Sprawozdanie
Projekt „Demographic Simulator”

Mateusz Smoliński

30 maja 2019

Temat:	**Wersja:**
Podsumowanie projektu	1.0
Autor:	**Data:**
Mateusz Smoliński e-mail: 01131251@pw.edu.pl	30.05.2019

Spis treści

1 Wstęp

Sprawozdanie powstało w celu podsumowania projektu indywidualnego. Celem projektu było zapoznanie się z językiem programowania C# oraz utworzenie prototypowej aplikacji w tym języku. Do ćwiczeń z językiem oraz napisania aplikacji miało służyć środowisko programistyczne Microsoft Visual Studio.

Tematem aplikacji była symulacja demograficzna na prostym, wyznaczonym geometrycznie obszarze. Otrzymuje plik z danymi o wyznaczonym obszarze oraz znajdujących się na nim miastach. Użytkownik może zmieniać różne parametry wewnątrz programu, od których zależy przyrost naturalny oraz częstotliwość występowania zdarzeń losowych. Po zamieszczeniu wszystkich potrzebnych informacji możliwe jest uruchomienie symulacji z możliwością dostosowania szybkości.

Projekt został zrealizowany zgodnie z wytycznymi założonymi na początku – spełnia wszystkie funkcje wymienione w karcie projektu. Zmiany w stosunku do wczesnej dokumentacji to przede wszystkim lekko zmieniona koncepcja interfejsu, wprowadzania danych z pliku oraz zmiany strukturalne w kodzie programu.

Program jest dostępny w serwisie GitHub pod następującym adresem: `https://github.com/matsmolinski/DemographicSimulator`

2 Zmiany w prowadzeniu projektu

2.1 Terminarz

Większość odbiorów zostało zrealizowanych zgodnie z terminem założonym na początku projektu. Przedstawienie projektu aplikacji zostało przesunięte o blisko dwa tygodnie z powodów zdrowotnych. Zatwierdzenie aplikacji oraz wysłanie sprawozdania końcowego odbędzie się przed terminem ostatecznym ustalonym na karcie projektu.

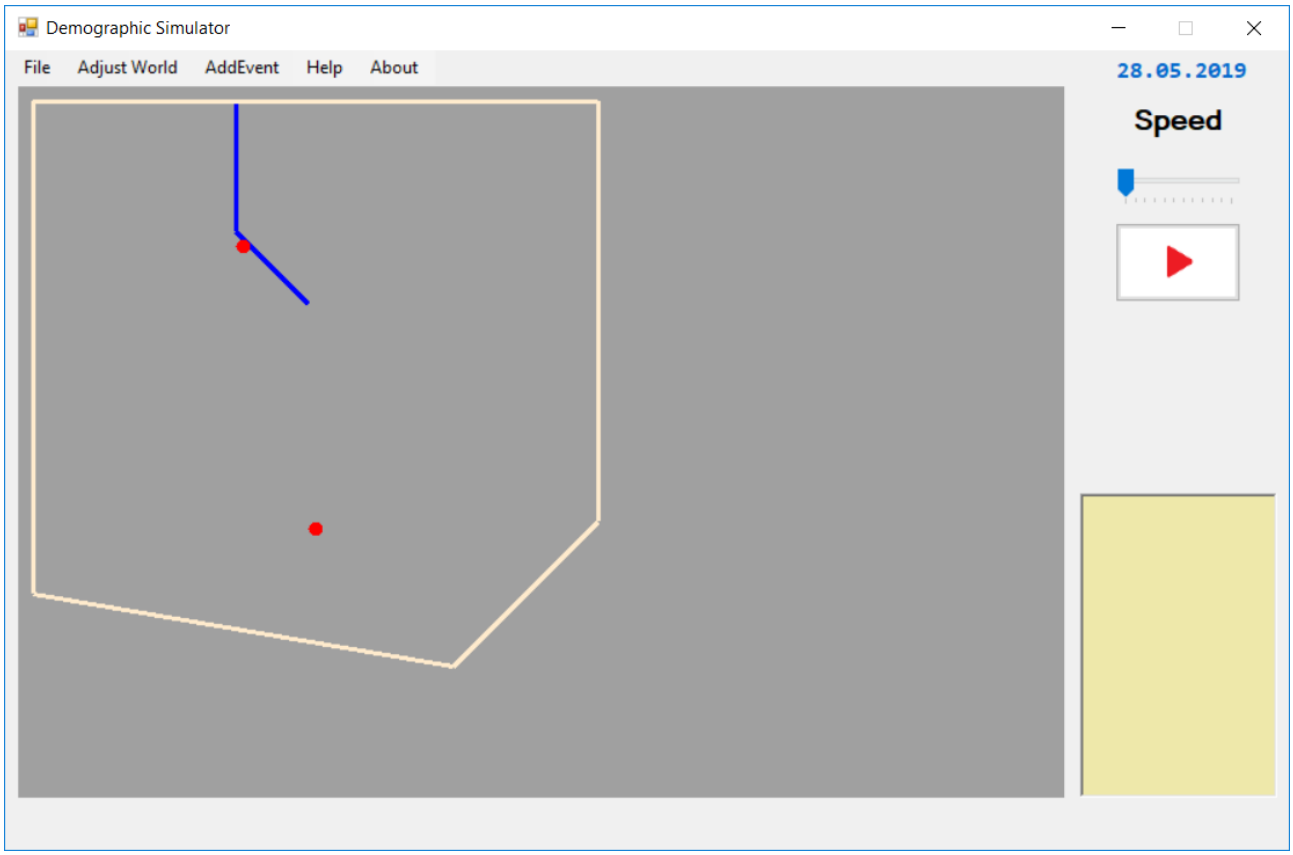
2.2 Środowisko programistyczne

Całość projektu została przeprowadzona za pomocą narzędzia Microsoft Visual Studio. W trakcie projektu została przetestowana nowa wersja Microsoft VS 2019, jednak ze względów licencyjnych ostatnie prace nad aplikacją zostały ulokowane w wersji z 2017 roku.

3 Zmiany względem specyfikacji funkcjonalnej

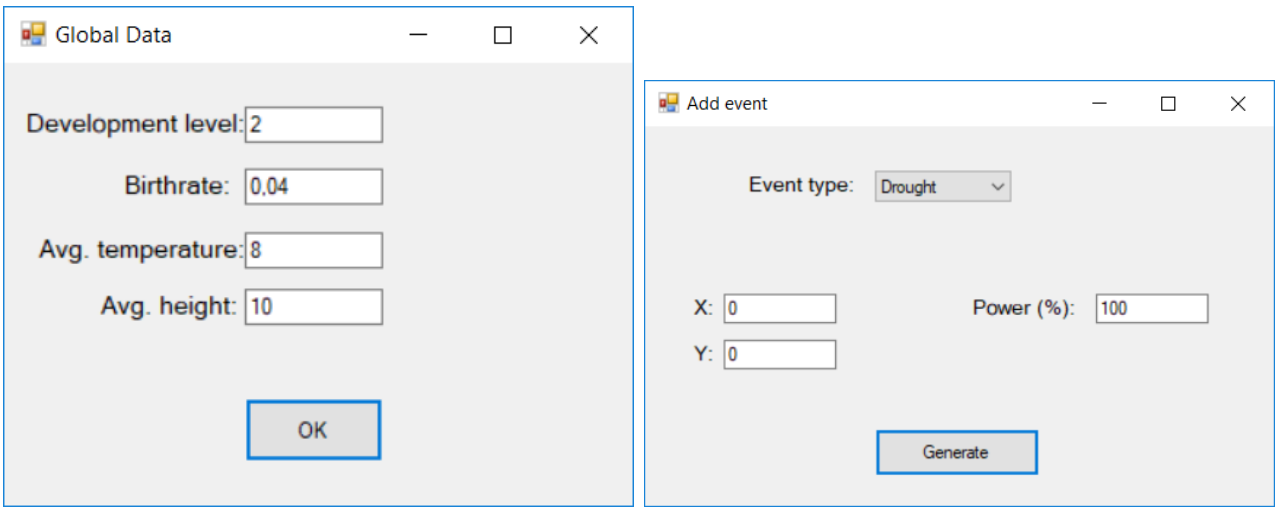
3.1 Graficzny interfejs użytkownika

Wygląd programu nie uległ dużej zmianie względem wstępnych projektów. Rysowanie na mapie zostało zautomatyzowane, przez co kolory i proporcje lekko różnią się od projektu. Dodany został także przycisk uruchamiający i zatrzymujący symulację, przypadkowo pominięty w pierwszej wersji okna.



Rysunek 1: Okno główne

Dodane zostały również dwa okna: generujące zdarzenia oraz zmieniające parametry mapy, których wstępna wizualizacja nie została przedstawiona w specyfikacjach, a same klasy zostały pominięte w modelu UML.



(a) Dostosowanie mapy (b) Dodawanie zdarzeń

Rysunek 2: Okna pomocnicze

3.2 Plik wejściowy

Zmiany w dostarczanym pliku wejściowym ograniczyły się do zrezygnowania z podawania oddzielnej temperatury do każdego z miast – opcjonalnym argumentem przy wprowadzaniu miast pozostała wysokość. Dodatkowo, zmieniono nazwy parametrów globalnych:

- zamiast temperatura_globalna należy podać srednia_temperatura,
- zamiast wysokosc_globalna należy podać srednia_wysokosc.

3.3 Sytuacje wyjątkowe

Do sytuacji wyjątkowych zostały dodane ograniczenia planszy – parametry muszą się zawierać w obszarze mapy, gdzie X ma być mniejszy niż 600, a Y mniejszy niż 500. Ponadto, program uniemożliwi podania współczynnika rozwoju cywilizacyjnego powyżej 10, przyjętego za maksymalną wartość tego parametru.

4 Zmiany względem projektu aplikacji

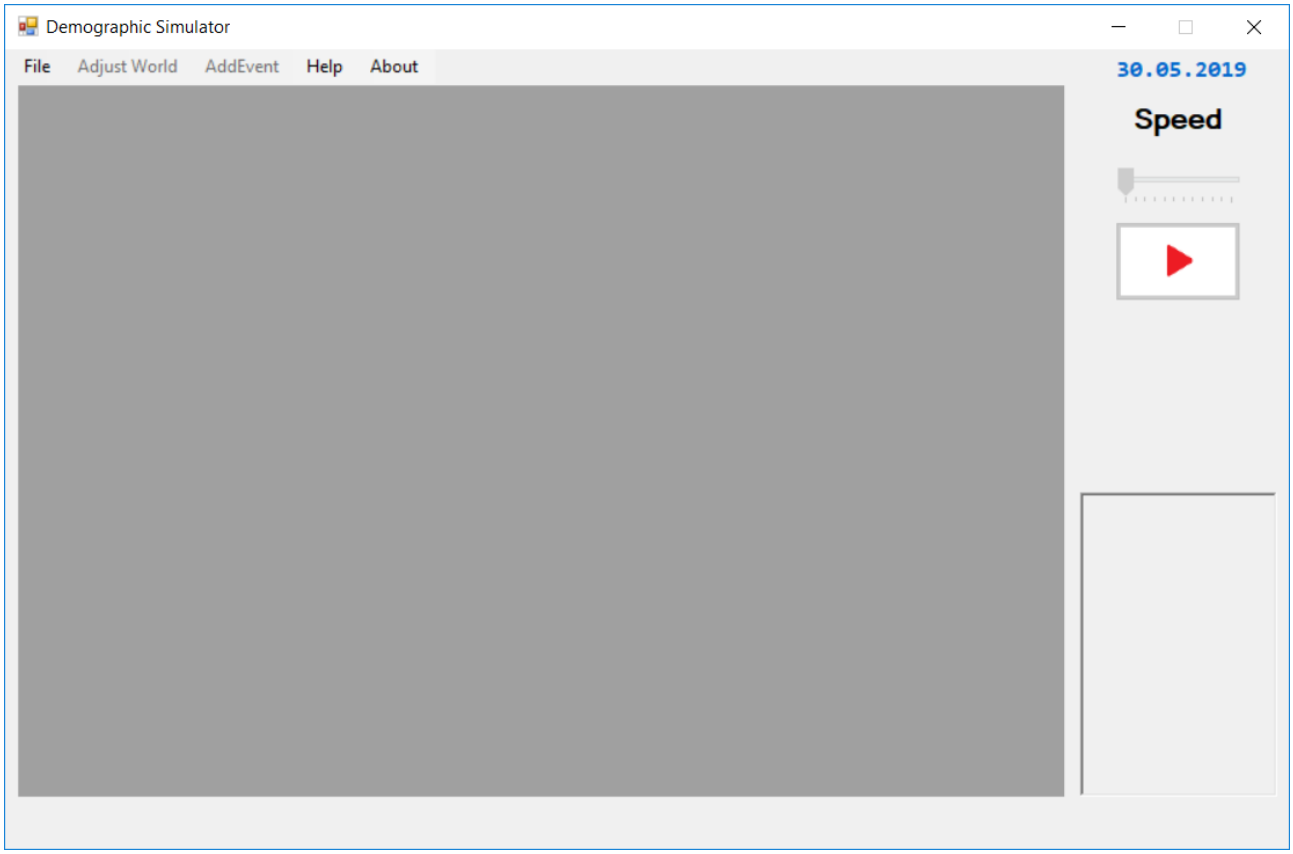
Wiele metod umieszczonych na diagramie klas zmieniło parametry bądź też zwracane typy. Do najistotniejszych przykładów można zaliczyć:

- Metoda interfejsu *IGameEvent* o nazwie *ProceedEvent* zmieniła się na typ void, uzyskała także parametry w postaci punktu centralnego i mocy (liczby całkowitej od 0 do 100). Metoda oddaje także parametry referencyjne z liczbą ofiar i/lub migrantów.
- Metoda *MakeTimeJump* nie przyjmuje już liczby miesięcy do przesunięcia czasowego, a zawsze wykonuje przeskok jednego miesiąca.

Powstały także nowe metody, upraszczające inne metody, które były bardziej złożone. Przykładem takiego podziału jest *ReadData*, z którego zostały wydzielone metody do odczytu poszczególnych linii pliku konfiguracyjnego.

5 Działanie programu

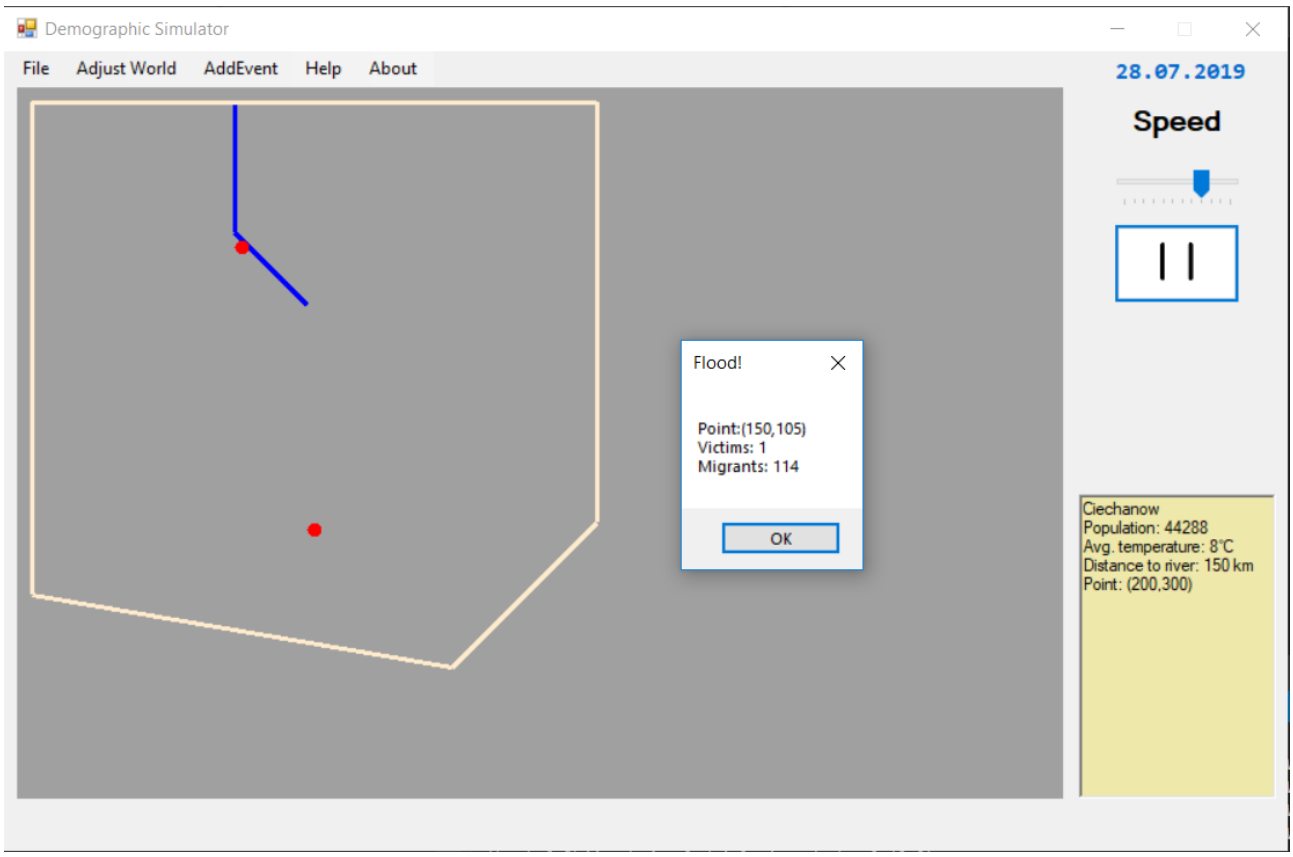
Uruchomienie programu powoduje otwarcie głównego okna programu, przedstawionego w poprzedniej sekcji. Wygląd okna będzie jednak neutralny, dopóki nie zostanie wczytany poprawny plik konfiguracyjny:



Rysunek 3: Okno aplikacji przed uruchomieniem symulatora

Można tego dokonać wybierając opcję *Load file* z menu *File*. W przypadku podania niewłaściwego pliku lub pliku z niepoprawnymi wartościami program wypisze na ekranie stosowny komunikat do zaistniałej sytuacji, wskazując na źródło błędu, a w przypadku pojedynczych błędów w linijkach pliku także numery błędnych linii.

Czerwony przycisk start rozpoczyna działanie symulacji. Można kontrolować jej przebieg za pomocą znajdującego się nad nim suwaka. W przypadku wygenerowania zdarzenia program wyświetli komunikat z informacjami o zdarzeniu.



Rysunek 4: Zdarzenie wywołane przez program

Symulacja zostanie wznowiona po zamknięciu wyskakującego okna. Dla ułatwienia kontroli, możliwe jest użycie przycisku pauzy także w momencie odczytu wydarzenia, żeby uniemożliwić programowi generowanie kolejnych wydarzeń.

Użytkownik może poza symulacją lub w trakcie symulacji zmieniać parametry lub dodawać zdarzenia ręcznie za pomocą przycisków znajdujących się na górnym menu. Wywołują one okna opisane w poprzedniej sekcji.

Program może w dowolnym momencie zamknąć aplikację lub wczytać kolejny plik do symulacji – program w razie potrzeby zamknie samodzielnie wątki związane z poprzednią symulacją, przygotowując się do kolejnej.

6 Podsumowanie i wnioski

6.1 Założenia, które udało się spełnić

Kluczowym założeniem projektu indywidualnego było zapoznanie się z nowym językiem programowania. Dlatego przy planowaniu aplikacji starałem się ułożyć taki model oprogramowania, który umożliwi wykorzystanie wielu rozwiązań programistycznych, z którymi spotkałem się wcześniej w większym lub mniejszym stopniu, w danym projekcie. Do takich rozwiązań należały między innymi:

- wielowątkowość,
- przeciążanie metod i operatorów,
- zastosowanie interfejsów,
- obsługa zdarzeń.

Większość z nich znalazła zastosowanie w programie, innych rozwiązań nauczyłem się w trakcie rozwiązywania problemów. Przykładem takich rozwiązań mogą być parametry referencyjne metod, które umożliwiają na proste uzyskanie kilku konkretnych informacji z metody bez potrzeby tworzenia dodatkowych klas.

Projekt został także w pełni zrealizowany pod względem dokumentacji. Wszystkie dokumenty założone na początku jako produkty projektu zostały przygotowane i oddane.

6.2 Założenia, które nie zostały w pełni spełnione

Zapoznanie się z językiem programowania przy budowie pojedynczej aplikacji wiązało się z pewnymi ograniczeniami. Nie wszystkie mechanizmy charakterystyczne dla języka C# zostały w nim wykorzystane, z uwagi na upływający czas lub nieznalezienie dla nich zastosowania w danym projekcie. Są to między innymi:

- synchroniczna praca wątków,
- alternatywne wyrażenia (np. Lambda).

Innym problemem, który został zrealizowany, jednak w sposób odmienny od tego założonego na początku, były wyliczenia symulacyjne. Pierwotnie miały one być oparte bezpośrednio na analizie prostych danych statystycznych (takich jak średnie/maksymalne wartości wysokości, temperatur). Jednak przy ilości i złożoności tych zagadnień odbiegających od pierwotnej roli programu zdecydowałem się wyważyć te zdarzenia na podstawie testów, żeby uzyskać czytelne w programie i satysfakcjonujące symulacje na podstawie niewielkiej ilości danych. Skutkiem tej decyzji jest duża ilość liczb w programie, które nie są wprost opisane żadnymi demograficznymi właściwościami.

6.3 Wnioski

1. Język C# okazał się być bardzo ciekawym językiem programowania, pełnym rozwiązań niedostępnych w innych językach obiektowych, takich jak np. Java. Wiedza o tym języku wyniesiona z tego projektu może być znaczącym czynnikiem dla ukończenia studiów czy też znalezienia przyszłej pracy.

2. Sam koncept symulacji demograficznej był problemem bardzo złożonym i na potrzeby realizacji projektu musiałem przyjąć pewien poziom abstrakcji. Dlatego złożone i skomplikowane zdarzenia naturalne zostały zdefiniowane prostymi zależnościami oraz wartościami losowymi. Najważniejszy koncept tego problemu został zrealizowany – wzrost pewnych wartości powoduje faktyczne zwiększenie prawdopodobieństwa powiązanych z nim wydarzeń. Bardzo istotne było dla mnie umożliwienie samodzielnego funkcjonowania („życia") miast przedstawionych na mapie, co zostało zrealizowane.

3. Utworzony symulator może stanowić bazę dla przyszłych projektów. Wiele mechanizmów zastosowanych w projekcie może znaleźć zastosowanie np. w prostych grach strategicznych.

Data:	Autor:	Zakres:	Zatwierdził:	Wersja:
28.05.2019	MS	Utworzenie dokumentu, wstęp	MS	0.1
28.05.2019	MS	Opis zmian	MS	0.2
30.05.2019	MS	Prezentacja działania programu	MS	0.3
30.05.2019	MS	Sformułowanie podsumowania i wniosków	MS	1.0