Uniwersytet Warmińsko-Mazurski w Olsztynie
Wydział Matematyki i Informatyki

Kierunek: Informatyka

Krzysztof Pietraszko

Symulacja ruchu drogowego na skrzyżowaniu z sygnalizacją świetlną sterowaną algorytmem ewolucyjnym

Praca inżynierska wykonana
w Katedrze Metod Matematycznych Informatyki
pod kierunkiem
Dr Inż. Bartosza Nowaka

2019, Olsztyn

University of Warmia and Mazury in Olsztyn
Faculty of Mathematics and Computer Science

Computer Science

Krzysztof Pietraszko

Traffic simulation on junction with traffic lights controlled by an evolutionary algorithm

Engineer's Thesis is performed
in the Department of Mathematical
Methods in Computer Science
under supervision of
Dr Inż. Bartosz Nowak

2019, Olsztyn

Streszczenie

łebełebe

Abstract

webewebe

Spis treści

Wstęp

Komputery są obecnie niezwykle popularnymi urządzeniami znajdującymi zastosowanie w wielu dziedzinach życia. Są pomocne także w zadaniach inżynieryjnych takich jak problemy optymalizacyjne.

Problemy optymalizacyjne

Problemy w wielu obszarach matematyki, inżynierii, ekonomii, medycyny i statystyki mogą być przedstawione w kategoriach optymalizacji.
Określenie problemu optymalizacyjnego rozpoczyna się od ustalenia zbioru zmiennych niezależnych lub parametrów. Często formułuje się również ograniczenia, które wyznaczają dozwolone wartości zmiennych. Inną istotną składową problemu optymalizacyjnego jest funkcja celu, której wartość zależy od zmiennych. Rozwiązaniem takiego problemu jest zbiór dozwolonych wartości zmiennych, dla których funkcja przyjmuje optymalną wartość (minimalną lub maksymalną, zależnie od badanego zagadnienia)[1].

Do rozwiązywania problemów optymalizacyjnych często stosowane są metody sztucznej inteligencji. Dostarczają lepszych, szybszych i bardziej precyzyjnych rozwiązań niż konwencjonalne techniki, szczególnie w złożonych problemach inżynieryjnych. Spośród ich charakterystycznych cech warto wymienić następujące:
— Metody te „pamiętają" swoje poprzednie odkrycia,
— Dostosowują swoją wydajność (np. decydują o eksploracji lub eksploatacji), co przekłada się na unikanie utknięcia w minimum lokalnym,
— Mogą planować swoje działania długofalowo [2].

Algorytmy ewolucyjne

Przykładowym typem metod sztucznej inteligencji są algorytmy ewolucyjne. Poszukują one optymalnego rozwiązania problemu w sposób, który czerpie inspirację z mechanizmu ewolucji gatunków. Jednym z głównych czynników wyróżniających ten zbiór technik jest to, że jednocześnie zmieniana jest pewna populacja rozwiązań, a jej przekształcenia przypominają procesy zachodzące w przyrodzie. Zrozumienie algorytmów ewolucyjnych wymaga przyswojenia pewnych kluczowych pojęć, takich jak dostosowanie (ang. *fitness*) stanowiące ocenę danego rozwiązania, oraz krzyżowanie i mutacja będące mechanizmami przekształcania populacji.

Krzyżowanie tworzy nowe rozwiązanie łącząc cechy dwóch innych, podczas gdy mutacja w losowy sposób modyfikuje istniejące rozwiązanie. Przebieg algorytmu genetycznego rozpoczyna się od utworzenia początkowej populacji rozwiązań, które zazwyczaj są generowane losowo. W związku z tym zakłada się, że jest to różnorodny zbiór zawierający dobre i złe cechy. Algorytm następnie generuje szereg kolejnych populacji (pokoleń), z pomocą wyżej wspomnianych mechanizmów przekształcających [3]. Kluczowe jest tu zachowanie odpowiedniej równowagi między wykorzystywaniem znanych pozytywnych cech (a więc krzyżowaniem) a odkrywaniem nowych możliwości w przestrzeni dopuszczalnych rozwiązań (a więc mutacją). Istnieją modele mające na celu zmniejszenie tego problemu. Przykładowo działanie algorytmu można podzielić na fazy: pozyskiwania, akumulacji i wykorzystywania wiedzy [4]. Innym rozwiązaniem jest przyjęcie zmiennego współczynnika mutacji danego pewną funkcją. Taką technikę przyjęto w tej pracy.

Przykładowe zastosowania algorytmów ewolucyjnych

Do ciekawych zastosowań algorytmów ewolucyjnych należą m.in. antena statku kosmicznego [5], czy też wentylator silnika [6], których kształt został zaprojektowany przez taki algorytm. Obszary przejawiające duży potencjał to chociażby planowanie produkcji [7], przewidywanie zmian temperatury na Ziemi[8] i wiele innych [9].

W tej pracy omówiono wykorzystanie algorytmu ewolucyjnego do optymalizacji sygnalizacji świetlnej na skrzyżowaniu. Każda zmienna niezależna to czas, przez jaki wybrane sygnalizatory są zielone. Ewaluacja funkcji celu każdorazowo wymaga przeprowadzenia symulacji ruchu pojazdów, którą w tym celu zaimplementowano. Symulacja trwa dopóki określona liczba samochodów nie opuści skrzyżowania, a czas jej trwania decyduje o ocenie danego osobnika.

TODO: OPIS ROZDZIAŁÓW

Użyte technologie

Głównymi czynnikami decydującymi o doborze technologii były
— Prostota interakcji z kartą graficzną,
— Dobre narzędzia do debugowania kodu,
— Jakość i kompletność dokumentacji,
— Wydajność,
— Możliwość prostej implementacji interfejsu graficznego,
— Wcześniejsze doświadczenie,
— Szybkość kompilacji – istotna przy częstych zmianach w kodzie.

Unity

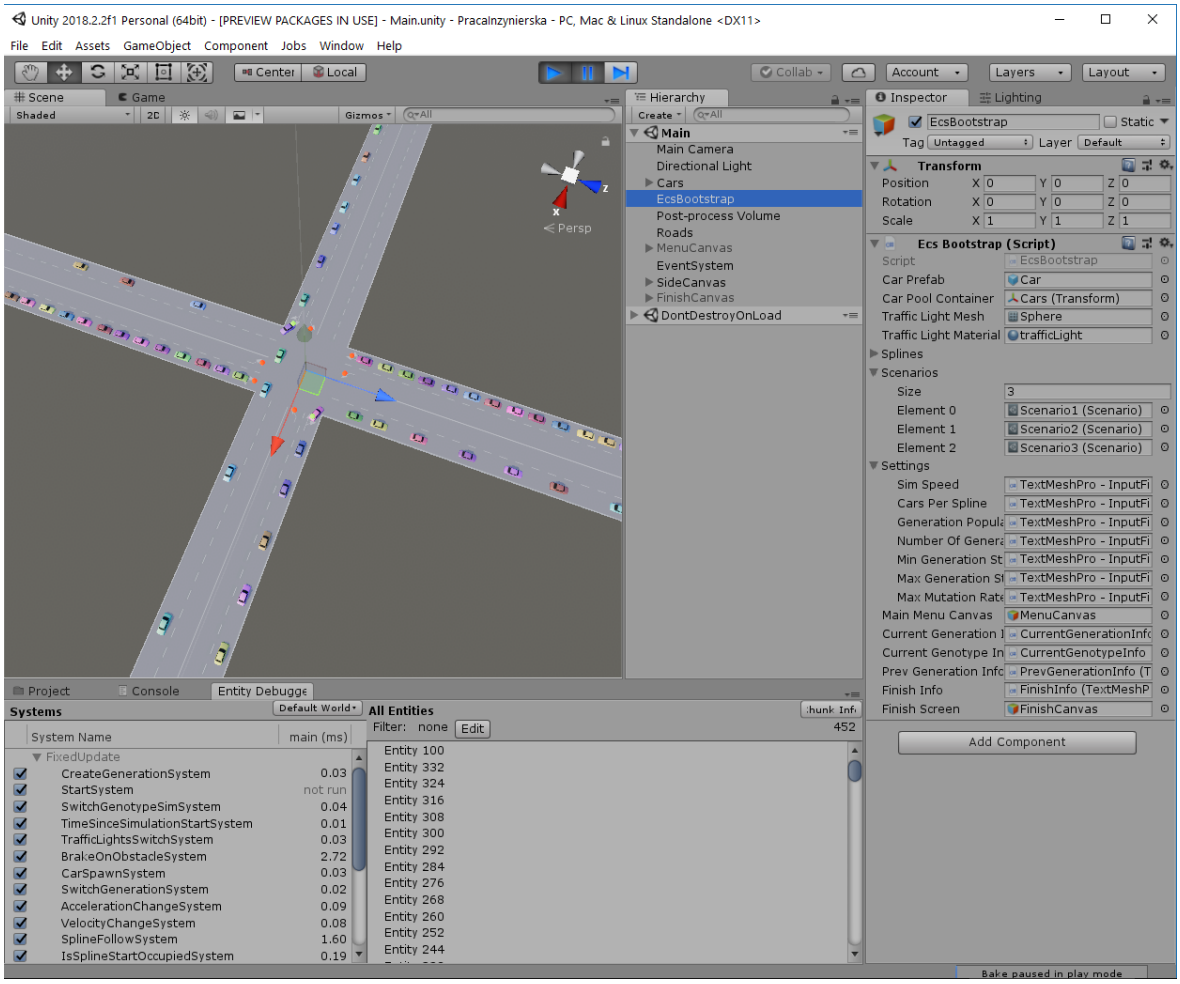
Unity to wieloplatformowy silnik autorstwa firmy Unity Technologies, którego głównym zastosowaniem jest tworzenie gier dwuwymiarowych i trójwymiarowych oraz symulacji. W porównaniu do podobnych narzędzi wyróżnia się prostotą obsługi, dużym naciskiem na edytory wizualne i zintegrowanym środowiskiem, które nie wymaga skomplikowanej konfiguracji. Wspiera wiele API graficznych (DirectX, OpenGL, Metal i Vulkan)[10] pod wspólnym interfejsem. Dzięki temu twórcy mogą w łatwy sposób tworzyć wersje swoich aplikacji na wiele platform bez uciążliwego dostosowywania swojego kodu. Podobne uproszczenia dostępne są również w zakresie obsługi urządzeń wejścia (jak mysz, klawiatura, kontrolery, ekrany dotykowe), a także dźwięku. Na wspomnienie zasługuje również wbudowany edytor WYSIWYG[1] interfejsu graficznego aplikacji. Silnik można też rozszerzać o własne narzędzia, na przykład graficzny edytor krzywych. Programować swoją aplikację można jedynie w języku C# [11]. Interfejs edytora Unity przedstawiono na rysunku 1.

Język C#

C# to silnie i statycznie typowany obiektowy język programowania zaprojektowany przez firmę Microsoft. Pozwala pisać kod z zachowaniem wszystkich założeń obiektowego paradygmatu programowania, czyli : abstrakcji, hermetyzacji, polimorfizmu i dziedziczenia. Mimo że C# to język obiektowy, posiada wiele funkcji inspirowanych językami funkcyjnymi, które m.in. pozwalają bardzo łatwo dokonywać filtrowania, przekształcania czy sortowania kolekcji obiektów. Ułatwia to pisanie czytelnego, a przy

[1] WYSIWYG (What You See Is What You Get) - „otrzymujesz to co widzisz"

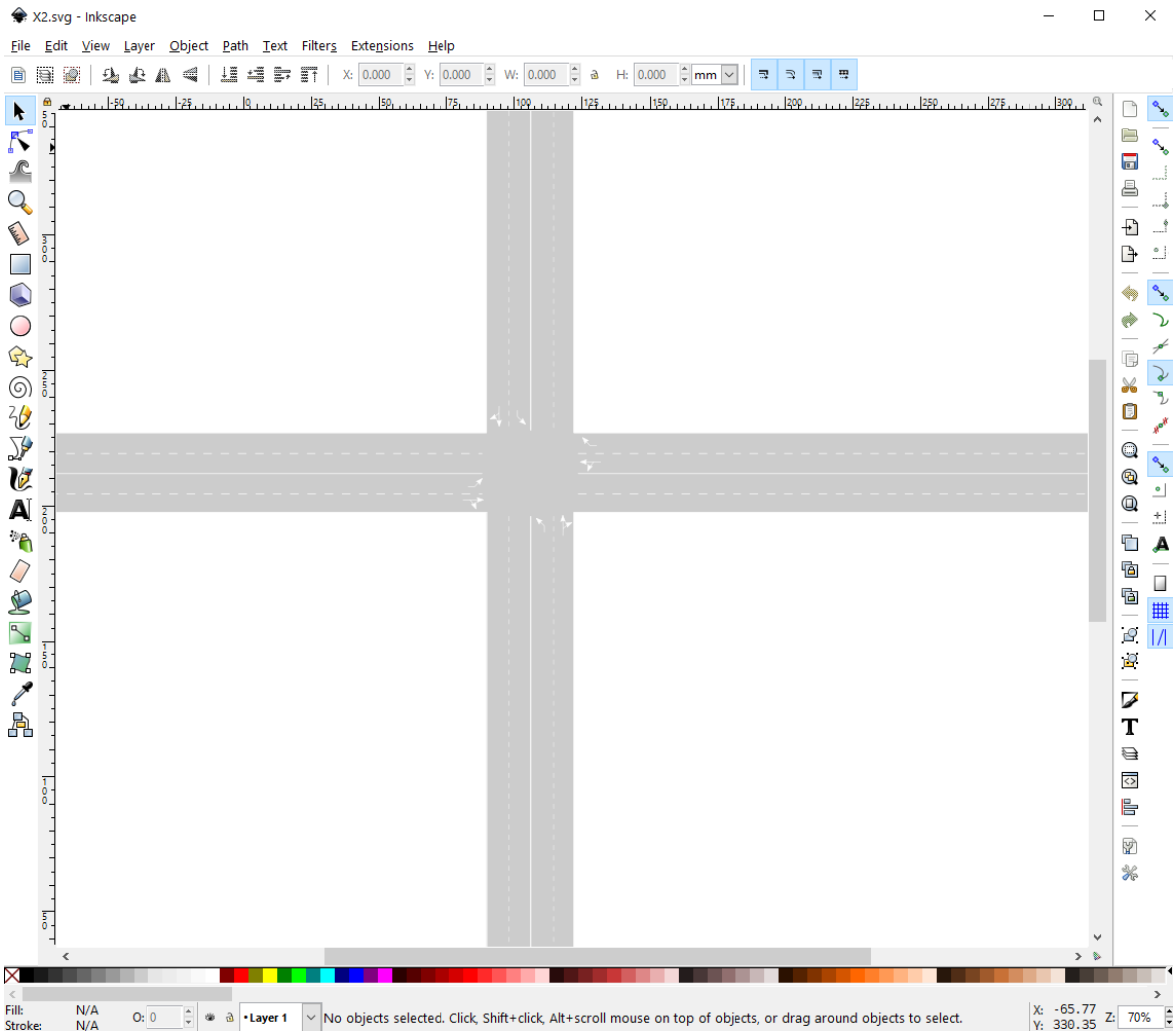
tym zwięzłego, kodu. Pierwotnie kod C# mógł być uruchamiany jedynie na systemie Windows. Zmieniło się to jednak w ostatnich latach dzięki środowiskom takim jak Mono oraz .NET Core i obecnie wspierane są inne platformy, takie jak Linux, macOS, iOS czy Android [12].



Rysunek 1. Interfejs edytora Unity

Inkscape

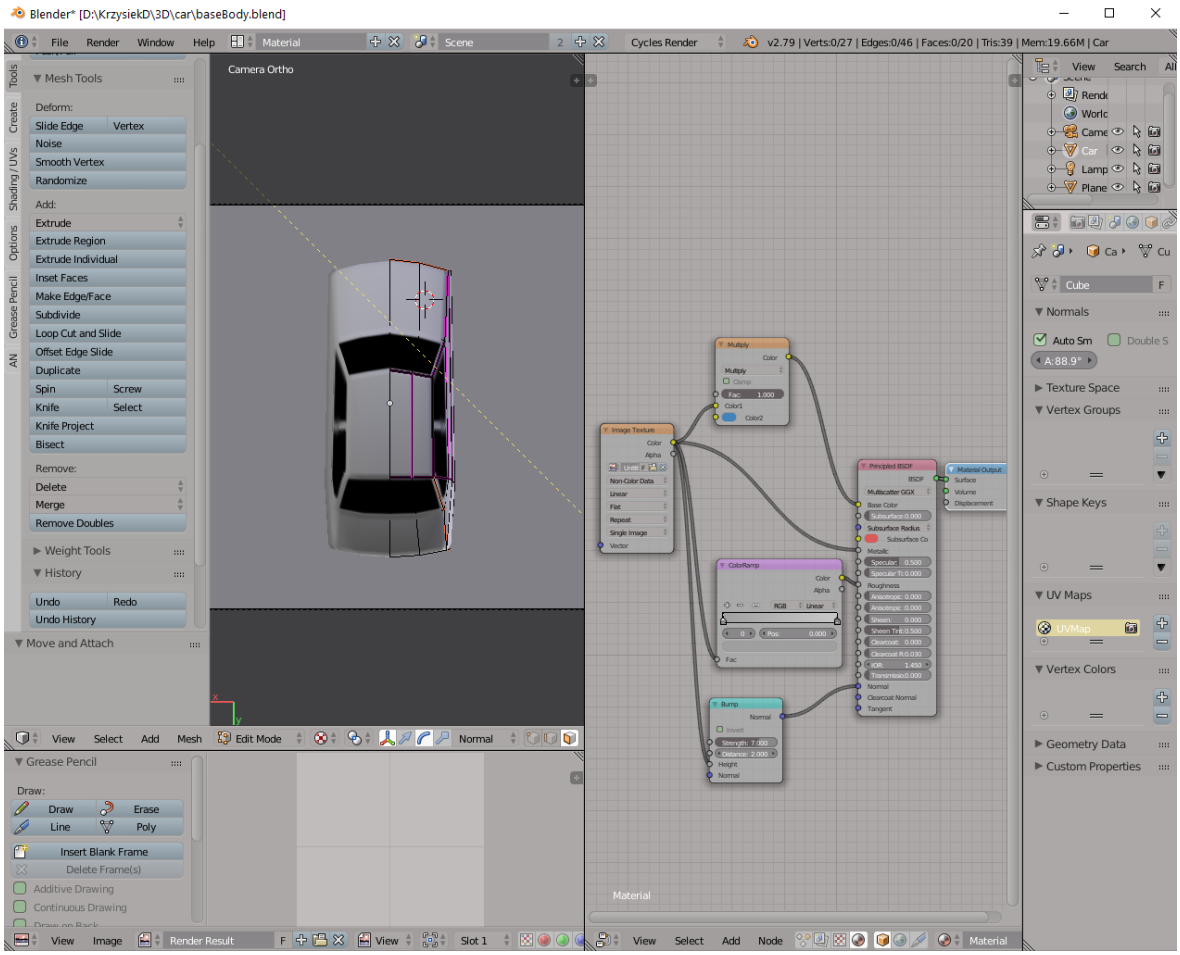
Inkscape to darmowy program do tworzenia grafiki wektorowej. Tworząc w nim obrazy korzysta się z podstawowych kształtów takich jak elipsy, wielokąty, krzywe czy strzałki. Inkscape używa formatu Scalable Vector Graphics (SVG). Mimo że program opiera się na formacie wektorowym, plikiem wynikowym rysunku w tej pracy był obraz rastrowy, który można eksportować w dowolnej rozdzielczości. Decyzja ta wynika z ograniczeń w importowaniu grafik wektorowych w silniku Unity. Za użyciem Inkscape przemawiała przede wszystkim mnogość narzędzi i ustawień do tworzenia kształtów i wyrównywania ich położenia. Wygląd interfejsu programu przedstawiono na rysunku 2. W tej pracy został użyty do stworzenia grafiki skrzyżowania.



Rysunek 2. Interfejs programu Inkscape

Blender

Blender to darmowy program do tworzenia grafiki trójwymiarowej stworzony przez organizację Blender Foundation. Posiada narzędzia do modelowania, teksturowania, renderowania i animacji. Oprócz tego można z jego użyciem wykonywać proste symulacje fizyczne, a nawet zrealizować postprodukcję filmów. Niewątpliwie zaletą programu jest połączenie ogromnej liczby narzędzi w jeden pakiet. Dodatkowe atuty to dobra integracja z silnikiem Unity, wygoda użytkowania i wydajność. W tej pracy posłużył do wykonania trójwymiarowego modelu samochodu oraz ikony aplikacji. Interfejs Blendera przedstawia rysunek 3.



Rysunek 3. Interfejs programu Blender

Funkcjonalność projektu

Projekt jest wysoce wyspecjalizowany – aplikacja przeprowadza wyłącznie proces symulacji i uczenia. Użytkownik może konfigurować ustawienia tego procesu.

Wymagania funkcjonalne

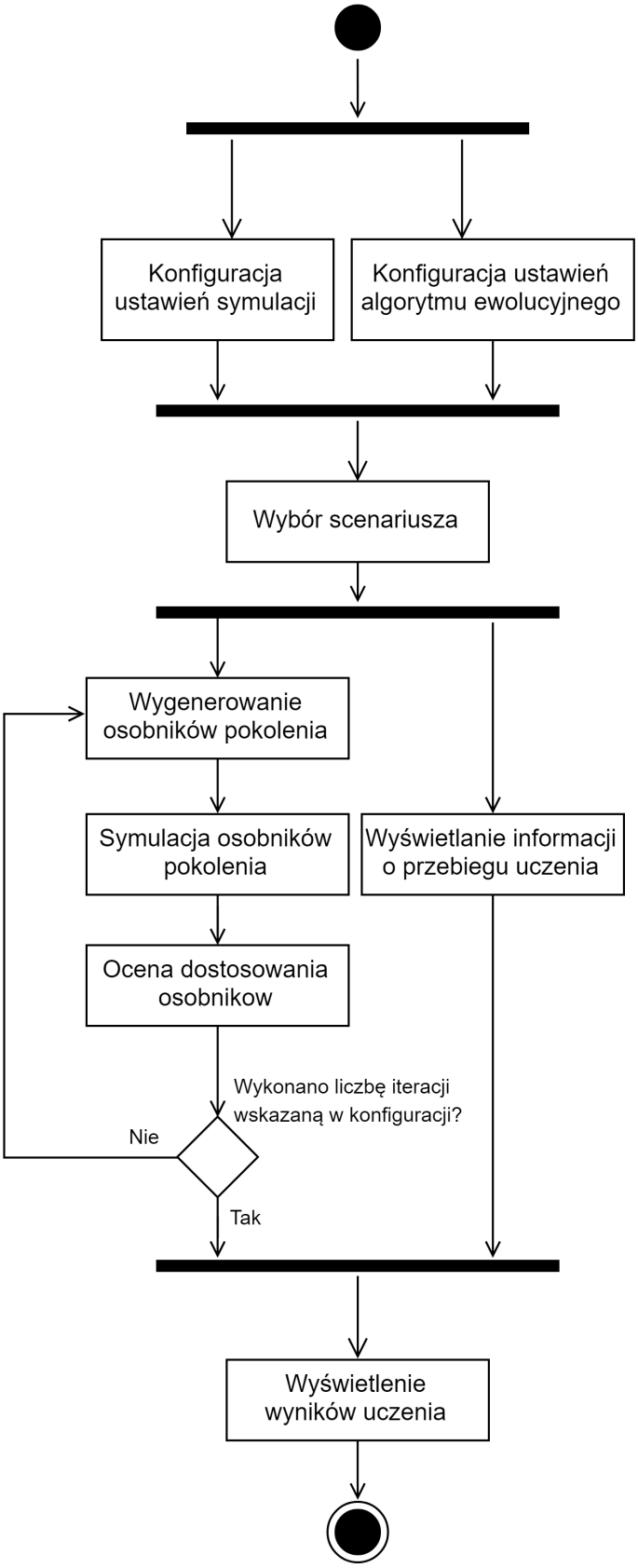
— Przeprowadzenie procesu uczenia optymalnych czasów świecenia sygnalizatorów na skrzyżowaniu za pośrednictwem algorytmu ewolucyjnego,
— Konfiguracja ustawień symulacji, będącej bazą procesu uczenia,
— Konfiguracja ustawień algorytmu ewolucyjnego,
— Wyświetlanie na bieżąco informacji o przebiegu uczenia ,
— Wyświetlenie wyników uczenia po jego zakończeniu.

Wymagania pozafunkcjonalne

— Stabilność – aplikacja musi pracować bezawaryjnie przez wiele godzin,
— Wydajność – aplikacja musi działać na tyle wydajnie, aby symulacja mogła być przeprowadzana kilkadziesiąt razy szybciej niż czas rzeczywisty.

Przypadki użycia

W projekt występuje tylko jeden przypadek użycia: „Przeprowadź proces optymalizacji sygnalizacji świetlnej”. Przedstawia go diagram aktywności na rysunku 4 oraz (hm) 1.



Rysunek 4. Diagram aktywności przypadku użycia „Przeprowadź proces uczenia sygnalizacji”

Tablica 1. Scenariusz przypadku użycia „Przeprowadź proces optymalizacji sygnalizacji świetlnej”

Nazwa	Przeprowadź proces optymalizacji sygnalizacji świetlnej
Aktorzy	Użytkownik
Opis	Funkcja pozwala użytkownikowi wywołać proces optymalizacji sygnalizacji świetlnej na skrzyżowaniu
Warunki początkowe	brak
Główny przepływ zdarzeń	1. Użytkownik wprowadza w menu głównym aplikacji ustawienia. 1.1 Użytkownik wprowadza ustawienia symulacji 1.2 Użytkownik wprowadza ustawienia algorytmu ewolucyjnego
Warunki końcowe	Zakończono proces optymalizacji i wyświetlono wyniki uczenia

Bibliografia

[1] Murray, W., Wright, M. H., Gill, P. E. *Practical Optimization.* Emerald Publishing Limited, 1982.

[2] Badar, A. Q. H., Umre, D. B. S., Junghare, D. A. S. Study of artificial intelligence optimization techniques applied to active power loss minimization.

[3] Cohoon, J., Karro, J., Lienig, J. Advances in evolutionary computing. Springer-Verlag, Berlin, Heidelberg, 2003.

[4] Berger-Tal, O., Nathan, J., Meron, E., Saltz, D. The exploration-exploitation dilemma: A multidisciplinary framework. *PLOS ONE 9*, 4 (2014).

[5] Globus, A., Hornby, G., Linden, D., Lohn, J. Automated antenna design with evolutionary algorithms.

[6] Leon, N., Uresti, E., Arcos, W. Fan-shape optimisation using cfd and genetic algorithms for increasing the efficiency of electric motors. *IJCAT 30* (11 2007).

[7] Wall, M. B. *A Genetic Algorithm for Resource-constrained Scheduling.* PhD thesis, A Genetic Algorithm for Resource-constrained Scheduling, Cambridge, MA, USA, 1996.

[8] Stanisławska, K., Krawiec, K., Kundzewicz, Z. W. Modeling global temperature changes with genetic programming. *Comput. Math. Appl. 64*, 12 (2012).

[9] Steinbuch, R. Successful application of evolutionary algorithms in engineering design. *Journal of Bionic Engineering 7*, 4 (2010).

[10] Unity Manual - Graphics API support. https://docs.unity3d.com/Manual/GraphicsAPIs.html (dostęp 17.01.2019).

[11] Programming in Unity. https://unity3d.com/programming-in-unity (dostęp 19.01.2019).

[12] Albahari, J. *C# 7.0 in a Nutshell.* O'Reilly UK Ltd., 2017.

Spis rysunków