
**VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ**
**BRNO UNIVERSITY OF TECHNOLOGY**

**FAKULTA INFORMAČNÍCH TECHNOLOGIÍ**
**FACULTY OF INFORMATION TECHNOLOGY**

**ÚSTAV INTELIGENTNÍCH SYSTÉMŮ**
**DEPARTMENT OF INTELLIGENT SYSTEMS**


# WEBOVÝ SIMULÁTOR MĚSTSKÉ DOPRAVY

**CITY TRAFFIC SIMULATOR FOR WEB**

**BAKALÁŘSKÁ PRÁCE**
**BACHELOR'S THESIS**


**AUTOR PRÁCE** — **KRISTIÁN KOVÁČ**
**AUTHOR**

**VEDOUCÍ PRÁCE** — **Ing. JAROSLAV ROZMAN, Ph.D.**
**SUPERVISOR**



**BRNO 2023**

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

# Zadání bakalářské práce

147929

Ústav: Ústav inteligentních systémů (UITS)
Student: **Kováč Kristián**
Program: Informační technologie
Specializace: Informační technologie

Název: **Webový simulátor městské dopravy**
Kategorie: Umělá inteligence
Akademický rok: 2022/23

Zadání:

1. Nastudujte makro a mikro simulaci dopravy. Nastudujte různé způsoby reprezentace silniční sítě a zaměřte se na reprezentaci pomocí křivek.
2. Navrhněte simulátor, který si z OpenStreet Mapy načte informace o silniční síti, tvaru křižovatek a rozmístění semaforů. Podle nastudovaných způsobů mikrosimulace modelujte pohyb všech vozidel. Na začátku simulace polohu vozidel a stav semaforů vygenerujte náhodně. Vozidla během cesty musí dodržovat pravidla silničního provozu. Umožněte běžné chování vozidel, tj. předjíždění pomaleji jedoucích vozidel a přejíždění z pruhu do pruhu.
3. Výstupem simulátoru bude průběžná poloha všech vozidel a stav všech semaforů. Tento výstup se bude generovat v zadaném časovém intervalu a bude se vykreslovat do OpenStreet mapy.
4. Vytvořený program implementujte a otestujte plynulost dopravy na části Brno - Střed. Výsledky zhodnoťte a navrhněte případná vylepšení.

Literatura:


- DOSTÁL, Martin. Simulátor městské dopravy. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.
- NĚMEC, František. Simulátor MHD linek v Brně. Brno, 2013. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.


Při obhajobě semestrální části projektu je požadováno:

- První dva body zadání

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Rozman Jaroslav, Ing., Ph.D.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 17.5.2023
Datum schválení: 3.11.2022

# Abstrakt

Cieľom tejto práce je vytvorenie simulátora mestskej dopravy, ktorý vizualizuje výsledné dáta na webovom rozhraní. Informácie o dopravnej sieti sú získané z projektu OpenStreetMap. Simulácia je založená na mikroskopickom modeli a zameriava sa predovšetkým na oblasť Brno - Střed. Vozidlá sú modelované ako samostatné entity, ktoré medzi sebou komunikujú a dodržiavajú pravidlá cestnej premávky. V práci sú popísané základné princípy modelovania a simulácie dopravy. Zaoberá sa aj niektorými existujúcimi mikroskopickými dopravnými simulátormi a ich vlastnosťami. V druhej časti práce je popísaný návrh a implementácia vytvorenej aplikácie. Tá sa zameriava predovšetkým na popis spracovávania OpenStreetMap dát, modelu vozidiel, štruktúry výsledkov simulácie a ich vizualizácie na webovom rozhraní.



# Abstract

The goal of this thesis is to create a city traffic simulator, which visualizes its results on a web interface. Information about the traffic network is obtained from the OpenStreetMap project. The simulation is based on a microscopic model and focuses mainly on the Brno - Střed area. Vehicles are modeled as separate entities that communicate with each other and follow the traffic rules. The thesis describes the basic principles of traffic modeling and simulation. It also deals with some existing microscopic traffic simulators and their properties. The second part of the thesis describes the design and implementation of the created application. It focuses mainly on the description of OpenStreetMap data processing, vehicle model, structure of simulation results and their visualization on a web interface.



# Kľúčové slová

simulácia, simulátor dopravy, mikroskopická simulácia, diskrétna simulácia, web, OpenStreetMap, Brno

# Keywords

simulation, traffic simulator, microscopic simulation, discrete simulation, web, OpenStreetMap, Brno


# Citácia

KOVÁČ, Kristián. *Webový simulátor městské dopravy.* Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jaroslav Rozman, Ph.D.

# Webový simulátor městské dopravy

## Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jaroslava Rozmana Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Kristián Kováč
17. mája 2023


## Poďakovanie

Rád by som poďakoval Ing. Jaroslavovi Rozmanovi Ph.D. za cenné rady, pripomienky a odborné vedenie tejto práce.

# Obsah

# Kapitola 1

# Úvod

V čoraz hustejšie osídlených mestách sa mestská doprava stala významnou problematikou. Mnoho ľudí na miesto využívania verejnej dopravy preferuje komfortnejšiu jazdu vlastným automobilom. Toto vedie k zvýšenému počtu vozidiel na cestách a tým aj ku zvýšenému dopravnému zaťaženiu miest. Výsledkom je zhoršenie celkovej dopravnej situácie, častejšie dopravné nehody a zvýšenie generovaných emisií v obytných oblastiach. V niektorých prípadoch sa vyťaženosť dopravnej siete dostáva na úroveň, kedy je potrebné niektoré cesty rozšíriť, alebo vytvoriť nové. Výstavba nových ciest je ale veľmi nákladná a nie vždy je možná. Preto je potrebné využiť existujúce cesty čo najefektívnejším spôsobom. Efektívne riadenie dopravy v mestách je preto kľúčové pre zlepšenie celkovej dopravnej situácie a kvality života obyvateľov.

Zložitosť mestských dopravných systémov a množstvo faktorov, ktoré na ne vplývajú spôsobuje, že je náročné vyhodnotiť účinnosť rôznych stratégií riadenia dopravy. Vykonávanie experimentov na skutočných cestách je časovo aj finančne náročné a nie vždy je možné. Z tohto dôvodu boli vytvorené dopravné simulátory, ktoré umožňujú vykonávať rôzne experimenty na virtuálnych modeloch reálnych dopravných sietí. Tie umožňujú analyzovať rôzne dopravné situácie na existujúcich cestách bez toho, aby bolo potrebné narušiť bežný priebeh dopravy. Taktiež sa dajú využiť pri návrhu nových častí dopravnej siete a tým znížiť riziko, že po ich finančne a časovo náročnej výstavbe nebudú mať očakávaný dopad na celkovú dopravnú situáciu.

Cieľom tejto práce je vytvorenie dopravného simulátora s webovým rozhraním. Simulátor umožňuje načítanie štruktúry dopravných sietí z projektu OpenStreetMap a vykonávanie simulácií na týchto sieťach. Následne sa výsledky simulácií prenesú do webového klienta, kde sa zobrazia na mape.

Druhá kapitola práce sa venuje popisu základných pojmov z oblasti modelovania a simulácií. Definuje pojmy ako systém, model systému a simulácia. Taktiež popisuje princíp simulácie diskrétnych udalostí, ktorý je využitý pre implementáciu samotného simulátora.

V tretej kapitole sú popísané rôzne typy dopravných simulátorov a ich využitie. Keďže sa práca zameriava na mikroskopickú simuláciu dopravy, v tejto kapitole sú popísané niektoré mikroskopické modely využívané dopravnými simulátormi. Ďalej sú v nej uvedené a popísané zdroje historických dopravných dát z Brna, ktoré by sa dali využiť pre ďalšiu kalibráciu simulátora.

Štvrtá kapitola sa zaoberá popisom existujúcich mikroskopických dopravných simulátorov. Sú v nej popísané ich výhody, nevýhody a prístupy k simulácii dopravy, ktoré využívajú.

Piata kapitola sa venuje projetu OpenStreetMap, ktorý je využitý pre načítanie dopravných sietí do simulátora. V tejto kapitole je predstavený dátový model, ktorý projekt využíva a popísané nástroje pre získanie dát z OpenStreetMap.

Šiestu kapitolu tvorí popis hlavných nástrojov a knižníc, ktoré boli využité pri implementácii simulátora. Je tu popísané, ako tieto nástroje fungujú a na čo slúžia.

Siedma kapitola popisuje návrh a implementáciu samotného simulátora, ktorý je výsledkom tejto práce. Sú v nej popísané princípy fungovania oboch častí simulátora, backendu aj frontendu. Taktiež sa zaoberá popisom komunikácie medzi týmito dvomi časťami.

# Kapitola 2

# Modelovanie a simulácia systémov

Modelovanie a simulácia systémov je v súčasnosti veľmi rozšírená oblasť, ktorá sa využíva v mnohých odvetviach. Vďaka možnosti vytvárania virtuálnych modelov reálnych systémov je na daných systémoch možné vykonávať rôzne experimenty, ktoré by boli v reálnom svete často nákladné, neetické alebo nemožné. V tejto kapitole sú popísané niektoré základné pojmy z oblasti modelovania a simulácií. Znalosť týchto pojmov je potrebná na pochopenie ďalších princípov, ktoré sú využívané v tejto práci.

## 2.1 Systém

Systém sa definuje ako zoskupenie objektov, ako sú napríklad ľudia, stroje alebo veci, ktoré spolupracujú za účelom dosiahnutia nejakého logického výsledku. To, čo reprezentuje termín systém, je v praxi často závislé od cieľov konkrétneho výskumu [9]. Príkladom jednoduchého systému môžu byť pokladne v supermarkete, pri ktorých sa zákazníci radia do fronty a čakajú na obsluhu. Systémy sa delia na diskrétne a spojité [17].

- **Diskrétne systémy** sú také, ktorých stav sa mení skokovo v určitých časoch simulácie. Príkladom diskrétneho systému môže byť už spomínaná pokladňa v supermarkete, keďže jej stav sa mení iba príchodom zákazníka, alebo jeho odchodom po tom, ako bol obslúžený.
- **Spojité systémy** sú také, ktorých stav sa mení súvisle s časom. Príkladom spojitého systému môže byť napríklad výstrel balistickej rakety, ktorá letí po určitej trajektórii.

Výsledné systémy sú často kombináciou diskrétneho a spojitého systému, avšak zväčša sa dajú klasifikovať podľa prevládajúceho typu [9].

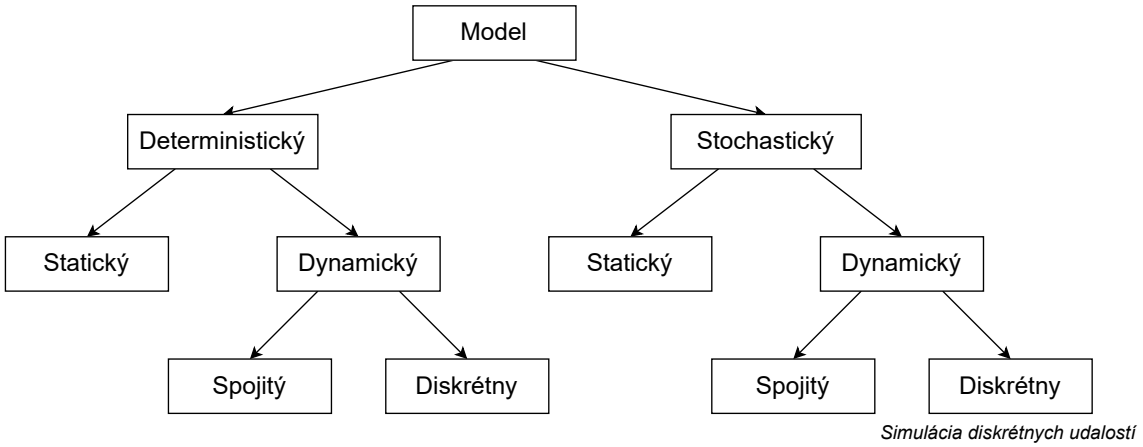
## 2.2 Model systému

Model je napodobenina systému iným, jednoduchším systémom, v našom prípade počítačovým programom. Model systému musí napodobňovať všetky vlastnosti systému, s ktorými pracujeme počas simulácie [17, 10]. V tejto časti je popísané delenie týchto modelov podľa ich vlastností.

- Model systému je **deterministický**, alebo **stochastický**. Deterministický model systému neobsahuje stochastické (náhodné) prvky. Napríklad, model pásového dopravníka, ktorý má konštantnú rýchlosť, zásobuje stroj s konštantným časom obsluhy a

nikdy nezlyhá, je deterministický. Na určitej úrovni detailov však majú všetky systémy nejaké stochastické prvky. Stroje zlyhávajú, ľudia nie sú roboti, požiadavky na obsluhu prichádzajú v náhodných časoch, atď.

- Model systému je **statický**, alebo **dynamický**. Statický model systému je taký, kde čas nie je významnou premennou. Napríklad, ak sa tento týždeň do štátnej lotérie zapojilo jeden milión ľudí, aká je pravdepodobnosť, že bude aspoň jeden víťaz? Model tohto systému by bol zrejme statický, lebo čas umiestnenia stávky každým človekom nie je významnou informáciou. Ak však máme záujem o pravdepodobnosť, že v nasledujúcich štyroch týždňoch nebude žiadny víťaz, potom musí byť tento model dynamický. Predpokladáme, že každý týždeň, kedy v prechádzajúcom týždni nikto nevyhral, počet hráčov narastie, pretože sa zväčšuje výška výhry. V dynamických modeloch vždy hrá čas významnú úlohu.
- Model systému je **diskrétny**, alebo **spojitý**. Diskrétny model je založený na pozorovaní systému v diskrétnych časových okamihoch, pričom sa predpokladá, že v danom časovom intervale sa systém správa konštantne. Spojité modely sú zvyčajne reprezentované jednou alebo viacerými diferenciálnymi rovnicami, ktoré modelujú spojitý časový vývoj systému.




Obr. 2.1: Ilustrácia delenia modelov podľa ich vlastností. Vytvorené podľa [10].

## 2.3 Modelovanie

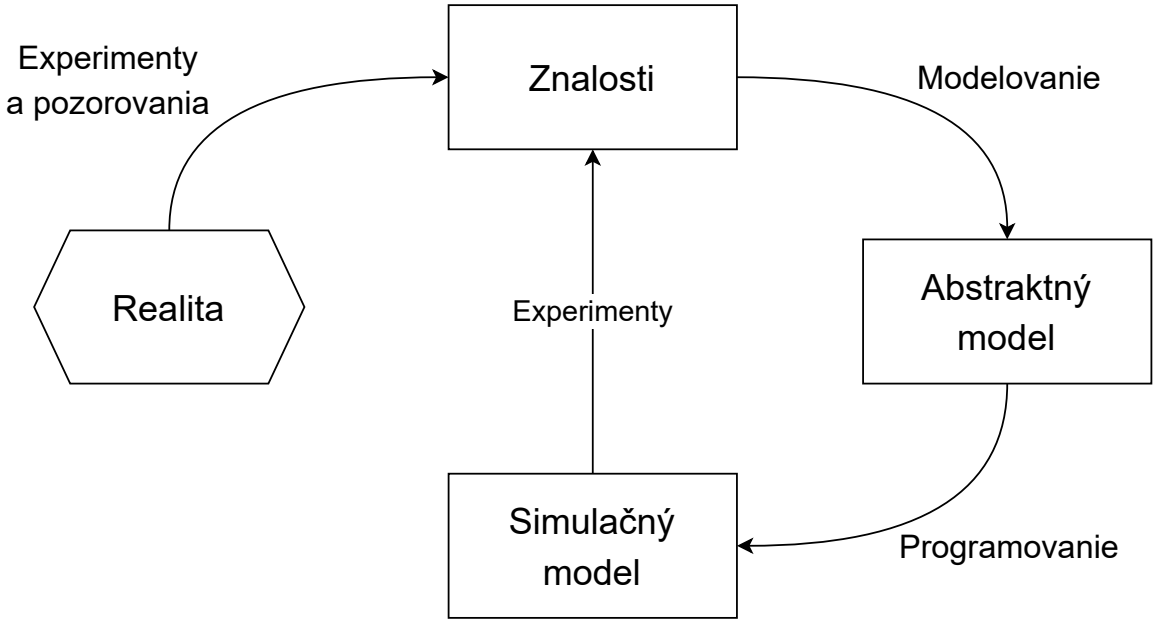
Modelovanie je proces vytvárania modelu systému. Aby bolo možné simulovať nejaký systém, je nutné vytvoriť jeho vhodný model. Vytváranie modelu by sa dalo zhrnúť do nasledujúcich krokov [17].

1. Tvorba **abstraktného modelu** systému. Tento model je založený na pozorovaní systému a pri jeho tvorení sa berú do úvahy len vlastnosti, ktoré sú pre simuláciu dôležité. Tým sa dosiahne zjednodušenie modelu na úroveň, že bude možné ho implementovať a zároveň bude stále reprezentovať všetky relevantné vlastnosti pôvodného systému.
2. Tvorba **simulačného modelu** na základe abstrakného modelu. Jedná sa o model, ktorý je implementovaný v nejakom programovacom jazyku a je možné ho použiť na

simuláciu pôvodného systému. Pri vytváraní sa už model ďalej nezjednodušuje a musí zahŕňať všetky vlastnosti abstrakného modelu.

## 2.4 Simulácia modelov

Simuláciou zvyčajne rozumieme vykonávanie simulačných experimentov na simulčných modeloch a analýzu výsledkov. Na základe týchto výsledkov sa buď overujú predpokladané vlastnosti pôvodného systému, alebo sa získavajú nové znalosti. Získané znalosti sa môžu následne použiť na úpravu a spresnenie abstraktného a simulačného modelu. Takto je možné iteratívne zlepšovať model až kým nebudú všetky vlastnosti pôvodného systému dostatočne presne reprezentované simulačným modelom [17, 6].




Obr. 2.2: Zobranie procesu modelovania a simulácie. Vytvorené podľa [17].

## 2.5 Simulácia diskrétnych udalostí

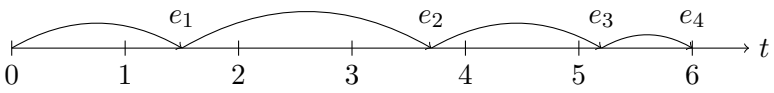
Táto kapitola obsahuje popis hlavných prvkov simulácie diskrétnych udalostí, ktoré sú použité na implementáciu simulátora a ich znalosť je nevyhnutná pre porozumenie jeho fungovania. Diskrétna simulácia udalostí je metóda simulácie, ktorá sa zaoberá modelovaním systému a jeho vývojom v čase pomocou reprezentácie, v ktorej sa stavové premenné menia náhle v oddelených časových bodoch [9]. Tieto časové body sú tie, v ktorých nastáva udalosť, ktorá môže zmeniť stav systému. Aj keď by diskrétna simulácia udalostí mohla byť teoreticky vykonaná manuálne, množstvo dát, ktoré reprezentujú väčšinu reálnych systémov, vyžaduje, aby sa simulácia vykonávala predovšetkým na počítačoch.

### 2.5.1 Mechanizmy posúvania času

Modelový čas je premenná, ktorá reprezentuje aktuálnu hodnotu času v modeli. Vzhľadom na dynamickosť modelov diskrétnej simulácie udalostí, je potrebné počas simulácie sledovať

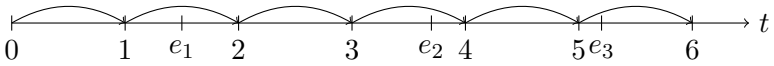
aktuálnu hodnotu modelového času a zvoliť vhodný mechanizmus na jeho posúvanie. Dvomi hlavými prístupmi na posúvanie modelového času sú:

- Posun času na základe nasledujúcej udalosti (*next-event time advance*). Pri tomto prístupe sa čas postupne posúva vždy k najbližšej naplánovanej udalosti. V tomto bode sa stav systému aktualizuje tak, aby sa zohľadnilo, že udalosť nastala a čas sa posunul na novú hodnotu.




Obr. 2.3: Ilustrácia posúvania času na základe nasledujúcej udalosti.

- Posun času o konštantnú hodnotu (*fixed-interval time advance*). Stav systému sa aktualizuje vždy v pravidelných časových intervaloch. Veľkosť intervalu závisí od typu konkrétneho modelu. V niektorých modeloch sa môže jednať o sekundy, v iných kľudne aj o roky. Tento prístup je v podstate špeciánym prípadom predchádzajúceho prístupu v prípade, že by sa udalosti naplánovali na rovnomerne rozložené časové body.




Obr. 2.4: Ilustrácia posúvania času o konštantný interval.

### 2.5.2 Komponenty simulácie diskrétnych udalostí

Aj keď simulácie diskrétnych udalostí môžu byť aplikované v rôznych oblastiach, všetky tieto simulácie zdieľajú niekoľko spoločných komponentov. V tejto časti si popíšeme tieto komponenty a ich úlohu v simulácii.

- **Stav systému** – Stav systému je definovaný množinou premenných, ktoré popisujú systém v určitom čase.
- **Simulačné hodiny** – Premená, ktorá udáva aktuálnu hodnotu simulačného času. Ten sa v prípade simulácie diskrétnych udalostí posúva vždy k najbližšej naplánovanej udalosti.
- **Fronta udalostí** – Fronta obsahujúca všetky udalosti, ktoré sa majú v budúcnosti vykonať.
- **Inicializačná procedúra** – Slúži na inicializáciu simulačného modelu v čase 0.
- **Procedúra časovania** – Podprogram, ktorý vyberie nasledujúcu udalosť z fronty udalostí a posunie simulačné hodiny na čas, kedy sa táto udalosť má vykonať.
- **Procedúry udalostí** – Podprogramy, ktoré aktualizujú stav systému v prípade, že nastane nejaká udalosť. Takýto podprogram je definovaný pre každý typ udalosti.
- **Generátor náhodných čísel** – Slúži na generovanie náhodných čísel z rôznych pravdepodobnostných rozdelení.

- **Štatistické procedúry** – Slúžia na zber rôznych údajov o stave systému počas simulácie, z ktorých sa následne vypočítavajú výsledné štatistiky. Tie sa používajú na vyhodnotenie správania systému.
- **Hlavný program** – Program, ktorý vyvolá procedúru časovania a odovzdá riadenie procedúre príslušnej udalosti. Taktiež je zodpovedný za kontrolu ukončovacích podmienok a zavolanie štatistických procedúr po skončení simulácie.

# Kapitola 3

# Simulácia dopravy

K simulácii dopravy sa vzťahuje proces modelovania a simulácie správania dopravných systémov. Tieto systémy zvyčajne zahŕňajú vozidlá rôznych typov, cesty, križovatky, semafory a podobne. Cieľom simulácie je analýza daného systému zá účelom jeho optimalizácie. Taktiež sa v mestách každoročne zvyšuje počet vozidiel, čo vedie ku komplikáciám, ktoré je výhodné predvídať a predčasne riešiť na základe týchto simulácií. V tejto kapitole sú popísané najpoužívanejšie typy dopravných simulácií a niektoré modely mikroskopických simulácií, ktoré sú využívané v tejto práci.

## 3.1 Typy simulácie dopravy

Existuje niekoľko typov simulácie dopravy, ktoré sa líšia v úrovni abstrakcie a zložitosti simulovaného toku dopravy. V tejto časti sú predstavené tri hlavné typy simulácie dopravy, makroskopická, mikroskopická a mesoskopická.

### 3.1.1 Makroskopická simulácia

V makroskopickej simulácii sa pohyb vozidiel modeluje na vysokej úrovni abstrakcie a primárne sa zameriava na správanie skupín vozidiel [11]. V makroskopických simuláciách sa zvyčajne pracuje s väčším množstvom dát agregovaných do štatistických modelov. Tieto modely zahŕňajú rôzne faktory, ako napríklad priepustnosť jednotlivých ciest, rýchlosť vozidiel, hustota premávky, doby čakania na križovatkách a podobne. Makroskopické modely sú zvyčajne deterministické, predpokladajú že tok dopravy je predvídateľný a môže byť modelovaný matematicky. Cieľom týchto simulácií je analýza priepustnosti dopravných úsekov, alebo aj celých dopravných sietí.

### 3.1.2 Mikroskopická simulácia

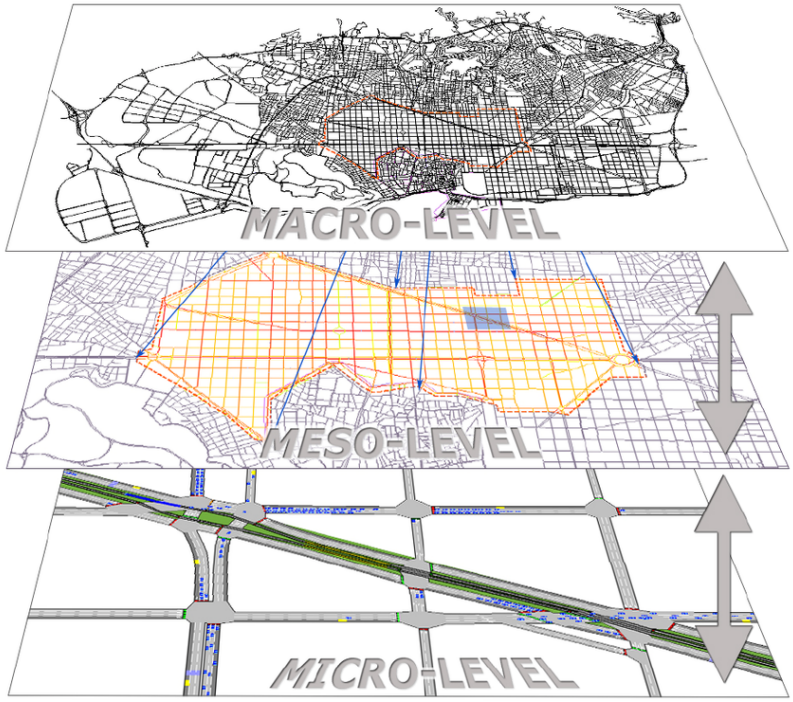
V mikroskopickej simulácii dopravy sa zohľadňujú jednotlivé vozidlá a ich interakcie s prostredím, ktoré zahŕňa ďalšie vozidlá, križovatky, semafory, jazdné pruhy a podobne [2]. Tieto simulácie často rozlišujú aj rôzne typy dopravných prostriedkov, ako sú osobné autá, autobusy, nákladné vozidlá alebo električky. Berú sa do úvahy aj vlastnosti jednotlivých vodičov, ich skúsenosti a povaha. Cieľom je analýza malých zmien v systéme na základe čo najdetailnejšieho napodobnenia reálneho správania vozidiel a vodičov. Mikroskopicé modely môžu byť taktiež použité pre analýzu vplyvu dopravy na životné prosredie, generovaného hluku, alebo objemu generovaných emisií [12].

### 3.1.3 Mesoskopická simulácia

Mesoskopická simulácia dopravy kombinuje prvky makroskopických a mikroskopických modelov a umožňuje simulovať tok dopravy na strednej úrovni abstrakcie. V tomto prípade sa simulujú jednotlivé vozidlá, ale popis ich správania a interakcií je založený na agregovaných (makroskopických) vzťahoch [2]. Mesoskopické modely sú stochastické, podobne ako mikroskopické, ale zároveň zahŕňajú niektoré zjednodušenia z makroskopických modelov.

### 3.1.4 Porovnanie typov simulácie dopravy

Voľba typu simulácie dopravy závisí od konkrétneho problému a od úrovne detailov, ktoré sú potrebné pre jeho analýzu. Makroskopické modely sú vhodné pre simulácie veľkého rozsahu, zatiaľ čo mikroskopické modely poskytujú podrobnejšie informácie o správaní jednotlivých vozidiel. Mesoskopické modely ponúkajú kompromis medzi týmito dvoma typmi, s dôrazom na simuláciu dopravných tokov v zložitých mestských prostrediach.




Obr. 3.1: Znázornenie porovnania jednotlivých typov simulácie dopravy a vzťahov medzi nimi. Prevzaté z [2].

## 3.2 Modely vozidiel v mikroskopických simuláciách dopravy

Kvôli detailnosti modelu vozidiel v mikroskopických simuláciách dopravy sa tieto modely zvyčajne dajú rozdeliť na viac menších modelov. Takto oddelené modely však nie sú samostatne použiteľné a preto je nutné zabezpečiť ich správne prepojenie a komunikáciu medzi nimi. V tejto časti sú predstavené tri základné modely, ktoré sú základom pre mikroskopické simulácie dopravy.

### 3.2.1 Model nasledovania vozidla (Car-following model)

Tento model definuje chovanie vozidiel, keď sa pred nimi nachádza iné vozidlo [7]. Jeho cieľom je regulovať rýchlosť vozidiel s ohľadom na zachovanie bezpečného odstupu medzi nimi. Pokiaľ je vozidlo v jazdnom pruhu ako jediné, jeho rýchlosť sa nastaví na maximálnu povolenú rýchlosť na danom úseku, poprípade na nižšiu rýchlosť, ktorá je danému vodičovi komfortná. V prípade, že sa pred vozidlom nachádza nejaké ďalšie, pomalšie idúce vozidlo, tak sa rýchosť vozidla upravuje tak, aby nedošlo ku kolízii. Takýmto spôsobom sa vozidlá radia do fronty, ktorá pozostáva z jedného vedúceho vozidla a jedného, alebo viac nasledujúcich vozidiel. Pri zrýchlení vedúceho vozidla, nasledujúce vozidlá tiež zrýchľujú až pokiaľ ich rýchlosť nepresiahne komfornú rýchlosť vodiča. V tom momente sa dané vozidlo odpája z fronty a pokračuje v pomalšej jazde. Pokiaľ je vozidlo zaradené do fronty a má možnosť zmeniť jazdný pruh, tým predbehnúť pomalšie vozidlo, tak sa predá riadenie modelu zmeny jazdného pruhu.

### 3.2.2 Model zmeny jazdného pruhu vozidla (Lane-changing model)

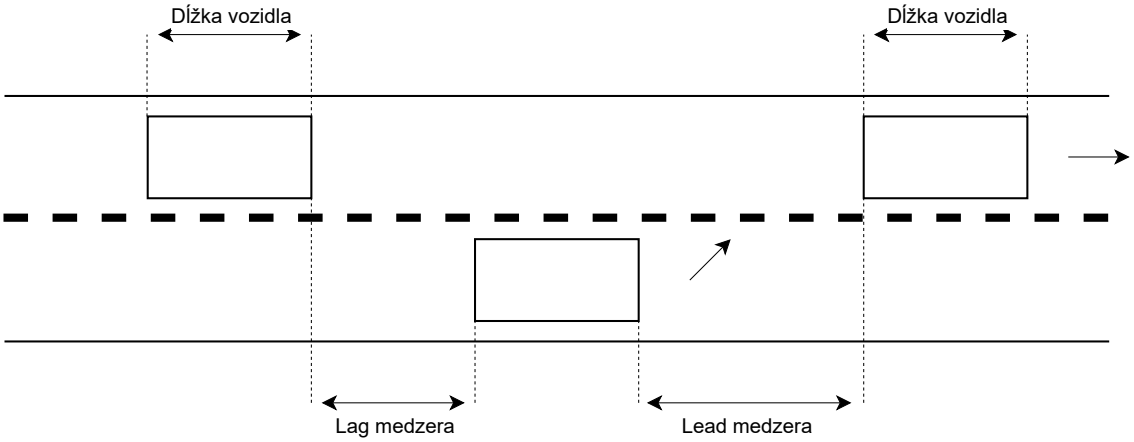
Lane-changing model definuje chovanie vozidiel pri prechádzaní medzi jazdnými pruhmi [6]. Zmena jazdného pruhu môže nastať v dvoch prípadoch. Prvým prípadom je voliteľná zmena jazdného pruhu. Tá nastáva vtedy, keď je vozidlo zaradené do fronty, má možnosť zmeniť jazdný pruh a tým predbehnúť pomalšie idúce vozidlo. Druhým prípadom je nutná zmena jazdného pruhu. Tá nastáva v prípade, kedy vozidlo musí zmeniť jazdný pruh, buď kvôli prekážke na vozovke, koncu jazdného pruhu, alebo kvôli nasledovaniu svojej naplánovanej trasy. Zmena jazdného pruhu by sa dala zhrnúť do nasledujúcich štyroch krokov:

1. Rozhodnutie, či sa má zmeniť jazdný pruh.
2. Výber cieľového jazdného pruhu.
3. Vyhodnotenie dostupnosti priestoru na zmenu jazdného pruhu.
4. Prechod vozidla do cieľového jazdného pruhu, pokiaľ je možné tento prechod vykonať bezpečne.

### 3.2.3 Model odhadu vzdialenosti (Gap-acceptance model)

Tento model sa využíva počas rozhodovania, či je dostatok priestoru na bezpečnú zmenu jazdného pruhu, alebo na prechod križovatkou. Pre správne vyhodnotenie situácie musí vozidlo vhodne komunikovať so svojim okolím a sledovať polohy vozidiel aj v dalších jazdných pruhoch. Model sleduje predovšetkým vzdialenosti medzi jednotlivými vozidlami a ich rýchlosti. Následne na základe týchto informácií vyhodnocuje, či je dostatok priestoru na bezpečný prechod. V prípade prechodu križovatkou, model vyhodnocuje, či sa za križovatkou nenachádza iné vozidlo, ktoré by bránilo prechodu. V tomto prípade musí vozidlo zastaviť a čakať na uvoľnenie priestoru za križovatkou.

Počas zmeny jazdného pruhu model sleduje vzdialenoti od vozidiel, ktoré sa nachádzajú v cieľovom jazdnom pruhu. V rámci modelu sa rozlišujú dva typy vzdialeností. Prvým je vzdialenosť medzi predným nárazníkom aktuálneho vozidla a zadným nárazníkom nasledovaného vozidla v cieľovom jazdnom pruhu. Táto vzdialenosť sa označuje aj ako „lead-gap“. Druhým typom je tzv. „lag-gap“. Ten je definovaný ako vzdialenosť medzi zadným nárazníkom aktuálneho vozidla a predným nárazníkom vozidla, ktorý je za aktuálnym vozidlom v cieľovom jazdnom pruhu [12].




Obr. 3.2: Ilustrácia vzdialeností „lead-gap“ a „lag-gap“ pri modeli odhadu vzdialenosti. Prevzaté z [12].

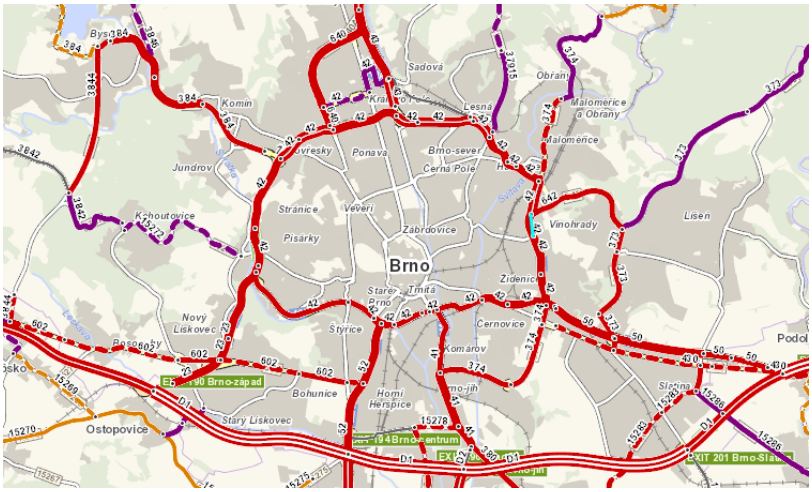
## 3.3 Dopravné dáta v Brne

Pre realistickú simuláciu je vhodné, aby sa simulátor opieral o reálne dáta. Tie sa po vhodnom spracovaní dajú využiť napríklad pre realistickejšie generovanie vozidiel a ich trás. Aj napriek tomu, že tieto dáta nie sú zahrnuté vo vytvorenom simulátore, v tejto časti sú popísané ich zdroje a možnosti využitia pre modelovanie územia mesta Brno.

### 3.3.1 Ředitelství silnic a dálnic České republiky

Ředitelství silnic a dálnic ČR (ŘSD ČR)[1] je štátny podnik, ktorý je zodpovedný za správu a údržbu ciest v Českej republike. V rámci svojej činnosti každých 5 rokov vykonáva sčítanie dopravy na vybraných cestách. Výsledkom sčítania dopravy je množstvo dát, ktoré obsahujú informácie o priemernom počte vozidiel, ktoré prešli daným úsekom cesty za jeden deň. Intenzita premávky je ďalej rozdelená do troch intevalov v priebehu dňa (06:00 - 18:00, 18:00 - 22:00, 22:00 - 06:00). Taktiež je z dát možné získať informácie o aké typy vozidiel sa jedná (osobné, nákladné, autobusy, atď.) a priemerné množstvo emisií vygenerované jedným vozidlom. Všetky dáta sú voľne dostupné vo formáte XLSX na webových stránkach ŘSD ČR [22]. Ich využitie v mikroskopickej mestskej simulácii je však obmedzené tým, že väčšina úsekov zahrnutých do sčítania sú mimo obytných častí miest.

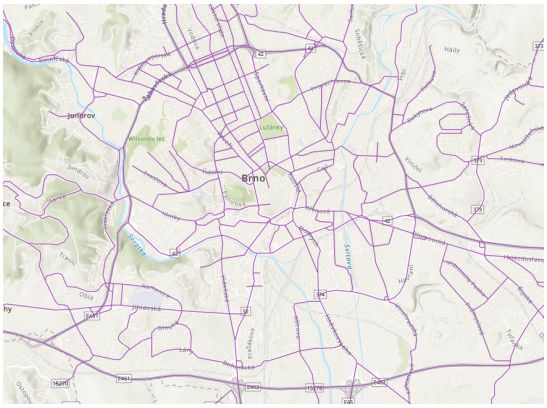
[1] https://www.rsd.cz




Obr. 3.3: Zobrazenie úsekov, ktoré boli v roku 2020 zahrnuté do sčítania dopravy v Brne.

### 3.3.2 Brněnské komunikace a.s.

Brněnské komunikace, a.s. (BKOM)[2] je spoločnosť, ktorá je zodpovedná za správu a údržbu ciest v Brne. Od roku 2016 každoročne vykonáva sčítanie dopravy na vybraných úsekoch v Brne [4]. Oproti celoštátnemu sčítaniu dopravy, ktoré vykonáva ŘSD ČR, sú zahrnuté úseky zväčša v obytných častiach mesta. Výsledné dáta obsahujú informácie o priemernom počte vozidiel, ktoré prešli danými úsekmi za jeden deň v tisícoch vozidiel. Tým pádom sú dáta síce menej detailné, ale zároveň kvôli dobrému pokrytiu mesta sú po vhodnom spracovaní dobre využiteľné pri tvorbe mestských simulácií.




Obr. 3.4: Zobrazenie úsekov, ktoré boli spoločnosťou Brněnské komunikace a.s. zahrnuté do sčítania dopravy v Brne.

| | |
|---|---|
| car_2016 | 9 |
| car_2017 | 9 |
| car_2018 | 9 |
| car_2019 | 9 |
| car_2020 | 10 |
| car_2021 | 11 |
| datum_exportu | May 12, 2023 |
| id | 315 |

Obr. 3.5: Zobrazenie formátu dát, ktoré boli získané zo sčítania dopravy v Brne spoločnosťou Brněnské komunikace a.s.

[2] https://www.bkom.cz/

# Kapitola 4

# Súčasný stav mikroskopických dopravných simulátorov

Mikroskopické dopravné simulátory hrajú dôležitú úlohu pri analýze dopravných systémov. Umožňujú skúmať netriviálne interakcie medzi jednotlivými vozidlami vo vysoko dynamickom dopravnom prostredí. Pomocou nich je možné na veľmi detailnej úrovni analyzovať dopravné toky, tvorbu doprávných zápch a účinnosť rôznych zásahov do dopravného systému. Existuje mnoho simulátorov, ktoré sa líšia v rôznych aspektoch. Niektoré z nich sú voľne dostupné pre širokú verejnosť (SUMO, A/B Street), iné sú určené primárne pre komerčné použitie a spoplatnené (VISSIM, AIMSUN). Taktiež sa tieto simulátory líšia v spôsoboch implementácie, detailnosti modelovania dopravného systému a výkonnosti. Vždy je teda potrebné zvážiť, ktorý simulátor je vhodný pre dané použitie. V tejto kapitole sú popísané dva open-source simulátory z ktorých sú niektoré princípy použité aj v implementácií tejto práce.

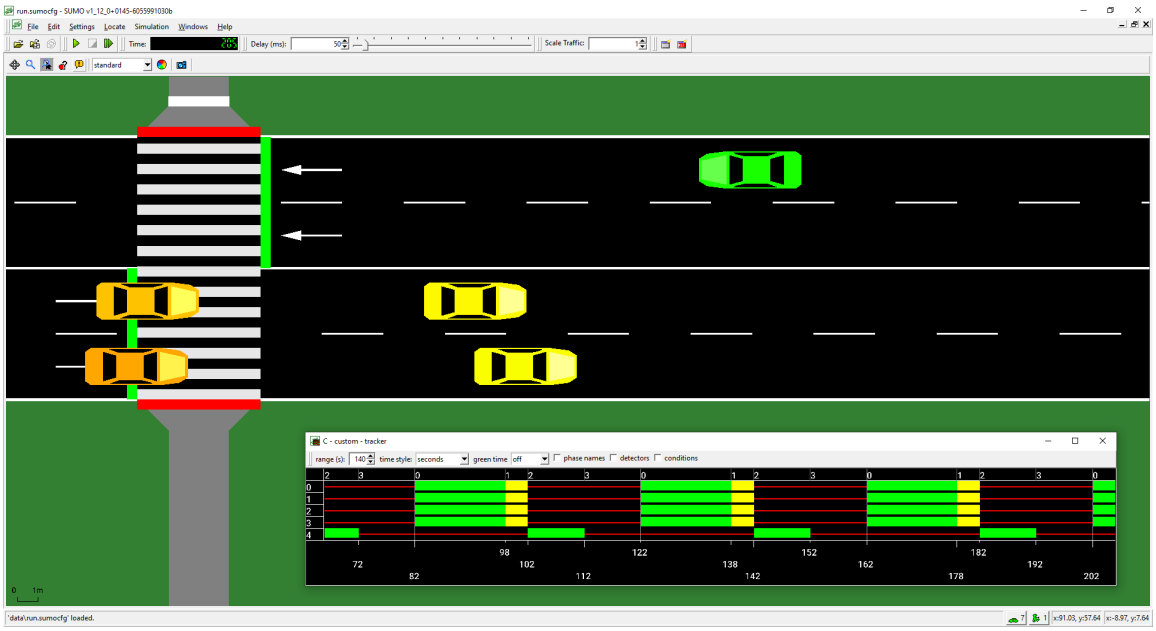
## 4.1 SUMO (Simulation of Urban MObility)

SUMO je open-source balík pre dopravnú simuláciu [3]. Je napísaný v jazyku C++ a poskytuje výkonnú platformu pre simuláciu dopravného toku, generovania emisií a spotrebu energií. Projekt bol založený v roku 2001 nemeckým strediskom pre letectvo a kozmonautiku (DLR). Odvtedy sa SUMO vyvinulo do plnohodnotného nástroja pre dopravnú simuláciu, ktorý je dodnes používaný v rôznych výskumných projektoch po celom svete.

Jedná sa o mikroskopický simulátor spolu s množstvom nástrojov pre spracovanie dát a ich vizualizáciu. Každé vozidlo je definované minimálne jedinečným identifikátorom, časom vytvorenia a trasou, ktorú má prejsť. V prípade potreby môžu byť vozidlá popísané detailnejšie. Vozidlám môže byť priradený typ, ktorý definuje ich fyzické rozmery a hodnoty premenných pre jazdé modely. Taktiež môže byť vozidlám priradený jeden z dostupných emisných tried pre modelovanie vznikajúcich emisií a spotreby paliva. Okrem vozidiel je možné do simulácie zahrnúť aj chodcov, ktorí sa pohybujú po chodníkoch a určitým spôsobom interagujú s vozidlami pri prechádzaní cez cesty.

Dopravná sieť je reprezentovaná pomocou grafu, v ktorom každý vrchol reprezentuje križovatku a každá hrana reprezentuje cestu medzi križovatkami. Takáto dopravná sieť môže byť buď vygenerovaná pomocou aplikácie `netgen`, alebo načítaná zo súboru. Načítanie sietí je možné z formátov iných simulátorov, ako napríklad VISUM, Vissim alebo MATsim. Taktiež je možné načítať mapy z formátov ako OpenStreetMap a Shapefile.

Na pozadí sa využíva diskrétna simulácia s pevným časovým krokom, ktorá je prevolene nastavená na 1 sekundu. Pozícia každého vozidla je reprezentovaná jazdným pruhom, na ktorom sa nachádza a jeho vzdialenosťou od začiatku tohto pruhu. Existujú dve verzie simulátora. Jedna je čiste konzolová aplikácia pre efektívnu simuláciu a druhá je grafická aplikácia, ktorá umožňuje aj vizualizáciu simulácie. Kľúčovou vlastnosťou systému SUMO je jeho schopnosť zvládnuť aj pomerne rozsiahle simulácie. Dokáže simulovať desiatky tisíc vozidiel a niekoľko stoviek svetelných križovaniek v reálnom čase.



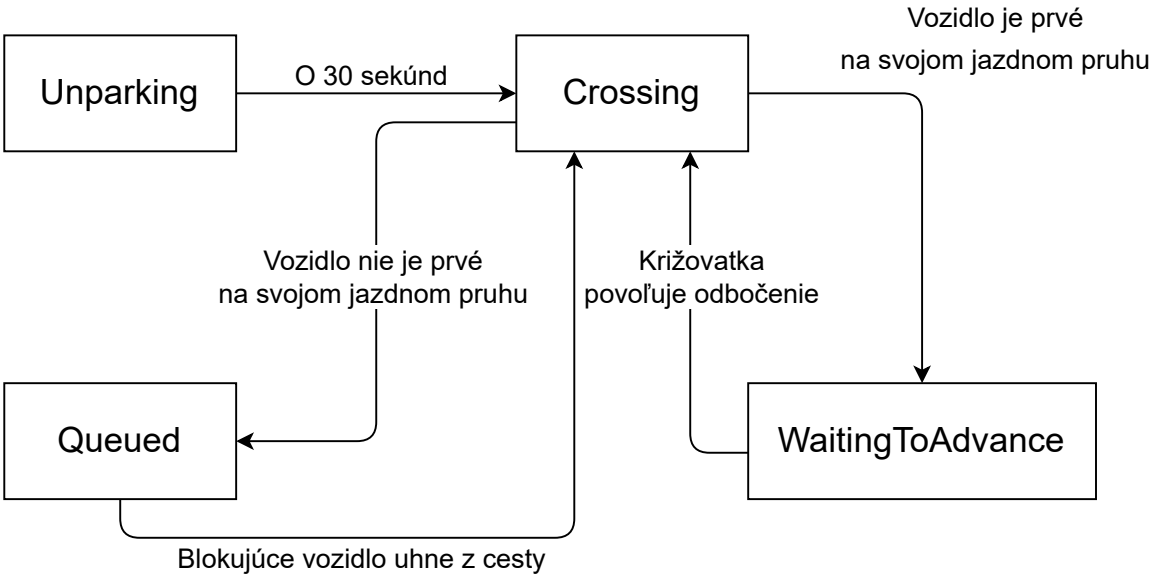
Obr. 4.1: Užívateľské rozhranie simulátora SUMO. Prevzaté z [3].

## 4.2 A/B Street

A/B Street je open-source dopravný simulátor napísaný v jazyku Rust [5]. Jedná sa o menej známy projekt, ktorý ale obsahuje množstvo zaujímavých riešení, ktoré sú dobre zdokumentované. Projekt je primárne zameraný na simuláciu dopravy v mestách a na detailné modelovanie vozidiel, chodcov a cyklistov na mikroskopikej úrovni.

Simulátor pracuje s dopravnými sieťami, ktoré sú uložené v súboroch so špeciálnym formátom. Tieto súbory sú vytvárané z OpenStreetMap dát pomocou nástroja `osm_convert`, ktorý je súčasťou projektu. Tento prístup je zvolený preto, aby sa jednotlivé mapy mohli predspracovať a nebolo tak potrebné ich spracovávať pri každom spustení simulátora.

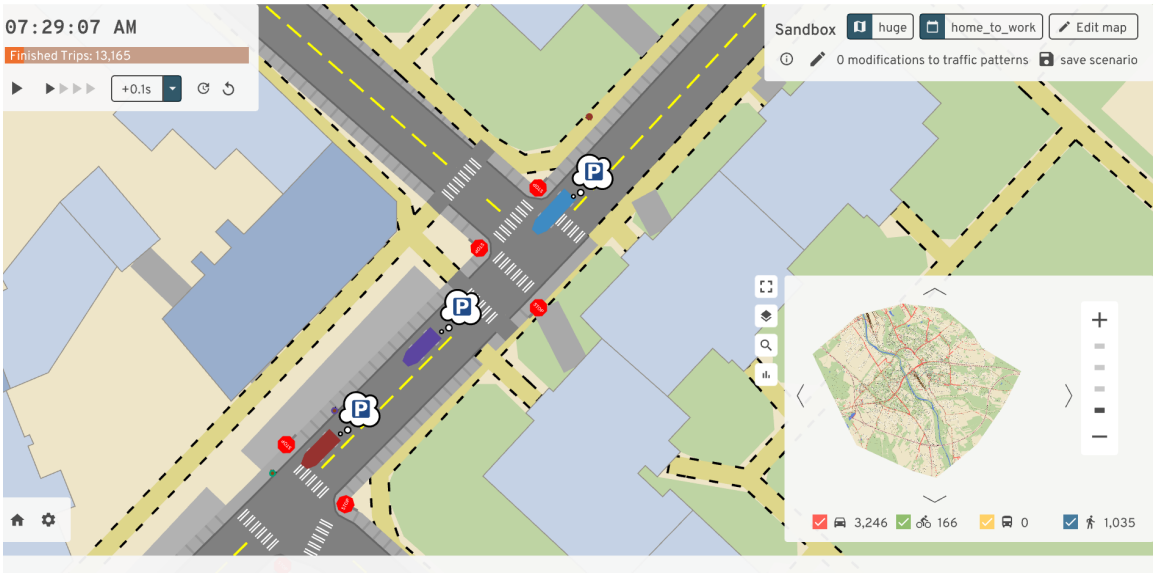
V simulátore A/B Street sa vozidlá môžu nachádzať v troch rôznych stavoch, `Crossing`, `Queued` a `WaitingToAdvance`. V stave `Crossing` je vozidlo vtedy, keď prechádza cestou a nič ho neobmedzuje. Po prejdení celej dĺžky cesty sa jeho stav zmení na `WaitingToAdvance`, čo znamená, že sa nachádza na križovatke a čaká na jej uvoľnenie. Do stavu `Queued` sa vozidlo dostane vtedy, keď vo svojom jazdnom pruhu dobehne pomalšie vozidlo a musí prispôsobiť svoju rýchlosť. Prechody medzi jednotlivými stavmi vozidiel sú znázornené na obrázku 4.2.




Obr. 4.2: Znázornenie stavov vozidiel a prechodov medzi nimi v simulátore A/B Street. Obrázok prevzatý z [5].

V tomto simulátore sa neberie ohľad na zrýchlovanie a spomalovanie vozidiel. Vozidlá sa pohybujú konštantnou rýchlosťou a ich zrýchlenie a zastavenie je okamžité. Tento spôsob modelovania je zvolený na základe predpokladu, že v mestských oblastiach je oproti rýchlostným cestám a diaľniciam, zrýchlenie a spomalenie vozidiel zanedbateľné.

Samotná simulácia je založená na simuácií diskrétnych udalostí. Na miesto prepočítavania pozícií agentov v každom kroku sa využíva systém udalostí, ktoré sa postupne plánujú do prioritnej fronty. Každý agent zostáva určitý čas v nejakom stave a naplánuje si udalosť na prechod do iného stavu v určitom čase. Takýmto spôsobom sa potenciálne zvyšuje efektivita simulácie, pretože sa nevykonávajú nepotrebné výpočty.




Obr. 4.3: Užívateľské rozhranie simulátora A/B Street.

# Kapitola 5

# OpenStreetMap

OpenStreetMap (OSM) je bezplatná a open-source platforma pre tvorbu a zdieľanie geografických dát. Bola spustená v roku 2004 a odvtedy sa stala jednou z najvýznamnejších mapovacích projektov na svete, ktorú používajú milióny ľudí. Zakladateľom projektu je Steve Coast, ktorý sa zo začiatku zameriaval na mapovanie územia Spojeného kráľovstva, kde síce v tej dobe existovalo veľké množstvo zozbieraných dát, ale neboli dostupné pod voľnou licenciou. V roku 2006 bola založená nezisková organizácia OpenStreetMap Foundation s cieľom podporovať rast, vývoj a distribúciu geografických dát každému, kto ich chce používať a zdieľať. V decembri 2006 spoločnosť Yahoo poskytla OpenStreetMap voľný prístup ku svojim satelitným snímkam, ktoré boli použité na vytvorenie podkladových máp. V nasledujúcich rokoch sa projekt postupne rozšíril do celého sveta [14].

Tvorba a úprava dát v OpenStreetMap je inšpirovaná Wikipédiou, čo znamená že je zachovávaná kompletná história úprav a každý používateľ má možnosť prispieť k vylepšeniu týchto dát. Počas rokov sa vytvorilo aj množstvo nástrojov, ktoré umožňujú jednoduchý prístup k týmto dátam a ich úprave. Najznámejšími z nich sú napríklad JOSM, iD alebo Potlatch.

## 5.1 Dátový model OpenStreetMap

V dátovom modeli OpenStreetMap sú prvky reálneho sveta reprezentované pomocou štyroch hlavných typov objektov; značkami, uzlami, cestami, vzťahmi [13].

- **Značka (tag)** – Značka je tvorená dvojicou kľúč-hodnota, ktorá slúži na definovanie vlastností ďalších objektov. Z toho vyplýva, že značka nemôže existovať samostatne, ale musí byť vždy priradená k nejakému uzlu, ceste alebo vzťahu. Aj keď existuje množstvo značiek, ktoré sú štandardizované a ich použitie je dobre zdokumentované, každý používateľ má možnosť vytvárať aj ľubovoľné nové značky. Toto spôsobuje, že v databáze OpenStreetMap je aktuálne viac ako 140 miliónov rôznych značiek[1].

  ```
  <tag k="maxspeed" v="30"/>
  ```

  Obr. 5.1: Príklad značky, ktorá definuje maximálnu povolenú rýchlosť na ceste.

- **Uzol (node)** – Uzol je najzákladnejším prvkom v OSM dátovom modeli. Reprezentuje jediný bod v priestore, ktorý je definovaný svojimi zemepisnými súradnicami a

[1] https://taginfo.openstreetmap.org/reports/database_statistics

jedinečným identifikačným číslom. Účely daného uzla môžu byť definované pomocou ďalších značiek. Napríklad uzol reprezentujúci svetelnú križovatku môže mať priradenú značku `highway: traffic_signals`.

```
<node id="21289029" version="3" timestamp="2016-08-12T08:28:44Z"
uid="123456" user="jannovak" changeset="41405556" lat="49.2005222"
lon="16.6143808"/>
```

Obr. 5.2: Príklad uzla pre znázornenie jeho parametrov.

- **Cesta way** – Týmto objektom sú okrem samotných ciest reprezentované aj všetky ďalšie lineárne prvky, ako napríklad rieky, steny budov alebo chodníky. Cesta je usporiadaným zoznamom uzlov a je väčšinou doplnená o ďalšie značky definujúce jej vlastnosti. Každá cesta má definovaný svoj smer práve podľa poradia jej uzlov. Táto vlastnosť je kľúčovou napríklad pri určovaní počtu jazdných pruhov v oboch smeroch, ktoré sú definované značkami `lanes: forward` a `lanes: backward`. Podľa uzavretosti sa cesty rozdelujú do dvoch skupín, otvorené a uzavreté. Otvorené cesty sú tie, ktoré začínajú v jednom bode a končia v inom. Typicky do tejto skupiny patrí väčšina ciest, koľajníc a riek. Uzavreté cesty sú naopak také, ktorých prvý a posledný uzol je identický. Pomocou týchto ciest sa zvyčajne mapujú objekty, ktoré tvoria nejaký priestor budovy, pozemky, kruhové objazdy, poprípade iné okružné cesty.

```
<way id="4619504" version="5" timestamp="2022-07-17T13:21:58Z"
uid="123456" user="jannovak" changeset="123714801">
    <nd ref="26261749"/>
    <nd ref="3881134222"/>
    <nd ref="26261750"/>
    <tag k="highway" v="residential"/>
    <tag k="lit" v="yes"/>
    <tag k="name" v="Helceletova"/>
    <tag k="surface" v="asphalt"/>
</way>
```

Obr. 5.3: Príklad cesty reprezentujúcej ulicu Helceletova, ktorá je vybavená pouličným osvetlením a má asfaltový povrch. Jej uzly sú definované pomocou elementov `<nd>`, ktoré obsahujú odkazy na uzly s identifikátormi `ref`.

- **Vzťah (relation)** – Objekty typu vzťah sa používajú pri modelovaní logických alebo geografických vzťahov medzi OSM objektmi. Môžu byť použité na reprezentáciu komplexných štruktúr, ako sú napríklad trasy zložené z viacerých segmentov, alebo parky s rôznymi časťami. Vzťah je definovaný svojim jedinečným identifikačným číslom a obsahuje odkazy na jeden alebo viacero iných objektov.

```
<relation id="4282644" visible="true" version="1" changeset="27528560"
timestamp="2014-12-17T12:52:49Z" user="jannovak" uid="123456">
    <member type="way" ref="317661146" role="outer"/>
    <member type="way" ref="317660963" role="inner"/>
    <tag k="landuse" v="farmland"/>
    <tag k="ref" v="9345665"/>
    <tag k="source" v="lpis"/>
    <tag k="type" v="multipolygon"/>
</relation>
```

Obr. 5.4: Príklad vzťahu, ktorý obsahuje dve cesty a reprezentuje poľnohospodársky pozemok.

Objekty typu uzol, cesta a vzťah majú sadu spoločných atribútov, ktoré slúžia predovšetkým na identifikáciu zmien a používateľov, ktorí ich vykonali. Tieto atribúty sú nasledovné:

- **id** – jedinečný identifikátor objektu
- **version** – verzia objektu. Táto hodnota sa inkrementuje pri každej zmene daného objektu.
- **changeset** – identifikátor skupiny zmien, ktoré boli vykonané jedným používateľom v priebehu krátkeho časového obdobia.
- **timestamp** – časová pečiatka, ktorá reprezentuje čas poslednej úpravy objektu.
- **user** – meno používateľa, ktorý vykonal poslednú zmenu objektu. Každý používateľ má možnosť si toto meno kedykoľvek zmeniť.
- **uid** – identifikátor používateľa, ktorý vykonal poslednú zmenu objektu. Táto hodnota je unikátna pre každého používateľa a nemôže byť zmenená.
- **visible** – určuje, či je objekt zmazaný z databázy. Táto informácia sa využíva pre zachovanie histórie zmien daneého objektu aj po jeho zmazaní.

## 5.2 Formáty OpenStreetMap dát

OpenStreetMap poskytuje rôzne formáty súborov pre ukladanie a distribúciu geografických dát. Každý formát je vhodný pre rôzne účely a všetky majú svoje vlastné výhody a nevýhody. Dvomi najdôležitejšími formátmi sú OSM XML a PBF.

- **OSM XML** je pôvodným a zároveň najrozšírenejším formátom súborov OpenStreetMap, ktorý je založený na formáte XML. Hlavnou výhodou tohto formátu je ľahká čitateľnosť a možnosť jednoduchého spracovania pomocou už existujúcich XML parserov. Súbory tohto formátu sú však pomerne veľké a teda nie sú vhodné pre väčšie mapy.
- **PBF** (Protocol buffer Binary Format) je binárny súborový formát, ktorý obsahuje OpenStreetMap dáta v kompaktnejšej podobe. Tento formát sa používa najmä na rýchle načítanie a ukladanie dát. Zároveň je oproti OSM XML menej pamäťovo náročný a umožňuje spracovávanie a ukladanie aj veľkých mapových súborov.

## 5.3 Spôsoby získávania dát

V tejto časti sú popísané najčastejšie používané metódy získavania dát z databázy OpenStreetMap. OpenStreetMap dáta je možné získať niekoľkými spôsobmi, pričom každý z nich má svoje výhody a nevýhody. Zvolený spôsob väčšinou závisí od typu a množstva dát, ktoré používateľ potrebuje.

### 5.3.1 OpenStreetMap API

OpenStreetMap API je oficiálne rozhranie, ktoré umožňuje používateľom získavať a upravovať dáta v databáze OpenStreetMap. Dáta sa získavajú na základe definovaného obdĺžnika, ktorý si používateľ definuje v požiadavke pomocou súradníc jeho se strán. Zo zvoleného obdĺžnika sa stiahnu všetky uzly, cesty a vzťahy, ktoré sa v ňom nachádzajú. Limitáciou tohto rozhrania je, že je v jednej požiadavke možné získať maximálne 50 000 uzlov. Preto sa tento spôsob primárne používa v OpenStreetMap editoroch, kde pre úpravu mapy používateľ väčinou potrebuje len malé množstvo dát, ktoré nie je potrebné filtrovať.

```
GET /api/0.6/map?bbox=left,bottom,right,top
```

Obr. 5.5: Tvar požiadavky na získanie OSM dát v definovanom obdĺžniku, kde `left`, `bottom`, `right` a `top` reprezentujú súradnice strán tohto obdĺžnika.

### 5.3.2 Overpass API

Overpass API je verejné rozhranie, ktoré umožňuje používateľom získavať špecifické podmnožiny OpenStreetMap dát na základe zadaných kritérií [15]. Tieto kritéria sú definované pomocou dotazovacieho jazyka Overpass QL (Overpass Query Language)[2]. Pomocou Overpass QL je možné vytvoriť požiadavku, ktorá vráti presne tie dáta, ktoré používateľ potrebuje. Vďaka tomu sa znižuje objem prenesených dát a zároveň si používateľ nemusí zo získaných dát filtrovať tie, ktoré pre svoje účely nepotrebuje. Narozdiel od hlavného OpenStreetMap API, ktorý je optimalizovaný primárne pre úpravu dát, Overpass API je optimalizované pre ich získavanie.

### 5.3.3 Planet.osm

Planet.osm je projekt, ktorý umožňuje stiahnutie máp väčších územných celkov (štátov, kontinentov, alebo celého sveta) vo formáte OSM XML alebo PBF [16].

Na rozdiel od Overpass API, pomocou ktorého je možné vždy stiahnuť najaktuálnejšie dáta, v prípade Planet.osm sú dáta aktualizované len raz za týždeň. Planet.osm je vhodný pre použitie v prípade, keď je potrebné stiahnuť veľké množstvo mapových dát. Komprimovaná verzia mapy celej planéty vo formáte OSM XML má veľkosť 126.2 GB a vo formáte PBF 68.6 GB (údaje z 01.05.2023).

[2] https://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL

# Kapitola 6

# Použité nástroje

Pre implementáciu simulačného backendu bol zvolený objektovo orientovaný jazyk Python. Hlavnými výhodami tohto jazyka sú, že je multiplatformový a má veľké množstvo knižníc, ktoré umožňujú rýchle a pohodlné vytváranie aplikácií. Frontend aplikácie je implementovaný v jazyku TypeScript, ktorý je nadstavbou jazyka JavaScript s pridanou statickou typovou kontrolou. V tejto časti sú popísané najdôležitejšie knižnice, ktoré sú použité v implementácií simulátora.

## 6.1 PyOsmium

PyOsmium je Pythonová knižnica a nástroj pre spracovanie a úpravu OpenStreetMap dát. Knižnica je postavená na knižnici Osmium [18], ktorá je napísaná v jazyku C++. Umožňuje čítanie, zápis a úpravu dát vo formáte OSM XML, PBF a niekoľkých ďalších. Taktiež umožňuje filtrovanie týchto dát. PyOsmium je navrhnutý tak, aby bol čo najrýchlejší a najefektívnejší, čo z neho robí vhodný nástroj pre prácu nad veľkými súbormi [19]. Princíp spracovávania dát je založený na postupnom prechádzaní všetkých objektov v súbore, a postupné volenie funkcií, ktoré sa majú nad objektami vykonať. Keďže sú v OSM dátovom modeli len tri typy objektov (uzly, cesty a relácie), pre ich spracovanie zvyčajne stačí definovať tri funkcie.

## 6.2 Simulačný framework SimPy

V tejto časti je predstavený framework SimPy základy jeho použitia [8]. SimPy je simulačný framework pre jazyk Python, ktorý je určený pre vytváranie diskrétnych simulácií udalostí. Je obzvášť vhodný pre simuláciu systémov, ktoré zahŕňajú viacero interagujúcich agentov, ako je to aj v prápade simulácie dopravy. Simpy umožňuje definovať procesy, udalosti, zdroje a následne simulovať ich interakcie.

### 6.2.1 Procesy

Na popis aktivít, ktoré sa vykonávajú v čase sa používajú procesy. Proces je definovaný ako sekvencia udalostí, ktoré sa vykonávajú v rámci definovaného časového intervalu. V SimPy je proces implementovaný ako generátor v jazyku Python, ktorý obsahuje kód reprezentujúci danú aktivitu. Každý proces vykonáva sériu inštrukcií a môže byť pozastavený čakaním na udalosť, alebo na uvoľnenie nejakého zdroja. Procesy v SimPy umožňujú modelovanie paralelných aktivít, ktoré prebiehajú v rôznych častiach systému. Každý proces

môže obsahovať viacero udalostí a zdrojov, s ktorými môže interagovať. Vďaka tomu je možné simulovať zložité systémy s množstvomm interakcií medzi jednotlivými procesmi.

### 6.2.2 Udalosti

Udalosť reprezentuje zmenu stavu systému, ktorá nastane v konkrétnom čase. Jedná sa o atomickú operáciu, ktorá sa celá vykoná v jednom okamihu modelového času. Udalosti môžu byť aktivované procesmi, ale aj inými udalosťami. Naplánované udalosti sú uložené v tzv. kalendári udalostí, kde sú radené podľa toho, kedy majú nastať. Typickým príkladom udalosti v dopravnej simácii je napríklad príchod vozidla na križovatku alebo zmena svietiacej farby na semafore.

### 6.2.3 Zdroje

SimPy rozlišuje tri typy zdrojov, ktoré umožňujú modelovanie problémov, kde je potrebné zdieľať obmedzené prostriedky medzi viacerými procesmi.

- `Resource` - pre reprezentáciu objektov, ktoré môžu byť použité súčasne iba obmedzeným počtom procesov.
- `Container` - pre reprezentáciu objektov, ktoré majú obmedzenú kapacitu a je možné ich plniť a vyprázdňovať nejakými homogénnymi prvkami.
- `Store` - pre reprezentáciu objektov, ktoré môžu mať neobmedzenú kapacitu a je možné ich plniť a vyprázdňovať prvkami rôznych typov.

## 6.3 Leaflet

Leaflet je open-source, výkonná a ľahko použiteľná JavaScriptová knižnica pre tvorbu interaktívnych máp pre webové aplikácie [1]. Bola vydaná v roku 2011 a odvtedy sa stala jednou z najpoužívanejších knižníc pre prácu s mapami.

Je navrhnutá tak, aby bola modulárna, čo umožňuje jej jednoduché rozšírenie o nové funkcionality pomocou pluginov. K vykreslovaniu máp používa dlaždicový prístup, kde je mapa rozdelená na množstvo malých štvorcových dlaždíc, ktoré sa načítavajú podľa potreby. Tento prístup umožňuje zobrazovať veľké mapy bez nutnosti načítania celého mapového súboru naraz. Leaflet podporuje množstvo poskytovateľov máp, ako napríklad OpenStreetMap, Mapbox, alebo Google Maps. Taktiež umožňuje vytváranie vlastných vrstiev dlaždíc, vykreslovanie rôznych začiek a iných interaktívnych prvkov na mape.

## 6.4 React

React je JavaScriptová knižnica pre tvorbu užívateľských rozhraní [21]. Je vyvýjaná spoločnosťou Meta Platforms (predtým Facebook) a komunitou samostatných vývojárov. Bol prvýkrát nasadený v roku 2011 a v roku 2013 bol uvoľnený ako open-source projekt. Je určená primárne na tvorbu jednostránkových webových aplikácií.

Jednou z hlavných výhod Reactu je jeho komponentový prístup k tvorbe užívateľského rozhrania. To znamená, že je celá aplikácia rozdelená na množstvo menších znovupoužiteľných komponentov, ktoré sú nezávislé na sebe a môžu byť použité v rôznych častiach

aplikácie. Každý komponent je zodpovedný za vykreslenie svojho vlastného užívateľského rozhrania na základe vstupných parametrov a jeho vnútorného stavu.

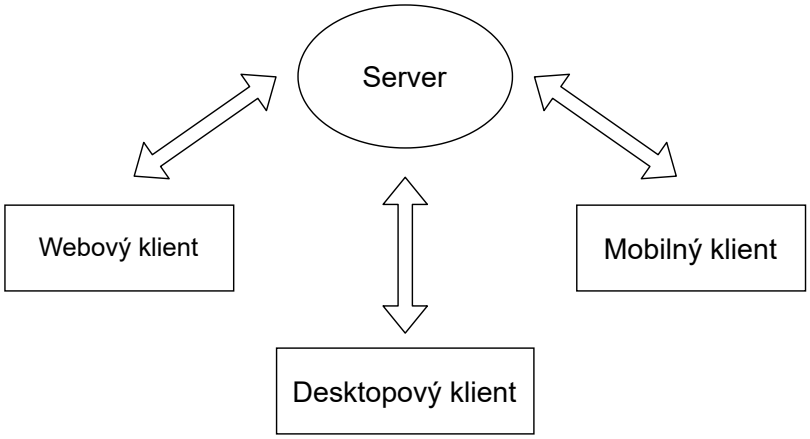
Ďalšou výhodou Reactu je využívanie virtuálneho DOM (Document Object Model). Jedná sa o abstraktnú reprezentáciu DOM, ktorá je uložená v pamäti a je synchronizovaná s reálnym DOM. Keď nastane nejaká zmena v DOM, je najprv vykonaná na jeho virtuálnej kópii, ktorá je následne porovnaná s reálnym DOM. Ten je následne v prípade potreby aktualizovaný. Tento prístup umožňuje znížiť počet operácií nad DOM a tým zvýšiť celkový výkon a efektivitu aplikácie.

React je taktiež veľmi dobre integrovateľný s inými knižnicami. Vďaka tomu existuje veľká komunita vývojárov, ktorí tieto knižnice vytvárajú a udržujú.

# Kapitola 7

# Implementácia simulátora

Jedná sa o webový simulátor, v ktorom sa samotná simulácia odohráva na backende a frontend slúži iba na vizualizáciu jej výsledkov. Tento prístup umožňuje implementáciu viacero klientskych aplikácií, ktoré pracujú s rovnakým simulačným jadrom a môžu pracovať na rôznych platformách. Backend simulátora je implementovaný v jazyku Python s využitím knižníc `PyOsmium`, `SimPy` a `Flask`. Na frontend je použitá JavaScriptová knižnica `React` a knižnica `Leaflet` pre prácu s mapou. V tejto kapitole je popísaný návrh a implementačné detaily oboch častí simulátora.




Obr. 7.1: Architektúra dopravného simulátora, kde sa celá simulácia odohráva na serveri a klientske aplikácie slúžia iba na analýzu a vizualizáciu jej výsledkov.

## 7.1 Návrh simulátora

Rovnako ako pri všetkých projektoch, tak aj pri dopravnom simulátore je počas návrhu vhodné rozložiť problém na viac menších častí. Implementácia webového dopravného simulátora sa dá rozložiť na nasledujúce časti:

1. Analýza a spracovanie mapových dát získaných z OpenStreetMap. Toto zahŕňa návrh štruktúr pre reprezentáciu ciest, jazdných pruhov a križovatiek. Tieto štruktúry je nutné navrhnúť tak, aby boli prehľadné a intuitívne na použitie počas samotnej simulácie.

2. Návrh spôsobu reprezentácie vozidiel a implementácia ich modelov. V tejto časti je nutné zabezpečiť správnu komunikáciu vozidiel s dopravnou sieťou a s ostatnými vozidlami.

3. Návrh a implementácia spôsobu voľby trasy vozidiel, ich vytvárania a odstraňovania z dopravnej siete. Trasy vozidiel sú generované postupne počas simulácie. Vždy po opustení križovatky sa vygeneruje smer odbočenia na ďalšej križovatke. Toto umožňuje budúce rozšírenie simulátora o voľbu trás na základe pravdepodobností odbočenia do jednotlivých smerov, odvodených z reálnych dát.

4. Vytvorenie API pre získavanie výsledkov simulácie. Výsledkom simulácie sú priebežné stavy vozidiel a semaforov na dopravnej sieti. Keďže sa môže jednať o veľké množstvo dát, ktoré sa posielajú po sieti, je nutné zabezpečiť ich kompaktnú reprezentáciu.

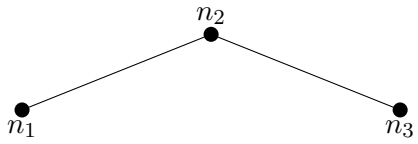
5. Návrh a implementácia frontendu. Súčasťou je možnosť nastavenia parametrov simulácie a následná vizualizácia výsledkov simulácie.

## 7.2 Backend simulátora

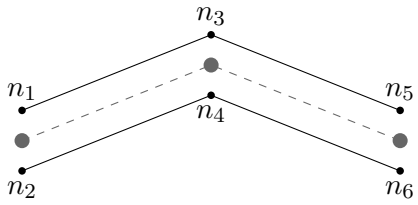
Backend tvorí samotné jadro simulátora. Je zodpovedný za všetky simulačné výpočty a distribúciu výsledných dát ku klientom. Jeho úlohou je spracovanie mapových dát z projektu OpenStreetMap do podoby, ktorá je vhodná pre simuláciu. Parametre simulácie sú definované používateľom a obsiahnuté v HTTP požiadavku na API. Po prijatí požiadavku sa na dopravnej sieti vytvoria vozidlá a spustí simulácia. Postupne sú zaznamenávané priebežné stavy vozidiel, ktoré sa po dokončení simulácie zabalia do kompaktnej podoby a pošlú v odpovedi klientovi.

### 7.2.1 Reprezentácia ciest a jazdných pruhov

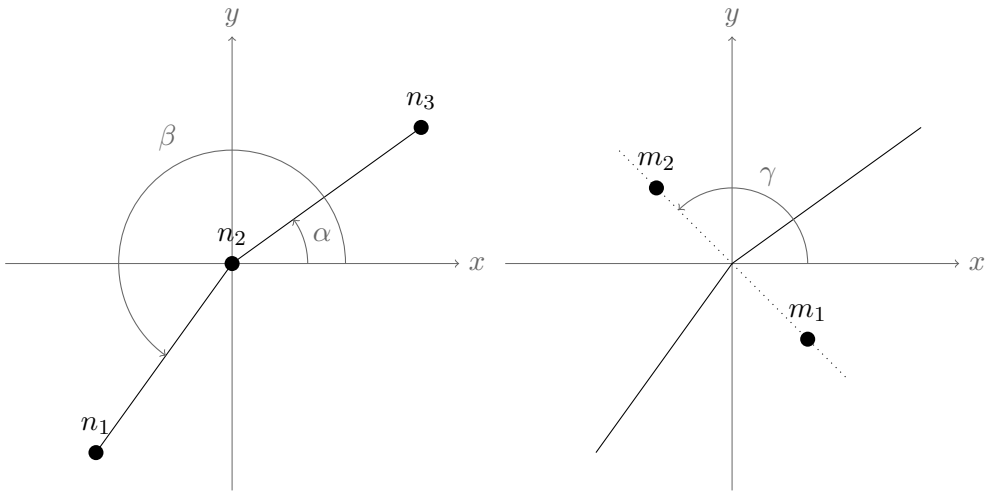
Podobne ako v dátovom modeli OpenStreetMap, je dopravná sieť reprezentovaná pomocou uzlov a trás medzi nimi. Pre simuláciu vozidiel však musíme tieto cesty ešte rodeliť do viacerých jazdných pruhov.




Obr. 7.2: Reprezentácia dvojprúdovej ciesty v OpenStreetMap.




Obr. 7.3: Reprezentácia dvojprúdovej cesty v simulátore.




Obr. 7.4: Výpočet súradníc uzlov jazdných pruhov na základe uzlov ciest.

Výpočet súradníc uzlov jednotlivých jazdných pruhov ($m_1$, $m_2$) je založený na výpočte priemeru uhlov medzi úsečkami tvoriacimi cestu a kladnou poloosou osy $x$.

$$\alpha = atan2(n_{3y}, n_{3x})$$
$$\beta = atan2(n_{1y}, n_{1x})$$
$$\gamma = \frac{\alpha + \beta}{2}$$

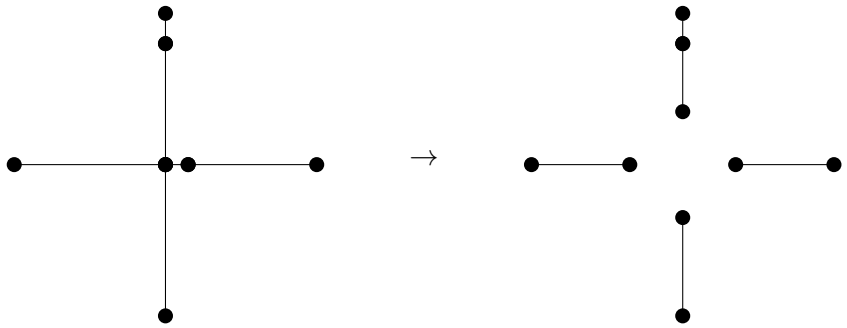
Výsledkom je uhol $\gamma$ definujúci úsečku na ktorej sú rozmiestnené uzly všetkých jazdných pruhov. V prípade, ak sa jedná o prvý alebo posledný uzol cesty, ktorý má iba jeden susedný uzol, je uhol $\gamma$ nastavený na 90° a teda úsečka prechádzajúca uzlami jazdných pruhov je kolmá na pôvodnú cestu.

### 7.2.2 Spracovanie dát z OpenStreetMap

Spracovanie OpenStreetMap dát vykonáva objekt triedy `Parser`, ktorý načítava dáta z OSM súboru a podľa nich postupne vytvára objekty typu `Node`, `Way`, `Lane` a `Crossroad`. Cieľom je previesť vstupné dáta do podoby, ktorá je jednoducho použiteľná pre simuláciu.

- `Node` – Trieda `Node` reprezentuje uzly jednotivých ciest, ktoré sú definované svojim jedinečným identifikačným číslom, súranicami a informáciou, či sa na danom uzle nachádza svetelná križovatka. Objekty si uchovávajú referencie na cesty, na ktorých sa nachádzajú. Pri priradení cesty k uzlu sa automaticky vypočítavajú súradnice uzlov jazdných pruhov patriacich k danému uzlu.
- `Way` – Táto trieda reprezentuje cesty medzi križovatkami. Každá cesta teda začína aj končí križovatkou. Pre jednodouchý prístup k jednotlivým uzlom cesty si trieda uchováva zoznam ich referencií, ktorý je usporiadaný podľa ich poradia na danej ceste. Počas vytvárania cesty sa automaticky vytvárajú aj jazdné pruhy, ktoré sú reprezentované objektami triedy `Lane`. Keďže v OpenStreetMap môžu byť križovatky aj uprostred ciest, je potrebné tieto prípady skontrolovať a previesť dané cesty do podoby, ktorú používa simulátor. Po načítaní všetkých ciest sa teda pre každú z nich skontroje, či nejaký jej nekoncový uzol nepatrí aj do inej cesty. V tom prípade

je nutné cestu rodeliť a medzi novovytvorenými cestami vytvoriť križovatku. Dáta z OpenStreetMap taktiež neberú do úvahy priestor uprostred križovaniek. Preto je potrebné koncové uzly jazdných pruhov posunúť tak, aby sa tento priestor vytvoril. Keď sú však uzly na konci cesty príliš blízko k sebe, posunutie iba posledného uzla by nestačilo. V takom prípade sa predposledný, alebo aj ďalšie uzly musia vymazať.



Obr. 7.5: Vizualizácia odsunutia koncových uzlov od stredu križovatky a vymazania nepotrebného uzla.

- `Lane` – Trieda `Lane` reprezentuje jeden jazdný pruh na ceste. Pre každý uzol cesty sú vypočítané pozície uzlov, ktoré reprezentujú uzly jednotlivých jazdných pruhov na danej ceste. Tvar jazdného pruhu je definovaný práve týmito uzlami. Počas vyvárania jazdných pruhov je však nutné dbať na správne poradie uzlov, ktoré musia byť usporiadané podľa smeru jazdy.
- `Crossroad` – Triedou `Crossroad` je reprezentovaný priestor medzi cestami. V kontexte simulátora teda tento objekt reprezentuje ako križovatky, tak aj spriestor, kde sa na ceste mení počet jazdných pruhov.

Pre niektoré výpočty je potrebné poznať dĺžky ciest, respektíve dĺžky jednotlivých jazdných pruhov. Tie sú reprezentované zoznamom geografických súradníc, ktoré reprezentujú pozície ich uzlov. Vzdialenoť medzi dvoma uzlami je vypočítaná pomocou tzv. Haversinovho vzorca [20]. Ten slúži na výpočet vzdialenosti dvoch bodov na zemskom povrchu, pričom zohľadňuje jeho zakrivenie. Vzorec je definovaný nasledovne:

$$d = 2r \cdot \arcsin(\sqrt{\sin^2(\frac{\phi_2 - \phi_1}{2}) + \cos\phi_1 \cdot \cos\phi_2 \cdot \sin^2(\frac{\lambda_2 - \lambda_1}{2})})$$

kde $r$ je polomer Zeme, $\phi_1$ a $\phi_2$ sú zemepisné šírky a $\lambda_1$ a $\lambda_2$ zemepisné dĺžky bodov. Výsledok $d$ je vzdialenosť medzi bodmi v rovnakých jednotkách, ako je polomer Zeme $r$.

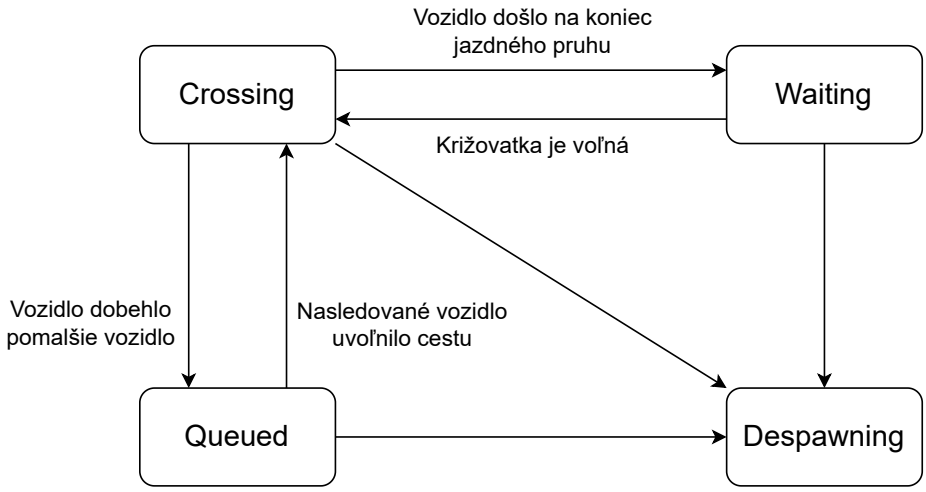
### 7.2.3 Impementácia modelu vozidiel

Model vozidiel je implementovaný v triede `Car`. Táto trieda obsahuje všetky potrebné informácie o vozidle, ako sú jeho rozmery, aktuálny jazdný pruh, pozícia na jazdnom pruhu, rýchlosť a stav. Vozidlá môžu byť v štyroch rôznych stavoch:

- `Crossing` – V tomto stave je vozidlo vtedy, ak voľne prechádza určitým úsekom vo svojom jazdnom pruhu. Vozidlo jazdí rýchlosťou, ktorá je pre daného vodiča pohodlná

a nie je obmedzené žiadnymi prekážkami. Vozidlo v tomto stave môže dobehnúť pomalšie idúce vozidlo v rovnakom jazdnom pruhu. Vtedy sa jeho stav mení na `Queued`. Ak sa vozidlo dostane na koniec svojho jazdného pruhu, kde pre pokračovanie musí počkať na uvolnenie križovatky, jeho stav sa mení na `Waiting`.

- `Queued` – Vozidlo v tomto stave je zaradené do fronty vozidiel a jeho rýchlosť definuje prvé vozidlo danej fronty. Pri zrýchlovaní a spomanovaní vozidiel, ktoré sú vo fronte pred ním, sa jeho rýchlosť vhodne prisôsobí. Ak sa však rýchlosť vozidiel pred aktuálnym vozidlom zvýši nad úroveň komfortnej rýchlosti vodiča, tak sa z fronty odpojí a zmení svoj stav na `Crossing`. Vozidlo v stave `Queued` sa taktiež pokúsi predbehnúť pomalšie idúce vozidlo. V tomto prípade sa využije model odhadu vzdialenosti vozidiel, ktorý je popísaný v kapitole 3.2.3. Tento model vyhodnotí či je predchádzanie možné. Ak je možné vozidlo bezpečne predbehnúť, tak sa na samotné predbiehanie využije model zmeny jazdného pruhu popísaný v kapitole 3.2.2.
- `Waiting` – Tento stav majú vozidlá, ktoré sa nachádzajú na konci svojho jazdného pruhu a čakajú na uvoľnenie križovatky. Vozidlá pri prechádzaní križovatou väčinou nepotrebujú aby boli všetky jazdné pruhy voľné. Preto sa na základe aktuálneho a cieľového jazdného pruhu rozhoduje, na uvoľnenie ktorých jazdných pruhov bude vozidlo čakať. Po uvoľnení všetkých potrebných jazdných pruhov sa vozidlo prepne do stavu `Crossing` a pokračuje v jazde. Pri prechode vozidla z jednej cesty križovatky na druhú sa zároveň vygeneruje trasa, ktorou bude vozidlo pokračovať na ďalšej križovatke. Toto je nutné urobiť preto, aby sa vozidlo mohlo už počas jazdy smerom k nasledujúcej križovatke prisôsobiť a prípadne sa zaradiť do správneho jazdného pruhu.
- `Despawning` – Týmto stavom sú označené vozidlá, ktoré už dosiahli svoj cieľový bod a sú pripravené na odstránenie zo simulácie. Cieľovým bodom rozumieme destináciu vozidla, alebo bod, v ktorom sa vozidlo dostalo do situácie, ktorú nedokáže vyriešiť. Tieto nevyriešiteľné situácie vznikajú predovšetkým kvôli chybným mapovým dátam. Príkladom môže byť napríklad slepá jednosmerná ulica.




Obr. 7.6: Znázornenie stavov vozidla a prechodov medzi nimi.

Keďže sa jedná o udalosťami riadenú simuláciu a stavy jednotlivých vozidiel sa neprepočítavajú v pravidelných simulačných krokoch, vozidlá nie sú samé o sebe schopné rozpoznať

nečakanú zmenu stavu prostredia. Preto sú pre každé vozidlo vytvorené dve udalosti. Prvá je vyvolaná samotným vozidlom vtedy, keď zmení svoj vlastný stav. Toto je potrebné pre to, aby túto udalosť mohli sledovať vozidlá, ktoré sa nachádzajú za daným vozidlom. Tie si tak môžu aktualizovať aj svoj stav s ohľadom na zmenu stavu vozidla pred nimi. Druhá udalosť je vyvolávaná prostredím vozidla. Táto je sledovaná samotným vozidlom a pri vyvolaní si prisôsobí svoj vlastný stav. Vyvolať sa môže napríklad v prípade, keď sa pred vozidlo zaradí nejaké ďalšie a je potrebné prehodnotiť jeho stavové premenné.

### 7.2.4 Reprezentácia križovatiek

Objekt križovatky si uchováva referencie na všetky cesty, ktoré do nej vstupujú. Pre všetky tieto cesty sú vypočítané ich vzájomné smery. Z hľadiska simulácie je toto dôležitý aspekt, pretože vďaka nim je možné rozhodovať, že z ktorých jazdných pruhov môže vozidlo odbočiť na ktorú cestu. Podľa tejto informácie sa vozidlá radia do správnych jazdných pruhov ešte pred vstupom do križovatky.

V každej križovatke sú vytvorené objekty jazdných pruhov, ktoré spájajú jazdné pruhy vstupujúcich ciest s jazdnými pruhmi vystupujúcich ciest. Tieto jazdné pruhy sú vytvorené na základe informácie o možnosti odbočenia z jednej cesty na druhú, ktorá je získaná z OpenStreetMap dát.

Samotný prechod vozidiel cez križovatku je založený na zaberaní zdrojov typu `Resource`, ktoré sú vytvorené pre každý jazdný pruh v križovatke. Keď sa vozidlo nachádza na hlavnej ceste a pokračuje jazdu po nej, tak si zdroj jazdného pruhu, ktorou bude prechádzať na križovatke zaberie niekoľko sekúnd pred tým, ako do samotnej križovatky vstúpi. Tým si zabezpečí, že v križovatke bude mať voľnú cestu a nebude musieť čakať na iné vozidlá. Po prejdení križovatky si vozidlo daný zdroj uvoľní a pokračuje v jazde. V prípade, keď je vozidlo na vedľajšej ceste alebo na ňu odbočuje, tak si zdroj jazdného pruhu pokúsi zabrať až keď sa nachádza na hranici križovatky.

Pri niektorých križovatkách nie je možné určiť hlavné a vedľajšie cesty. To môžne nastať buď z dôvodu chybných mapových dát, alebo z dôvodu, že to tak je aj v skutočnosti. V takomto prípade sa vozidlá riadia pravidlom pravej ruky. Tiež sa pokúšajú zabrať zdroje jazdných pruhov až keď prijdú na hranicu križovatky. Vtedy si navyše skontrolujú všetky jazdné pruhy, ktoré prichádzajú z pravej strany a overia, či sa na nich nenachádza vozidlo, ktoré je dostatočne blízko križovatky. Ak sa tam také vozidlo nacháza, tak sa čaká na jeho prejdenie križovatkou. V opačnom prípade si vozidlo zdroj jazdného pruhu zaberie a pokračuje v jazde.

Pri zaberaní zdroja jazdného pruhu si však vozidlá musia overiť, či sa daný jazdný pruh nekríži s iným, ktorý je už zabratý iným vozidlom. V tom prípade sa čaká na uvoľnenie všetých jazdných pruhov, ktoré sa krížia s tým, ktorým chce vozidlo prejsť. Týmto spôsobom je zabezpečené, že sa v križovatke nebudú krížiť cesty dvoch vozidiel, čo by spôsobilo ich kolíziu.

V prípade svetelných križovatiek je princíp podobný, avšak tie si v určitých časových intervaloch zablokujú zdroje jazdných pruhov, ktoré majú červenú. Vozidlá si tieto zdroje nemôžu zabrať a musia čakať pred križovatkou na prepnutie svetiel. Jazdné pruhy sú na každej križovatke rozdelené do dvoch skupín, ktoré majú spoločné svetelné signály. Tieto skupiny sa určujú podľa vzájomnej polohy ciest na ktorých tieto jazdné pruhy začínajú. Jazdné pruhy z dvoch protiľahlých ciest patria do rovnakej skupiny.

### 7.2.5 Kalendár udalostí

Celú simuláciu je potrebné zaznamenávať, aby sa následne mohla poslať na frontend. K tomu slúži kalendár udalostí. Ten rozlišuje dva typy udalostí, zmenu stavu vozidla a zmenu stavu svetelnej križovatky.

Pri každej zmene stavu vozidla sa do kalendára pridá nový záznam s jeho novým stavom. Zaznamenaná udalosť sa skladá z nasledujúcich informácií:

- `time` – čas, kedy sa udalosť stala
- `car_id` – identifikátor vozidla, ktorého sa udalosť týka
- `way_id` – identifikátor cesty, na ktorej sa vozidlo nachádza
- `lane_id` – identifikátor jazdného pruhu, na ktorom sa vozidlo nachádza
- `position` – pozícia vozidla v rámci jazdného pruhu
- `speed` – aktuálna rýchlosť vozidla

V prípade zmeny stavu svetelnej križovatky sa do kalendára pridá záznam s nasledujúcimi informáciami:

- `time` – čas, kedy sa udalosť stala
- `crossroad_id` – identifikátor svetelnej križovatky
- `green_lanes` – zoznam jazdných pruhov, ktoré majú zelenú

Sekvencia týchto informácií je dostatočná pre vizualizáciu pohybu vozidiel a stavov svetelných križovatiek počas celej simulácie. Výhodou tohto prístupu je, že sa záznamy tvoria iba pri zmene stavu vozidla, respektíve križovatky a nie v každom simulačnom kroku. Výsledkom je značné zníženie množstva dát, ktoré je potrebné spracovať a preniesť na frontend.

### 7.2.6 API dopravného simulátora

K získaniu dát z dopravného simulátora slúži API, vytvorené pomocou knižnice `Flask`, ktoré má jediný endpoint. Na tento endpoint klient posiela požiadavky typu `GET` s požadovanými parametrami simulácie a v odpovedi dostáva dáta, ktoré sú potrebné na jej vizualizáciu. Súčasťou týchto dát sú informácie o všetkých cestách, jazdných pruhoch, križovatkách a zoznam všetkých záznamov z kalendára udalostí. Kedže výsledkom niektorých simulácií môže byť veľké množstvo dát, tak sa posielajú v kompaktnej binárnej podobe. Takto sa zníži množstvo dát prenesených po sieti. Štruktúra týchto dát je znázornená na obrázku A.1.

### 7.2.7 Parametre simulácie

Pri spustení samotnej simulácie má používateľ možnosť zadať niekoľko parametrov. Tieto parametre sú nasledovné:

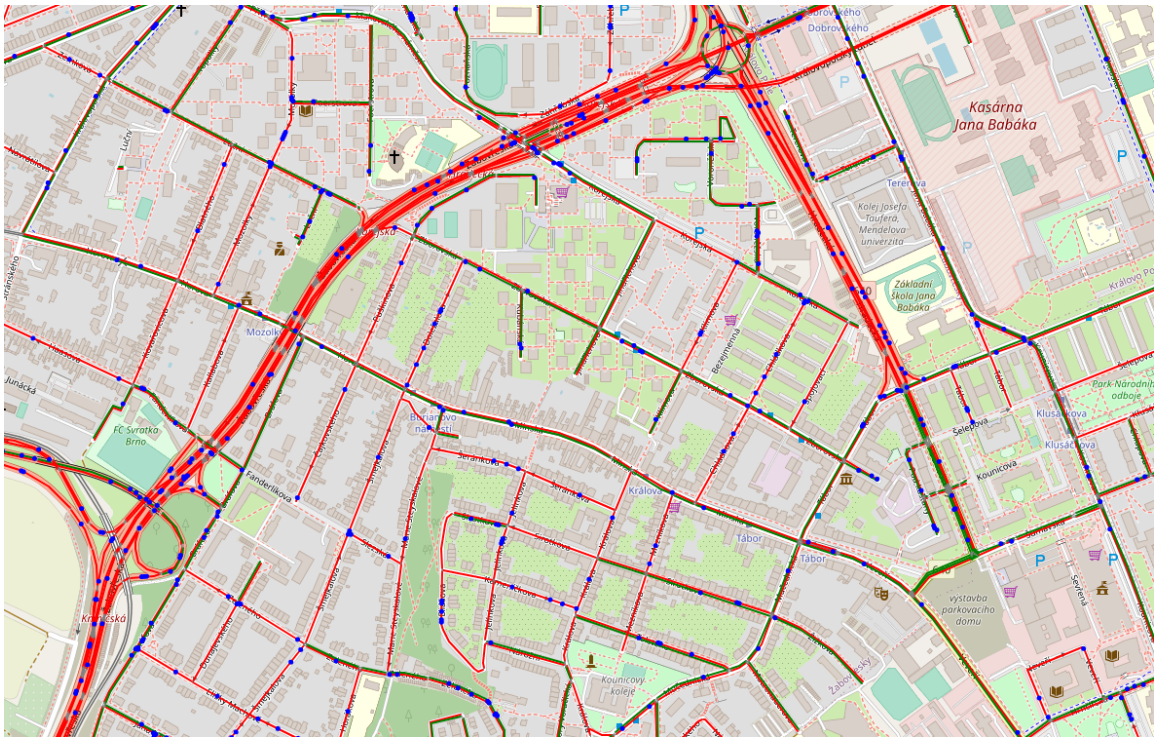
- počet vozidiel, ktoré sa majú v simulácii nachádzať

- dĺžka simulovaného časového úseku
- seed pre generátor náhodných čísel

Nastavením počtu vozidiel má používateľ možnosť simulovať rôzne hustoty premávky. Týmto spôsobom je napríklad možné simulovať dopravu v rámci rôznych častí dňa, alebo rôzne dopravné špičky. Je však potrebné dbať na to, že zvyšovaním počtu vozidiel na mape sa zvyšuje čas potrebný na simuláciu a aj množstvo vygenerovaných dát. Keďže sa vozidlá generujú rovnomerne na celej mape, tak je vhodné zvoliť dĺžku simulovaného časového úseku tak, aby sa hustota premávky na jednotlivých cestách stihla ustáliť. To znamená, aby sa v simulácií mohlo premietnuť napríklad to, keď je nejaká križovatka nesprávne navrhnutá a vznikajú na nej dopravné zápchy. Umožnenie zvolenia seedu je z dôvodu, aby si používateľ mohol zopakovať rovnakú simuláciu viackrát, poprípade aby ju mohol jednoducho zdieľať.

## 7.3 Frontend simulátora

Výsledky simulácie sa zobrazujú na webovom rozhraní. Po otvorení stránky má používateľ možnosť si zvoliť požadované parametre simulácie. Následne sa odošle požiadavka na API, ktorá simuláciu s danými parametrami vykoná a vráti výsledky. Frontendová časť aplikácie tieto dáta spracuje a zobrazí používateľovi. Najprv sa vykresluje dopravná sieť, ktorá sa skladá z ciest, jazdných pruhov a križovatiek. Potom sa v určitých časových intervaloch vykreslujú pozície jednotlivých vozidiel.



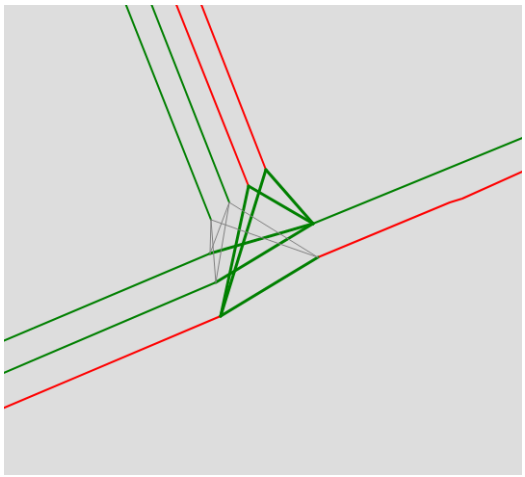

Obr. 7.7: Vizualizácia simulácie v rámci webového rozhrania. Na obrázku je zobrazená OpenStreetMapa, v ktorej sú vykreslené cesty a nimi prechádzajúce vozidlá.

### 7.3.1 Vykreslovanie ciest a jazdných pruhov

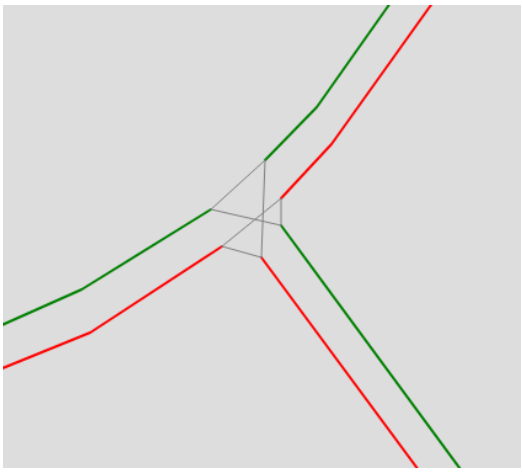
Keďže sa zobrazená mapa aktualizuje v pravidelných časových intervaloch, vykreslovanie veľkého množstva ciest môže byť náročnou operáciou, ktorá by mohla znížiť celkový výkon aplikácie. Cesty a jazdné pruhy sú ale statickými objektami, ktoré sa počas simulácie nemenia. Preto je možné ich vykresliť len raz, tesne pred začiatkom zobrazovania vozidiel. Toto je dosiahnuté použitím memoizovaného komponentu `Roadnet`. Memoizovaný komponent je taký, ktorý si prepočítava svoj výstup len vtedy, keď sa zmenia jej vstupné parametre. Keďže vstupným parametrom pri každej aktualizácií mapy je objekt, ktorý reprezentuje dopravnú sieť a nikdy sa nemení, tak sa všetky výpočty vykonávajú len raz. Samotné vykreslovanie je implementované použitím komponentu `Polyline` z knižnice `Leaflet`, ktorý vykreslí úsečku medzi zadanými bodmi, reprezentovanými svojimi zemepisnými súradnicami. Tieto body sú získané z objektov `Lane`, ktoré obsahujú zoznam uzlov a ich súradníc reprezentujúcich daný jazdný pruh.

### 7.3.2 Vykreslovanie križovatiek

Križovatky sú vykreslované rovnakým spôsobom ako jazdné pruhy. Jediným rozdielom sú križovatky so svetelnou signalizáciou, pri ktorých je potrebné vyznačiť aj stav semaforov. Toto je dosiahnuté tým, že sa jazdné pruhy, ktoré majú v danom čase zelenú, vykreslia zelenou farbou, ostatné sú vykreslené šedou farbou ako pri križovatkách bez semaforov. Informácia o tom, ktoré jazdné pruhy majú zelenú je zahrnutá v záznamoch kalendára udalostí.



Obr. 7.8: Reprezentácia svetelnej križovatky. Zelené jazdné pruhy v križovatke znázorňujú smery, v ktorých majú vozidlá zelenú.



Obr. 7.9: Reprezentácia križovatky bez semaforov. V takýchto križovatkách sa vozidlá riadia hlavnými cestami, alebo pravidlom pravej ruky.

### 7.3.3 Vykreslovanie vozidiel

Pozície jednotlivých vozidiel sú v rámci udalostí zaznamenávané len v prípadoch, keď za zmení ich stav. Preto je pred samotným vykreslením nutné vypočítať pozície vozidiel medzi jednotlivými udalosťami. V prvom kroku sa zoznam udalostí rozdelí na viacero menších zoznamov, ktoré obsahujú udalosti pre jednotlivé vozidlá. Následne sa podľa aktuálneho

simulačného času $t$ nájdu dve udalosti ($e_{prev}$, $e_{next}$), medzi ktorými sa tento čas nachádza. Podľa rozdielu časov týchto dvoch udalostí sa vypočíta pozícia $p$ vozidla na danom jazdnom pruhu. Táto hodnota je v percentách, kde 0 % znamená začiatok jazdného pruhu a 100 % jeho koniec.

$$\Delta t = time(e_{next}) - time(e_{prev})$$
$$\Delta p = position(e_{next}) - position(e_{prev})$$
$$p = position(e_{prev}) + (t - time(e_{prev})) * \Delta p$$

Keďže pre vykreslenie vozidla je potrebné poznať jeho geografické súradnice, tak je nutné túto hodnotu vypočítať z pozície vozidla na jazdnom pruhu. K tomuto výpočtu je potrebné poznať dĺžku celého jazdného pruhu $l$, ktorý sa vypočíta nasledovne:

$$l = \sum_{i=0}^{n} s_i$$

kde $s_i$ je dĺžka jednotlivých segmentov a $n$ je ich celkový počet na danom jazdnom pruhu. Pozícia vozidla $d$ v kilometroch od začiatku jazdného pruhu sa vypočíta vynásobením percentuálnej pozície s celkovou dĺžkou daného jazdného pruhu.

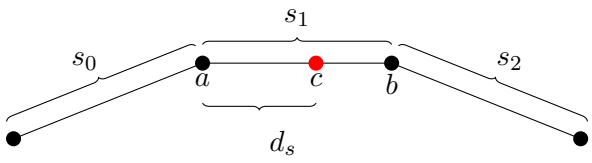
$$d = \frac{p}{100} * l$$

Následne sa iteráciou po jednotlivých segmentoch jazdného pruhu a postupným sčítavaním ich dĺžok nájde segment, na ktorom sa vozidlo nachádza $s_v$. Pozícia vozidla na tomto segmente sa vypočíta ako rozdiel medzi celkovou dĺžkou jazdného pruhu a súčtom dĺžok všetkých predchádzajúcich segmentov.

$$d_s = d - \sum_{i=0}^{v-1} s_i$$

Z tejto pozíce sa pomocou lineárnej iterpolácie medzi súradnicami začiatočného a koncového bodu segmentu ($a$, $b$) vypočítajú súradnice $c$ vozidla.

$$c = a + (b - a) * \frac{d_s}{s_v}$$




Obr. 7.10: Výpočet súradníc vozidla na základe jeho pozície na jazdnom pruhu.

# Kapitola 8

# Záver

Cieľom práce bolo navrhnutie a implementácia mikroskopického dopravného simulátora s webovým rozhraním. Simulátor mal byť schopný simulovať vozidlá, ktoré dodržiavajú pravidlá cestnej premávky. Zároveň mala byť celá simulácia založená na mapových dátach z projektu OpenStreetMap. Všetky tieto požiadavky boli v práci splnené.

V prvej časti práce sú popísané teoretické základy z oblasti modelovania a simulácie dopravy. Taktiež sú popísané jednotlivé typy dopravných simulátorov a detailne je vysvetlený princíp mikroskopických dopravných simulátorov, na ktorom je táto práca založená. Ďalej je vysvetlený dátový model OpenStreetMap a spôsoby získavania dát z tohto projektu. V tejto časti sú taktiež predstavené dopravné dáta z mesta Brno, ktoré by mohli byť v budúcnosti využité na rozšírenie simulátora.

Druhá časť práce sa venuje návrhu a implementačným detailom vytvoreného dopravného simulátora. Je popísaná jej architekúra, najdôležitejšie komponenty simulácie a spôsob komunikácie medzi klientskou a serverovou časťou aplikácie.

Aj keď je výsledný simulátor funkčný a spĺňa požiadavky, ktoré sú na neho kladené, stále je priestor na jeho vylepšenie a rozšírenie. Hlavným vylepšním by bola aplikácia už spomínaných dopravných dát na vytvorené modely. Taktiež by bolo vhodné brať do úvahy reálne rozmery vozidiel aj pri ich vykreslovaní, nie len pri výpočte ich pozície. V neposledom rade by sa do simulácie dali zakomponovať aj ďalší účastníci cestnej premávky, napríklad vozidlá mestskej hromadnej dopravy, cyklisti alebo chodci. Všekty tieto vylepšenia by viedli k ešte väčšej realistickosti simulácie a tým aj k jej väčšej vypovedacej hodnote.

# Literatúra


[1] Agafonkin, V. *Leaflet* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://leafletjs.com/.

[2] Barcelo, J., García, D. a Perarnau, J. Methodological notes on combining macro, meso and micro models for transportation analysis. Január 2005.

[3] Behrisch, M., Bieker Walz, L., Erdmann, J. a Krajzewicz, D. SUMO – Simulation of Urban MObility: An Overview. In:. Október 2011, sv. 2011. ISBN 978-1-61208-169-4.

[4] Brněnské komunikace, a. *Intenzita dopravy - Intenzita vozidel / Vehicle traffic intensity* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://gis.brno.cz/ost/edas/public/17eb2c51-d790-4e0d-ba8f-6d0255484ac1.

[5] Carlino, D. *A/B Street* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://a-b-street.github.io/docs/.

[6] Dostál, M. *Simulátor městské dopravy*. 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[7] Jiménez, T., Mussi, P. a Siegel, G. A road traffic simulator: Car-following and lane-changing. In:. Január 2000, s. 241–245.

[8] Klaus G. Müller, T. V. *SimPy* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://simpy.readthedocs.io.

[9] Law, A. *Simulation Modeling and Analysis*. 5. vyd. McGraw-Hill Education, 2015. McGraw-Hill series in industrial engineering and management science. ISBN 9781259010712.

[10] Leemis, L. M. a Park, S. K. *Discrete-Event Simulation: A First Course*. USA: Prentice-Hall, Inc., 2005. ISBN 0131429175.

[11] Mohan, R. a Ramadurai, G. State-of-the art of macroscopic traffic flow modelling. *International Journal of Advances in Engineering Sciences and Applied Mathematics*. 1. vyd. September 2013, zv. 5, s. 10. DOI: 10.1007/s12572-013-0087-1.

[12] Němec, F. *Simulátor MHD linek v Brně*. 2014. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií.

[13] OpenStreetMap Wiki. *Elements* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://wiki.openstreetmap.org/wiki/Elements.

[14] OpenStreetMap Wiki. *History of OpenStreetMap* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://wiki.openstreetmap.org/wiki/History_of_OpenStreetMap.

[15] OpenStreetMap Wiki. *Overpass API* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://wiki.openstreetmap.org/wiki/Overpass_API.

[16] OpenStreetMap Wiki. *Planet.osm* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://wiki.openstreetmap.org/wiki/Planet.osm.

[17] Peringer, P. *Modelování a simulace IMS: Studijní opora*. Fakulta informačních technologií VUT v Brně, 2021.

[18] Topf, J. *Osmium* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://osmcode.org/libosmium/.

[19] Topf, J. *PyOsmium* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://osmcode.org/pyosmium/.

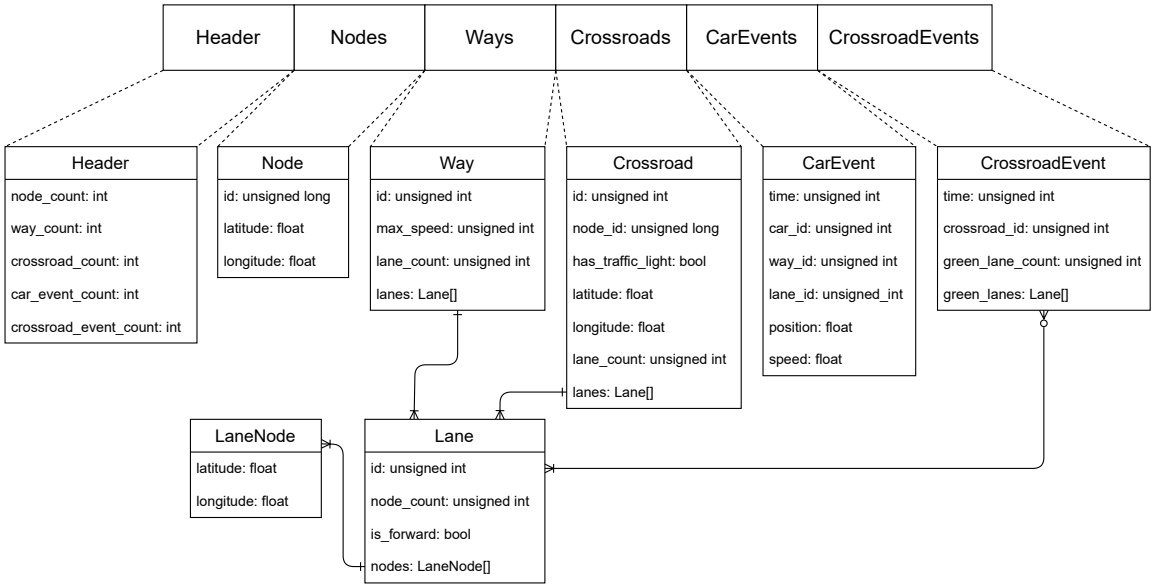
[20] Wikipedia. *Haversine formula — Wikipedia, The Free Encyclopedia* [online]. 2023. [Online; accessed 15-May-2023]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Haversine_formula&oldid=1154480125.

[21] Wikipedia. *React (software) — Wikipedia, The Free Encyclopedia* [online]. 2023. [Online; accessed 14-May-2023]. Dostupné z: http://en.wikipedia.org/w/index.php?title=React%20(software)&oldid=1153998051.

[22] ČR Ředitelství silnic a dálnic. *Sčítání dopravy* [online]. 2023 [cit. 2023-05-12]. Dostupné z: https://www.rsd.cz/silnice-a-dalnice/scitani-dopravy.

# Príloha A

# Štruktúra prenášaných dát na frontend




Obr. A.1: Vizualizácia štruktúry binárnych dát získaných z API dopravného simulátora. V havičke (`Header`) je uložený počet uzlov, ciest, križovaniek a udalostí, ktoré po samotnej hlavičke nasledujú. Toto je potrebné z dôvodu, aby sa počas následného spracovávania vedelo, kde jednotlivé časti dát začínajú a končia. Podobne to platí aj v prípade zanorených štruktúr, ako sú napríklad jazdné pruhy (`Lane`) v rámci ciest (`Way`), alebo križovaniek (`Crossroad`).

# Príloha B

# Obsah SD

Priložená SD karta obsahuje:

```
+-- src                    - zdrojové súbory aplikácie
|   |
|   +-- server             - zdrojové súbory serverovej časti aplikácie.
|   |
|   +-- client             - zdrojové súbory klientskej časti aplikácie.
|   |
|   +-- README.md          - návod na spustenie aplikácie.
|
+-- doc                    - technická správa
    |
    +-- src                - zdrojové súbory technickej správy
    |
    +-- projekt.pdf        - technická správa vo formáte PDF
```