Gra w życie Conwaya

Ksawery Mielczarek

Reguły gry

- Dwuwymiarowy automat komórkowy
- Warunki początkowe determinują ewolucję

- Komórki „martwe” lub „żywe” zmieniają swój stan co krok czasowy:
  - Komórki martwe ożywają, jeśli mają **dokładnie 3 żywych sąsiadów**
  - Komórki żywe przeżywają wtw. gdy mają **2 lub 3 żywych sąsiadów** (w przeciwnym przypadku umierają z „osamotnienia” lub „przeludnienia”

Pomysł implementacji

- Plansza gry to np.array.
- Komórki **żywe ~ 1**; komórki **martwe ~ 0**
- W każdym kroku w komórce kładziemy sumę jej sąsiadów i **10-krotność** jej wartości w poprzednim kroku,
- Wartość w komórce (0-18) jednoznacznie określa jej ewolucję.

Pomysł implementacji

Żywi sąsiedzi	Suma w komórce	Czy będzie żyć?	Suma w komórce	Czy będzie żyć?
	Komórka martwa		Komórka żywa	
0	0	Nie	10	Nie
1	1	Nie	11	Nie
2	2	Nie	12	Tak
3	3	Tak	13	Tak
4	4	Nie	14	Nie
5	5	Nie	15	Nie
6	6	Nie	16	Nie
7	7	Nie	17	Nie
8	8	Nie	18	Nie

Implementacja

```python
def step(board):
    new_board = board.copy()
    new_board += 9 * board
    for i in [-1, 0, 1]:
        for j in [-1, 0, 1]:
            new_board += np.roll(board, shift = (i, j), axis = (0, 1))

    new_board[new_board == 3] = 20
    new_board[new_board == 13] = 20
    new_board[new_board == 14] = 20
    new_board[new_board < 20] = 0
    new_board[new_board == 20] = 1

    return new_board
```

Kod kroku czasowego

```python
space = np.zeros([50, 50])
```

Pusta przestrzeń

Warunki początkowe dla różnych układów

```
glider = space.copy()
glider[23:26, 25] = 1
glider[25, 26] = 1
glider[24, 27] = 1
```

```
immortal = space.copy()
immortal[22, 21] = 1
immortal[22:24, 23] = 1
immortal[24:27, 25] = 1
immortal[25:28, 27] = 1
immortal[26, 28] = 1
```

```
diehard = space.copy()
diehard[25, 21:23] = 1
diehard[24, 22] = 1
diehard[24, 26:29] = 1
diehard[26, 27] = 1
```

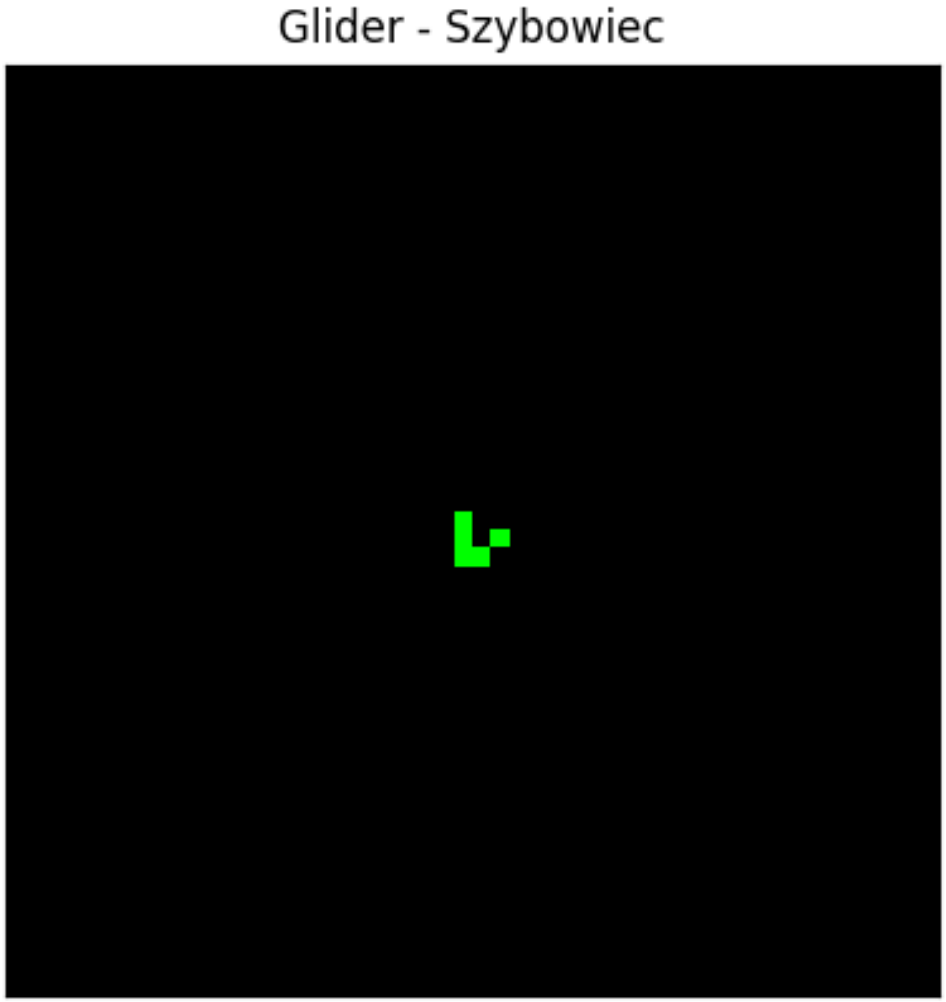
```
tumbler = space.copy()
tumbler[22:27, 24] = 1
tumbler[22:27, 26] = 1
tumbler[22:24, 23] = 1
tumbler[22:24, 27] = 1
tumbler[25:28, 28] = 1
tumbler[25:28, 22] = 1
tumbler[27, 23] = 1
tumbler[27, 27] = 1
```

```
pulsar = space.copy()
pulsar[24, 21:24] = 1
pulsar[21:24, 24] = 1
pulsar[19, 21:24] = 1
pulsar[21:24, 19] = 1

pulsar += np.roll(np.rot90(pulsar, 1), shift = 1, axis = 0)
pulsar += np.roll(np.rot90(pulsar, 2), shift = (1, 1), axis = (0, 1))
pulsar += np.roll(np.rot90(pulsar, 3), shift = 1, axis = 1)
pulsar[pulsar == 2] = 1
```

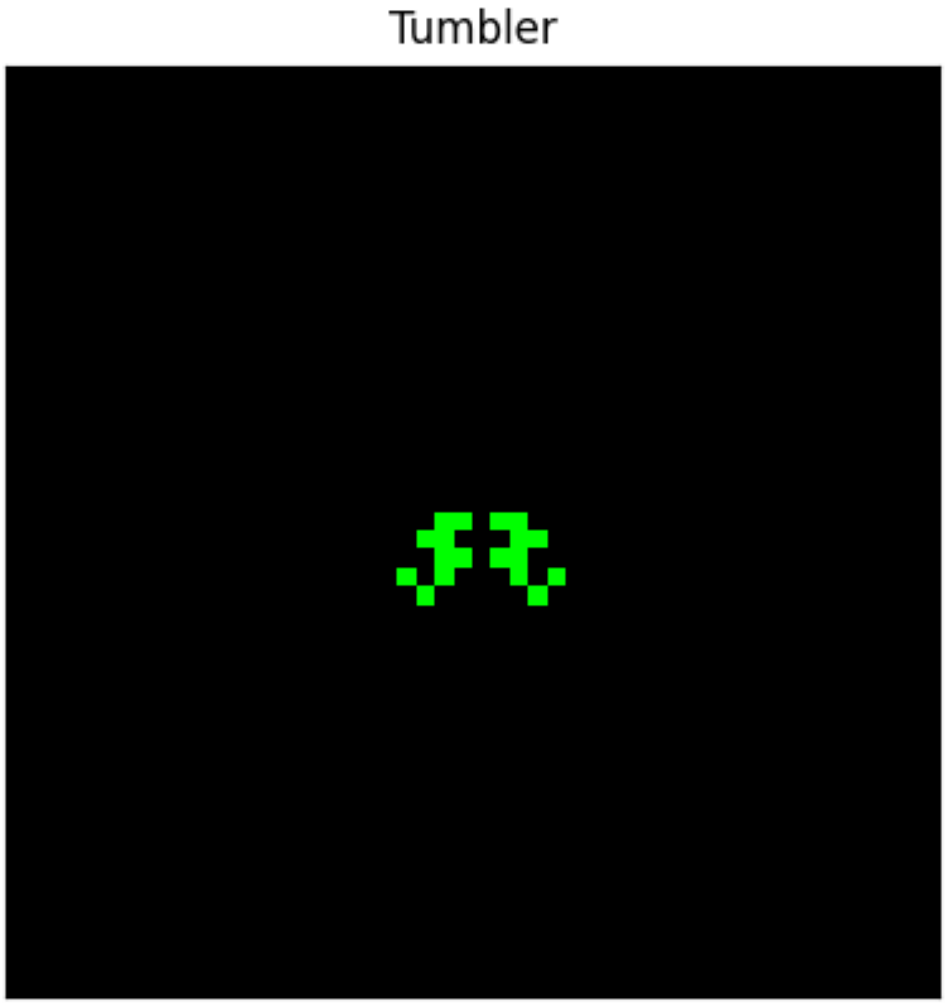
Tworzenie animacji

```
# declare animation function

cmap = colors.ListedColormap(['black', 'lime'])
bounds = [0, 1, 2]
norm = colors.BoundaryNorm(bounds, cmap.N)

fig = plt.figure()
im = plt.imshow(board, interpolation = 'none', cmap = cmap, norm = norm)
im.axes.get_xaxis().set_visible(False)
im.axes.get_yaxis().set_visible(False)
im.axes.set_title("Eskadra")

def init():
    im.set_data(board)
    return im,

def animate(i):
    board = im.get_array()
    board = step(board)
    im.set_array(board)
    return im,

# animate and save it

anim = ani.FuncAnimation(fig, animate, frames = 350, interval = 200, blit = 'True')
anim.save('Eskadra.gif', writer = 'ffmpeg', fps = 10)
```

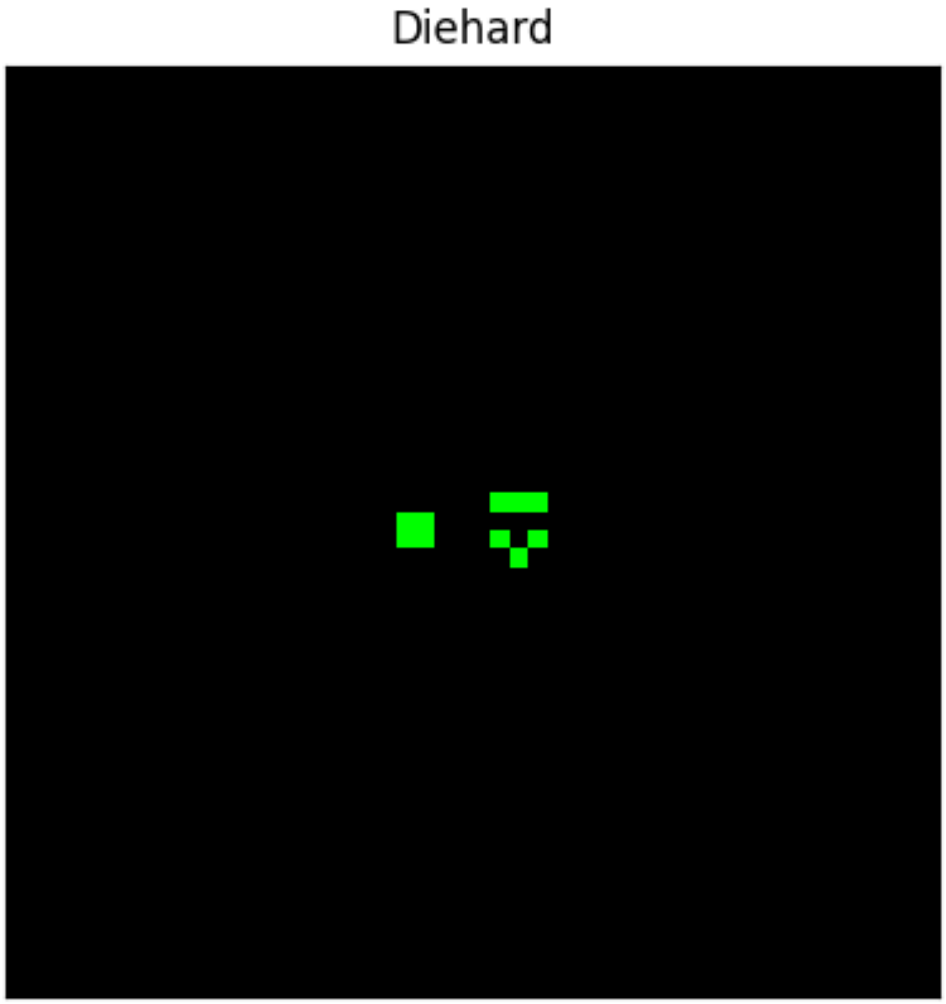
Animacje



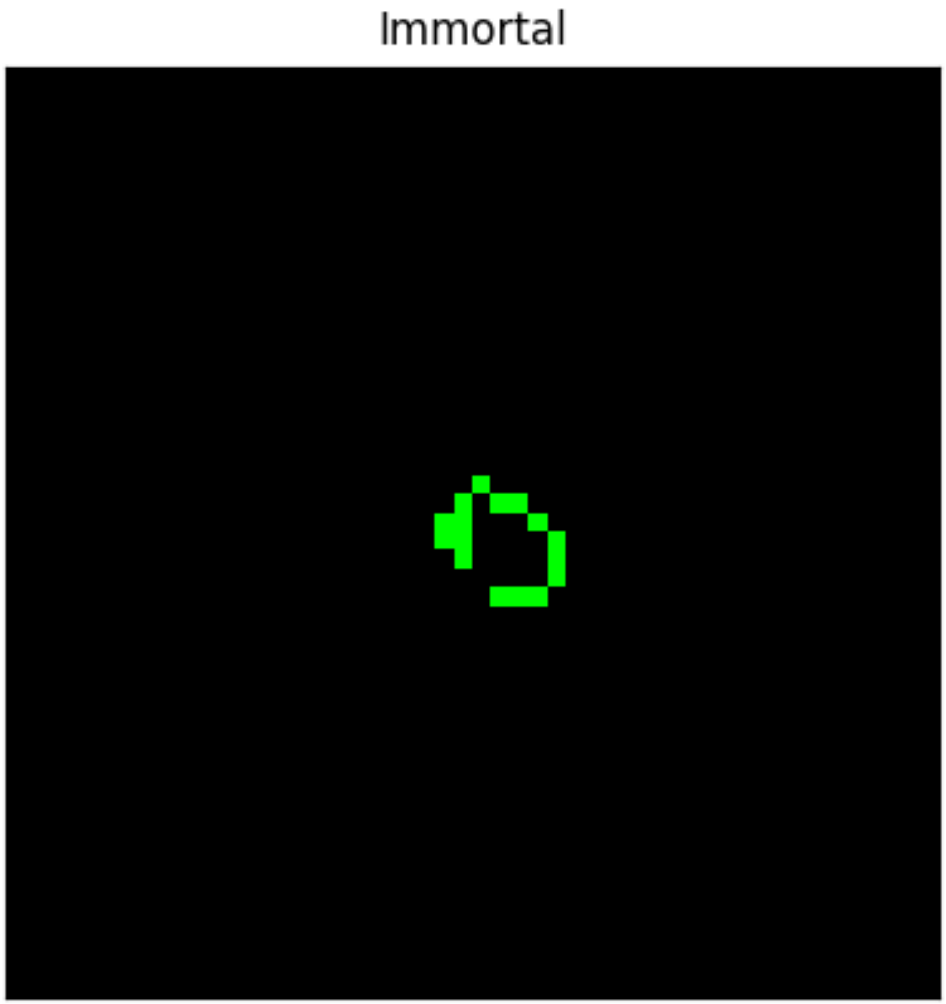
Animacje



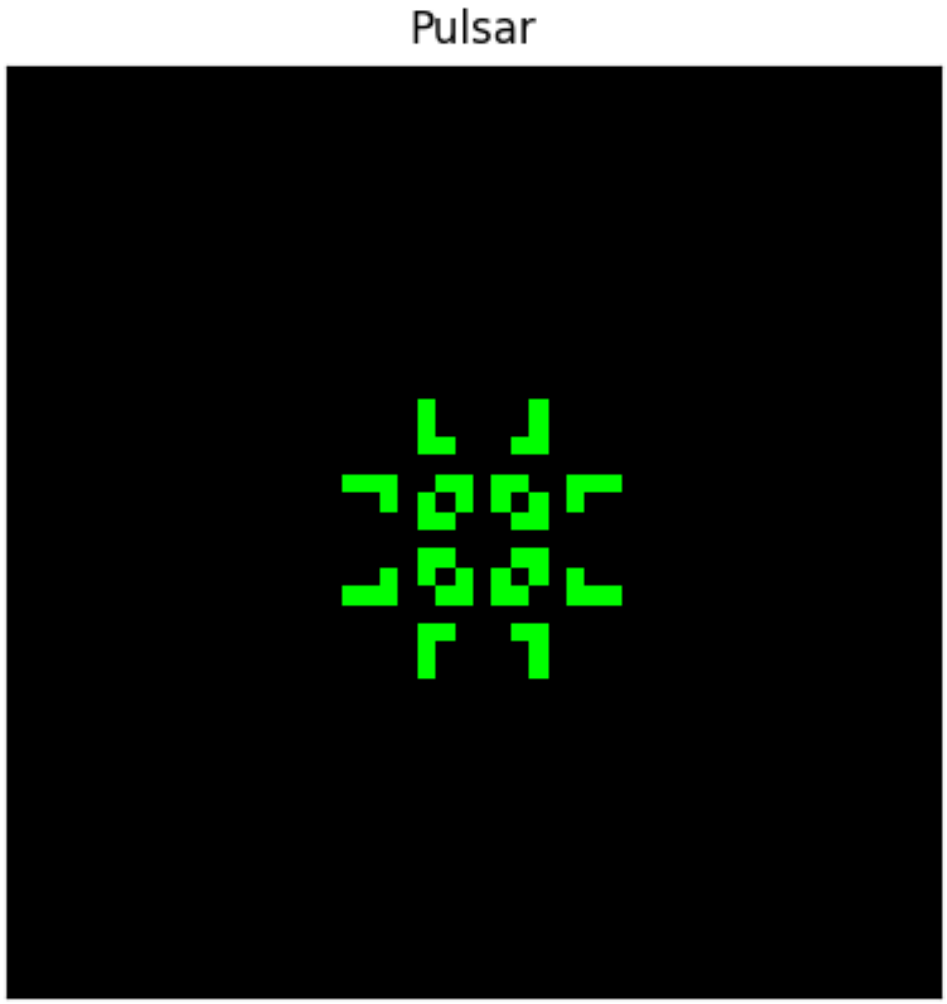
Animacje



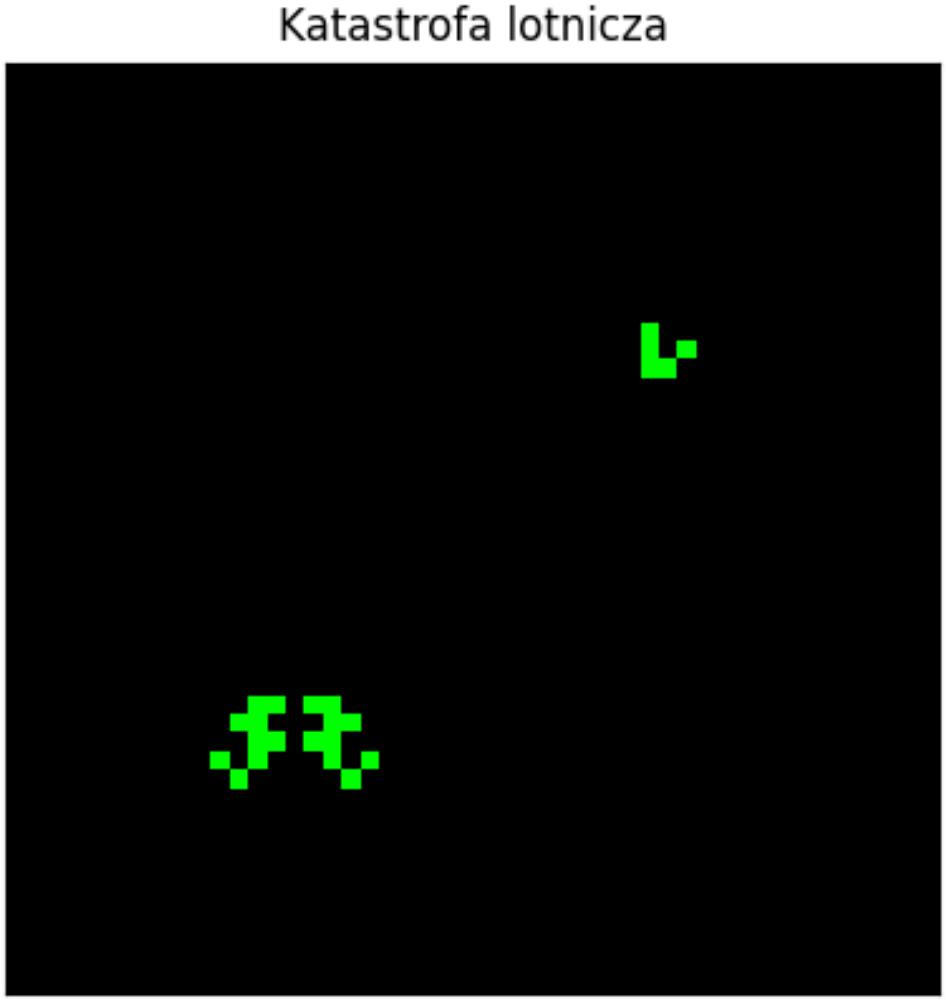
Animacje



Animacje



Animacje



Animacje