**Dokumentacja wstępna**

**Prowadzący projekt:**

Dr. Inż. Rafał Biedrzycki, email: R.Biedrzycki@elka.pw.edu.pl

czwartek 14:30 - 16:00, pok. 24

**Treść projektu:**

Mamy N miast w Polsce (np. 15), pomiędzy którymi trzeba przewieźć towary. Czas przewozu jest ograniczony - towar musi dotrzeć do celu najpóźniej o 6 rano (wyrusza o 22). Kierowcy muszą odpoczywać. Przy dalekich podróżach można się zamienić pojazdami. Przy bliższych podróżach pojazd musi wrócić do miejsca, z którego wyruszył. Każdy pojazd jest postrzegany jako samodzielny agent, który decyduje o trasie, któremu mogą przydarzyć się różne rzeczy, np. korek lub awaria. Oprogramowanie powinno składać się z części serwerowej i klienckiej. Część serwerowa dokonuje symulacji i zapisuje wyniki do bazy. Część kliencka (na przeglądarkę internetową) umożliwia zadawanie warunków symulacji (ilość towarów do przewiezienia, ładowność pojazdu, reguły rządzące zachowanie agenta, prawdopodobieństwa zdarzeń) jak i wizualizację przebiegu symulacji (pojazdy poruszają się po mapie (niekoniecznie rzeczywistej, choć byłoby miło użyć danych rzeczywistych (przez API do ulubionej mapy))). Przed rozpoczęciem realizacji projektu proszę zapoznać się z zawartością strony.

Odpowiedzi na pytania do dr. inż. Biedrzyckiego:

1. Język programowania: C++ (większościowo)
2. Bazy danych: mySQL
3. Aplikacja powinna umożliwiać szeroki zakres regulacji parametrów i składowych symulacji (np. jak zmieni się przebieg transportów jeżeli dodamy/ujmiemy, którąś z dróg, punkt przeładunkowy)
4. Pojazdy mają co do zasady wracać do siebie do bazy, tak aby kierowcy mogli wrócić do domu. W tym celu mogą się oni wymieniać samochodami w połowie trasy.
5. “Serwer” powinien być zrealizowany jako serwer sieciowy umożliwiający korzystanie z aplikacji klienckiej na innej maszynie.
6. Dokumentacja powinna zawierać elementy opisu dla klienta (w większości) jak i dla potencjalnego dewelopera (w mniejszej części)

**Klient – funkcjonalności:**

Aplikacja na przeglądarkę internetową (funkcjonalności okien):

Client\_Menu

Nowy

Załaduj

Opcje

Autorzy

Wyjdź

. . .

?

Client\_New

Miasta

Agent

Param mapy

Param czas

Inne

Warszawa

Kraków

Gdańsk

Łódź

. . .

l. poj.

l. kier.

Symuluj

Zapisz Param.

Załaduj Param.

Wróć

Parametryzacja symulacji:

1. Miasta:
   * Lista miast z parametrami (ilość pojazdów i kierowców w danym mieście)
2. Agent:
   * Prawdopodobieństwo awarii
   * Maksymalny czas jazdy bez przerwy
   * Ładowność pojazdu
   * Maksymalna prędkość
3. Parametry mapy:
   * Prawdopodobieństwo korków
   * Prawdopodobieństwo wypadków
4. Parametry czasowe
   * Okno czasowe symulacji
   * Krok symulacji\*
5. Inne

Client\_Options

Grafika

Dźwięk

Ust. Domyśl.

Kat. Domyśl.

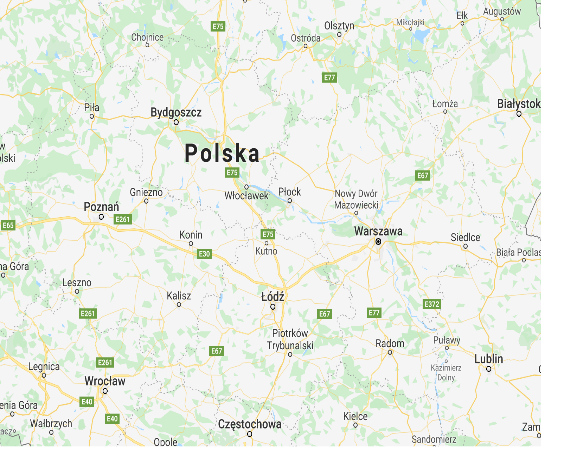
Inne

Katalog domyślny (miejsce zapisania symulacji):

C:\Users\Nitrox\Documents\

Wróć

Client\_Sim



Menu

Zapisz symulacje

Załaduj symulacje

Cofnij

Korek

Blokada

Awaria

Inne



Mapa czasu rzeczywistego:

1. Podejrzenie stanu drogi
   * Ustaw korek
   * Ustaw blokadę
2. Podejrzenie stanu agenta:
   * Zaplanowanie wypadku/awari
   * Stan kierowcy (godziny jazdy, ładowność itd.)

**Klient – technologie:**

1. Google Maps API
2. HTML5, CSS
3. Python - HTTP

**Serwer – funkcjonalności i technologie:**

1. C++11 Boost Beast – obsługa HTTP
2. Agent jako osobny wątek/proces:
   * Agenci sami wyznaczają swoje trasy
   * Trasy zostają wysłane do serwera, który, posiadając wiedzę o wszystkich trasach, optymalizuje plan
3. Forma komunikacji agent-serwer zrealizowana przy pomocy formatu JSON
4. Serwer analizuje plik wejściowy, odseparowuje dane dotyczące poszczególnych agentów i uruchamia wątki z interesującymi je danymi
5. W trakcie symulacji serwer może otrzymać prośbę o aktualizację symulacji dotyczącą danego agenta

**Agent – funkcjinalności:**

1. Agent jako osobny wątek
2. Agent otrzymuje od serwera parametry/wymagania związane z nim samym
3. Po zaplanowaniu trasy agent odsyła ją (w formie pewnego zbioru próbek)
4. Serwer może ponownie wywołać agenta do zaplanowania trasy (albo jej części) jeżeli na etapie optymalizacji stwierdzi, że jest to wymagane
5. Trasowanie zrealizowane przez optymalizację pewnego grafu (skończonego)