

# Opracowanie projektu

## Gra w życie wg. Conway'a

Autor: Bartosz Beksa

Przedmiot: Programowanie w języku Ruby

Prowadzący: Piotr Arłukowicz

Data: 2022.06.11

## Główne założenia programu

1. Pola są żywe kiedy mają dwóch lub trzech żywych sąsiadów,
2. Pola umierają gdy mają więcej niż trzech lub mniej niż dwóch żywych sąsiadów,
3. Nowe żywe pola pojawiają się gdy pole jest martwe i ma trzech żywych sąsiadów,
4. Plansza jest hash'em zawierającym, każde pole żywe z koordynatami [X, Y]

## Opis podstawowy

### Opis reguł gry:

Gra toczy się na planszy podzielonej na kwadratowe komórki. Każda komórka ma ośmiu sąsiadów, czyli komórki przylegające do niej bokami i rogami. Każda komórka może znajdować się w jednym z dwóch stanów: może być albo żywa (włączona), albo martwa (wyłączona). Stany komórek zmieniają się w jednostkach czasu. Stan wszystkich komórek w jednostce czasu jest używany do obliczenia stanu wszystkich komórek w następnej jednostce. Po obliczeniu wszystkie komórki zmieniają swój stan dokładnie w tym samym momencie. Stan komórki zależy tylko od liczby jej żywych sąsiadów. W grze w życie nie ma graczy w dosłownym tego słowa znaczeniu udział człowieka sprowadza się jedynie do ustalenia stanu początkowego komórek.

# Najważniejsze miejsca w kodzie

## Rysowanie planszy:

Do stworzenia okna z planszą gry użyłem biblioteki Ruby 2D służącej łatwego do tworzenia aplikacji, gier i wizualizacji 2D.

Tło okna jest ustalone na kolor zapisany w formacie RGB #570861 .


Funkcja dzieląca planszę na kwadratowe pola o wyznaczonej wielkości (SQUARE\_SIZE).

```
def draw_lines
  (Window.width / SQUARE_SIZE).times do |x|
    Line.new(
      width: 1,
      color: Color.new('black'),
      y1: 0,
      y2: Window.height,
      x1: x * SQUARE_SIZE,
      x2: x * SQUARE_SIZE,
    )

    (Window.height / SQUARE_SIZE).times do |y|
      Line.new(
        width: 1,
        color: Color.new('black'),
        x1: 0,
        x2: Window.width,
        y1: y * SQUARE_SIZE,
        y2: y * SQUARE_SIZE,
      )
    end
  end
end
```

*Rysunek 1: funkcja rysująca linie na planszy [źródło: opracowanie własne]*

Linie rysowane są w pętłach najpierw poziomo później pionowo w odpowiedniej od siebie odległości ustalonej w wcześniej w kodzie na 32 jednostki, ich kolor to czarny, a grubość to jedna jednostka.

Funkcja zmieniająca kolor żywych czyta po kolei dane z hash'a do którego zapisywane są koordynaty i w ich miejscu rysuje kwadrat w innym kolorze od tła w tym przypadku jest to fuksja .

```
def draw_alive_squares
  @grid.keys.each do |x, y|
    Square.new(
      color: Color.new('fuchsia'),
      x: x * SQUARE_SIZE,
      y: y * SQUARE_SIZE,
      size: SQUARE_SIZE
    )
  end
end
```

Rysunek 2: funkcja zmieniająca kolor żywych pól [źródło: opracowanie własne]

## Obliczanie i tworzenie nowej klatki:

Plansza podstawowo nie zawiera żadnych żywych pól do włączania lub wyłączania pól zastosowana jest metoda `switch`.

```
def switch(x, y)
  if @grid.has_key?([x, y])
    @grid.delete([x, y])
  else
    @grid[[x, y]] = true
  end
end

on :mouse_down do |event|
  grid.switch(event.x / SQUARE_SIZE, event.y / SQUARE_SIZE)
end
```

Rysunek 3: funkcja do włączania i wyłączania pól oraz jej wykorzystanie [źródło: opracowanie własne]

Przy kliknięciu na dane pole dla podanych współrzędnych, czyli w miejscu gdzie użytkownik kliknął sprawdzane jest czy pole było zapisane w hash'u i jeżeli jest to zostaje usunięte, a jeżeli nie ma go to zostaje wpisane.

Do wyliczania nowej klatki tworzona jest nowa plansza do której wpisywane zostają pola włączone wyliczone na podstawie aktualnej planszy.

Dla każdego pola sprawdzane jest czy jest aktualnie żywe i ile ma żywych sąsiadów.

Jeżeli pole było włączone i ma 2 lub 3 żywych sąsiadów to pozostaje przy życiu lub jeżeli pole było wyłączone i ma dokładnie 3 żywych sąsiadów to staje się żywe, w każdym innym przypadku pole w nowej klatce będzie martwe.

```

def calculate_frame
  if @playing
    new_grid = {}

    (Window.width / SQUARE_SIZE).times do |x|
      (Window.height / SQUARE_SIZE).times do |y|
        alive = @grid.has_key?([x, y])

        alive_neighbours = [
          @grid.has_key?([x - 1, y - 1]),
          @grid.has_key?([x, y - 1]),
          @grid.has_key?([x + 1, y - 1]),
          @grid.has_key?([x + 1, y]),
          @grid.has_key?([x + 1, y + 1]),
          @grid.has_key?([x, y + 1]),
          @grid.has_key?([x - 1, y + 1]),
          @grid.has_key?([x - 1, y]),
        ].count(true)

        if (alive && alive_neighbours.between?(2, 3)) || (!alive &&
alive_neighbours == 3)
          new_grid[[x, y]] = true
        end
      end
    end

    @grid = new_grid
  end
end

```

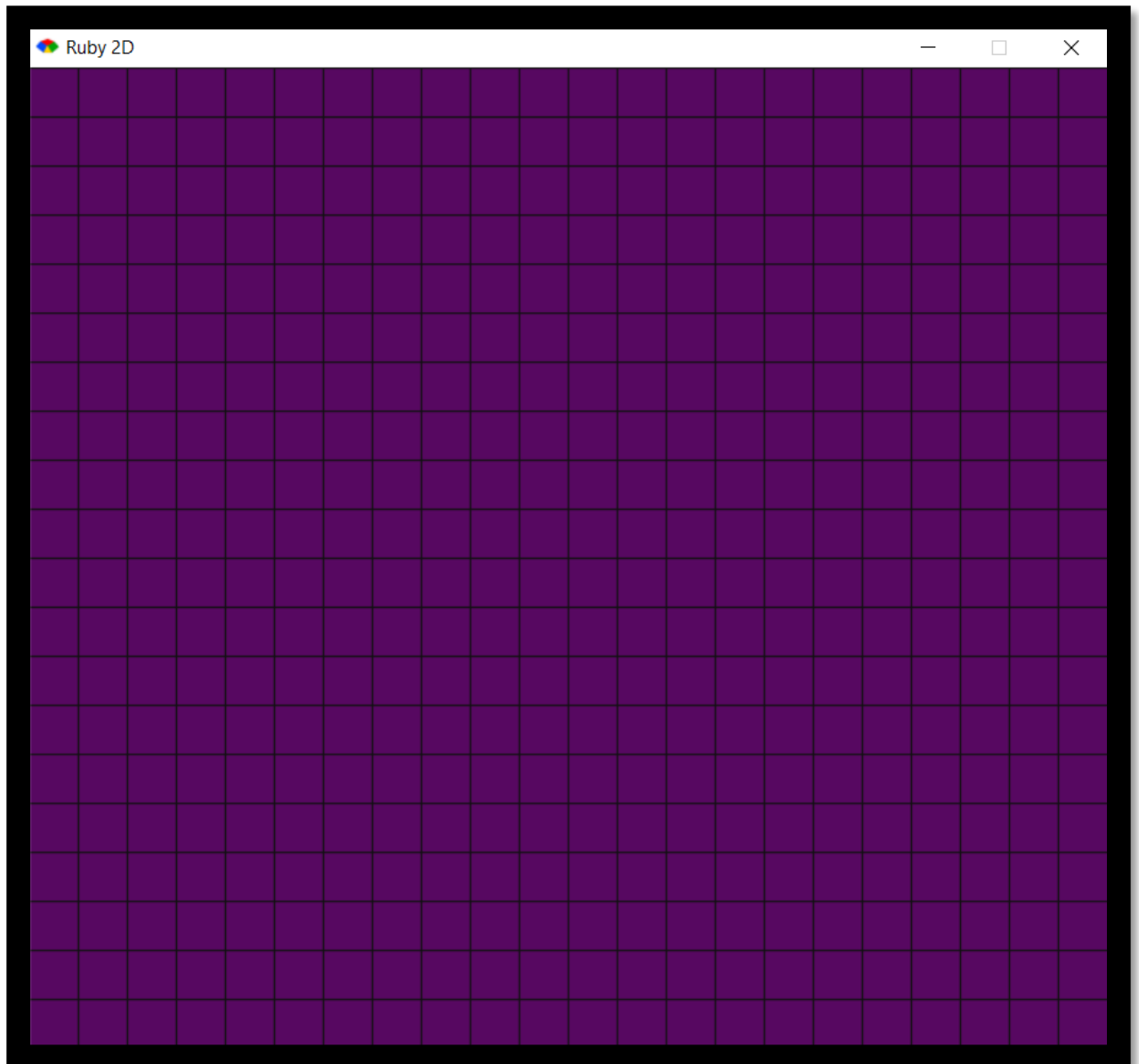
Rysunek 4: funkcja obliczająca nową klatkę [źródło: opracowanie własne]

Poprzednia plansza jest nadpisywana przez new\_grid i zostaje wyświetlony nowy układ żywych pól.

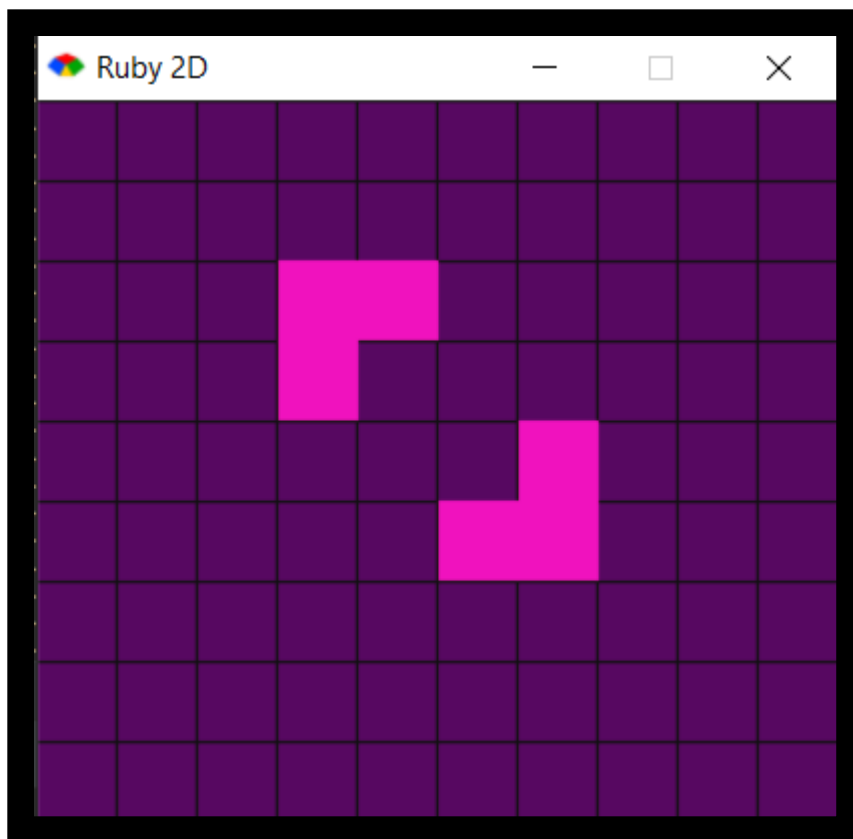
# Uruchamianie i opis sterowania

## Uruchamianie:

Program może zostać uruchomiony z kilkoma opcjami (sl1, sl2, sl3, sl4, osc1, osc2, osc3, osc4, osc5, gl) kilka pól będzie wtedy żywych od początku tworząc jeden z przygotowanych wzorców oraz plansza będzie miała zmienioną wielkość, program włączony bez żadnej opcji wyświetli pustą planszę bez żadnego żywego pola.



Rysunek 5: program uruchomiony bez żadnych opcji [źródło: opracowanie własne]



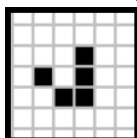
Rysunek 6: program uruchomiony z opcją 'osc3' [źródło: opracowanie własne]

## Sterowanie:

Aby włączyć lub wyłączyć jedno z pól na planszy wystarczy w nie kliknąć.

Kilka klawiszy ma też swoją funkcję:

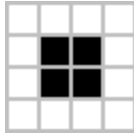
- c – wyczyszczenie planszy (wyłączenie wszystkich pól)
- q – wyjście z gry
- spacja – uruchomienie gry
- Klawisze numeryczne (wszystkie od 0 do 9) tworzą na planszy jeden z przygotowanych wzorców
  - 0 - Glider (szybowiec)



Rysunek 7: szybowiec [źródło: wikipedia.org]

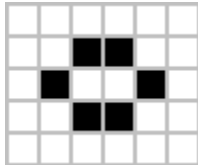
## Statyczne formy życia:

- 1 - Klocek (blok)



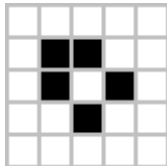
Rysunek 8: klocek [źródło: wikipedia.org]

- 2 – ul (Beehive)



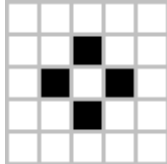
Rysunek 9: ul [źródło: wikipedia.org]

- 3 – łódź (Boat)



Rysunek 10: łódź [źródło: wikipedia.org]

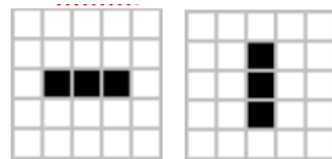
- 4 – Koniczynka (Tub)



Rysunek 11: Koniczynka [źródło: wikipedia.org]

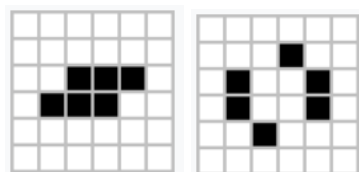
## Oscylatory:

- 5 – Blinker



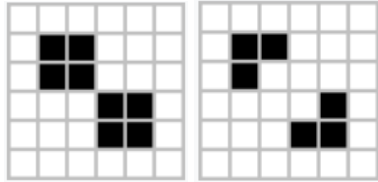
Rysunek 12: Blinker [źródło: wikipedia.org]

- 6 – Toad



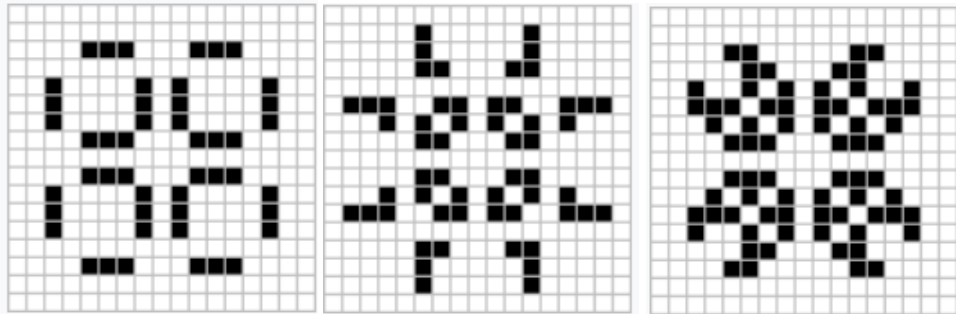
Rysunek 13: Toad [źródło: wikipedia.org]

- 7 – Beacon



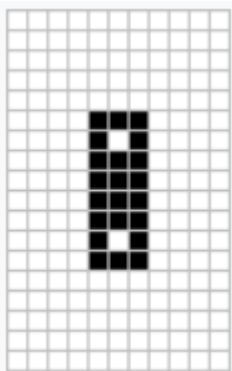
Rysunek 14: Beacon [źródło: wikipedia.org]

- 8 – Pulsar



Rysunek 15: Pulsar [źródło: wikipedia.org]

- 9 – Pentadecathlon (15 różnych stanów)



Rysunek 16: Pentadecathlon [źródło: wikipedia.org]