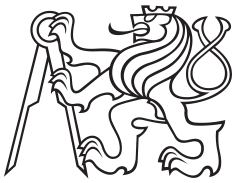
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra počítačové grafiky a interakce

Škálovatelná simulace dopravy v systému VRUT

Vít Neckář

Školitel: doc. Ing. Jiří Bittner, Ph.D.
Leden 2020

Poděkování

Děkuji panu docentovi Ing. Jiřímu Bittnerovi, Ph.D. za trpělivost a vedení této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

Abstrakt

Tato práce popisuje aktuální implementaci chování řidičů v systému VRUT a navrhuje její vylepšení.

Klíčová slova: VRUT, behaviorální model, vozidla, křižovatky, simulace

Školitel: doc. Ing. Jiří Bittner, Ph.D.
Praha 2,
Karlovo náměstí 13,
E-421

Abstract

This thesis describes the current implementation of drivers' behaviour in VRUT and suggests its improvement.

Keywords: VRUT, behavioral model, vehicles, junctions, simulation

Title translation: Scalable Traffic Simulation for VRUT

Obsah

Obrázky

Tabulky

Kapitola 1
Úvod

V dnešní době mechanizace a nahrazování člověka stroji je stále více lidských činností prováděno počítači a umělou inteligencí, což platí i pro řízení automobilů. S rostoucím počtem vozidel na silnicích roste i riziko možných nehod, ať už těch malých, kde jsou škody pouze na majetku, po ty velké, které končí v lepším případě zraněním svých účastníků, v horším případě ztrátou na životě.

1.1 Motivace

Člověk není dokonalý a každý z nás dělá chyby. Výrobci automobilů se snaží nebezpečným situacím vyvarovat a již nyní mají moderní vozidla ve výbavě různé technologie, jež pomáhají řidičům v bezpečné jízdě a vyhnutí se nehodě (např. asistent jízdy v pruzích, automatické nouzové brzdění, adaptivní tempomat apod.).

Automatizace ale může jít ještě dál, a to k autonomnímu vozidlu. Několik značek již své autopiloty testuje na silnicích, a i když jsou někteří lidé skeptičtí, další výrobci se přidávají.

Před zkouškami v reálném prostředí je nutné testovat autonomní vozidla v simulátorech. Jedním z nich je VRUT od ŠKODA Auto a Katedry počítačové grafiky a interakce ČVUT. Náplní tohoto projektu je vylepšení stávající implementace modulů Traffic a VehicleSimulator, přesněji rozšíření modulu Traffic, aby lépe modeloval chování řidičů ve složitějších dopravních situacích, jako jsou víceúrovňové křižovatky a předjíždění, zjednodušená simulace pro vozidla ve větší vzdálenosti od kamery a implementace zjednodušeného kinematického modelu v modulu VehicleSimulator. Ten bude aplikován na vzdálenější vozidla za účelem snížení výkonnostních nároků simulátoru.

1.2 Cíle práce

Cílem práce je prostudovat a zanalyzovat metody používané pro simulaci dopravy v menších a středně rozsáhlých simulacích dopravních systémů, popsat aktuální implementaci předjíždění a projíždění křižovatek a navržení nové implementace.

Kapitola 2

Metody používané pro simulaci dopravy

Existuje několik přístupů k simulaci dopravy. Tato kapitola přístupy popisuje.

2.1 Simulační modely

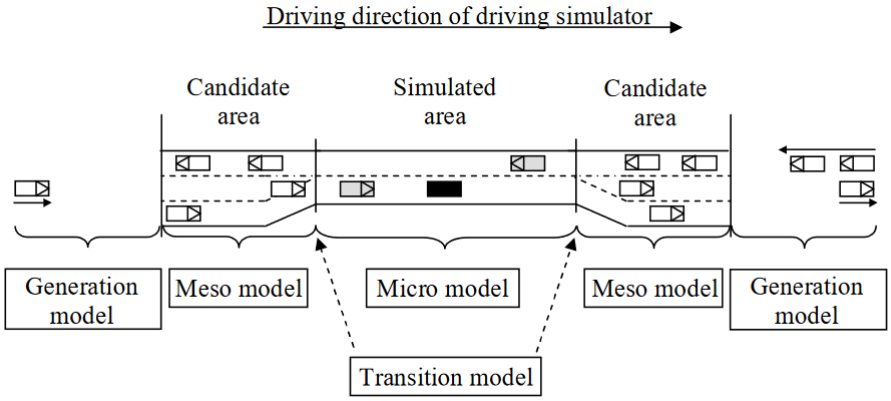
Články [3] a [4] mluví o simulačních modelech, které dále rozdělují do několika kategorií podle vrstvy, ve které simulují.

Nejpropracovanější je mikroskopický model. Každé vozidlo je bráno nezávisle a má své vlastní detailní jízdní vlastnosti, model rozhodování řidiče při interakci s ostatními a další vlastnosti, díky kterým simulace vypadá realisticky, což je nesporná výhoda. Nevýhodou je omezená škálovatelnost, jelikož pro větší počet vozidel je simulace výpočetně velmi náročná. Asi nejznámějším příkladem toho modelu je simulátor SUMO (Simulation of Urban MObility) [6].

Méně detailním, ale stále propracovaným modelem, je mezoskopický model. Vozidla mohou být simulována po skupinách i jako jednotlivci, ale jejich vzájemná interakce už není tak detailní jako u mikroskopického modelu. Interakce je založena na základě rychlosti, hustotě a průjezdnosti dopravy v daném úseku.

Nejméně detailním modelem je model makroskopický. Nestará se o vozidla jako jednotlivce, ale bere dopravu jako celkovou průjezdnost úsekem. Makroskopický model má tedy výhodu ve své menší náročnosti, nemůžeme v něm ale jedno vozidlo reprezentovat jako jednotku ovlivňující celek.

Modely se můžou ještě dále kombinovat, o čemž mluví práce [3] jako o hybridním modelu. Pro blízké okolí řidiče se používá detailní a náročný mikroskopický model, pro větší vzdálenosti, kde není přesná simulace potřeba, se používá méně detailní makroskopický model, čímž se ušetří drahocenný výkon. Mezi těmito dvěma modely může být ještě model mezoskopický, pomocí kterého můžeme mít plynulý přechod vozidel. Právě přechod mezi úrovněmi je důležitý, nechceme-li, aby simulace vypadala nevěrohodně.



Obrázek 2.1: Schéma rozdělení simulace na vrstvy z článku [5]. Černé vozidlo je řízeno uživatelem, šedivá vozidla jsou simulována a bílá vozidla jsou kandidáti k simulaci.

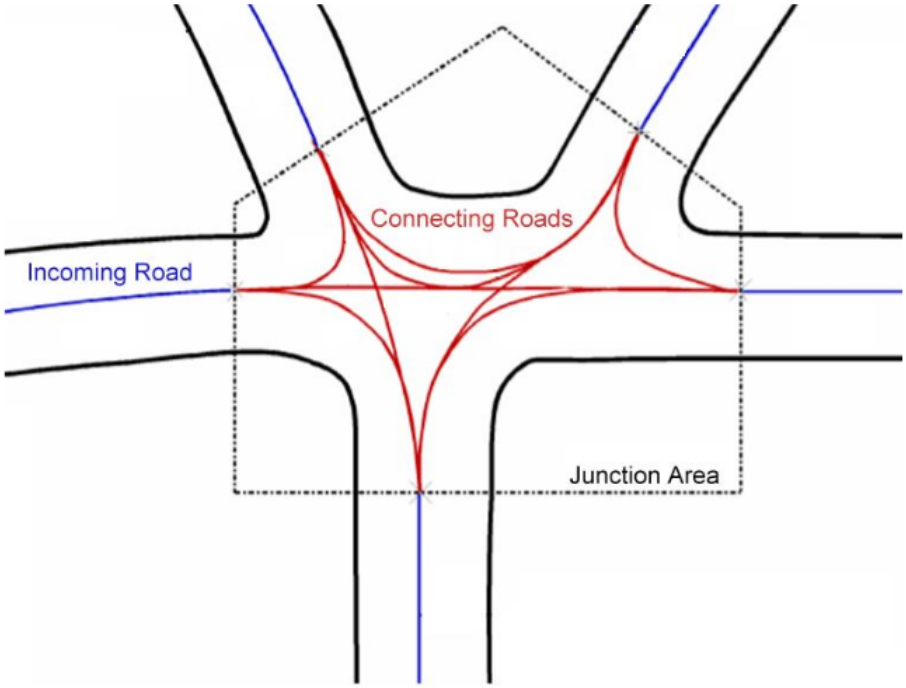
2.2 Datové struktury

Hlavní simulační datovou strukturou bývá (a je tomu tak i ve VRUTu) graf reprezentující silniční síť. Jednotlivé uzly reprezentují křižovatky, hrany mezi nimi jsou silnice. Pokud z uzlu nevychází žádná hrana, jedná se o slepou ulici, uzel se dvěma body znázorňuje jen jeden úsek z větší silnice a může znázorňovat zatáčku nebo se může využívat k lepšímu výpočtu při hledání cesty či realistickém průjezdu grafem [7]. Dále v sobě uzly mohou mít uložené informace, jako je třeba maximální rychlost.

Jak bylo již zmíněno, hrany mezi uzly reprezentují silnice nebo úseky silnic. Každý pruh má svoji vlastní hranu, jednoproudé silnice mají tedy dvě hrany, víceproudé dálnice jich mají více. Tato topologie značně zjednodušuje simulaci. Při jízdě autonomního vozu se nemusí počítat offset od středu silnice.

V simulátorech se často využívá hierarchická reprezentace. OpenDRIVE® [8], což je specifikace formátu pro popis silničních sítí, má na vrcholu hierarchie silnice a křižovatky. Středem silnice vede referenční linka. Tato linka popisuje vlastnosti celé silnice, jako jsou například dopravní značky, tvar nebo převýšení. Silnice se skládají z pruhů a navzájem se spojují vztahem předchůdce-následovník. V místě, kde se setkávají 3 a více silnic je nutno použít křižovatku. Ty se skládají z tras, kterými může vozidlo křižovatku projet.

Projekt OpenStreetMap, zaměřený na tvoření a vizualizaci topografických map, používá k reprezentaci dat uzly, v nichž jsou uložené geografické souřadnice. Jako samostatné mohou představovat body zájmu nebo hory. Cesta je uspořádaný list uzlů a reprezentuje jak lineární vlastnosti typu silnic nebo řek, tak i oblasti, například parky nebo jezera. Posledním prvkem jsou relace, neboli uspořádané seznamy uzlů, cest a dalších relací, kterým lze být přiřazena nějaká vlastnost.



Obrázek 2.2: Obrázek ukazující reprezentaci křižovatek a silnic v OpenDRIVE [8]

Vozidla jako taková jsou reprezentovány instancí své třídy. Každé vozidlo může mít jiné vlastnosti, jako jsou maximální rychlost, akcelerace, agresivita řidiče apod. Dále je důležitá aktuální pozice na silnici, která se aktualizuje každou jednotku času simulace (krok). Vozidla mají také svůj stav, ve kterém právě jsou, např. "Stojí", "Jede", "Předjíždí", "Je předjížděn"atd.

2.3 Mezoskopický model

Jak bylo již napsáno v sekci 2.1, mezoskopické modely vyplňují mezeru mezi makroskopickými a mikroskopickými modely. Existuje několik různých metod jejich implementace. Jedna možnost je reprezentovat silniční síť jako uzly a hrany a vozidla sdružovat do skupin, které cestují po hranách z uzlu do uzlu. Jiný způsob je rozdělit hrany na segmenty a skupiny vozidel nechat pohybovat po nich. Třetí přístup je modelovat samostatná vozidla na hranách makroskopicky a v uzlech používat frontu na simulaci křižovatek. Tento poslední způsob používá model Mezzo, který zde chci popsat blíže [10].

2.3.1 Mezzo

Mezzo je mezoskopický model vyvinutý tak, aby se mohl jednoduše spojit s jiným mikroskopickým modelem.

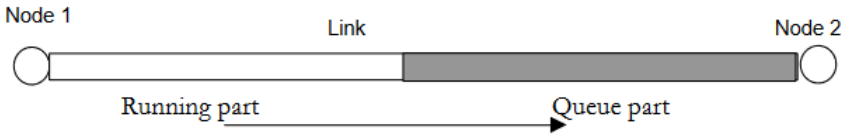
Silniční síť se skládá z uzlů, které reprezentují křižovatky, nájezdy a sjezdy z dálnic, a jednosměrných hran, představujících cesty mezi uzly. Na většinu silnic stačí dvě hrany, jedna na každý směr.

Hrany jsou rozděleny do dvou částí - část pro jízdu a část pro kolonu. Kolona začíná na konečném uzlu hrany a roste směrem k počátečnímu, když do hrany více vozidel přijíždí, než z ní vyjíždí. Pokud by například na konečném uzlu byla světelná křižovatka a na semaforu by padla červená, část s kolonou se bude zvětšovat.

Vozidlo se pohybuje po části pro jízdu rychlostí určenou z hustoty provozu na hraně. Když je hustota nízká, rychlost vozidla bude vyšší a naopak při vysoké hustotě provozu bude rychlost nízká. Tato rychlost se používá k výpočtu nejdřívějšího času opuštění hrany. Vozidla jsou podle tohoto parametru na hraně seřazena.

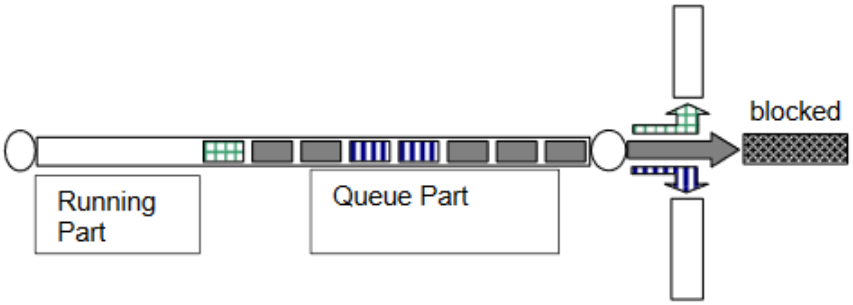
Kolona je definována tak, aby v čase t obsahovala pouze vozidla s nejdřívějším časem opuštění menším než t. Jinými slovy ta vozidla, která by hranu již opustila, kdyby na konečném uzlu nebylo nějaké zdržení.

Hustota provozu na hraně se nesmí počítat dvakrát, bere se v potaz tedy pouze v části pro jízdu. Pro vozidla v koloně se nepočítá, jelikož toto zdržení je započteno již v čekání v koloně. Pokud by se hustota počítala pro všechna vozidla na hraně, zdržení způsobené vozidly v koloně by se počítalo dvakrát.



Obrázek 2.3: Reprezentace hrany v modelu Mezzo [10]

Konečný uzel napojuje hranu k jiným hranám. Každé toto napojení se nazývá odbočení (i v případě, že vozidlo jede rovně). Odbočením projede za určitý čas jen omezený počet vozidel (kapacita odbočení). V reálném světě určuje množství projetých vozidel délka zelené na semaforu, přednosti nebo vozidla směřující jinam blokující přístup k odbočení. V Mezzo je kapacita odbočení reprezentována frontou. Vozidla jsou postupně vybírána z kolony před uzlem a posílána do další hrany, pokud je tam pro ně místo. Každé odbočení má vlastní server, který se stará pouze o vozidla, která potřebují tímto odbočením projet. Je nutné ale brát v potaz, že fronta je jen jedna, a tak mohou vozidla vepředu, jejichž hrana je plná, blokovat vozidla za sebou, jejichž odbočení je průjezdné. Zároveň je možné, že vozidlo, které je ve frontě například na druhém místě, jeho odbočení je průjezdné, avšak vozidlo před ním nemůže pokračovat, tak by mohlo křižovatkou projet svým odbočovacím pruhem. Mezzo je ale mezoskopický model a pruhy nijak nereprezentuje. Je však možné pro každé odbočení nadefinovat look-back limit, neboli maximální počet vozidel od začátku kolony, pro které server kontroluje, jestli nechtějí daným odbočením projet.



Obrázek 2.4: Reprezentace křižovatky a odbočení v modelu Mezzo [10]. Pokud by byl look-back limit na odbočení doleva a doprava (kostičkované, resp. pruhované) nastavený na hodnotu 4, budou moci pruhovaná vozidla směřující doprava projet, kostičkované směřující doleva už ale ne.

Server odbočení je modelován stochastickým procesem. Tento proces vygeneruje časový interval, za jaký server zpracuje jedno vozidlo. Tím se reprezentuje kapacita odbočení. Pokud je odbočení víceproudé, má každý pruh vlastní server, přestože samotné pruhy reprezentovány nejsou. Tím se zařídí větší přesnost kapacity odbočení. Víceproudým odbočením totiž mohou ve stejný okamžik projet dvě vozidla, což na jednoproudém není možné. Průjezd vozidel na dvou různých pruzích lze též považovat za nezávislý, takže by měl být v modelu reprezentován dvěma různými procesy.

2.4 Simulace pomocí modelování chování řidičů

Aby simulace dopravy vypadala reálně, je nutné, aby se agenti v simulaci chovali uvěřitelně. Toho se dá dosáhnout aplikací kvalitních behaviorálních modelů. Na druhou stranu ale tyto modely nesmějí být příliš komplikované, aby jejich výpočet nebyl moc časově náročný. Ideální praxe je nalezení kompromisu mezi počtem parametrů těchto modelů a akceptovatelnou simulací. Tyto parametry by také měly být součástí parametrů třídy vozidla nebo řidiče, aby s nimi uživatel mohl jednoduše experimentovat a testovat je.

Nejdůležitějším behaviorálním modelem je model následovníka. Ten určuje, jak rychle pojede a jak se bude agent chovat za autem, které jede ve stejném pruhu. Většinou mívá tři stavy, volný, kde jede sám a nikoho nenásleduje, následovník, kdy je někdo před ním, a nouzové brzdění, kdy se snaží vyhnout srážce.

Dalším důležitým modelem je model přizpůsobení rychlosti, který určuje, jak rychle agent pojede na základě rychlostního limitu, stavu vozovky nebo jestli neprojíždí zatáčkou.

Další model, model změny pruhu, je důležitý v simulacích s víceproudými silnicemi. Agent přejede do jiného pruhu, pokud je to potřeba, pokud je k přejíždění ochotný a pokud to je možné.

Podobný předchozímu modelu je předjížděcí model. Pouze model změny

pruhu většinou nestačí, je potřeba i model, který obstará celý manévr.

Posledním zde zmíněným modelem je model zachování odstupu. Silně závisí s modelem změny pruhu a předjížděcím modelem a bývá většinou jejich součástí. Agent musí vědět, že do daného chtěného pruhu může přejet pouze v případě, že za auty před ním a za ním zbyde dostatečně velká mezera pro bezpečnou jízdu. Tato mezera se může měnit na základě situace, například pokud má agent ve svém pruhu překážku nebo potřebuje najet na sjezd, může akceptovat i menší mezeru jako přijatelnou.

2.4.1 Následovník

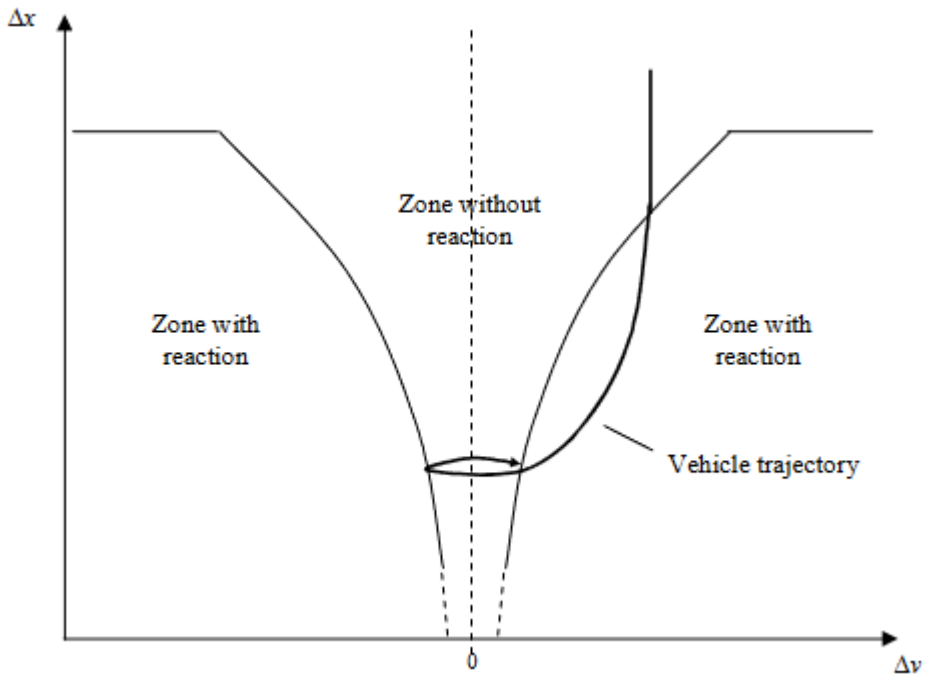
Autonomní vozidlo (agent) je následovníkem, pokud jede před ním vozidlo, které ho svojí rychlostí omezuje. Znamená to tedy, že kdyby agent jel svojí požadovanou rychlostí, vedlo by to ke srážce. Pokud před sebou žádné omezující auto nemá, cestuje svojí požadovanou rychlostí. Jak se bude jako následovník chovat, určuje již daná implementace modelu. Tyto klasifikace se dělají na základě logiky, na které modely pracují.

Nejstudovanější rodinou modelů je Gazis–Herman–Rother (GHR). Ty vidí vztah mezi vůdcem a následovníkem jako momentální akceleraci následovníka, která se počítá z jeho aktuální rychlosti, rychlostním rozdílu oproti sledovanému vozidlu a odstupu mezi vozidly.

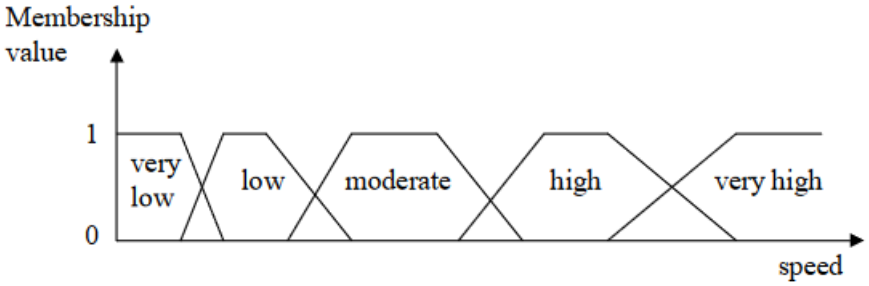
Další skupinou jsou modely držící bezpečnou vzdálenost. Zde je nejznámější model podle Gippse (1981). Ten mezeru mezi vozidly drží takovou, aby při náhlém a rychlém zastavení prvního vozidla stihl agent bezpečně s prodlevou zareagovat a zastavit také.

Psycho-fyzické modely využívají prahy nebo akční body. Pokud je této hranice dosaženo, agent změní své chování a může tím reagovat na změny aktuální dopravní situace kolem sebe. Tyto hranice a reakce, které vyvolávají, se dají zobrazit do grafu, který má na osách rychlost, resp. vzdálenost mezi vozy, viz obrázek 2.1.

Existují ještě modely využívající fuzzy logiku. Člověk o rychlosti a vzdálenostech nemyslí v absolutních číslech, ale pouze relativně. Tomu se snaží přiblížit tyto modely. Využívají k tomu funkce, které absolutní hodnoty namapují na slova, reprezentující fuzzy množiny. Ty se mohou i překrývat. Nespornou výhod fuzzy modelů je jednoduchost vytvářeních podmínek, nevýhodou je ale náročná kalibrace.



Obrázek 2.5: Psycho-fyzický model (Zdroj: Olstam, 2009)



Obrázek 2.6: Fuzzy model - příklad funkce mapující rychlost (Zdroj: Olstam, 2009)

Jak je vidět, následovnických modelů je hned několik. Různým simulačním aplikacím se mohou hodit různé modely a záleží také na lokalitě, pro kterou je aplikace vyvíjena. Indickým uživatelům, kteří jsou na silnicích zvyklí na větší anarchii, se může zdát simulace realistická, zatímco jejich středoevropským kolegům nikoliv. Konfigurace modelu se také může měnit na základě simulované dopravní situace. Řidiči mohou mít v zácpě jinou reakční dobu, než když mají volnou cestu.

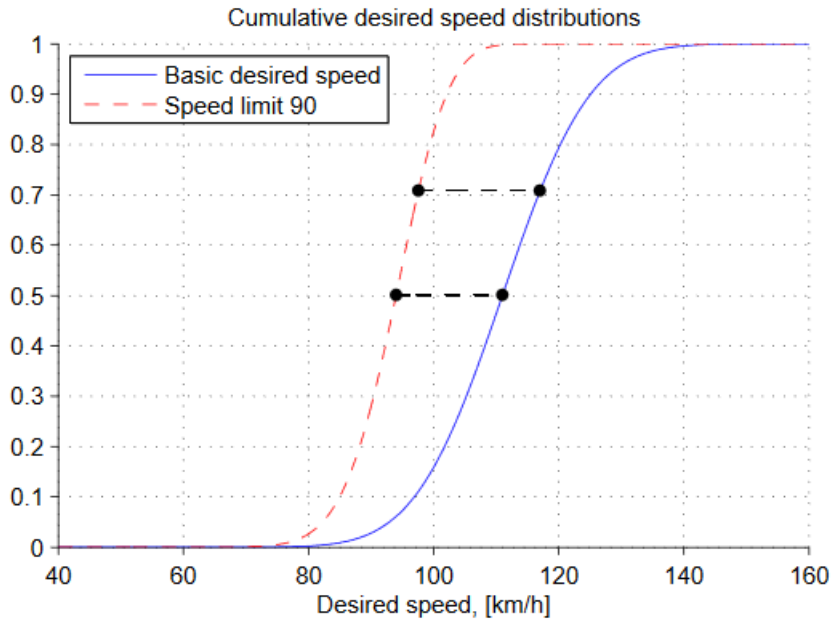
2.4.2 Přizpůsobení rychlosti

Ve většině dopravních simulací má každý agent svojí požadovanou rychlost, která se, stejně jako v reálném světě, mění na momentální situaci. Pokud

jede řidič po dálnici nebo rychlostní silnici, bude jeho požadovaná rychlost pravděpodobně kolem rychlostního limitu. Na venkovských, méně udržovaných silnicích, ale může požadovaná rychlost záviset více na vozovce. K modulaci této požadované rychlosti slouží model přizpůsobení rychlosti.

Pro silnice, kde hlavně záleží na rychlostním limitu, je modelování jednodušší. Jedním možným způsobem je každému řidiči přidělit pro každý rychlostní limit danou požadovanou rychlost. To nám zajistí jednoduchý, ale flexibilní model, který dobře simuluje rozdíly mezi jednotlivými limity pro různé řidiče. Podobným, ale méně flexibilním způsobem, je definování distribuce relativních rychlostí. Požadovanou rychlost poté získáme sečtením rychlostního limitu a dané relativní rychlosti. Jiný způsob násobí rychlostní limit parametrem, který znázorňuje, jak řidič respektuje maximální povolenou rychlost.

Na venkovských cestách bude kromě rychlostního limitu požadovaná rychlost záviset také např. na šířce a stavu vozovky. Každému řidiči je přiřazena základní požadovaná rychlost, která je následně upravena na požadovanou rychlost snížením svého mediánu podle podmodelů (každý podmodel představuje jeden faktor, např. šířku silnice). Distribuční křivka ještě může být otočena okolo svého mediánu, aby postihovala převážně agenty s vyšší požadovanou rychlostí.



Obrázek 2.7: Příklad posunu a rotace křivky distribuční funkce rychlosti [4]

2.4.3 Změna pruhu

Dalším důležitým behaviorálním modelem je model změny pruhu. Když se řidič rozhoduje, jestli změní pruh, musí vzít v potaz několik věcí. Gipps (1986) uvádí 3 otázky: Potřebuje řidič změnit pruh? Chce řidič změnit pruhu?

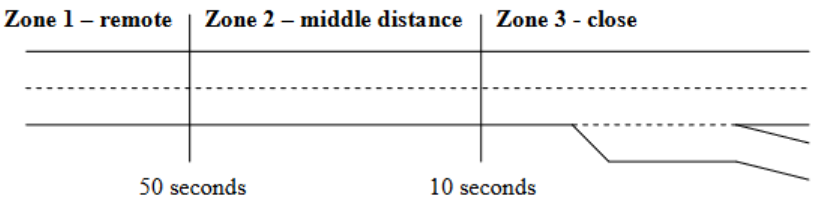
Dovoluje momentální situace řidiči změnit pruh? Právě z Gippsova modelu vychází i většina nových modelů.

V Gippsově modelu je změna pruhu nemožná, pokud není v požadovaném pruhu dostatek místa na zařazení. V situacích, kde ale řidič pruh změnit musí, jelikož je v jeho pruhu překážka, řidiči v jeho požadovaném pruhu vytvoří dostatečně velkou mezeru, aby zaseknutému řidiči pomohli se zařadit.

Jiné řešení navrhl Hidas (2002), který vidí rozdíl mezi nutnou změnou pruhu a výhodnou změnou pruhu. Nutná změna pruhu znamená, že se řidič vyhýbá například překážce. Výhodná změna může být za účelem vyšší rychlosti. Hlavní rozdíl je v toleranci zachování odstupu v cílovém pruhu. Okolní řidiči také můžou řidiče před sebe pustit, pokud je změna pruhu nutná.

Toledo (2005) navrhl model, jež na rozdíl od dřívějších modelů, které vybíraly pouze sousedící pruhy, vybírá ze všech pruhů stejného směru. Tím si řidič vybere pro sebe nejvhodnější pruh, za jehož získáním může nastat několik změn pruhu.

Nutnost roste na základě blížící se překážky, což se dá modelovat například oblastmi (Gipps), kde se chování řidiče mění na základě toho, jak blízko je daná oblast k překážce.



Obrázek 2.8: Nutnost roste na základě blížící se překážky, což se dá modelovat například oblastmi (Gipps).

Řidičova touha po změně pruhu jde modelovat různými způsoby. Jedním způsobem je model následovníka. Agent si vybere takový pruh, kde je nejméně ovlivňována jeho požadovaná rychlost. Jiným způsobem je funkce vyvíjející tlak. Její výstup roste s klesající vzdáleností od překážky nebo pomalejšího auta a rostoucím rychlostním rozdílem mezi vozidlem agenta a překážkou.

2.4.4 Předjíždění

Model pro předjíždění obstarává celý manévr nutný pro předjetí jiného vozu. Celý model se dá rozdělit do několika podmodelů. První je dodržování silničních pravidel. Řidiči by měli dodržovat plné čáry, v reálném světě se toto ale vždy neděje a my můžeme stejné chování nasimulovat v aplikaci. Další podmínkou je dostatečně velká vzdálenost k protijedoucímu autu, aby se agent stihl včas zařadit. Agent také musí jet v dostatečně výkonném autě, aby manévr zvládl. Poslední zmíněnou podmínkou je samotné rozhodnutí řidiče, jestli se mu délka eventuálního předjíždění nezdá příliš dlouhá. Toto rozhodnutí se může lišit řidič od řidiče a stejný řidič může v různých okamžicích jednu mezeru odmítnout a tu samou později přijmout.

Základem předjíždění je přijmutí nabízené mezery. Při studiích (Bottom a Ashworth 1978 a Daganzo 1981) se zjistilo, že řidiči se mezi sebou v akceptování mezery příliš neliší, liší se ale velikost mezery na základě situace pro daného řidiče. Z toho vyplývá, že realitu nejlépe reprezentují modely, ve kterých se řidič o předjížděcím manévru rozhoduje nezávisle na předchozích manévrech. Těmto modelům se říká nekonzistentní.

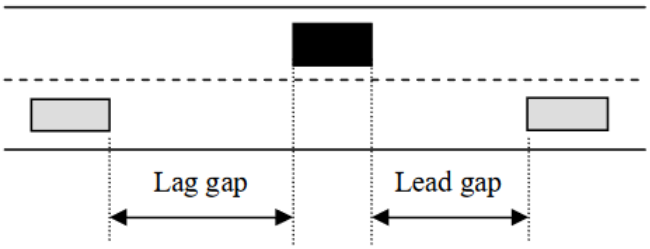
Dalšími faktory v předjíždění mohou být řidičova rychlost, vůz, kterým jede, rychlost předjížděného vozidla, typ předjížděného vozu nebo způsob předjíždění.

2.4.5 Zachování odstupu

Dobrý model zachování odstupu je důležitý jak v modelech změny pruhu, tak v modelech předjíždění. I když je změna pruhu nutná, nemusí být možná, jelikož v požadovaném pruhu jezdí auta a agent nemá možnost se zařadit. Je nutné definovat minimální mezeru, do které je řidič se ochotný zařadit. Tato minimální mezera se ale těžko měří a je u každého řidiče jiná. Navíc může každý řidič v různých okamžicích akceptovat různě velké mezery. Existují modely, které mají definovanou jednu kritickou mezeru pro všechny. Každý řidič má ale různé kritické hodnoty pro mezeru před a za a i pro změnu pruhu doleva nebo doprava.

Jiné modely místo mezer při změně pruhu řeší nutné zpomalení pro bezpečné zařazení, a to jak pro vůz agenta, tak i pro vůz, co je v požadovaném pruhu za agentem.

Ve skutečnosti řidiči s blížící se překážkou akceptují menší mezery, resp. větší nutné zpomalení. V modelech se toto zmenšování/zvětšování definuje lineárně.



Obrázek 2.9: Ilustrace mezer před a za vozidlem při změně pruhu z článku [4].

Kapitola 3

Simulování složitějších dopravních situací v systému VRUT

Tato kapitola vychází převážně z práce [3] a dokumentace VRUTu [2].

3.1 VRUT - Úvod

VRUT (Virtual Reality Universal Toolkit) je modulární software pro vizualizaci a editaci 3D dat, který vznikl ve spolupráci ŠKODA Auto a Katedry počítačové grafiky a interakce ČVUT.

Samotné jádro aplikace obsahuje pouze základní funkčnost, kterou rozšiřují moduly, jako je právě Traffic nebo VehicleSimulator. Důraz je kladen na univerzálnost, kde se uživatel sám rozhodne, které funkcionality potřebuje a které ne. Eventuálně může být VRUT využit jako sandbox, na němž programátor vytvoří řešení svého problému a nemusí se starat např. o implementaci načítání vstupu, rendering a podobně [2].

Aplikace se neustále vyvíjí a i když se dokumentace k ní aktualizuje, informace v ní bývají zastaralé.

VRUT je napsaný v C++, využívá wxWidgets pro uživatelské rozhraní, OpenGL3 pro rendering grafického rozhraní, OpenAL pro rendering zvuku a fyzikální engine ODE. Aby byla zachována modularita, moduly jsou nezávislé na jádru a komunikují s jádrem pomocí událostí.

3.2 Modul Traffic

Původní implementaci a k ní dokumentaci [3] modulu Traffic vytvořil Jaroslav Minařík v rámci práce "Simulace okolních dopravních dějů". Tato implementace byla později upravena a nějaké části, jako například průjezd křižovatkami, byly vynechány. K této nové verzi modulu jsem již ale nenašel žádnou dokumentaci. Pokusím se zde popsat, jak funguje modul Traffic v aktuálním stavu.

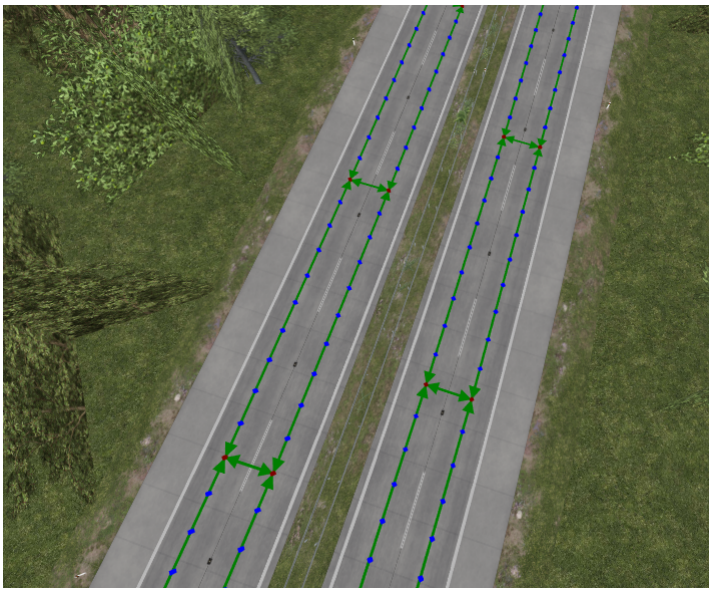
3.2.1 Graf silnic

Traffic je modul starající se o chování řidičů na silnici. Ti se ale nedokážou ve virtuálním světě orientovat, pokud jim nedáme nějaké orientační body. K tomu slouží datová struktura nazvaná "graf silnic". Jeden jízdní pruh je definován uzly spojenými orientovanými hranami. Agent se při jízdě pohybuje od uzlu k uzlu a díky tomu, že jsou hrany orientované, ví, jaký je směr.

Aktuální implementace grafu silnic je velmi ořezaná oproti své původní verzi a podporuje pouze dálnice s výjezdy a nájezdy. Uzly mají v sobě uložené informace o své poloze, šířce vozovky, maximální povolenou rychlost a svůj identifikátor. Identifikátory uzlů v grafu musí tvořit spojitý rozsah od 1 do celkového počtu uzlů.

Uzly se spojují do pruhů. Kvůli přehlednosti se doporučuje mít uzly v jednom pruhu označené po sobě jdoucími a spojitý rozsah tvořícími identifikátory. Pruh má v sobě uložené, jestli je levý nebo pravý, počet uzlů, který ho tvoří, jestli se jedná o dálnice, výjezd nebo nájezd a jaká je na něm úroveň autonomního řízení (rozsah 0-4, kde 0 je žádná automatizace a 4 kompletní).

Pruhy, které vedou stejným směrem, je nutné ještě spojit dohromady, aby bylo možné přejíždění z pruhu do pruhu.

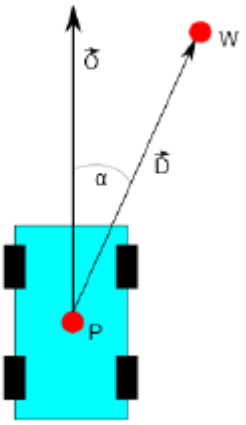


Obrázek 3.1: Graf silnic

3.2.2 Orientace na silnici

Agenti musí vědět, jak nadefinovaný graf používat. Na začátku simulace se každému řidiči nalezne bod grafu, který je k němu nejblíže, a zároveň segment grafu, ve kterém uzel leží, je orientován podobně jako automobil. Tím zařídíme, aby se vozidlo dostalo k co nejbližší silnici a zároveň na ní nenajelo do protisměru. Když vozidlo projede daným bodem, je vybrán bod na něj v pruhu navazující.

Agent ještě musí vědět, jak s vozidlem zatočit. Mějme vektor orientace vozidla $\vec{O}$, bod polohy vozidla P, následující bod cesty W, vektor $\vec{W} = P - W$, který označuje požadovanou orientaci vozidla a úhel α, který označuje odchylku mezi vektory $\vec{O}$ a $\vec{W}$. Agent se bude snažit otáčet volantem tak, aby kola svírala s požadovaným vektorem orientace co nejmenší odchylku a tím vůz dostal správný směr. Jinak řečeno, snaží se minimalizovat úhel α.



Obrázek 3.2: Obrázek znázorňující, jak získat požadovanou orientaci vozidla (Zdroj: Minařík, 2014)

3.2.3 Přizpůsobení rychlosti

Agenti by měli dodržovat maximální povolenou rychlost na silnicích. Jelikož je tato informace uložena v uzlech grafu, řidič si jí jednoduše přečte při projetí uzlem. Porovná jí se svojí aktuální rychlostí; pokud jede pomaleji, přidá hodnotu síly na plynovém pedálu, pokud jede rychleji, hodnotu síly ubere. V případě velkého překročení nastaví hodnotu síly na plynovém pedálu na nulu a naopak přidá na brzdě.

Navíc má každý agent parametr "willingness"(hodnoty od $0.5 - 1$), díky kterého nepojedou všichni řidiči uniformě stejnou rychlostí. Tímto parametrem se vynásobí maximální povolená rychlost a řidič se bude řídit podle této nově získané rychlosti. Pokud je tedy povolená rychlost $100km/h$ a řidič má parametr "willingness"nastavený na 0.8, pojede rychlostí $100 * 0.8 = 80km/h$. Kdyby byla hodnota parametru větší než 1, řidič by jel rychleji, než je nejvyšší povolená rychlost, čehož lze taky využít.

3.2.4 Zachování odstupu

Stejně jako v reálném světě, vozidla by si mezi sebou měla udržovat dostatečný odstup. To zařídíme tím, že si agent kontroluje, jestli není moc blízko vozidlu před sebou. Pokud ano (15-20 metrů) a jeho relativní rychlost k autu před

ním je větší, ubere na plynovém pedálu. V případě velmi malé vzdálenosti (<15 metrů), přibrzdí. Tím se vyhneme případným kolizím.

3.2.5 Předjíždění

Mechanika předjíždění byla upravena a nesouhlasí s tou, která je popsána v původní práci.

Předjíždět se dá pouze levým pruhem, a to pouze za předpokladu, že vede stejným směrem. V případě, že je před agentem pomalejší vozidlo a agent ho chce předjet, podívá se na další uzel v levém sousedním pruhu a nastaví si ho jako následující. Navíc se stav agenta nastaví na "OVERTAKING"a stav předjížděného vozu na "OVERTAKEN". Vozidlo následně přejede do levého pruhu a pokračuje cestu v něm. Jakmile se dostane před předjížděné vozidlo, nastaví se oba stavy zpátky na "NORMAL". Návrat do svého pruhu vozidlo řeší až v tomto stavu a je vysvětlen níže.

3.2.6 Změna pruhu

Agenti mění pruh pouze v případě, že je to potřeba. Do levého pruhu vjíždí pouze v případě předjíždění nebo vyhýbání se překážce. Do pravého pruhu se následně vrací podobným způsobem, jako když předjížděli.

Agent si ve stavu "NORMAL"kontroluje, jestli není napravo od něj existující pravý pruh. Pokud ano (např. poté, co někoho předjel), nastaví si další uzel v pravém pruhu jako svůj následující.

3.2.7 Napojování

Napojování na dálnici je ve VRUTu implementováno zvlášť. Je to z toho důvodu, že se při nájezdu na dálnice kontroluje více věcí, než při obyčejné změně pruhu. Agent si navíc zjišťuje, jestli je vzdálenost od vozidla, které se k němu blíží, dostatečně velká na to, aby se stihl zařadit a zrychlit.

3.3 Modul VehicleSimulator

Modul VehicleSimulator se stará o věrohodné fyzikální chování vozidel, k čemuž využívá knihovnu ODE. Ze získaných vstupních hodnot, které jsou natočení volantu, polohy plynového a brzdového pedálu a vyhodnocené polohy vozidla, transformuje karoserii automobilu a kola a nakopíruje je do scény.

3.3.1 Pacejka model

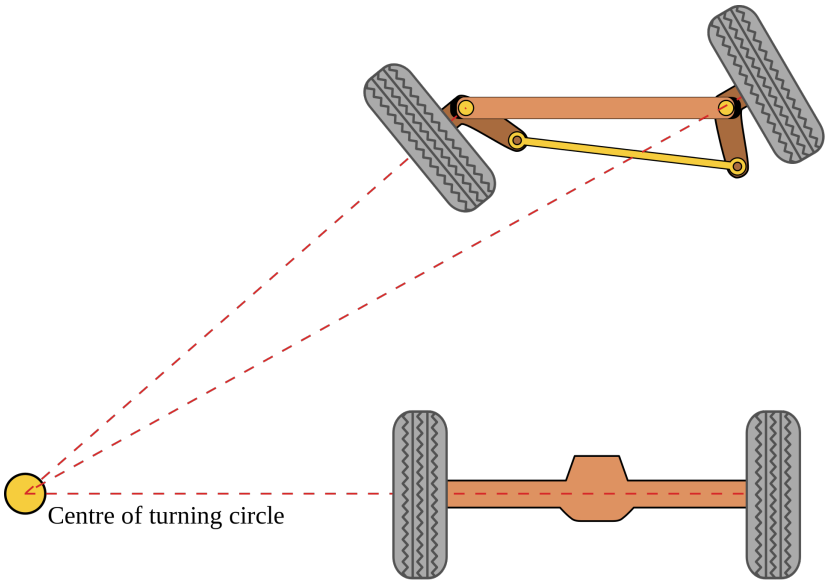
Pacejka model, také známý jako Magický vztah, určuje fyzikální chování pneumatik. Vztahu se říká magický, protože i když není postavený na žádném fyzikálním základě, přesto funguje pro mnoho konstrukcí pneumatik.

Magický vztah přijímá dva typy vstupních parametrů. Jedny jsou statické parametry popisující povrch pneumatiky (a1-a15, b1-b15). Druhý typ jsou

parametry počítané při každém kroku, což je slip angle, slip ratio a zatížení kola. Výstupem vztahu jsou maximální síly F_x a F_y, které může pneumatika vyvinout v podélném a příčném směru, aniž by došlo ke smyku.

3.3.2 Řízení Ackermannovým způsobem

Vozidla se řídí Ackermannovým způsobem. Ten definuje, jak se natáčí kola automobilu. Je pro něj charakteristické, že se kola při zatáčení natočí každé pod jiným úhlem tak, aby nedocházelo ke smyku.



Obrázek 3.3: Ackermannův způsob řízení. Každé kolo se při zatáčení natočí pod trochu jiným úhlem tak, aby nedocházelo ke smyku.

Kapitola 4

Rozbor a návrh řešení

Tato kapitola obsahuje návrh úpravy reprezentace grafu silnic a jak mohou agenti projíždět křižovatkami. Dále navrhuje zjednodušený fyzikální model pro vzdálenější vozidla.

4.1 Rozšíření grafu silnic

Aktuální reprezentace počítá pouze se silnicemi, pruhy a uzly. Křižovatky vůbec nebere v potaz a bez její úpravy vozidlo těžko pozná, že se na křižovatce nachází, natož jak se na ní má chovat.

Nabízí se inspirace formátem OpenDRIVE® nastíněném v sekci 2.2, tedy oddělit reprezentaci křižovatek od silnic. Zjednoduší se tím i uživatelská orientace v datech. Křižovatky se skládají z průjezdů, jejichž reprezentace je podobná pruhům, ze kterých se skládá silnice. Průjezdy jsou reprezentovány jako posloupnost uzlů a hran. Průjezd také drží informaci, který silniční pruh jaké silnice do něj ústí a do kterého pruhu následuje.

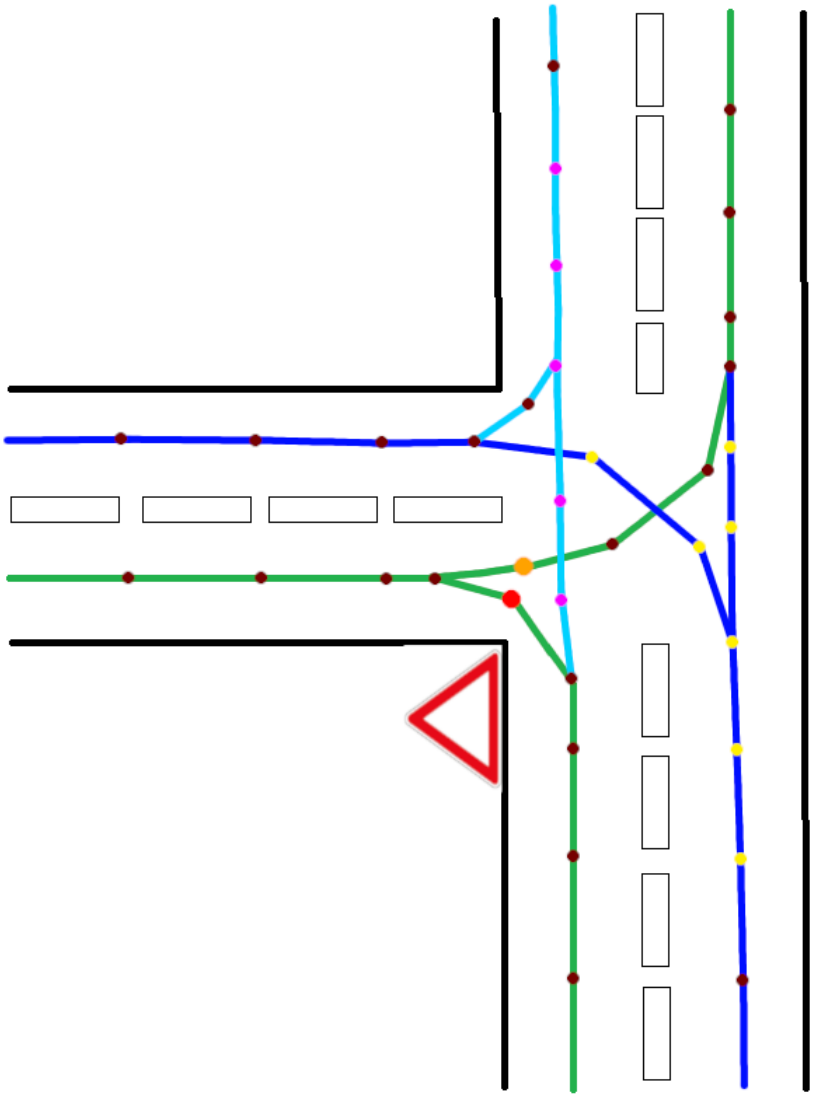
4.2 Křižovatky

Hlavním problémem při projíždění křižovatek je, komu má dát řidič přednost. I v reálném světě má s tímto mnoho lidí problém. Vozidlo tedy musí poznat, že je na křižovatce, a určit, koho na křižovatce pustit a koho ne. Co se týče předností, existují dva typy křižovatek. Ty, kde není určena přednost a řidič dává přednost zprava, a ty se značením o přednostech, kde se řidič řídí značkami.

Na křižovatkách bez značení platí pravidlo přednosti zprava. V případě odbočení doleva platí také přednost protijedoucímu. Jelikož se v České republice jezdí vpravo, jsou při odbočení vlevo protijedoucí vozidla také vpravo a odbočující vozidlo kříží jejich dráhu. Při odbočení doprava vozidlo nikomu přednost nedává.

Křižovatky se značením fungují na podobném principu jako ty bez značení. Rozdíl je ten, že ze dvou křížících se silnic je jedna hlavní a druhá vedlejší, což musí agent vzít v potaz.

Jelikož ale agenti ve VRUTu pracují zatím hlavně s grafem silnic a značky nevidí, řešení křižovatky na typy nerozděluje. Potřebné informace může získat právě z hierarchické reprezentace grafu silnic. Stačí uzlu, do kterého vozidlo vjíždí, nadefinovat jiné uzly, popřípadě pruhy, kterým má vozidlo dávat přednost. Jelikož mají uzly v sobě uloženou referenci na své sousedy, lze do této množiny přednostních uzlů přidat i uzly v blízkém okolí definovaného uzlu. Získáme tím referenci na oblast těsně před křižovatkou a zároveň i uvnitř ní, aniž bychom museli tyto uzly explicitně definovat. Informace, zda některým uzlem z této množiny zrovna neprojíždí nějaké vozidlo, je v simulaci reprezentována také. Uzly, před kterými má vozidlo přednost, nemusíme řešit. Vůz je nutné zastavit pouze v případě, že některým z přednostních uzlů projíždí jiné vozidlo.



Obrázek 4.1: Obrázek znázorňující schéma křižovatky s přednostmi. Vozidlo přijíždí zleva po zelené části grafu. Pokud se vydá po větvi začínající červeným bodem, bude dávat přednost pouze vozidlům projíždějícím růžovými body na světlé modré časti grafu. Pokud bude dále pokračovat po větvi začínajícím oranžovým bodem, bude dávat přednost jak vozidlům na růžových bodech, tak vozidlům na žlutých bodech v tmavě modré části grafu.

4.3 Fyzikální model

Aktuální řešení implementace se potýká s výkonnostními problémy. Momentální fyzikální model využívající Pacejka model a knihovnu ODE je velmi náročný a nehodí se při simulaci více vozů najednou. Pro vozy vzdálenější od kamery by byl vhodnější jednoduší model.

Prvním navrženým zjednodušením je úprava modulu VehicleSimulator. Knihovna ODE, počítání Pacejkova magického vztahu a řízení Ackermannovým způsobem je pro vzdálenější vozidla zbytečně náročné a může se nahradit jednoduchou kinematikou. Celé vozidlo můžeme brát jako těleso pohybující se po podložce s nulovým odporem, které má pouze směr a rychlost. K tomu musíme ještě řešit orientaci, aby nám vozidla nejezdila bokem. To můžeme vyřešit tím, že čelní část automobilu bude vždy mířit tam, kam vozidlo jede.

Druhým možným zjednodušením je nastavení konstantní rychlosti všem agentům, například na maximální povolenou rychlost. Tím se vyhneme nutnosti hlídat odstupy a předjíždění.

4.4 Mezoskopický model

Pro simulaci vozů, které jsou od řízeného tak daleko, že už nejsou ani vidět, lze upustit od mikroskopického modelu úplně a použít model mezoskopický.

Vzhledem k aktuální reprezentaci grafu silnic se nabízí podobné řešení, jaké využívá model Mezzo, který popisuji v sekci 2.3.1. Graf je hierarchicky rozdělen na silnice a křižovatky, což se může použít při mezoskopické simulaci. Na části grafu reprezentovaném silnicemi lze vozidla simulovat jako jednotky datového toku pohybující se rychlostí určené z hustoty provozu na dané silnici. Pro spočtení potřebujeme znát délku dané silnice. Tato informace ale u silnic není explicitně zadaná. My ji můžeme buď do definice přidat, nebo ji lze spočítat. To by šlo vydělením počtu uzlů v celé silnici počtem pruhů. Tato relativní délka ale může být velmi nepřesná, jelikož ne všechny pruhy vedou po celé délce silnice. Další možnost je například sečíst vzdálenosti mezi jednotlivými sousedními body od prvního do posledního v pruhu, který vede od jedné křižovatky ke druhé. Tato délka sice též nemusí být úplně přesná, k výpočtu hustoty by ale měla být dostačující.

Pro simulaci křižovatek jako fronty vozidel čekajících na odbočení se nám hodí nadefinované průjezdy. Každý průjezd může mít vlastní server. Některé ale budou společné, protože průjezdy mohou mít společný svůj počáteční uzel. Každý server bude mít nadefinovaný ještě vlastní look-back limit, pokud se pruhy před křižovatkou rozdělují. Look-back limit se může nadefinovat stejně jako v Mezzo, tedy na počet vozidel, nebo na délku, aby byl přesnější. Dva kamiony zabírají na silnice více místa, než dvě osobní vozidla.

Kapitola 5
Implementace

Následující kapitola popisuje úpravy modulu Traffic.

5.1 XML reprezentace grafu silnic

Graf silnic je datová struktura, pomocí které se navigují autonomní vozidla po silnici. Tato struktura je uložená v XML souboru, který je nutné ručně upravovat.

5.1.1 Původní implementace

Původní implementace pracovala se dvěma hlavními elementy: silnice (*road*) a spojení (*connections*).

Silnice (*road*) představuje nejvyšší vrstvu hierarchie dat. Reprezentuje jednu silnici a jejími elementy jsou pruhy (*lane*). Má dva atributy: svůj globální identifikátor (*id*) a typ komunikace (*type*). Ten nabývá hodnot od 0 do 4, kde 0 je dálnice a 4 okresní cesta.

Pruh (*lane*) sdružuje skupinu uzlů (*node*). Je nutné ho definovat se třemi povinnými atributy: počet uzlů v pruhu (*nodes*), typ (*type*), kde 0 je normální jízdní pruh, 1 značí sjezd z dálnice a 2 nájezd, a identifikátor (*id*), který ale jen určuje pořadí pruhu od krajnice, například 0 pro pravý pruh, 1 pro levý, -1 pro sjezd nebo nájezd. Tato hodnota tedy není jednoznačný identifikátor. Dále má pruh dva nepovinné atributy: úroveň povoleného autonomního řízení (*level*) nabývající hodnoty od 0 (pro žádnou automatizaci) až do 5 (plná automatizace) s výchozí hodnotou 3 a povolení přejetí do vedlejšího pruhu (*o*), kde je výchozí hodnota 0, znamenající zákaz změny pruhu, a další možné hodnoty jsou: 1 - povolená změna doleva, 2 - změna doprava, 3 - změna na obě strany.

Základním elementem grafu silnic je uzel (*node*). Reprezentuje jeden uzel grafu a vzhledem k tomu, jak se VRUT rozrůstal a simulace se stávala náročnější a přesnější, má hned několik povinných i nepovinných atributů. Těmi povinnými jsou: globální identifikátor pro celý graf (*id*), pozice v 3D prostoru (x, y, z), maximální povolená rychlost (*sl*) a šířka pruhu (w). Nepovinnými atributy jsou: jméno (*name*), úroveň automatizace (*level*), která funguje stejně jako stejnojmenný atribut v pruhu a získává se z něho i výchozí

hodnota, povolení změny pruhu (o), kde to funguje totožně, dva atributy pro poloměr zatáčky (r, $r2$), rychlostní limit ($vmax$), rychlostní limit pro průjezd zatáčkou ($vdop$) a 5 atributů pro rychlostní profily ($vprof0$, $vprof1$, ..., $vprof4$).

```
<road id="5" type="2">
  <lane id="0" nodes="22" type="1" level="4" o="0">
    <node id="1" x="147" y="69" z="0.7" o="0" sl="30" w="3.9" />
    ...
  </lane>
  ...
</road>
...
```

Druhý hlavní element je spojení (*connections*). Sdružuje pod sebe všechny nadefinované spojení mezi uzly, tedy elementy sekvence (*sequence*), skupiny spojení (*connectiongroup*) a spojení pruhů (*interconnection*). Existují čtyři typy spojení mezi uzly, a to dopředné, zpětné, levé a pravé. Dopředné pro zjištění následujícího uzlu na cestě, zpětné převážně pro výpočet křivky průjezdu grafem a udržení informace o topologii. Levá a pravá spojení se používají například pro změnu pruhu.

Sekvence představuje spojení mezi několika po sobě jdoucími uzly. Má 4 atributy: ID prvního uzlu ($from$), ID posledního uzlu (to), směr spojení (dir) a uzavření do smyčky ($closed$). Tento element vytvoří sekvenci spojení uzlů $from \leftrightarrow from + 1$, $from + 1 \leftrightarrow from + 2$, ..., $to - 1 \leftrightarrow to$. Pokud je atribut $closed$ nastavený na 1, vytvoří se i přímé spojení $to \leftrightarrow from$. Směr (dir) definuje typ spojení, jaká se budou mezi uzly vytvářet. Tento atribut může nabývat čtyř hodnot, od 0 do 3, v praxi se ale používají pouze hodnoty 2 a 3. Pokud bude mít směr spojení hodnotu 2, vytvoří se sekvence dopředných spojení $from \rightarrow from + 1$, $from + 1 \rightarrow from + 2$, ..., $to - 1 \rightarrow to$ a sekvence zpětných spojení $to \rightarrow to - 1$, $to - 1 \rightarrow to - 2$, ..., $from + 1 \rightarrow from$. V případě nastavení atributu na hodnotu 3 se tato spojení vytvoří v opačné orientaci, tedy $from \rightarrow to$ budou zpětná a $to \rightarrow from$ budou dopředná. Možnost mít sekvenci pouze dopředných nebo zpětných spojení není podporována.

Spojení pruhů (*interconnection*) je element podobný svojí funkcí elementu sekvence, vytváří ale mezi uzly levá a pravá spojení. Jeho atributy jsou ID počátečního uzlu prvního pruhu ($from$), ID počátečního uzlu druhého pruhu (to), počet spojení určených k vytvoření ($count$) a směr tvorby spojení (dir). Tento poslední atribut může mít 4 hodnoty. Pokud se rovná 0, vytváří se levá spojení $from \rightarrow to$, $from + 1 \rightarrow to + 1$, ..., $from + count - 1 \rightarrow to + count - 1$. Pro hodnotu 1 jsou tato spojení pravá. V případě, že je atribut nadefinovaný na 2, vytvoří se oboustranná spojení, kde ta ve směru $from \rightarrow to$ jsou levá a ve směru $to \rightarrow from$ pravá. Nastavíme-li atribut na 3, spojení se vytvoří obráceně ($from \rightarrow to$ pravá, $to \rightarrow from$ levá).

Element skupiny spojení (*connectiongroup*) je jednoduchý element používaný ke sdružení spojení stejného typu, definovaného atributem *type*. Ten může nabývat hodnoty 0 pro dopředná spojení, 1 pro zpětná, 2 pro spojení

do uzlu v levém pruhu a 3 pro spojení s uzlem v pravém pruhu. Samotná spojení jsou reprezentovaná elementem *connection* s atributy ID počátečního uzlu *from*, ID koncového uzlu *to* a nepovinným *type*. Poslední jmenovaný je stejný jako atribut *type* v *connectiongroup*. Pokud se nevyužije, pro spojení se použije jako výchozí hodnota právě ta definovaná výše.

```
<connections>
  <sequence from="12" to="26" dir="2" closed="0" />
  ...
  <interconnection from="12" to="27" count="14" dir="2" />
    ...
  <connectiongroup type="0">
    <connection from="26" to="42" type="0" />
    ...
  </connectiongroup>
  ...
</connections>
...
```

5.1.2 Identifikátory

Při editaci XML dat byly ve staré implementaci velkým problémem globální identifikátory uzlů. Vzhledem k tomu, že ID v jednom pruhu musely tvořit spojitý rozsah, tak pokud chtěl uživatel například přidat jeden uzel, musel následně ručně upravovat všechny identifikátory v celém grafu, které měly vyšší číslo, než ten přidaný. Z toho důvodu jsem upravil reprezentaci tak, aby byly identifikátory uzlů v pruhu pouze lokální. Globální identifikátory se přiřazují až následně automaticky během načítání dat do VRUTu.

Další identifikátory činící problémy byly ty u pruhů, jelikož se vůbec nejednalo o identifikátory, přestože se tak nazývaly. Tento problém jsem vyřešil přidáním atributu *lid* do elementu *lane* a lokální identifikátory ukládal do něj.

Důsledek těchto úprav se následně projevil u elementů řešících spojení uzlů, kde k identifikaci uzlů již nestačilo pouze jejich ID, ale bylo potřeba zde brát v potaz i ID silnice a pruhu. Následující příklad ukazuje elementy *sequence*, *interconnection* a *connection* po úpravách.

```
<sequence road="5" lane="0" from="0" to="21" dir="2" closed="0"/>
...
<interconnection road="5" fromlane="0" tolane="1" from="0" to="0"
                                                  count="22" dir="2" />
...
<connection fromroad="1" fromlane="0" from="24" toroad="3"
                                       tolane="0" to="0" type="0"/>
...
```

5.1.3 Křižovatky

Pro reprezentaci křižovatek byl do původní implementace přidán nový hlavní element *junction* s atributy globálního identifikátoru (*id*), typu (*type*), úrovně povoleného autonomního řízení (*level*) a nepovinného jména (*name*). Hodnoty typu vychází z hodnot, které může nabývat atribut typ u silnice. Element křižovatka pod sebe spojuje další elementy, kterými jsou průjezd křižovatkou (*path*), napojení na silnice (*lanelink*) a přednosti (*priorities*).

Element průjezdu křižovatkou (*path*) je podobný elementu pruhu (*lane*) a stejně jako on sdružuje skupinu uzlů (*node*). Nese tři atributy: Svůj identifikátor v rámci křižovatky (*id*), počet uzlů (*nodes*), úroveň povoleného autonomního řízení (*level*) s výchozí hodnotou v křižovatce a typ (*type*), který má hodnotu 0 pro směr rovně, 1 pro odbočení doleva a 2 pro odbočení doprava.

Napojení na silnici (*lanelink*) je jednoduchý element s jedním atributem (*path*) reprezentujícím identifikátor napojovaného průjezdu křižovatkou. Jeho potomky jsou elementy předchůdce (*predecessor*) a následovníka (*successor*).

Předchůdce (*predecessor*) představuje napojení průjezdu se silnicí, která míří do křižovatky. K tomu potřebuje atributy k identifikaci silnice (*road*), pruhu (*lane*) a uzlu (*node*), kterému vytvoří oboustranné spojení s prvním uzlem průjezdu.

Následovník (*successor*) funguje podobně jako předchůdce. Spojuje poslední uzel průjezdu s uzlem silnice vyjíždějící z křižovatky definovaným atributy *road*, *lane* a *node*.

Element předností (*priorities*) je rodičem skupiny elementů *priority*, které dohromady, společně se svým potomkem *check*, reprezentují všechny přednosti v křižovatce.

Přednost (*priority*) shromažďuje všechny uzly, které mají před daným průjezdem přednost. Má atributy *path*, jenž představuje identifikátor průjezdu, a *node*, což je identifikátor uzlu, na kterém má vozidlo počkat v případě dávání přednosti. Přednostní uzly jsou reprezentovány jako seznam potomků *check*.

Posledním zde diskutovaným elementem je *check*. Představuje uzel, který má agent zkontrolovat pro případná blížící se vozidla. Atributy elementu jsou *path*, což je identifikátor průjezdu, ve kterém přednostní uzel leží, a *node*, což je identifikátor daného uzlu.

Pro plnou podporu křižovatek v grafu silnic bylo ještě potřeba upravit element *sequence*, aby uměl kromě spojování sekvencí uzlů v pruzích spojovat i uzly v průjezdech.

```
<sequence junction="0" path="0" from="0" to="4" dir="2"
                                                closed="0" />
```

V následující ukázce reprezentace křižovatky je u předností definováno, že vozidlo na průjezdu 0 bude čekat před uzlem 2, pokud se k uzlu 0 na průjezdu 3, respektive k uzlu 1 na průjezdu 4, blíží vozidlo.

```
<junction id="0" type="2" level="4">
  <path id="0" nodes="6" type="2">
    <node id="0" x="320" y="10" z="0.7" o="0" sl="30" w="3.9" />
    ...
  </path>
  ...
  <lanelink path="0">
    <predecessor road="5" lane="0" node="21" />
    <successor road="1" lane="0" node="0" />
  </lanelink>
  ...
  <priorities>
    <priority path="0" node="2" >
      <check path="3" node="0" />
      <check path="4" node="1" />
    </priority>
    ...
  </priorities>
  ...
</junction>
...
```

5.2 Úprava tříd grafu silnic

Hierarchická reprezentace silnic se promítla i do tříd v kódu. Vedle úpravy staré třídy *RoadGraphNode* byly přidány třídy *RoadGraphRoad*, *RoadGraphLane*, *RoadGraphJunction* a *RoadGraphPath*. Úpravám se nevyhnula ani třída *AICar*.

5.2.1 RoadGraphNode

RoadGraphNode je základní stavební jednotka grafu silnic. Jedná se o třídu reprezentující jeden uzel grafu. K informacím, které tato třída nesla, bylo nutno přidat další pro uložení dat o křižovatce. Jedná se o boolean určující, zda je uzel v křižovatce, identifikátor křižovatky, typ křižovatky, identifikátor průjezdu, typ průjezdu a lokální identifikátor uzlu v rámci křižovatky. Dále bylo potřeba přidat seznam navazujících uzlů, jelikož stará implementace podporovala pouze jeden navazující uzel, což na křižovatce nestačí.

5.2.2 RoadGraphRoad

Tato třída reprezentuje nejvyšší vrstvu hierarchie grafu, tedy silnici. Je v ní uložený identifikátor silnice a její typ. Též nese seznam pruhů, které jsou součástí dané silnice.

5.2.3 RoadGraphLane

Jedná se o jednoduchou třídu určenou pro reprezentaci pruhu. Informace v ní uložené jsou identifikátor pruhu, pořadí pruhu (staré ID), typ pruhu, úroveň povoleného autonomního řízení, maximální povolená rychlost a informace, zda lze v pruhu předjíždět. Pruh také nese seznam uzlů, které jsou v něm obsaženy.

5.2.4 RoadGraphJunction

Pro reprezentaci křižovatky se používá jednoduchá třída, kde je uložen identifikátor křižovatky, její typ a seznam průjezdů.

5.2.5 RoadGraphPath

Poslední zde zmíněná třída je *RoadGraphPath*, která reprezentuje jeden průjezd křižovatkou. Je v ní uložený identifikátor průjezdu, jeho typ, úroveň autonomního řízení, rychlostní limit a seznam uzlů, které v průjezdu leží. Dále je v něm také uložený seznam přednostních uzlů a identifikátor uzlu, na kterém má vozidlo v případě dávání přednosti čekat.

5.3 Chování vozidel

Jedno vozidlo je ve VRUTu reprezentováno třídou $AICar$. Její hlavní funkce je $DoJob$, která se stará o rozhodování vozidla. V této funkci se také přepíná stavový automat, který definuje chování vozidla v každém konkrétním stavu. Tyto stavy jsou například $NORMAL$, kdy se vozidlo snaží pohybovat po grafu od uzlu k uzlu a dodržovat rychlost, $WAIT$, což je stav, kdy se vozidlo snaží napojit na silnici a kontroluje, jestli je pruh volný, nebo $OVERTAKING$, kdy vozidlo předjíždí jiné vozidlo a kontroluje si, jestli se může vrátit do svého pruhu.

V rámci podpory křižovatek byl upraven stav $NORMAL$ a přidán stav $GIVEWAY$.

Do stavu $NORMAL$ byla přidána kontrola, zda vozidlo nenajíždí do křižovatky. Tuto informaci lze získat z aktuálního uzlu. Pokud vozidlo opravdu vjelo do křižovatky, zjistí si, na jakém je průjezdu, které uzly mají přednost a kde má počkat. Vozidlo se přepne do stavu $GIVEWAY$ v případě, že aspoň na jednom přednostním uzlu jede jiné vozidlo. Přednostní uzly jsou definované v aktuálním průjezdu a jsou za ně považovány i sousedi přednostních uzlů do vzdálenosti dvou hran.

Ve stavu $GIVEWAY$ se nastaví rychlost vozidla na 0 a kontrolují se přednosti, stejně jako je tomu ve stavu $NORMAL$. Pokud ani na jednom z přednostních uzlu není žádné vozidlo, automobil se přepne zpátky do stavu $NORMAL$ a pokračuje ve své cestě.

Jelikož byla do grafu silnic přidána možnost více vstupních a výstupních cest u jednoho uzlu, musela se vozidlům přidělit funkce rozhodnutí se, kam

pokračovat. To se vyřešilo podobně, jako je tomu u sjíždění agentů z dálnice. Vozidlo se při příjezdu do uzlu, ze kterého je více možných cest, vybere náhodně jeden z následujících uzlů a pokračuje po něm dále svoji cestu.

Tím vyvstal další problém, a to výpočet křivky průjezdu grafem. Pro něj se využívalo statických informací z uzlů, kdy každý uzel znal svého přímého předchůdce a následovníka. Jelikož je ale nově možné mít více předchůdců nebo následovníků, musí si vozidla pamatovat, kterými uzly projela, a tato informace se nově pro výpočet využívá. Vozidla si pamatují dva uzly nazpátek, dohromady tedy mají v jeden okamžik informaci o čtyřech uzlech, to jest dva dozadu, jeden, ke kterému se blíží, a jeho následovník.

Kapitola 6
Testování a výsledky

K otestování a demonstraci implementace byl vytvořen kompletní graf silnic na mapě *krizovatka*. XML soubor s daty má přibližně 1500 řádků. Testování bylo vykonáno na stolním počítači s procesorem Intel Core i5-7600K s frekvencí 3.80GHz, 32GB RAM a grafickou kartou NVIDIA GeForce GTX 1070. VRUT byl zkompilován v programu Visual Studio 2017 na operačním systému Microsoft Windows 10. Testy byly zaměřeny na schopnost vozidel projet křižovatkou a náročnost simulace na základě počtu simulovaných vozidel na křižovatce.

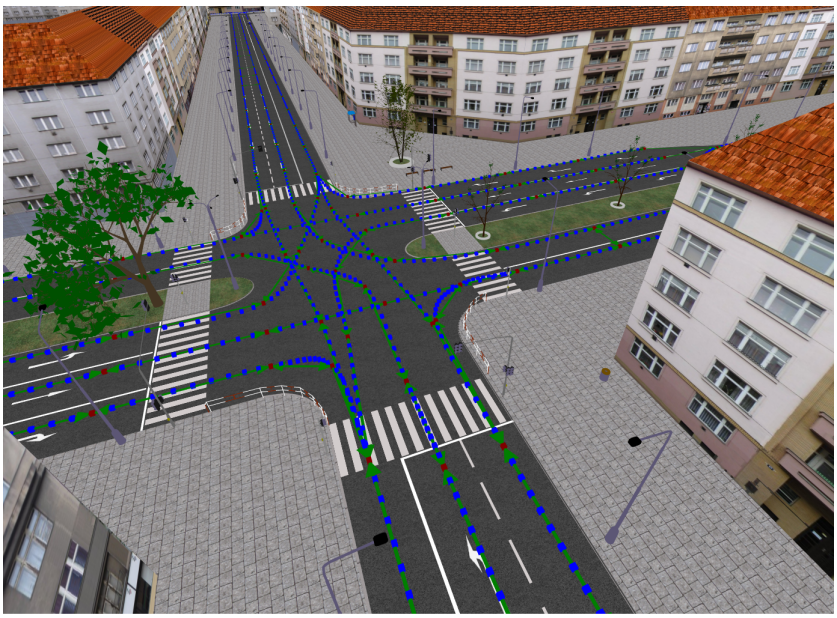
6.1 Křižovatky

Mapa *krizovatka* obsahuje velkou komplexní křižovatku uprostřed své trati. Na ní proběhla zatěžkávací zkouška implementace průjezdu křižovatkami. Pokud jsou správně nadefinované přednosti a pozice uzlů jsou nastavené tak, aby vozy nemusely moc ostře zatáčet, jsou vozidla schopná bezpečně křižovatku projet. Jelikož má i uživatelem řízené vozidlo přiřazený bod pro lokalizaci, autonomní vozidla dokáží brát v potaz i jej.

Problém nastává na určitých uzlech, kde v případě, že vozidlo zastaví, nedokáže se již znovu rozjet. Tento problém souvisí s výpočtem pohodlné rychlosti v křivce a částečně se vyřešil manuálním vyplněním této hodnoty v daném uzlu. Problém též může nastat, pokud jsou dva průjezdy křižovatkou vycházející ze stejného uzlu příliš blízko u sebe. Poněvadž berou vozidla informace o okolním provozu pouze z grafu silnic, auto na vedlejším průjezdu nevidí a mohou do něj narazit. To lze vyřešit definováním průjezdů v dostatečné vzdálenosti, aby se na ně vedle sebe vešly dvě vozidla.

6.2 Náročnost simulace

Co se týče testování hardwarové náročnosti implementace, počet snímků za sekundu je nepřímo úměrný počtu simulovaných vozidel. Nejvyšší náročnost nastává v uzlu, který má nastavenou kontrolu předností, tedy těsně před křižovatkou. Při velmi nízkém počtu snímků simulace nestíhá a přestává fungovat správně. Tabulka 6.1 ukazuje závislost počtu snímků za sekundu na parametru



Obrázek 6.1: Detail mapy *krizovatka*. Na této křižovatce probíhalo testování přednostní.

odeTimeStep a počtu simulovaných vozidel. Parametr *odeTimeStep* určuje velikost kroku simulace v milisekundách a ovlivňuje tím přesnost výpočtu simulace.

	odeTimeStep		
Počet vozidel	2	2.5	3
2	220	230	240
3	215	220	225
5	185	190	195
7	170	180	185
10	155	170	175
11	140	160	175
12	120	150	165
13	30	140	160
14	10	120	155
15	10	75	150
20	7	8	15

Tabulka 6.1: Tabulka ukazující závislost počtu snímků za sekundu na parametru *odeTimeStep* a počtu simulovaných vozidel.

Kapitola 7

Závěr

Cílem projektu bylo nastudovat metody používané v dopravních situacích se zaměřením na simulaci chování řidičů, navrhnutí řešení vylepšení modulů Traffic a VehicleSimulator a implementace a otestování tohoto řešení.

Podařilo se vytvořit funkční řešení autonomního průjezdu křižovatkami. Vozidla si navzájem dávají přednost a umí projíždět i uzly s více možnými následovníky. Za tímto účelem byla předělána reprezentace grafu silnic tak, aby podporovala křižovatky, byla přehlednější a lépe se editovala. K demonstraci byl vytvořen graf silnic na mapě *krizovatka*.

Na vytváření VRUTu se v průběhu času vystřídalo mnoho programátorů a délka zdrojového kódu tak narostla do ohromných rozměrů. To, ve spojení se složitostí vytváření testovacích dat, mělo za následek, že se nestihla vytvořit implementace zjednodušené fyzikální simulace v modulu VehicleSimulator. Zůstala tedy pouze ve stádiu návrhu.

7.1 Plány do budoucna

Traffic je velmi komplexní modul a stále je zde mnoho funkcí, které do něho lze přidat.

7.1.1 Semafory

Implementace vůbec nepočítá se světelnými křižovatkami. Jejich zakomponování do simulace může být další krok, jak aplikaci vylepšit. Kromě normálních předností daných značkami by poté musela vozidla řešit i zda jim svítí na semaforu zelená.

7.1.2 Editor grafu silnic

Jestli je ale něco, co by velmi zjednodušilo práci, je to interaktivní editor grafu silnic. Ruční editace XML dat je časově i psychicky náročná a při jakékoliv změně je nutné restartovat celou aplikaci. Editor by mohl tyto problémy vyřešit a ušetřil by tak dalším programátorům, kteří budou na modulu pracovat, mnoho času a psychického vypětí.

Literatura

[1] Paden, B., Čáp, M., Yong, S. Z., Yershov, D. & Frazzoli, E. *A survey of motion planning and control techniques for self-driving urban vehicles.* IEEE Transactions on intelligent vehicles, 1(1), 33-55, 2016.

[2] *Dokumentace aplikace VRUT.* ŠKODA Auto a.s., ČVUT FEL, 2017

[3] Jaroslav Minařík. *Simulace okolních dopravních dějů.* Diplomová práce ČVUT FEL, 2014.

[4] Olstam, J. *Simulation of Surrounding Vehicles in Driving Simulators* Linköping University, 2009

[5] Olstam, J., Lundgren, J., Adlers, M. & Matstoms, P. *A Framework for Simulation of Surrounding Vehicles in Driving Simulators* ACM, 2008

[6] Krajzewicz, D., Hertkorn, G., Wagner, P. & Rössel C. *SUMO (Simulation of Urban MObility)* German Aerospace Centre, Centre for Applied Informatics (ZAIK), 2002

[7] Lauinger, J. H. *Development of Traffic Simulation in a Game Environment* Technische Universität Darmstadt, 2016

[8] *OpenDRIVE® - Format Specification, Rev. 1.5* VIRES Simulationstechnologie GmbH, 2019

[9] *OpenStreetMap* https://www.openstreetmap.org/

[10] Burghout, W. *Hybrid microscopic-mesoscopic traffic simulation* Royal Institute of Technology, 2004