Game of life:
  - Primeros tres simular una matriz sin bordes, toroide
  - Primeros tres en poder generar osciladores
  - Primeros tres en poder generar pistola de planeadores

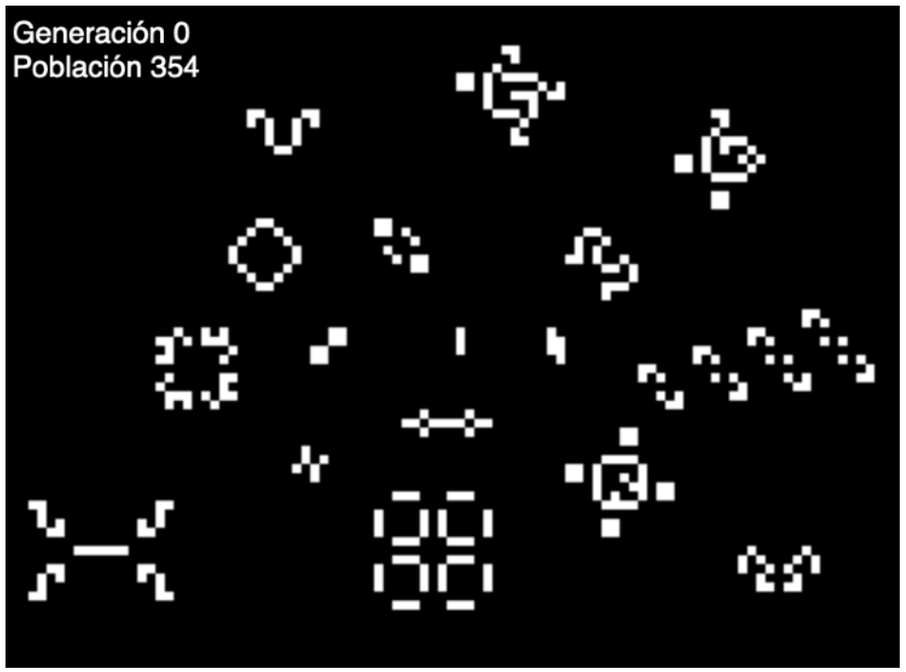
Punto extra

- Toroide



Punto extra

- Osciladores



- Planeadores

Pistola de planeadores de Gosper (*Gosper Glider Gun*)

https://es.wikipedia.org/wiki/Juego_de_la_vida