

Game of Life

wersja asynchroniczna – projekt studencki

Autor:
Tomasz Bajorek

1. Wstęp

Celem projektu było stworzenie aplikacji, realizującej asynchroniczny automat komórkowy „Life”. Jego zasady wymyślił w 1970r. Brytyjski matematyk John Conway.

2. Założenia

Przed realizacją projektu przyjęto następujące założenia:

1. aplikacja działa w środowisku przeglądarki internetowej i jest umieszczona na dostępnym serwerze WWW;
2. umożliwia definiowanie początkowego zagęszczenia żywych organizmów w chwili $t = 0$;
3. posiada przyciski, pozwalające sterować przebiegiem symulacji: start/stop, następny krok oraz restart z zadanymi ustawieniami;
4. prezentuje dwuwymiarową planszę, a której wizualizowany jest przebieg symulacji;
5. w każdej chwili jest podawana aktualna ilość żywych komórek;
6. aplikacja zawiera wykres prezentujący historię poprzednich wartości liczności populacji;
7. algorytm Life działa w sposób asynchroniczny;
8. w symulacji powinny być zastosowane okresowe warunki brzegowe.

3. Life – zasady

Deterministyczny automat komórkowy Life jest zdefiniowany na dwuwymiarowej sieci kwadratowej. Każda komórka może przyjmować dwa stany: 0 – martwa lub 1 – żywa. Jej otoczeniem są wszystkie komórki, sąsiadujące z nią bokiem lub rogiem, co daje obszar otoczenia Moore'a o promieniu 1.

Zachowanie automatu w kolejnych chwilach czasowych jest determinowane przez następujące reguły:

- ✓ każda żywa komórka, która w czasie t ma dokładnie dwóch lub trzech żywych sąsiadów, będzie żywa w czasie $t+1$;
- ✓ każda martwa komórka, która w czasie t ma dokładnie trzech żywych sąsiadów, będzie żywa w czasie $t+1$;
- ✓ wszystkie pozostałe komórki, do których w czasie t nie odnoszą się powyższe zasady, będą martwe w czasie $t+1$.

4. Realizacja

4.1. Wykorzystane technologie

Aplikacja została zrealizowana przy pomocy następujących języków i technologii: HTML5, JavaScript oraz CSS. Konsekwencją wyboru biblioteki React¹ jako głównego narzędzia wspomagającego pisanie programu, stała się jego komponentowa struktura. Dodatkowo do wspomagania przepływu danych między komponentami zgodnie z regułami wzorca Flux² zastosowano bibliotekę Reflux³.

Aplikacja została stworzona z wykorzystaniem odmiany ES2015 języka JavaScript. Do kompilacji kodu, wykorzystano narzędzie browserify⁴ wraz z wtyczką babelify⁵. Kod CSS powstaje po kompilacji plików źródłowych SASS⁶. Za zarządzanie projektem produkcję plików wyjściowych odpowiada Gulp⁷. Obsługę zależności aplikacji od zewnętrznych bibliotek i pakietów dostarcza narzędzie npm⁸. Ikony przycisków zostały stworzone przy pomocy biblioteki FontAwesome⁹. Do rysowania wykresu populacji użyto biblioteki Highcharts¹⁰. Projekt jest kontrolowany przez system zarządzania wersjami Git¹¹ oraz hostowany na serwerze github.com.

4.2. Struktura projektu

W głównym katalogu aplikacji znajdują się następujące podkatalogi i pliki:

- *app* – kod źródłowy aplikacji, napisany w języku JavaScript (ES2015);
- *css* – wynikowy kod CSS po kompilacji, używany przez przeglądarkę internetową;
- *font* – pliki biblioteki FontAwesome;
- *js* – kod JavaScript po kompilacji, używany przez przeglądarkę internetową;
- *sass* – kod źródłowy arkuszy stylów, napisany w SASS oraz kod źródłowy biblioteki FA;
- *.gitignore* – lista katalogów i plików, które nie mają być indeksowane przez Git;
- *gulpfile.js* – plik konfiguracyjny narzędzia Gulp;
- *index.html* – plik aplikacji, otwierany przez przeglądarkę internetową;
- *package.json* – zbiór zależności dla narzędzia npm, pozwalający przy jego pomocy zainstalować projekt lokalnie w środowisku deweloperskim lub produkcyjnym.

4.3. Koncepcja działania

Aplikacja składa się z komponentów, będących klasami, dziedziczącymi po klasie *Reflux.Component*. Model aplikacji, zawierający dane i operujący na nich dziedziczy po klasie *Reflux.Store*. Zdarzenia w aplikacji są reprezentowane przez akcje biblioteki Reflux. Z modelu dane są dystrybuowane do komponentów.

¹ <https://facebook.github.io/react/>

² <https://facebook.github.io/flux/>

³ <https://github.com/reflux/refluxjs>

⁴ <http://browserify.org/>

⁵ <https://github.com/babel/babelify>

⁶ <http://sass-lang.com/>

⁷ <http://gulpjs.com/>

⁸ <https://www.npmjs.com/>

⁹ <http://fontawesome.io/>

¹⁰ <https://www.highcharts.com/>

¹¹ <https://git-scm.com/>

Logika automatu komórkowego została zawarta w klasie Model. Zapewnia ona poprawną obsługę planszy, definiuje mechanizm jej inicjalizacji losowymi wartościami, kontroluje proces symulacji, a także pozwala ustawić takie parametry, jak rodzaj otoczenia lub funkcja zawierająca zbiór reguł, pozwalających wyliczyć stan komórki w następnej chwili.

Inicjalizacja planszy zadaną gęstością odbywa się poprzez ustawienie wszystkich komórek jako martwe, a następnie losowanie tylu z nich, które mają być żywe, aby osiągnąć zadaną liczebność.

Plansza jest przedstawiona jako dwuwymiarowa tablica w języku JavaScript. Model zawiera funkcję transformującą współrzędne spoza zakresu planszy tak, aby zapewnić periodyczne warunki brzegowe dla symulacji.

4.4. Interfejs aplikacji

Centralnym elementem aplikacji jest plansza, wizualizująca symulację o rozmiarze 700 x 700 pikseli. Użytkownik może wybrać, jaka ilość komórek ma podlegać symulacji. W przypadku wyboru różnej wielkości wymiarów komórki będą prostokątami.

W lewej części aplikacji znajduje się panel, pozwalający sterować parametrami symulacji, takimi, jak ilość komórek w pionie i w poziomie oraz gęstość żywej populacji przed startem symulowania. Minimalne wymiary planszy to 2 x 2, a maksymalne 280 x 280. Gęstość musi się znaleźć w przedziale domkniętym od 0 do 1. Panel boczny zawiera również informacje o aktualnej chwili czasowej oraz bieżącej liczności populacji. Poniżej użytkownik ma do dyspozycji przyciski, sterujące przebiegiem symulacji:

- start/stop – pozwalający rozpocząć lub zatrzymać proces symulacji;
- następny krok – wykonujący tylko jeden krok symulacji, dostępny tylko wtedy, gdy nie jest ona uruchomiona;
- reset – umożliwiający zresetowanie planszy z nowymi warunkami początkowymi w momencie, gdy nie jest włączona symulacja.

Na dole panelu znajdują się linki do kodu źródłowego i dokumentacji oraz informacje o autorze aplikacji.

W dolnej części okna jest położony wykres, prezentujący zmiany liczności populacji w czasie. Zawiera on maksymalnie 100 ostatnich chwil czasowych.

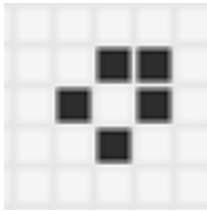
5. Obserwacje

5.1. Charakterystyczne kształty

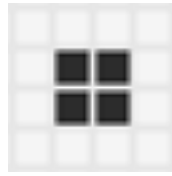
Podczas wielokrotnego uruchamiania automatu zaobserwowano kilka charakterystycznych kształtów, tworzonych przez grupy komórek. Można wyróżnić ich dwa rodzaje: stałe elementy oraz statki kosmiczne.

5.1.1. Elementy stałe

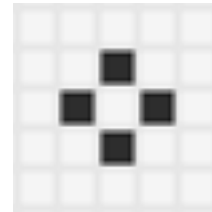
Na poniższych rysunkach przedstawiono zaobserwowane grupy niezmiennie.



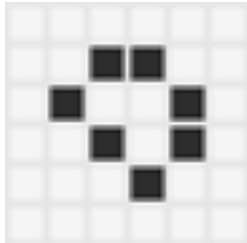
Rysunek 1 Łódź



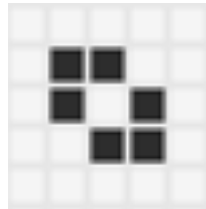
Rysunek 2 Blok



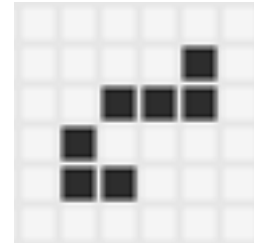
Rysunek 3 Koniczynka



Rysunek 4 Bochenek



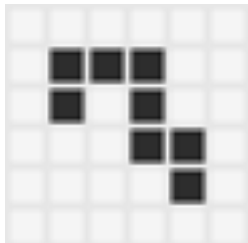
Rysunek 5 Statek



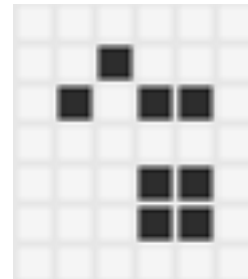
Rysunek 6 Zjadacz

5.1.2. Statki

W czasie przeprowadzonych symulacji znaleziono jeden statek kosmiczny, poruszający się ukośnie po planszy. Poniższy rysunek przedstawia dwie fazy, w jakich można było zaobserwować ów element.



Rysunek 7 Statek kosmiczny - faza I



Rysunek 8 Statek kosmiczny - faza II

5.2. Zachowanie się planszy

Po licznych obserwacjach planszy można stwierdzić, iż automat komórkowy Life w wersji asynchronicznej dąży do labiryntowego rozłożenia żywych komórek. Liczność populacji w tym stanie osiąga około 40%. Jest to ilość, do której dąży automat.

Jeżeli ustawiona zostanie mniejsza liczność populacji żywych komórek, będzie ona rosła do uzyskania tej wartości. Wartość minimalna dla planszy o startowych wymiarach 100 x 100 komórek, dla której zazwyczaj populacja zaczyna dążyć do okolic 40% to 6%. Dla 5% to, czy liczba komórek zacznie zmierzać do wspomnianej granicy, czy też ustali się na niższym stałym poziomie, utrzymując same "skamieliny", zależało od ustawienia początkowego żywych komórek. W przypadku mniejszych wartości życie zamierało całkowicie lub kończyło się na kilku stałych grupach komórek.

Dla analizowanej planszy 100 x 100 automat będzie dążył do okolic 40% również dla większej gęstości początkowej. Im wyższa wartość zostanie ustawiona dla chwili $t=0$ (przed rozpoczęciem symulacji), tym większy spadek zanotuje populacja w chwili $t=1$. Jednakże już od następnej chwili czasowej rozpocznie się wzrost do wspomnianej powyżej liczności komórek.

Testując dane dla mniejszych plansz, np. 50 x 50 komórek potrzeba było trochę większej gęstości (ok. 10%), aby populacja prawie zawsze zaczynała dążyć do okolic 40%.

Jeżeli gęstość startowa zostanie ustawiona na poziomie około 40%, wartość ta zostanie zachowana w kolejnych krokach. Będzie się natomiast zmieniać ustawienie komórek.

6. Wnioski

Podsumowując niniejszy projekt można wyciągnąć kilka wniosków:

- ✓ wersja asynchroniczna zachowuje się w inny sposób, niż wersja synchroniczna;
- ✓ mimo różnicy można zaobserwować grupy komórek, charakterystyczne także dla wersji synchronicznej, jednak zazwyczaj są one nietrwałe, gdyż zostają wchłonięte przez rozrastający się obszar żywych komórek; wyjątek stanowią stany niezmiennie planszy, powstałe przy ustawieniu odpowiednio małej gęstości początkowej;
- ✓ liczność populacji w okolicy 40% to wartość, do której dąży automat, dla większości gęstości początkowych.