

# Specyfikacja funkcjonalna – Wireworld

Krzysztof Dąbrowski i Jakub Bogusz

26 kwietnia 2019

# Spis treści

<b>1</b>	<b>Cel projektu</b>	<b>2</b>
<b>2</b>	<b>Opis ogólny problemu</b>	<b>3</b>
2.1	Wstęp . . . . .	3
2.2	Wire World . . . . .	3
2.2.1	Symulacja . . . . .	3
2.2.2	Struktury . . . . .	4
2.3	Game of life . . . . .	4
2.3.1	Symulacja . . . . .	4
2.3.2	Struktury . . . . .	5
2.4	Interfejs użytkownika . . . . .	5
<b>3</b>	<b>Działanie programu</b>	<b>6</b>
3.1	Opis komunikacja z użytkownikiem . . . . .	6
3.2	Wygląd interfejsu użytkownika . . . . .	6
3.2.1	Po włączeniu programu . . . . .	6
3.2.2	W trakcie symulacji . . . . .	6
3.2.3	Symulacja wstrzymana . . . . .	6
3.2.4	Tryb automatu Game of Live . . . . .	7
3.3	Plik wejściowy . . . . .	7
3.3.1	Przykład . . . . .	7
3.3.2	Format pliku . . . . .	7
<b>4</b>	<b>Wyniki działania programu</b>	<b>8</b>
<b>5</b>	<b>Sytuacje wyjątkowe</b>	<b>9</b>
5.1	Zmiana domyślnego zachowania . . . . .	9
5.2	Błędy . . . . .	9
5.2.1	Błędy pliku wejściowego . . . . .	9
5.2.2	Błędy losowe . . . . .	10

# Rozdział 1

## Cel projektu

Celem projektu jest implementacja automatu komórkowego Wire World w języku Java z interfejsem graficznym zaimplementowanym przy pomocy biblioteki JavaFX. Gotowy program ma pozwalać użytkownikowi przeprowadzać symulacje zgodne z określonymi zasadami. Parametrami generacji użytkownik będzie mógł sterować ręcznie z wbudowanego menu opisanego niżej [TU DAĆ LINK DO SEKCJI Z GRAFIKĄ I OPISEM CZĘŚCI]. Interfejs będzie wyświetlał na bieżąco podgląd kolejnych generacji. Użytkownik będzie miał możliwość wstrzymania symulacji oraz zmianę stanu planszy przy pomocy narzędzi edycji.

Ponadto będzie istnieć również możliwość przełączenia trybu symulacji z Wire World na automat komórkowy Game of life.

## Rozdział 2

# Opis ogólny problemu

### 2.1 Wstęp

Projekt skupia się na realizacji 3 głównych aspektów problemu. Są to odpowiednio automaty komórkowe „Game of Life” i „Wireworld” oraz wizualna prezentacja działania tych automatów.

### 2.2 Wire World

Wire World jest automatem komórkowym wymyślonym przez Briana Silvermana w roku 1987. Jest często używany do symulacji elementów elektronicznych operujących na wartościach bitowych. Pomimo prostoty reguł, jakie nim rządzą, za pomocą Wireworld można nawet stworzyć działający komputer.

#### 2.2.1 Symulacja

**Stany** Komórka może znajdować się w jednym z czterech stanów:

- pusta,
- głowa elektronu,
- ogon elektronu,
- przewodnik.

**Pokolenie** to zbiór stanów wszystkich komórek w danej chwili. Gdy stan pokolenia jest ustalony, możliwe jest utworzenie nowego (potomnego) pokolenia komórek, powstających według poniższych zasad.

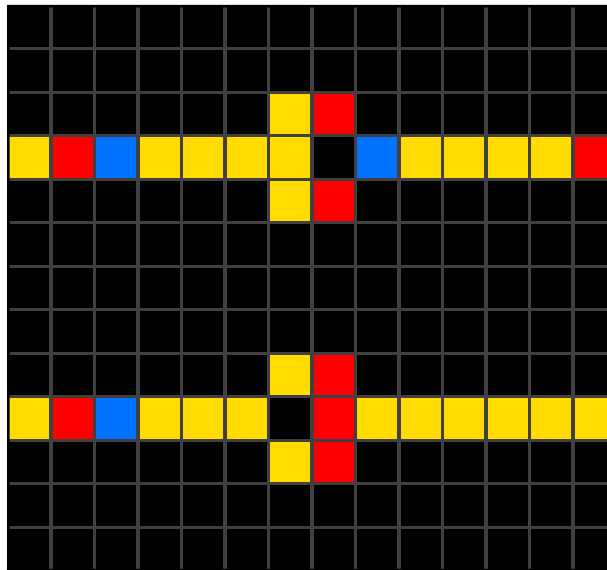
**Reguły** Następne pokolenie generowane jest zgodnie z regułami:

- Jeżeli komórka jest pusta, to pozostaje pusta niezależnie od jej otoczenia,

- Jeżeli komórka jest głową elektronu, to zmienia się w ogon elektronu,
- Jeżeli komórka jest ogonem elektronu, to zmienia się w przewodnik,
- Jeżeli komórka jest przewodnikiem i sąsiaduje z jedną lub dwoma komórkami będącymi głowami elektronu, to zmienia się w przewodnik.

### 2.2.2 Struktury

Symulacja przeprowadzona zgodnie z powyższymi regułami może prowadzić do powstania ciekawych obiektów zwanych strukturami.



Rysunek 2.1: Przykłady struktur - dioda przewodząca i nieprzewodząca

## 2.3 Game of life

Game of life jest automatem komórkowym wymyślonym przez brytyjskiego matematyka Johna Horton Conway w 1970 roku. Polega na symulacji kolejnych pokoleń życia komórek według następujących zasad.

### 2.3.1 Symulacja

**Stany** Komórka może znajdować się w jednym z dwóch stanów:

- żywa,
- martwa.

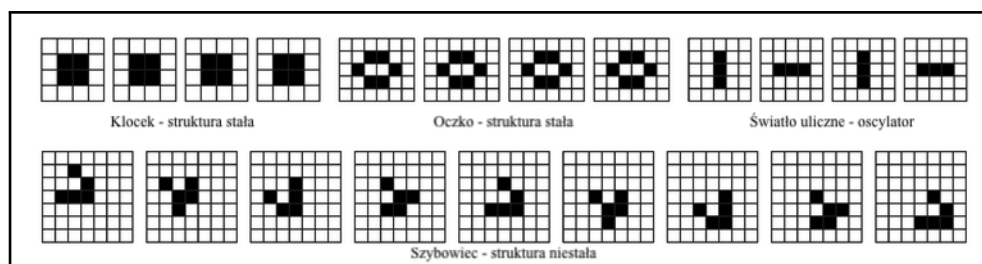
**Pokolenie** to stan wszystkich komórek w danej chwili. Gdy stan pokolenia jest ustalony, możliwe jest utworzenie nowego (potomnego) pokolenia komórek, powstających według poniższych zasad.

**Reguły** Następne pokolenie generowane jest zgodnie z regułami:

- Jeżeli komórka była martwa i miała dokładnie 3 żywych sąsiadów, w następnym pokoleniu staje się żywa,
- Jeżeli komórka była żywa to pozostaje żywa jeśli miała dwóch lub trzech żywych sąsiadów. W przeciwnym razie staje się martwa.

### 2.3.2 Struktury

Symulacja przeprowadzona zgodnie z powyższymi regułami może prowadzić do powstania ciekawych obiektów zwanych strukturami.



Rysunek 2.2: Przykłady struktur

Reguły symulacji umożliwiają również tworzenie dużo bardziej skomplikowanych struktur (jak na przykład maszyna Turinga – <https://youtu.be/My8AsV7bA94>).

## 2.4 Interfejs użytkownika

Ważnym aspektem programu jest sposób interakcji z użytkownikiem. Realizowana aplikacja będzie obsługiwana poprzez graficzny interfejs użytkownika.

Interfejs pozwoli na poniższe funkcjonalności:

- Bieżący podgląd stanu symulacji
- Możliwość dostosowania prędkości symulacji
- Zapis aktualnego pokolenia do pliku
- Wczytanie pokolenia z pliku
- Edycje poszczególnych komórek w trakcie symulacji
- Wstawianie figur wybranych z przybornika

## Rozdział 3

# Działanie programu

### 3.1 Opis komunikacja z użytkownikiem

Po programie zostanie wyświetlony graficzny interfejs, który pozwoli użytkownikowi sterować parametrami w dowolny sposób, zgodny z regułami symulacji. W razie wystąpienia błędów program będzie informować użytkownika wyświetlając nowe okno zawierające opis błędu.

W ramach interfejsu użytkownik będzie mógł wykonywać czynności opisane w punkcie 2.4 oraz przełączać między automatami Game of Life i Wireworld.

### 3.2 Wygląd interfejsu użytkownika

Poniżej przedstawiony jest szkic interfejsu użytkownika w różnych stanach.

#### 3.2.1 Po włączeniu programu

Większość kontrolek jest **wyłączonych** do momentu wczytania planszy z pliku lub wygenerowania losowej. Program domyślnie wyświetla okno symulacji automatu Wireworld.

#### 3.2.2 W trakcie symulacji

Można zmienić tempo symulacji oraz wstrzymać generację następnych pokoleń, co umożliwi edycję aktualnego pokolenia.

#### 3.2.3 Symulacja wstrzymana

Możliwe jest edytowanie komórek przy pomocy narzędzie i wstawianie figur.

### 3.2.4 Tryb automatu Game of Live

## 3.3 Plik wejściowy

Plik wejściowy pozwala na wczytanie stanu planszy. Dzięki temu użytkownik ma kontrolę nad początkiem symulacji, oraz może kontynuować symulację z zapisanego wcześniej etapu.

### 3.3.1 Przykład

5 3	– rozmiar (x y)
1 0 0 1 1	– Wartości poszczególnych komórek
0 1 1 0 1	– 1 - żywa
0 0 0 1 1	– 0 - martwa

### 3.3.2 Format pliku

#### Kodowanie

Ponieważ plik powinien zawierać tylko liczby arabskie i odstępy możliwe jest dowolne kodowanie kompatybilne z ASCII.

**Sugerowane kodowania to:** ASCII, UTF-8, ISO 8859, Windows-1250

#### Opis formatu

Plik w pierwszej linii powinien zawierać 2 liczby. Pierwsza z nich oznacza rozmiar planszy w poziomie, druga w pionie.

Następnie plik powinien zawierać tyle linii jaki został podany rozmiar w pionie. W każdej z tych linii powinno być tyle 0 lub 1 ile wynosi rozmiar w poziomie. Zero oznacz komórkę martwą, a jeden komórkę żywą.



## Rozdział 4

# Wyniki działania programu

Wyniki działania programu będą zależeć od preferencji użytkownika - od podanych [argumentów](#).

- wyświetlić wybraną ilość pokoleń w konsoli,
- wygenerować plik lub pliki .png z reprezentacjami graficznymi kolejnych pokoleń,
- wygenerować plik .gif przedstawiający życie cywilizacji,
- wygenerować plik lub pliki .txt reprezentujący konkretny stan cywilizacji, mogący służyć za plik wejściowy.

## Rozdział 5

# Sytuacje wyjątkowe

Czasem działanie programu może ulec zmianie na skutek nieprawidłowych danych wejściowych, niestandardowych ustawień wprowadzonych przez użytkownika lub z przyczyn losowych. Ten rozdział opisuje jak program zachowa się w takiej sytuacji, oraz co może ją wywołać.

### 5.1 Zmiana domyślnego zachowania

**Zbyt szeroka plansza** W przypadku gdy użytkownik włączy wyświetlanie kolejnych stanów w konsoli ale rozmiar planszy będzie zbyt szeroki by możliwe było jej wyświetlenie bez zawijania wierszy program wyświetli komunikat o niemożliwości wyświetlenia planszy w konsoli. Kolejne pokolenia nie będą wyświetlane w oknie wiersza poleceń ale generacja plików wynikowych nie ulegnie zmianie.

Treść komunikatu: „Wybrana szerokość planszy jest zbyt duża by możliwe było wyświetlenie kolejnych pokoleń w oknie konsoli.”

### 5.2 Błędy

Opis błędów, które mogą wystąpić w trakcie działania programu.

#### 5.2.1 Błędy pliku wejściowego

**Podany plik nie istnieje** Jeśli ścieżka podana przez użytkownika jest błędna program wyświetli komunikat o braku możliwości otwarcia wskazanego pliku i zakończy pracę.

Treść komunikatu: „Nie udało się otworzyć wskazanego pliku.”

**Pusty plik** W przypadku gdy plik wskazany przez użytkownika będzie pusty program powiadomi o tym i zakończy pracę.

Treść komunikatu: „Wskazany plik wejściowy jest pusty.”

**Rozmiar planszy nie będący liczbą** Jeśli w pierwszej linii pliku znajdować się będą wartości inne niż liczby program nie będzie w stanie wczytać rozmiaru planszy. W takiej sytuacji wyświetli odpowiedni komunikat i zakończy pracę. Treść komunikatu: „Nie udało się wczytać rozmiaru planszy.”

**Niedodatni rozmiar planszy** Jeśli jeden z wymiarów planszy nie będzie dodatnią liczbą całkowitą program zasygnalizuje błąd i zakończy pracę. Przykładowy komunikat: „Szerokość musi być większa od 0. Podana szerokość to -5.”

**Brak nowej linii po rozmiarze planszy** W przypadku gdy po wysokości planszy w pliku będzie inny znak niż przejście do nowej linii program zasygnalizuje błąd i zakończy pracę. Treść komunikatu: „Spodziewany koniec lini po wymiarze planszy.”

**Błąd przy wczytywaniu stanu komórki** Jeśli nie uda się wczytać stanu komórki, na przykład ponieważ w pliku jest za mało linii lub jedna z linii jest zbyt krótka, program wypisze, w którym miejscu pliku wystąpił błąd i zakończy pracę. Przykładowy komunikat: „Wystąpił błąd przy próbie przeczytania znaku w linii: 3 kolumnie: 8.”

**Nieprawidłowy znak w pliku** Jeśli podczas czytania pliku program napotka nieprawidłowy znak wypisze na jakiej pozycji w pliku napotkany został nieprawidłowy znak, jaki to znak, oraz czego spodziewał się program. Przykładowy komunikat: „Niewspierany znak napotkany w linii: 2 kolumnie: 1. Spodziewana wartość: 0 lub 1. Napotkana wartość: T”

### 5.2.2 Błędy losowe

**Brak pamięci operacyjnej** Gdyby w systemie zabrakło pamięci program nie będzie w stanie funkcjonować poprawnie. Program wyświetli komunikat o błędzie i przerwie pracę. Treść komunikatu: „Program nie uzyskał pamięci od systemu operacyjnego. Spróbuj uruchomić program ponownie za pewien czas.”