Specyfikacja funkcjonalna automat komórkowy **"Wire World"**

Bartosz Michałowski

1 kwietnia 2019

Spis treści

1 Opis ogólny **3**
1.1 Nazwa programu 3
1.2 Poruszany problem 3

2 Opis funkcjonalności **4**
2.1 Wymagania funkcjonalne 4
2.2 Jak korzystać z programu 4
2.3 Uruchomienie programu 4

3 Opis danych wejściowych i interfejsu **5**
3.1 Struktura pliku wejściowego 5
3.2 Przechowywanie danych w programie 5
3.3 Dane wyjściowe 6
3.4 Interfejs graficzny 6

4 Komunikaty dla użytkownika i testowanie **7**
4.1 Przykładowe komunikaty skierowane do użytkownika 7
4.2 Ogólny przebieg testowania 7

1 Opis ogólny

1.1 Nazwa programu

Nazwa programu: `WireWorld`

1.2 Poruszany problem

Automatem komórkowym określa się system składający się z pojedynczych komórek, znajdujących się obok siebie. Każda z komórek może przyjąć jeden ze stanów, przy czym liczba stanów jest skończona, ale dowolnie duża. Stan komórek zmieniany jest synchronicznie w zależności od komórek otaczających oraz przyjętych reguł. Poruszanym problemem jest implementacją takowego automatu komórkowego **Wire World** autorstwa Briana Silvermana i stworzenie modułu wizualizującego planszę w czasie rzeczywistym. Program zostanie napisany w języku **Java**. W programie nie przewiduje się zastosowania konstrukcji dostępnych w **Javie wersji 8** i wyższej.

2 Opis funkcjonalności

2.1 Wymagania funkcjonalne

- Wczytywanie do programu początkowej konfiguracji z pliku o wybranym formacie
- Przeprowadzanie żądanej liczby generacji.
- Wizualizacja w czasie rzeczywistym zmieniając generacje ręcznie lub włączając tryb auto-przełączania
- Zapisanie bieżącej generacji do pliku, który może być później wczytany
- Tworzenie i edytowanie generacji w trybie rzeczywistym za pomocą interfejsu graficznego
- Wstawianie gotowych struktur odpowiadających za bardziej zaawansowane struktury

2.2 Jak korzystać z programu

Obsługę programu umożliwia intuicyjny i czytelny interfejs graficzny. Jest on przeznaczony do wczytania z pliku odpowiedniej generacji komórek, narysowania jej ręcznie, bądź edytowania bieżącej. Z poziomu interfejsu użytkownik może importować gotowe struktury i sterować przebiegiem procesu generacji. Przez proces ten rozumie się przeprowadzenie symulacji, przechodzenie pomiędzy kolejnymi stanami i definiowanie określonych atrybutów, takich jak rozmiary planszy, czy miejsce zapisu wybranej generacji.

2.3 Uruchomienie programu

Program przeznaczony jest do uruchomiania za pomocą pliku `WireWorld.jar`.

3 Opis danych wejściowych i interfejsu

3.1 Struktura pliku wejściowego

W Wireworld wyróżniamy następujące stany komórek:

- `EMPTY` pusta komórka
- `CONDUCTOR` komórka przewodząca elektrony
- `HEAD` głowa elektronu
- `TAIL` ogon elektronu

Oraz następujące struktury:

- `ELECTRON` elektron
- `DIODE` dioda
- `OR` bramka logiczna OR
- `AND` bramka logiczna AND
- `XOR` bramka logiczna XOR
- `CLOCK` prosty generator elektronów

Plik tekstowy musi zawierać w pierwszej kolejności wymiary planszy, na której przeprowadzane będą generacje, jako dwie kolejne liczby całkowite. Następnie w oddzielnych liniach należy zamieścić instrukcje modyfikujące komórki siatki zgodnie z następującym wzorcem:

`NAZWA_STRUKTURY/STANU POZYCJA_X POZYCJA_Y ORIENTACJA`

Orientacja jest argumentem opcjonalnym i dostępnym jedynie dla struktur, a w wypadku próby przypisania jej do stanu zostanie ona pominięta. Za rotacje odpowiadają następujące wartości argumentu:

- **L** struktura zwrócona w stronę lewej krawędzi siatki
- **U** struktura zwrócona w stronę górnej krawędzi siatki
- **R** struktura zwrócona w stronę prawej krawędzi siatki
- **D** struktura zwrócona w stronę dolnej krawędzi siatki

Domyślnie struktury zwrócone są w stronę prawej krawędzi planszy.

3.2 Przechowywanie danych w programie

Wszelkie informacje w programie przechowywane są w polach obiektów odpowiednich klas.

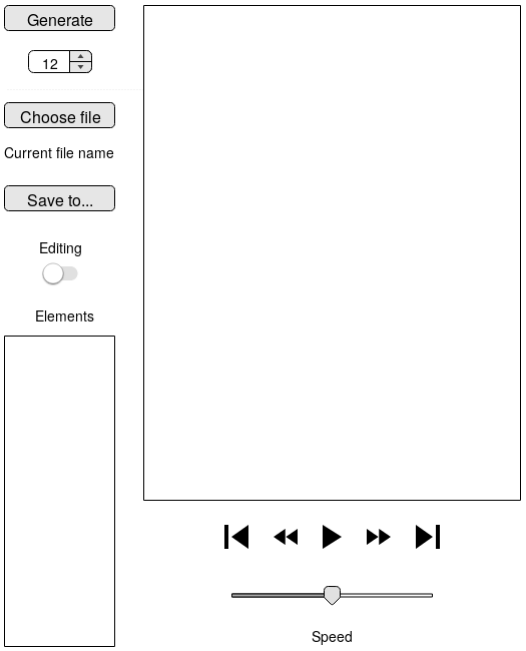
3.3 Dane wyjściowe

Z poziomu interfejsu graficznego istnieje możliwość stworzenia pliku zawierający wybraną generacje w formie analogicznej do formy pliku wejściowego, w miejscu zdefiniowanym przez użytkownika.

3.4 Interfejs graficzny

Interfejs graficzny daje użytkownikowi szeroki zakres opcji umożliwiających modyfikacje warunków w jakich nastąpi generacja. Należą do nich:

- Wczytywanie pierwszej generacji z pliku
- Tworzenie pierwszej generacji z dostępnych elementów
- Wybór liczby pożądanych generacji
- Zapisanie aktualnego stanu siatki do pliku
- Poruszanie się po kolejnych generacjach
- Automatyczne przechodzenie przez generacje przy wybranej prędkości



Rysunek 1: Szkic poglądowy interfejsu graficznego

4 Komunikaty dla użytkownika i testowanie

4.1 Przykładowe komunikaty skierowane do użytkownika

- Pomyślnie utworzono generację
- Nie udało się wczytać pliku wejściowego - spowodowane brakiem uprawnień
- Nie udało się utworzyć pliku wyjściowego - spowodowane brakiem uprawnień
- Błędny format pliku wejściowego - plik wejściowy zawiera niedozwolone znaki

4.2 Ogólny przebieg testowania

Na ogół procesów testowania będą składały się odpowiednie fazy testowe. Pozwolą one w sposób optymalny sprawdzać tworzony program. Wodzącymi narzędziami użytymi do weryfikowania kodu będą `JUnit` i `AssertJ`. Interfejs graficzny zostanie sprawdzony ręcznie, podczas tworzenia programu. Celem procesu testowania jest zapewnienie obsługi zdarzeń, w których program może zachowywać się w sposób niekontrolowany.