Symulacja ruchu samochodów na drugiej obwodnicy Krakowa.

Marcin Kroczek, Michał Wąsik, Paweł Świder

Projekt:

Model Nagela-Schreckenberga:

Będziemy używać modelu Nagela w celu symulacji ruchu aut. Dokonamy paru poprawek:

1. zmiana wielkości kratki do odpowiadającej warunkom miejskim (ok 5m)
2. różne modele samochodów
3. dodanie skrzyżowań
4. dodanie wjazdów i zjazdów

Mapa:

W celu wizualizacji symulacji został pobrany fragment mapy Krakowa zawierający drugą obwodnicę oraz tereny przyległe. Fragmenty mapy wygenerowany za pomocą programu Mapertive z OpenStreetMap. Wygenerowano tilesy[1] w wielkościach od 13 do 18.

Powstałe w ten sposób fragmenty zostały złożone i umożliwiają przesuwanie oraz zmianę powiększenia mapy bez utraty jakości. Mapa zawiera współrzędne w granicach: (0,0) – (16384, 16384). Jedna spółrzędna odpowiada jednemu pikselowi na mapie przy największym powiększeniu mapy.

Z tego zmierzono długość mostu dębnickiego w rzeczywistości oraz w aplikacji i uzyskano następującą równość:

$$1p \approx 0.39m$$

Korzystając z tej informacji możemy dostosowywać długość jednej komórki na drodze. Docelowo przyjęliśmy że jedna komórka zajmuje ok. 5m – jest jednak wiele komórek których długość różni się od podanej wyżej są to m.in. komórki na skrzyżowaniach oraz komórki łączące fragmenty dróg.

Punkt:

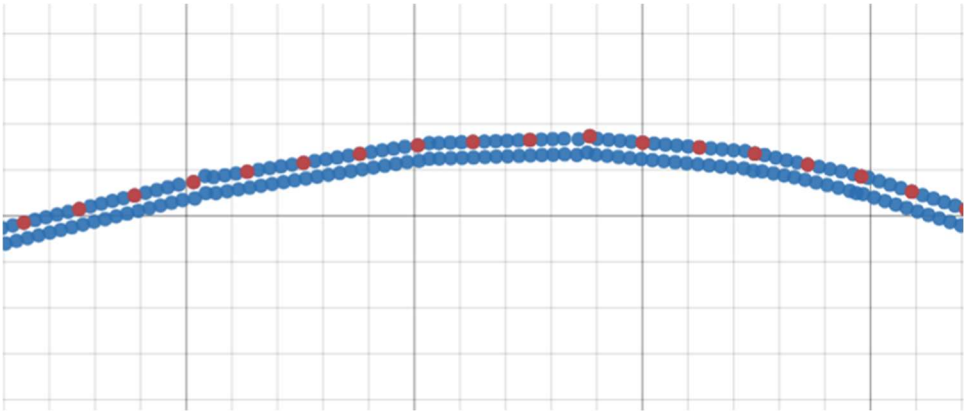
Podstawowym obiektem każdej CA jest pojedynczy punkt. W naszym przypadku jest on reprezentowany przez klasę Point. Sąsiedztwo z jakiego zdecydowaliśmy się skorzystać, to sąsiedztwo von Neumanna. Point może przyjmować 4 główne typy: zwykły (standardowy punkt należący do drogi), wjazd do symulacji, wyjazd z symulacji oraz skrzyżowanie. Rozróżnienie na te typy pozwala nam wprowadzić odpowiednią logikę biznesową w kodzie.

Drogi:

Drogi znajdujące się w symulacji muszą odpowiadać faktycznym drogom na obwodnicy, co oznacza że muszą mieć odpowiednią długość, a także wzajemne położenie. Do projektowania dróg posłużyliśmy się całym pakietem narzędzi geometrycznych, które wcześniej napisaliśmy (dzielenie obszaru na

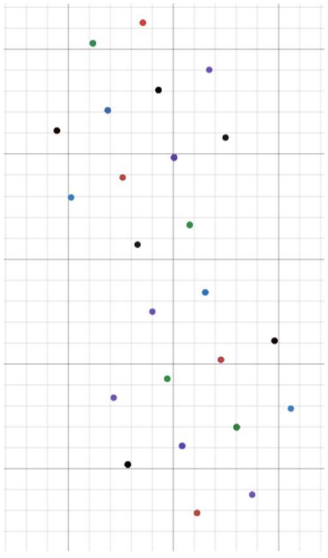
[1] Tiles - OpenStreetMap Wiki

punkty, szukanie prostej prostopadłej itd.). Każda droga ma z góry określoną liczbę pasów oraz przyporządkowaną przestrzeń, w której ma zostać wybudowana. Przestrzeń jest dzielona na pasy, a pasy na pojedyncze komórki. W naszej symulacji przyjęliśmy, że jedna komórka ma długość 5m. Drogi można łączyć zarówno ze sobą, jak i ze skrzyżowaniami.



Skrzyżowania:

W celu stworzenia skrzyżowania, wczytujemy wcześniej przygotowany plik “json”, który zawiera odpowiednie dane takie jak np. odpowiednie wjazdy i wyjazdy, współrzędne odpowiednich 4 celek itp. Każdy z pasów, z których auta wjeżdżają na skrzyżowanie ma zadeklarowane id dróg, na której auto może skończyć po przejechaniu skrzyżowania. Otrzymując dany plik kod generuje obiekt przechowujący odpowiednie komórki, która są ze sobą odpowiednio połączony, oraz obiekt umożliwia pojazdom wjazd na odpowiednią komórkę z danej drogi. Dodatkowo skrzyżowanie zapewnia prosty system świateł drogowych, który umożliwia przejazd jedynie samochodom z jednej, aktualnie posiadającej zielone światło drogi.



Przykładowo wygenerowane skrzyżowanie (Rysunek “desmos”)

Miasto:

Do przechowywania dróg oraz skrzyżowań służy nam miasto. Miasto udostępnia metody do pobrania wjazdów i wyjazdów z danej drogi, można również poprosić je o zwrócenie obiektu konkretnej drogi lub skrzyżownaia.

Samochody:

Po drogach naszej symulacji mogą przemieszczać się dwa typy pojazdów: auta osobowe oraz autobusy. Te pierwsze zajmują jedną kratkę (co odpowiada dł. 5m), autobusy natomiast zajmują 2 kratki.

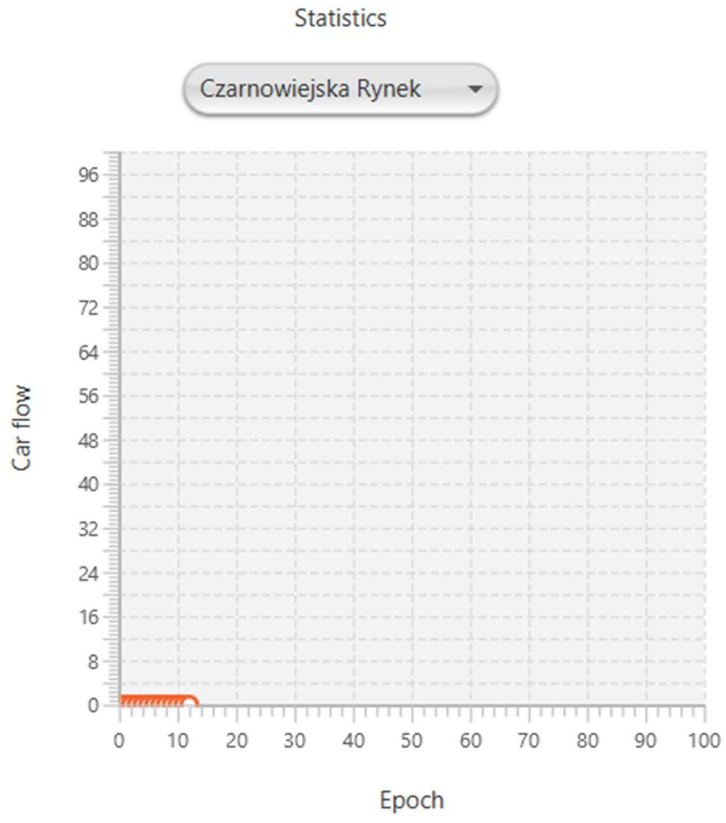
Program posiada funkcję umożliwiające zmianę parametrów dotyczących aut dla każdego wejścia w trakcie działania, nie zostały jednak one w 100% zsynchronizowane z silnikiem.

Statystyki:

Symulacja będzie umożliwiać zbieranie statystyk dotyczących

1. ile aut wjechało i ile wyjechało na obwodnicę (z wyszczególnieniem na każdy zjazd)
2. przepustowość skrzyżowań
3. średnia prędkość samochodów

Aplikacja wyświetla wykresy pokazujące ile samochodów w danej epoce wyjechało z danym zjeździe z symulacji.



Narzędzia:

Do modelowania będziemy używać języka Java. Wizualizację symulacji i statystyk przedstawimy za pomocą biblioteki JavaFX.

Dane:

Stworzyliśmy wiele plików z danymi gdzie przechowywaliśmy dane dotyczące symulacji. Przechowywaliśmy m.in.:

1) fragmenty mapy

2) dane gdzie znajduję się dane skrzyżowanie i jakie wjazdy/wyjazdy zawiera
3) dane dotyczące fragmentów dróg
4) szczegółowe dane dotyczące skrzyżowań

Walidacja modelu:

W celu zbadania jak dobrze model odwzorowuje rzeczywistość postanowiliśmy sprawdzić czas przejazdu pojedynczego auta przez zamodelowany fragment obwodnicy tj. od mosty Dębnickiego przez całą aleję 3 Wieszczy. Trasa ma 3.4km i zawiera zarówno odcinki proste jak i zakręty. Podczas przejazdu auto nie zatrzymywało się na światłach.

Dla długości standardowej komórki równej 13 oraz maksymalnej prędkości auta równej 4 przejechanie tego odcinku zajęło 170 iteracji. Jedna iteracja odpowiada jednej sekundzie ruchu.

Daje nam to średnią prędkość symulowane auto równą: 20m/s czyli 72km/h co jest prędkością nie odbiegającą od realiów krakowskich.

Symulacja pozwala na zmianę szybkości auta więc ta wartość może być lepiej ustalona.

Następne kroki:

Stworzony projekt posiada bardzo wiele możliwości potencjalnego rozwoju. Są to m.in.:

1) Całkowite zamodelowanie reszty obwodnicy
2) Dopracowanie skrzyżowań
3) Potencjalną zmianą modelu – wprowadzenie gęstszych komórek oraz aut posiadających wiele komórek. Pozwoli to na lepsze sterowanie symulacją.
4) Dodanie do obwodnicy kolejnych dróg i finalnie stworzenie symulacji całego Krakowa.
5) Dodanie możliwości wyprzedzania dla samochodów.

Wnioski:

Projekt okazał się dużo większy niż początkowo zakładaliśmy, na wielu etapach projektowania np:

- modelowaniu skrzyżowań – okazało się jedną z trudniejszych rzeczy której nie udało się zamodelować zgodnie z pierwotnymi założeniami i harmonogramem.
- stworzenie mapy – pobranie odpowiednich tekstur potrzebnych do generowania mapy okazał się bardziej czasochłonne niż sądziliśmy a pomysł żeby wyświetlać plik svg okazał się niewystarczający gdyż żaden dostępny program nie generował pliku który posiadałby satysfakcjonujące nasz powiększenie. Z tego powodu musieliśmy sami wygenerować odpowiednie pliki, przefiltrować (gdyż część z nich wygenerowała się nadmiarowo) i stworzyć wyświetlanie wskazanego fragmentu mapy.
- modelowanie dróg – największą trudnością którą udało się pokonać było zamodelowanie dróg tak żeby współrzędne ich komórek pokrywały się z tymi wyświetlanymi na mapie i żeby auta nie wyświetlały się np. na domach czy drzewach.

Literatura:

1. Multilane Vehicular Traffic Simulation Using the CA Theory, Ilya Furmanov, Natalia Churbanova, and Marina Trapeznikova

2. Multi-agent System Model for Urban Traffic Simulation and Optimizing Based on Random Walk, Yu Cheng, Tao Zhang, and Jianfei Wang