Zaawansowane Programowanie w C++

Wstępne sprawozdanie z Projektu

Temat projektu: Ewolucja pojazdów w 2D

Mateusz Krakowski

Jakub Marcowski

6 czerwca 2023

1 Oryginalny Temat Projektu

Oprogramowanie przeprowadzające ewolucję sztucznych pojazdów w 2D. Chodzi o wyewoluowanie pojazdu, który dotrze jak najdalej w zadanym czasie. Coś podobnego dostępne jest online. Przydatna może być biblioteka do symulacji fizyki, np. Box2D. Przed rozpoczęciem realizacji projektu proszę zapoznać się z zawartością strony.

2 Zarys projektu

Stworzyliśmy aplikację realizującej ewolucje sztucznych pojazdów. W tym celu skorzystaliśmy z algorytmu ewolucyjnego z selekcją turniejową i mutacją gaussowską. Samochód składa się z dwóch kół oraz szkieletu.
W skład genomu pojazdu wchodzi:

- rozmiar kół
- rozstawienie wierzchołków w szkielecie samochodu
- gęstości kół
- gęstość szkieletu samochodu

3 Funkcjonalności

3.1 Obserwacja zachodzącej mutacji samochodzików

Funkcjonalność w pełni zrealizowana, hiperparametry mutacji możemy zmienić w pliku config/EvolutionaryAlgorithmConfig.cc. Jeśli chcemy wymusić przejście do następnej generacji, możemy kliknąć przycisk "N".

3.2 Generator samochodzików z możliwością zmiany parametrów samochodzika (prędkość zadana, maksymalna gęstość, minimalna gęstość)

Samochodziki generowane są na podstawie chromosomu jaki podamy do generatora samochodzików. Maksymalne i minimalne wartości atrybutów chromosomu możemy zmieniać w config/EvolutionaryAlgorithmConfig.cc, natomiast prędkość zadaną samochodzika można zmienić w config/CarConfig.cc.

3.3 Generator mapy z możliwością zmiany parametrów (minimalna i maksymalna zmiana nachylenia terenu)

Mapa jest generowana losowo, hiperparametry generatora można zmieniać w pliku config/EvolutionaryAlgorithmConfig.cc.

3.4 Mechanizm ewolucji samochodzików z możliwością zmiany jego parametrów

Do realizacji tej funkcjonalności wybraliśmy algorytm ewolucyjny z mutacją gaussowską i selekcją turniejową. Hiperparametry algorytmu takie jak: maksymalne i minimalne wartości atrybutów chromosomu, prametry inicjalizacji i mutacji chromosomu, liczbę uczestników turnieju itd. możemy zmieniać w config/EvolutionaryAlgorithmConfig.cc.

3.5 Zapisywanie genomu i wyników samochodzików

Funkcjonalność włączamy lub wyłączamy zmieniając stałą SAVE_TO_FILE w pliku config/Config.cc, możemy również zmienić ścieżkę pliku w którym zapisywane mają być dane chromosomów.

3.6 Dodatkowo: Wykresy

Dodatkowo postanowiliśmy wprowadzić do programu rysowanie wykresu prędkości i obecnej pozycji samochodzików.

4 Instrukcja obsługi

4.1 Instalacja

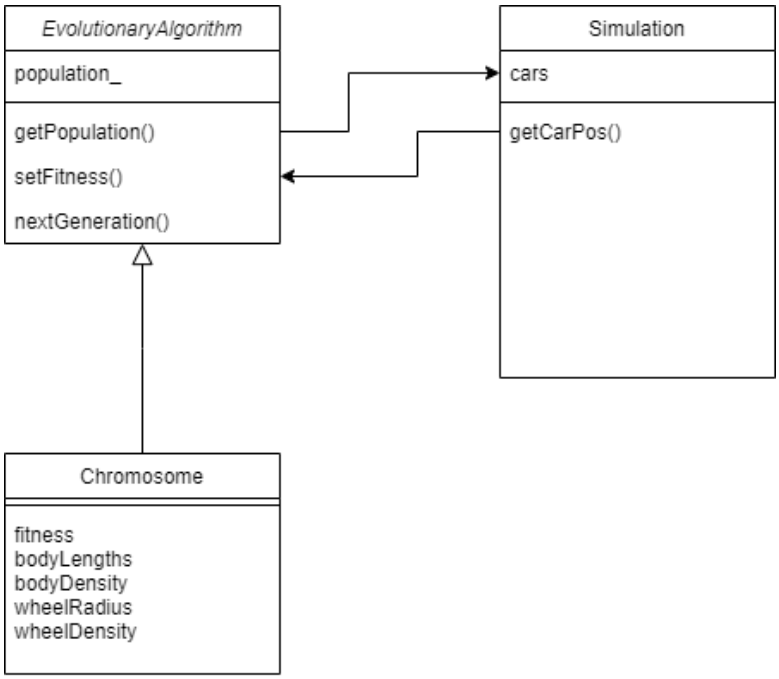
Postępuj zgodnie z tym, co zawarte w pliku README.md.

4.2 Funkcje programu

Następujące klawisze odpowiadają za:

- "Q wyjdź z programu
- "N wymuś przejście do następnej generacji
- "P zatrzymaj symulację symulację, ponowne kliknięcie wznawia symulację

5 Architektura aplikacji



5.1 EvolutionaryAlgorithm

EvolutionaryAlgorithm to klasa odpowiedzialna za ewolucję chromosomów. Do populacji dostajemy się przez funkcję getPopulation(), gdy kończymy iterację symulacji to podajemy fitness każdego z chromosomów przez funkcje setFitness() a następnie wykonujemy nextStep().

5.2 Shape

W pliku Shape.h zdefiniowane są struktury potrzebne do sytmulacji takie jak Box, Circle i Polygon.

6 Zadania

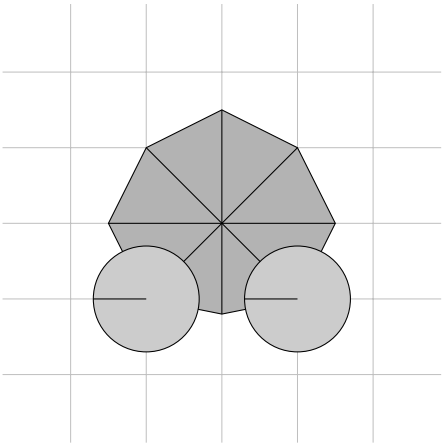
6.1 Wykonane zadania

Po lewej planowany czas, po prawej czas faktycznie spędzony

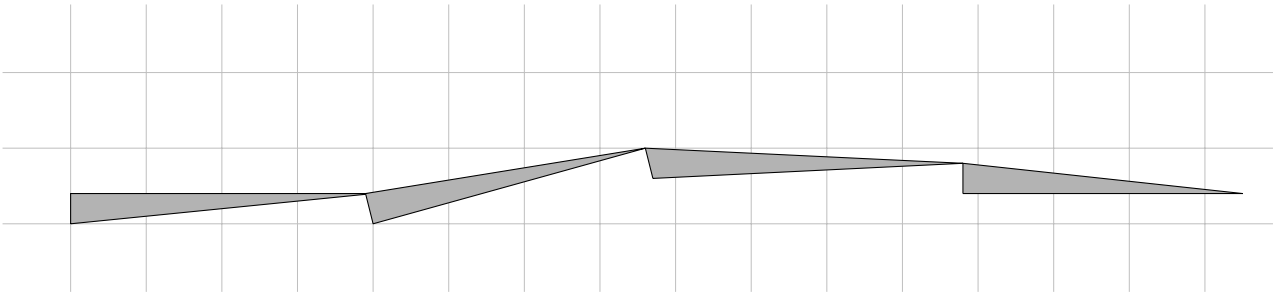
1. planowanie (10h -> 10h):
   - wybór odpowiednich bibliotek 2h -> 2h
   - rozpisanie zadań 6h -> 6h
   - podział odpowiedzialności 2h -> 2h
2. szkielet aplikacji (24h -> 36h):

- zapoznanie się z biblioteką 2DBox 8h -> 12h
- stworzenie dema aplikacji 16h -> 24h

3. implementacja samochodzika (24h -> 36h):
   - klasa samochodzika 12h -> 18h
     - koło 4h -> 6h
     - szkielet 4h -> 6h
   - generator samochodzików na podstawie podanego genomu 4h -> 6h



4. implementacja mapy (20h -> 28h):
   - mapa 8h -> 12h
     - podłoże 4h -> 4h
   - generator mapy 8h -> 12h



5. implementacja algorytmu ewolucyjnego (32h -> 31h):
   - klasa przechowująca populacje 8h -> 12h
   - funkcja inicjalizująca populacje 2h -> 3h
   - funkcja oceniająca samochód 6h -> 8h
   - mechanizm selekcji 8h -> 4h
   - mechanizm mutacji (mutacja gaussowska) 8h -> 4h

6. mechanizm eksportowania danych samochodzików do zewnętrznego pliku 8h -> 10h
7. rysowanie wykresów 0h -> 16h

6.2 Porównanie estymaty z czasem rzeczywiście spędzonym nad projektem

Według naszych pozytywnych estymacji, projekt powinien potrwać 118 godzin, po 59 godzin na członka zespołu. Faktycznie nad projektem spędziliśmy 167 godzin.