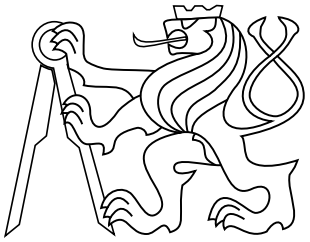
diplomová práce

Simulace okolních dopravních dějů

Bc. Jaroslav Minařík



Květen 2014

Ing. Jiří Bittner Ph.D.

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra počítačové grafiky

Poděkování

Děkuji panu Ing. Jiřímu Bittnerovi, Ph.D. a Mgr. Antonínu Míškovi, Ph.D za podporu při psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Abstrakt

Tato práce prezentuje chování autonomně řízených vozidel. Diskutuje spojení simulace okolního provozu s jízdním simulátorem ve virtuálním prostředí.

Cílem této práce je navrhnout modul simulace silničního provozu pro projekt VRUT, vznikající ve spolupráci katedry počítačové grafiky na fakultě elektrotechnické ČVUT. VRUT neboli "Virtual Reality Universal Toolkit" je schopen simulovat jízdu virtuálního vozidla, ovládaného přes vstupní zařízení člověkem, a vizualizovat ji na výstupním zařízení.

Modul který má vzniknout touto prací má za úkol simulovat okolní dopravní provoz a zajistit, aby se okolní vozidla chovala co nejvěrohodněji. Auta by měla zvládat jednoduchou jízdu po silnici, dodržování silničních pravidel, vyhýbání se kolizím, předjíždět, měnit jízdní pruhy a být schopna dodržet přednost na křižovatkách.

Klíčová slova

Vozidla; simulace; řízení; autonomní;

Abstrakt

This text presents steering behaviors that control autonomous vehicles populating roadways in virtual environments.

The goal of this work is to design and implement an autonomous vehicle simulation module for the Virtual Reality Universal Toolkit project. VRUT is a project being developed in cooperation with CTU and Škoda Auto. It is capable of simulation and visualization of a vehicle driven by user.

The module this work is discussing is supposed to autonomously control surrounding vehicles to simulate traffic driving. These vehicles are expected to behave as naturally as possible. They should be capable of keeping on track, following road rules and crossroad priorities, avoiding collisions, overtaking and switching lanes.

Keywords

Vehicle; simulation; driving; autonomous;

Obsah

Zkratky

Preliminary text...

VRUT	Virtual Reality Universal Toolkit
SW	Software
HW	Hardware
GUI	Graphical user interface, česky grafické rozhraní
XML	Extensible markup language, česky rozšiřitelný značkovací jazyk

1. Úvod

Autonomní řízení vozidel je v posledním desetiletí velmi často diskutovanou tématikou a zaobírá se jí mnoho společností nejen z herního průmyslu. Znalost o tom jak probíhá interakce mezi řidičem, vozem a dopravním systémem je nezbytná, zejména v moderní době, kdy množství řidičů a vozidel na silnicích roste. S nimi rostou i nároky na dopravní systémy, jejich kvality a spolehlivost.

Abychom získali povědomí o tom, jak tyto systémy fungují a jak ovlivňují řidiče, provádějí výzkumníci studie chování a expermenty, které mohou být uskutečňovány v reálných situacích, na testovacích tratích, nebo v jizdních simulátorech. Nejvěrohodnějším prostředím je samozřejmě reálná situace, která ovšem může být nepředvídatelná z hlediska například počasí, stavu silnice a dopravní situace. Nemožnost kontrolovat danou situaci limituje při návrhu experimentu v reálném prostředí s ekvivalentními podmínkami pro všechny subjekty testování. Testovací tratě nabízejí bezpečnější prostředí a možnost poskytnout testovacím řidičům podobné podmínky. Mají ale nedostatky, především pokud jde o rozmanitost a složitost simulované situace. Jízdní simulátor nabízí mnohem méně realistické prostředí, než reálný svět, testovací podmínky v něm však mohou být plně a bezpečně kontrolovány.

Jízdní simulátor je nástroj, který umožňuje řídit vozidlo ve virtuálním prostředí. Umí reprodukovat simulace od každodenních dopravních situací, po specifické situace, jako je například riskantní předjíždění. Důležitou komponentou tohoto nástroje je simulace chování okolních řidičů.

Tato práce pojednává o simulaci okolních dopravních dějů ve virtuálním prostředí a řeší způsoby, jakými je možno reprezentovat dopravní situace a jak tyto propojit s jízdním simulátorem. Zaměřuje se na techniky simulující jednoduché děje, jako je jízda několika vozidel v koloně, předjíždění a projíždění křižovatkou.

1.1. Motivace

Motivací k výběru tohoto tématu pro mě byla šance na využití výsledné implementace v praxi. Díky tomu, že katedra Počítačové grafiky a interakce spolupracuje se společností Škoda Auto na projektu zabývajícím se mimo jiné simulátorem jízdy, mám možnost pracovat na něčem, co může být užitečné v komerční sféře.

Navíc mě vždycky fascinoval virtuální svět a simulace jevů, žijících svým vlastním životem. Vozidla která se sama rozhodují na základě dopravní situace jsou přímo vzorovým příkladem takového virtuálního světa.

1.2. Cíle práce

Cílem této práce je prostudovat a analyzovat základní metody simulace dopravy ve virtuálním prostředí. Součátí práce je také implemenotovat zjištěné metody pro simulaci výše uvedených dějů do existujícího software pro simulaci jízdy VRUT (Virtual Reality Universal Toolkit) od společnosti Škoda Auto.

1.3. Struktura práce

Text je logicky rozdělen do dvou relativně oddělených celků: Teoretický rozbor simulace a popis vlastní implementace do existujícího software. Tyto části jsou ovšem silně propojeny, protože implementační část aplikuje znalosti získané z teoretického rozboru.

Kapitola 2 Simulace okolního provozu specifikuje problém a upřesňuje zadání. Zhodnocuje existující články, diskutující problematiku simulace okolních dopravních dějů a simulaci autonomně řízených vozidel obecně. Povídá také o základních metodách simulace a reprezentace příslušných datových struktur, potřebných k simulaci.

V další kapitole 3 Rozbor řešení jsou rozebrány konkrétní metody zvolené k realizaci výsledné simulace. Kapitola popisuje vybranou datovou strukturu sloužící k navigaci autonomně řízených vozidel a způsoby simulace jednotlivých jízdních scénářů. Vysvětluje také způsob generování vozidel do simulace a odstraňování vozidel pro simulaci nevhodných.

Kapitola 4 Implementace se zabývá napojením výsledné práce do projektu VRUT, detailně popisuje formu a způsob uložení příslušných dat, potřebných k simulaci. Dále obsahuje výčet a popis hlavních tříd, které jsou v rámci implementace vytvořeny.

Kapitola 5 Testování vyhodnocuje chování implementace pro různý počet simulovaných vozidel. Výsledkem je maximální počet autonomních vozidel, který je možný použít pro testované scénáře tak, aby simulace probíhala v reálném čase.

Závěrem tohoto textu kapitola 6 Závěr, která shrnuje práci a hodnotí dosažené výsledky.

2. Simulace okolního provozu

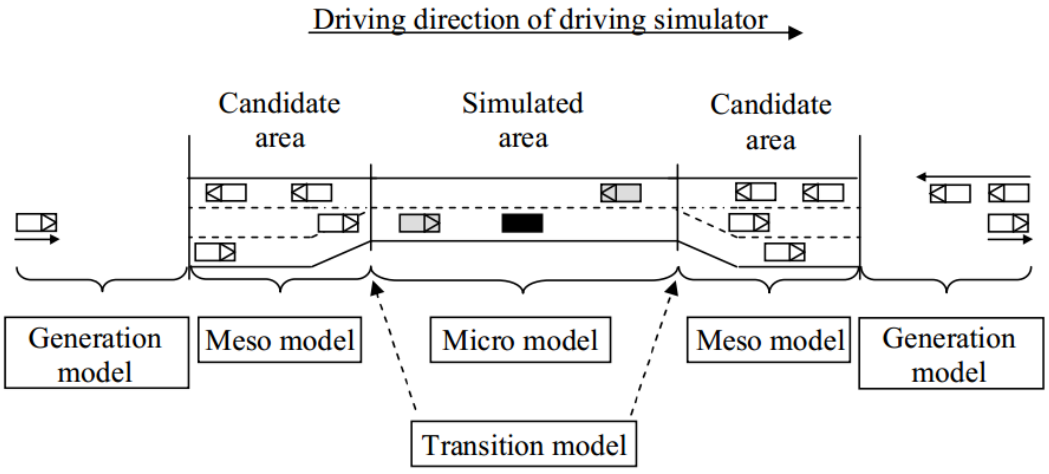
Počítačová simulace dopravy je široce používaná metoda v oboru modelování, plánování a vývoje dopravních sítí a systémů. Důležitým rozdílem mezi simulací okolních vozidel simulátoru a tradiční aplikací simulace je ten, že jedno z vozidel je řízeno člověkem. To zvyšuje nároky na model simulace individuálních subjektů, protože výstupem programu je vlastní chování okolních vozidel.

Při analýze tohoto problému a již existujících řešení jsem objevil několik článků, které se tématikou simulace autonomního dopravního provozu a jejího propojení se simulátory zabývají. Následuje rešerše, ve které jsou popsány přístupy a metody, které články diskutují.

2.1. Simulační modely

Článek [1] z roku 2008 popisuje framework pro generování a simulaci okolních vozidel v jízdním simulátoru. Zmiňovaný framework generuje dopravní provoz a simuluje realistické interakce mezi vozidly. Je založen na přístupu, ve kterém je simulováno pouze limitované okolí řízeného vozidla.

Předmětem zájmu je tedy bezprostřední okolí řidiče. Pouze v tomto okolí jsou okolní vozidla simulována. Toto okolí se pohybuje stejnou rychlostí a stejným směrem, jako řízené vozidlo. Ideou tohoto pohyblivého okolí je způsob, jak se vyhnout simulaci vozidel příliš vzdálených od řidiče, které nemají vliv na dopravu v jeho okolí, nejsou vidět, a tak je zbytečné plýtvat na nich výpočetním výkonem. Velikost okolí však musí být adekvátně zvolena. Především by nemělo být menší, než je dohled řidiče, aby se generovaná vozidla znenadání neobjevovala před řidičem. Okolí musí být dostatečně veliké,



Obrázek 1. Článek [1] popisuje metodu rozdělení simulace na vrstvy ve virtuálním okolí simulátoru. Černý vůz je řízen manuálně, šedé vozy jsou autonomně řízeny a bílé vozy jsou kandidáti pro simulaci.

aby byla simualce realistická a dovolovala řidičovi změny v rychlosti.

Framework ze článku [1] toto okolí složil celkem ze čtyř komponent: mikroskopický simulační model dopravy, mezoskopický simulační model, pravidla pro přechod mezi těmito dvěma modely a model pro generování nových vozidel (viz obrázek 1). Vozidla v simulovaném okolí jsou řízeny mikroskopickým modelem, který používá pokročilé metody pro stíhání vozidel, předjíždění a přizpůsobování rychlosti. Je důležité, aby se vozy v tomto okolí chovaly jako reální řidiči. Vozidla vzdálená od simulátoru nejsou až tak důležitá a jsou oddělena do zóny kandidátů pro simulaci. Jsou simulovány méně výpočetně náročným mezoskopickým modelem. Vozy za touto zónou jsou řízeny nejjednodušším, makroskopickým modelem.

Rozdíl mezi makroskopickým a mezoskopickým modelem je ten, že makroskopický model řeší provoz jako skupiny vozidel, řízeny jako celky, které jsou po trati distribuovány danými mírami. Mezoskopický model oproti němu bere simulovaná vozidla jako jednotlivce, jedoucí konstantní rychlostí. Tento model značně zjednodušuje mikroskopický přístup a slouží hlavně jako přechod mezi jím a markoskopickým modelem.

Nachází-li se vozidlo v zóně kandidátů a dosáhne hranice mezi mezo- a mikroskopickým modelem, je mu dovoleno vstoupit pouze v případě, že je dodstatečně vzdáleno od nejbližšího simulovaného vozidla ve svém jízdním pruhu. Tím je zajištěn plynulý a bezpečný přechod vozidla do simulované zóny.

Framework ovšem nepodporuje křižovatky, nájezdy, ani sjezdy a diskutuje pouze simulaci na jednoduché dvouproudé silnici. Simulovaná vozidla časem opustí simulační okno a systém zůstane prázdný, pokud nebudou nějakým způsobem generována vozidla nová. Článek [1] popisuje generování vozů následujícím způsobem. Pokud jede kandidát na vygenerování stejným směrem, jako řízený vůz, nachází se za ním a má vyšší rychlost, nebo je před vozidlem a má rychlost nižší, je vytvořen a přidán do zóny kandidátů. V opačných případech nemá cenu vozidlo generovat, protože by díky nevhodné rychlosti okamžitě ze simulace vyskočilo. Intuitivně, pro vozidla v opačném pruhu platí pravidla jednodušší, generuje se kterékoliv vozidlo, které vstoupí do kandidátní zóny.

Jak je vidět, simulace okolních dopravních dějů se dá kategorizovat do tří skupin, podle modelu, kterým řízení autonomní dopravy simuluje:

Mikroskopický model - Každé vozidlo se chová jako nezávislý agent a pro každého tohoto agenta jsou prováděny nezávislé výpočty chování, stavu a rozhodování. Oproti makroskopickému problému se zde však vyskytují problémy škálovatelnosti, protože takový přístup je náročný na výpočetní paměť a výkon. Vozidla se však chovají věrohodněji a uživateli přijde okolní doprava realističtější, protože každý vůz si počítá své vlastní jízdní vlastnosti.

Makroskopický model - Vozy jsou statisticky distribuovány definovanou, nebo dynamickou mírou. Výhodou tohoto modelu je snadné použití částicových systémů za cenu toho, že nám neposkytuje dostatečně konkrétní pohled na jedno individuální vozidlo, jakožto prvek, který sám o sobě nemá vliv na dopravu.

Hybridní model - Tento model se snaží kombinovat oba předchozí přístupy. Pro bezprostřední okolí řidiče používá model mikroskopicý, který je věrohodnější. Vzdálené vozy jsou pak řízeny makroskopickým modelem, který je výpočetně méně náročný. Mezi těmito úrovněmi může mít ještě třetí, mezoskopický model, který má na starosti plynulý přechod.

2.2. Existující řešení simulace dopravního provozu

Článek [2] nabízí náhled na několik známých, komerčních i nekomerčních, a široce používaných řešení simulace dopravy. Porovnává jejich schopnosti, rozdíly a diskutuje některé nápady, použité v příslušných řešeních. K jejich hodnocení používá následující metriky:

Open source - Je kladen důraz na to, aby bylo řešení volně dostupné, případně aby mělo přístupné zdrojové kódy ke studování a modifikaci.

Portabilita mezi operačními systémy - V moderní době s několika předními operačními systémy je důležité, aby software fungoval pod více platformami. Nemožnost programu fungovat například pod operačními systémy Linux a Mac OS X snižuje počet uživatelů a tím pádem popularitu software.

Dokumentace a uživatelské rozhraní - Zejména u komplexních softwarových systémů, jako jsou simulátory dopravy, je kladen důraz na to, aby byl výsledný produkt přehledný, jednoduchý k porozumění a měl uživatelsky přívětivé uživatelské rozhraní.

Schopnost vytvářet dopravní sítě - Požadavkem na dopravní simulátor je i to, aby disponoval možností automatického, nebo ručního vytváření dopravních sítí a scénářů k simulaci.

GUI simulace a kvalita grafické reprezentace - Jedná se o vizuální simulace a tak je důležité, aby výstupní vizualizace byla přehledná, pokud možno realistická, nebo aby alespoň odrážela realitu v přípustné formě.

Výstup - Ačkoliv se grafická forma reprezentace simulace dopravy zdá být nejlepší metodou k hodnocení průjezdu danou konkrétní dopravní sítí, k analytickým účelům je vhodné, aby měla apliakce možnost vygenerovat výstup do souboru v příslušných formátech.

Schopnost simulovat veliké dopravní sítě - V některých situacích je potřeba simulovat dopravu v celém městě, které může být veliké. Tento bod klade důraz na optimalizaci testovaného softwaru a hodnotí jednotlivé výkony při velikých zátěžích.

Výkon s ohledem na procesor a operační paměť - Souvisí s předchozím bodem, aplikace musí brát ohled na výpočetní výkon akutálně dostupné techniky.

2.2.1. SUMO - Simulation of Urban MObility

SUMO je open-source, portabilní balíček, simulující dopravní provoz mikroskopickým modelem. Umožňuje simulovat jak se daný požadavek na provoz, který se skládá z jednotlivých vozidel, pohybuje danou sítí silnic. Je sestaven tak, aby byl schopen zvládnout veliké silniční infrastruktury.

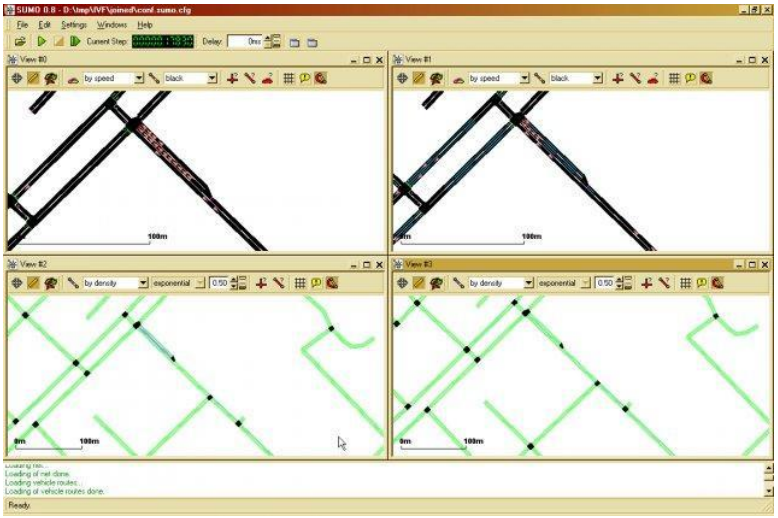
Uživatel je k definování dopravních infrastruktur nucen ručně upravovat XML soubory, reprezentující silniční sítě. Může ovšem využít automatický generátor, který umí vygenerovat tři základní scénáře: Mřížkovou síť, Pavoučí síť a náhodou síť.

Obrovskou výhodou tohoto řešení je, že je open-source a zdarma ke stažení.

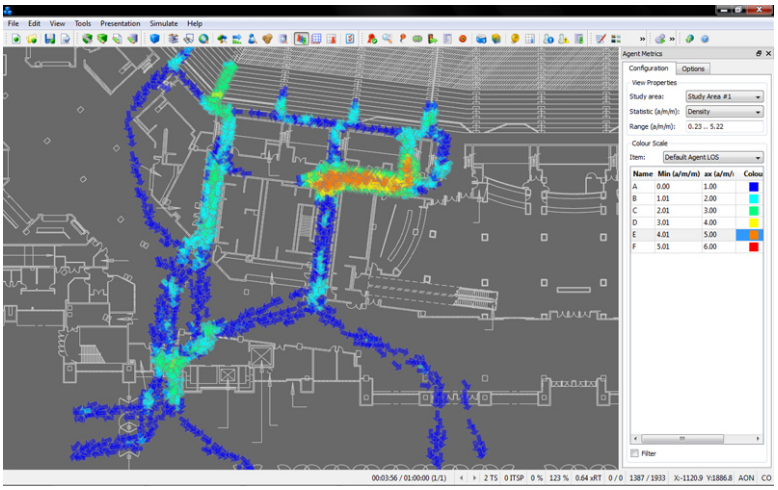
2.2.2. Quadstone Paramics Modeller

Quadstone Paramics je přední software v oblasti mikroskopické simulace dopravy a chodců, používaný k profesionálnímu plánování efektivních, úsporných dopravních infrastruktur, která umožňuje operativní hodnocení pro současné dopravní podmínky.

Software je plně škálovatelný a navržený tak, aby dovolil simulovat scénáře od těch nejmenších, jako jsou jednoduché křižovatky, až po komplexní městské dopravní systémy. Je používán ve více než 80 zemích světa lidmi z komerčních oblastí, pracovníky



Obrázek 2. Náhled uživatelského rozhraní projektu SUMO - Simulation of Urban MObility.

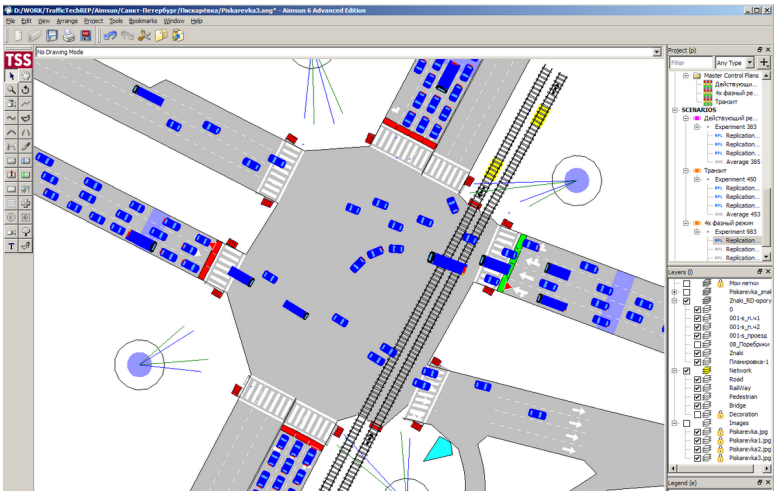


Obrázek 3. Náhled uživatelského rozhraní projektu Quadstone Paramics Modeller.

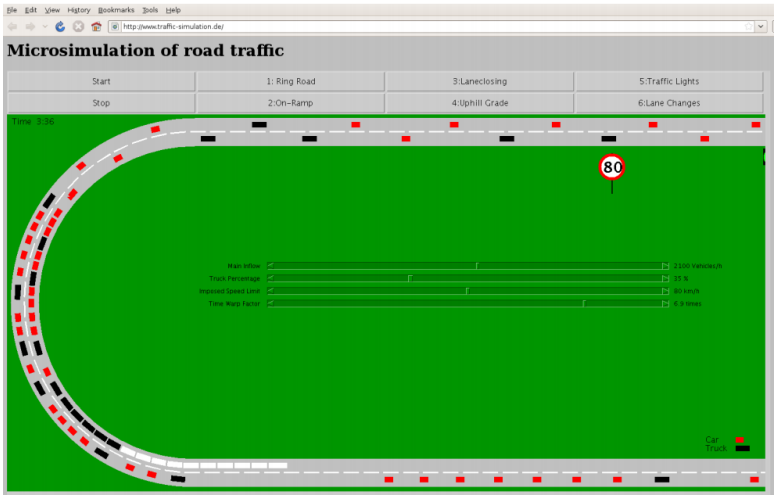
výzkumu dopravy a státem financovanými vládními agenturami. Paramics je považován za nejspolehlivější plánovací dopravní aplikací v komerční sféře. Nevýhodouu, která z toho vyplývá je, že není volně dostupný. Disponuje 30-denní trial verzí, kterou stojí za to vyzkoušet. V plné verzi tento software údajně disponuje nástrojem k automatickému generování dopravních infrastruktur. Možnost použít jej v této práci ovšem odpadá.

2.2.3. Aimsun

Aimsun je komerční modelovací software, který dovoluje uživateli modelovat jednoduché i komplexní dopravní scénáře. Je známý vyjímečným výkonem a rychlostí simulací. Disponuje mezoskopickým, mikroskopickým, i hybridním simulačním modelem. Má intuitivní uživatelské rozhraní ke kreslení dopravních spojení. Opět se jedná o komerční software, zdarma je k dispozici pouze 30-denní trial verze.



Obrázek 4. Náhled uživatelského rozhraní projektu Aimsun.



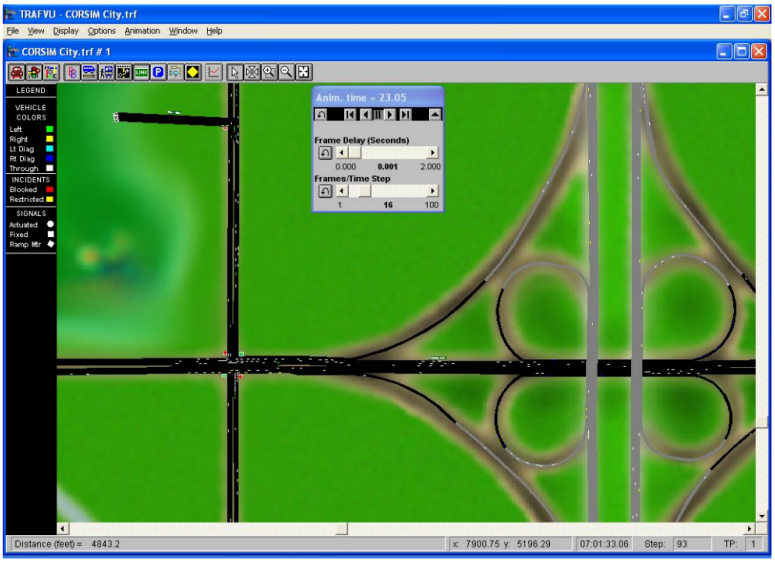
Obrázek 5. Náhled uživatelského rozhraní projektu Treiber's Microsimulation of Road Traffic.

2.2.4. Treiber's Microsimulation of Road Traffic

Osobní softwarový projekt vytvořený Martinem Treiberem pro výzkum dopravního modelování. Je dostupný na stránkách *http://www.traffic-simulation.de*. Jedná se komplexní Java applet, díky kterému je možno simulovat jednoduché dopravní scénáře přímo v prohlížeči. Je tedy platofrmově nezávislý. V této práci ovšem odpadává možnost využití, protože se jedná o neflexibiliní applet do prohlížeče.

2.2.5. Trafficware SimTraffic

Jedná se o simulační aplikaci, která je součástí komerčního Trafficware Synchro Studio balíčku. Tento balíček jeden z předních softwarů dopravního průmyslu. Aplikace slouží jako simulátor dopravy pro Trafficware studio, které disponuje mimo jiné i aplikací k synchronizaci světelných signalizací na křižovatkách.



Obrázek 6. Náhled uživatelského rozhraní projektu CORSIM TRAFVU.

2.2.6. CORSIM TRAFVU

Balíček pro simulaci dopravy pomocí mikroskopického modelu. Slouží k evaluaci světelných signalizačních systémů, dálničních a silničních sítí a kombinovaných struktur. Disponuje dvěma reprezentačními modely s názvy NETSIM a FRESIM. NETSIM model reprezentuje dopravu v městských infrastrukturách. FRESIM model reprezentuje dopravu na dálnicích a silnicích první třídy. Simulace modeluje pohyb jednotlivých vozidel a do toho zahrnuje i vliv fyzikálních podmínek a řidičova chování.

2.3. Vlastní řešení

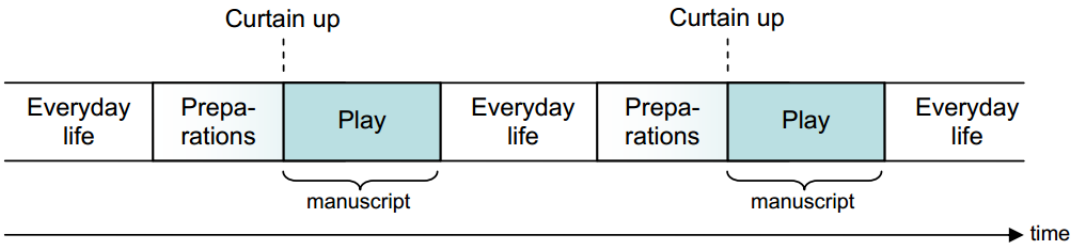
Žádné z diskutovaných řešení se do této práce nehodí a to buď z důvodu, že se jedná o komerční software, ale hlavně z toho důvodu, že neřeší ovládání jednotlivých vozidel na té nejnižší úrovni. Vozy jsou simulovány jako body, které perfektně sledují cestu, definovanou lomenými čarami, nebo křivkou, angl. *path following*. Tato práce se zaměřuje na simulaci vozidel s ohledem na fyzikální chování pneumatik, volantu a jízdními vlastnostmi vozidla obecně (angl. *steering behavior*).

Je tedy zapotřebí navrhnout model, který tyto problémy řeší. Následující kapitola 3 Rozbor řešení se věnuje analýze modelu, který bude vhodný pro výslednou implementaci práce.

2.4. Kombinace jízdního simulátoru a simulace okolní dopravy

Zásadním problémem simulací je kombinace autonomních vozidel, tedy simulace dopravního provozu, a řízených vozidel, tedy simulátorů. Předchozí sekce mluvila o existujících řešeních simulací dopravního provozu. Existuje velké množství simulátorů jízdy vozidlem. Příkladem může být Diplomová práce studenta ČVUT Simulátor jízdy městem [3].

Článek [4] k problému přistupuje z pohledu experimentů na chování řidičů. Řeší, jak se řidiči chovají ve specifických situacích a scénářích. Simulační scénář je specifikace silniční a dopravní situace na trati jízdního simulátoru. Je v ní zahrnut popis cesty



Obrázek 7. Ilsutrace z článku [1] znázorňující kombinaci tří komponent simulačního scénáře.

a prostředí, například specifikace geometrie vozovky, povrchu, počasí a okolních objektů, jako jsou stromy, domy, atd. Scénář také nese informaci o ostatních vozidlech v simulaci, spolu s jejich akcemi, pokud již byli předdefinovány.

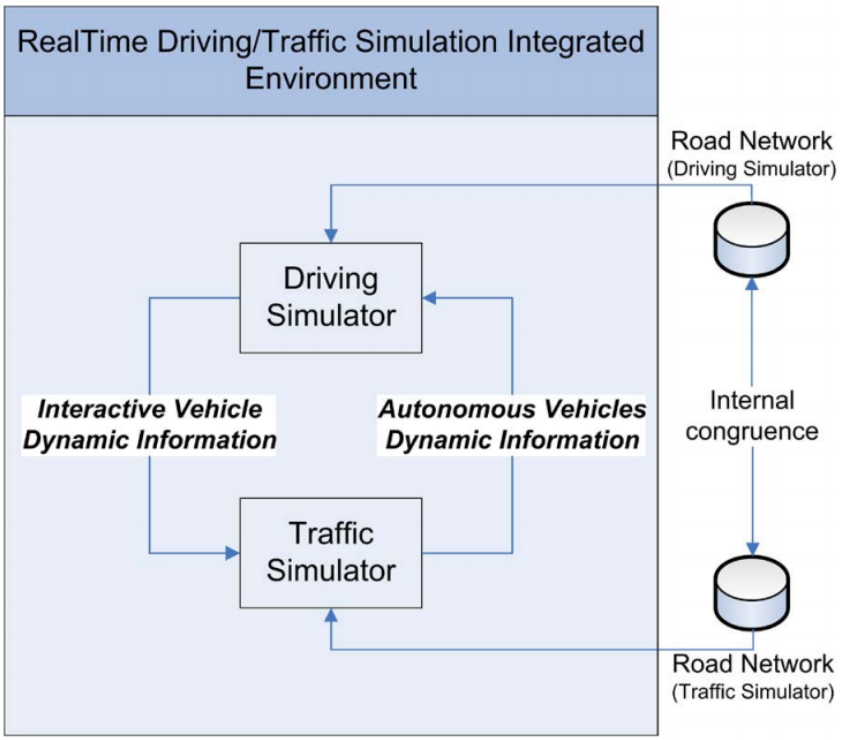
Článek navrhuje nástroj, ve kterém testovací scénáře znamenají krátké simulace konkrétní dopravní situace, kteoru by testovaný subjekt měl za úkol projet. Tyto scénáře jsou ručně vytvořeny testujícími osobami. Klade se důraz na reproducibilitu, aby mohl být scénář opakován s co nejméně odchylkami, a tak je důležité, aby vzniklé podmínky byly co možná nejvíce jednoznačné. Vzhledem k tomu, že dopravní provoz je velmi komplexní a dynamická záležitost, jedná se o velmi obtížný problém. Z toho duvůdu je doporučeno, aby měla simulace možnost ovládat úroveň autonomie okolních vozidel.

Obrázek 7 naznačuje, jak je simulační scénář rozdělen v čase na jednotlivé bloky. Článek se inspiroval divadelní terminologií a jednotlivé segmenty pojenoval *Everyday life* (normální, standartní, každodenní život), *Preparations* (přípravy na divadelní hru, tvorba scénáře, příprava herců) a *Play* (samotná divadelní hra). Blok *Everyday life* reprezentuje "normální"simulaci v aktuální scéně, s příslušnými jízdními podmínkami, bez žádných vyjímečných událostí. Blok *Preparations* znázorňuje časový blok, ve kterém je připravována nějaká konkrétní kritická situace (např. výjezd autobusu ze zastávky). Okolní vozidla (herci) jsou v tomto segmentu přemisťovány a celá scéna je připravována na simulaci dané situace. Blok *Play* je část, kde je simulovaná vlastní situace na scéně.

Článek tedy popisuje přístup, ve kterém je kladen důraz na detail a reproducibilitu jednoho krátkého dopravního scénáře. To přináší vysoké nároky na přípravu a důslednou produkci scény. Na druhou stranu je ovšem vhodný ke studii chování řidičů, jsou-li mu zadány podmínky a je pro něj připravena konkrétní scéna.

Článek [4] se na způsob propojení jizdního simulátoru a simulace okolní dopravy dívá podobně. Řeší tento problém jako spojení dvou naprosto nezávislých komponent a hledá jakoby vzájemnou symbiózu. Obrázek 8 naznačuje komunikaci mezi těmito komponentami. Je vidět, že jak dopravní simulátor, tak simulace okolního provozu používá vlastní reprezentaci dopravní infrastruktury (Road Network).

Integrované prostředí má za úkol zajistit efektivní komunikaci a spolehlivou výměnu dat mezi těmito dvěma aplikacemi. Musí v průběhu simulace zajišťovat, aby byly autonomní vozy řízeny mikroskopickým simulačním modelem spolu s pohybem řízeného vozu. Také simulátor musí posílat kinematická data (pozici, směr, rychlost, ...) simulační aplikaci. Ta potom může vypočítat kinematická data okolních simulovaných vozidel pro další krok a poslat je zpět simulátoru, který má za úkol aktualizovat tyto informace na výstupu.



Obrázek 8. Ilsutrace z článku [4] znázorňující schéma komunikace dopravního simulátoru a simulace okolního provozu. Z obrázku je vidět, že obě komponenty používají svoji vlastní reprezentaci dopravní infrastruktury (Road Network).

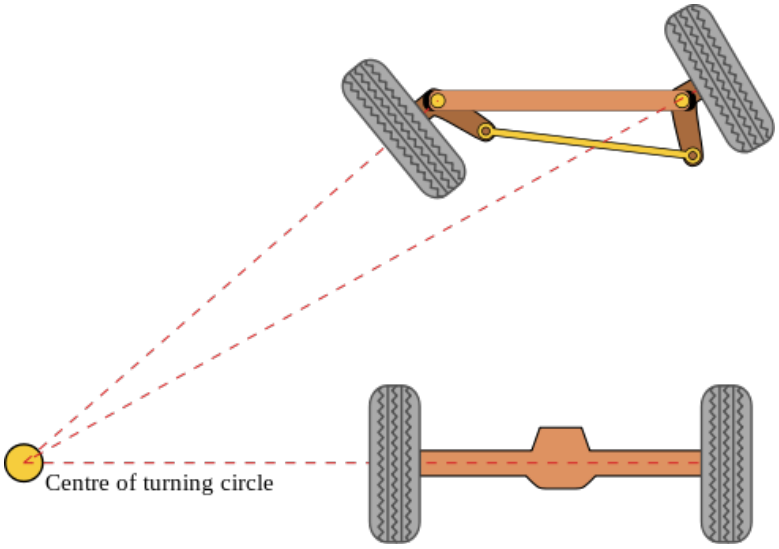
2.5. Řízení vozidla (Steering behavior)

Předchozí články řešily simulaci z globálního pohledu. Jednalo se o řízení dopravního provozu jako celku. Nyní se podívejme na specifické problémy, týkající se fyzikálního modelu řízení jednoho konkrétního vozidla, jakožto účastníka provozu. Zajímá nás, jakým způsobem je vozidlo naváděno a jak se naviguje po vozovce, jaké má povědomí o okolí a jak na něj reaguje.

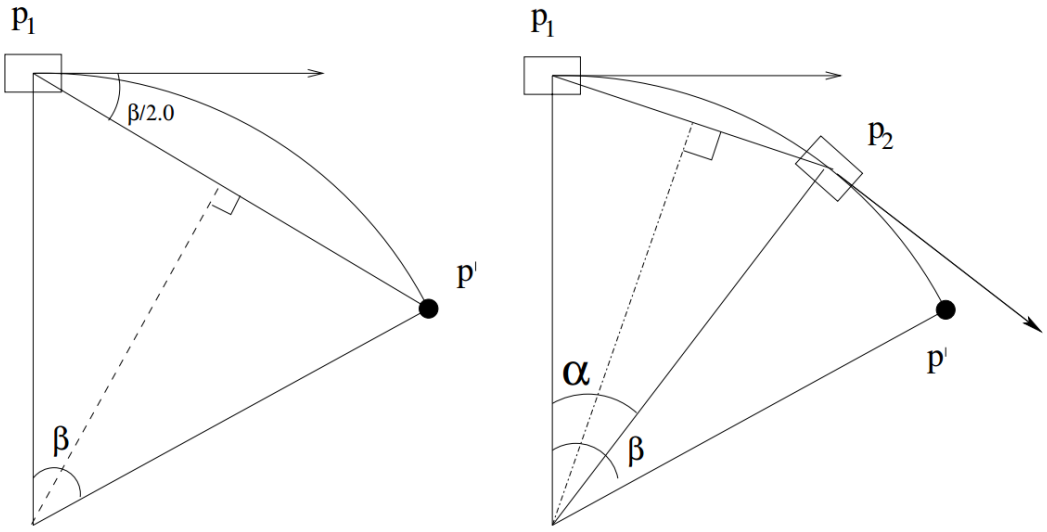
Ve virtuálním světě bývají vozovky a obecně objekty grafu scény definovány texturami, materiály a geometrií reprezentovanou polygony. Cesty jsou modelovány jako 3D stuhy, díky kterým je dána jednoznačná lokální orientace a mohou na ní být snadno počítány relativní vzdálenosti. Taková reprezentace je však k navádění vozidla po trati a hledání cest autonomně řízených vozidel nevhodná. Článek [5] definuje strukturu k navádění vozu jako množinu za sebou jdoucích bodů, tvořících cesty (paths). Vozidlo pak projíždí tuto sekvenci bodů, jeden za druhým. V každém kroku simulace je vypočítán úhel natočení kol tak, aby mířila na projížděný bod. Tento jednoduchý přístup nabízí snadné řešení problémů, jako je například změna jízdního pruhu. Ta proběhne tak, že vozidlo plynule přejde na sekvenci uzlů, definující sousední jízdní pruh.

Jelikož vozidla udržují kontakt s vozovkou koly, vztahují se na ně podmínky omezující pohyb. Vozidlo s pevnými nápravami se například nemůže pohybovat do stran (tak jako bychom chtěli při parkování).

Při průjezdu vozidla zatáčkou je pravé a levé kolo natočeno v odlišném úhlu, z důvodu splnění tzv. Ackermanovy podmínky. Ta říká, že střed otáčení musí ležet na prodloužené ose zadní nápravy. Tím se zajistí, aby se kola při jízdě pouze odvalovala a nevznikalo



Obrázek 9. Ackermanova geometrie (převzato z anglické verze Wikipedie)



Obrázek 10. Článek [5] popisuje jízdu zatáčejícího vozidla na kolech jako pohyb po kružnici, kde směr vozu je tangentou této kružnice. Vlevo je vidět trasu vozidla $p1$ spočítanou z projížděného bodu p'. Vpravo pak aktualizovaná pozice vozu $p2$.

nežádoucí smýkání po vozovce. Tento řídící mechanismus je pro simulaci ovšem příliš složitý a používá se v případech, kde se klade důraz na realističnost a přesnost fyzikálního modelu simulátoru.

Článek [5] popisuje pohyb vozidla jako pohyb po kružnici, která prochází polohou vozidla, jeho orientace je tangentou této kružnice, a protíná projížděný bod, viz obrázek 10.

3. Rozbor řešení

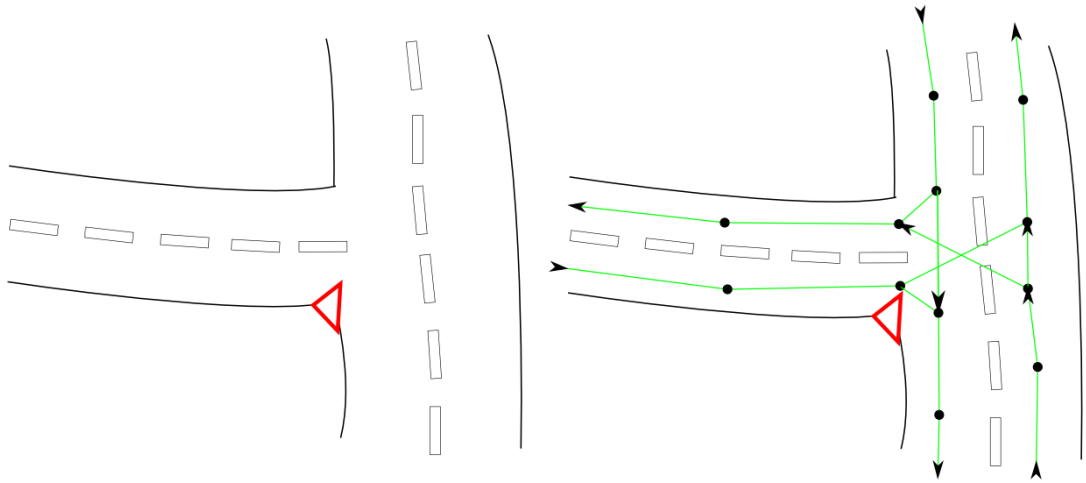
Cílem této práce je mimo jiné naimplementovat simulaci okolního dopravního provozu pro software VRUT (Virtual Reality Universal Toolkit). Tento modulární software disponuje mimo jiné již jízdním simulátorem (více viz sekce 4 Implementace). Jelikož žádné existující řešení (viz sekce 2.2) tomuto projektu nevyhovuje, nemusí se implementace starat o problémy při propojování (viz sekce 2.4) se simulátorem okolního provozu. Daný model simulace okolního provozu tedy může být navržen na míru.

Následující kapitola navrhuje vlastní řešení simulace okolní dopravy. Nejprve je nutno zadefinovat datovou strukturu, která bude reprezentovat prostředí pro pohyb autonomních vozidel. Potom je potřeba vozidla naučit po této struktuře jezdit a brát při tom ohled na dopravní předpisy a okolní vozidla a řešit dopravní situace.

3.1. Graf silnic

Aby mohla být autonomní vozidla naváděna po dopravní infrastruktuře, musíme jim předat informaci o tom, kde tyto cesty jsou a kudy vedou. Z toho vyplývá, že je potřeba navrhnout datovou strukturu, kterou jsme schopni reprezentovat dopravní svět, tedy silnice, křižovatky, jízdní pruhy a další periferie, jako jsou semafory, přechody a podobně. Intuitivně se nabízí graf, kde vrcholy a hrany definují umístění silnice. Jelikož u cesty potřebujeme znát směr, ve kterém je dovoleno jezdit, musí být hrany orientované.

Jeden jízdní pruh bude reprezentován sérií za sebou jdoucích vrcholů a orientovaných hran. Autonomní vozidlo se bude v simulační smyčce držet na těchto hranách, resp. bude se snažit dorazit k dalšímu uzlu grafu. Je potřeba držet informaci o sousedních



Obrázek 11. Vlevo je vidět analyzovaná geometrie silnice. Vpravo pak schéma, jak by měl vypadat graf silnic pro daný typ křižovatky. Černé body značí průjezdové body, zelené hrany pak logická spojení mezi nimi, tedy možnosti, kudy mohou vozy jezdit.

pruzích kvůli možnosti pruhy měnit, nebo předjíždět jiné vozy. Také je potřeba vědět, jakým směrem vede vedlejší pruh opačého směru, kvůli výpočtům při předjíždění.

Speciálním případem grafu budou křižovatky, větvení do více jízdních pruhů, nebo naopak sjednocování. Zde si musíme uvědomit, že sjednocují-li se dva jízdní pruhy, v kontextu grafu to znamená, že uzel v grafu bude mít několik vstupních hran. Uzel na křižovatce ponese informaci o tom, do kterých jízdních pruhů je možné odbočit. To znamená, že uzel v grafu bude mít několik výstupních hran. Na těchto uzlech také musíme udržovat informaci o tom, který pruh má před kterým přednost. Krom seznamu sousedních (navazujících) uzlů tedy také ponese seznam přednostních uzlů, které si aktuální vozidlo musí hlídat kvůli přednosti. Aby vůz věděl, zdali má kvůli přednosti zastavit, musíme také v každém uzlu udržovat seznam vozidel, které jsou na něj aktuálně naváděny. Takto si simulované vozidlo při vjezdu do křižovatky zjistí, kterým jízdním pruhům má dát přednost a zdali jsou v těchto pruzích momentálně jiná vozidla.

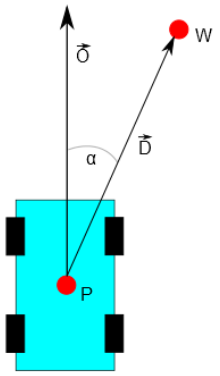
Protože nejsme schopni z geometrie trati vyčíst, zdali je pruh oddělen přerušovanou čarou a dá se tedy předjíždět, ponese v sobě uzel také informaci o možnosti, zda-li je v dané části silnice dovoleno předjíždět. Také nejsme schopni vyčíst z dopravních značek omezení rychlostí, takže tato informace taktéž musí být uložena v uzlu grafu. Celkem tedy uzel grafu ponese následující informace:

- Seznam navazujících uzlů. V případě obyčejného jízdního pruhu je tento seznam jednočlenný
- 3D souřadnice definující polohu uzlu v prostoru
- Omezení rychlosti
- Dovolení předjíždět
- Seznam uzlů, které mají před daným uzlem přednost
- Seznam vozidel, které do daného uzlu momentálně jedou

Jelikož stávající verze projektu VRUT žádnou takovou datovou strukturou nedisponuje, budeme si ji muset vytvořit sami z informací, které jsou nám k dispozici z modelu světa. Simulátor dopravních dějů tedy bude disponovat funkcionalitou, která, pokud pro načtenou mapu graf silnic neexistuje, parserem analyzuje geometrii světa a tam, kde je geometrie označena identifikátorem vyznačujícím vozovku, vytvoří uzly cesty. Otázkou zůstává, jestli pro každý jízdní pruh udržovat zvláštní seznam hran a uzlů, nebo bude pro celou silnici stačit jeden seznam, který by udržoval informaci o počtu, šířce a směru jízdních pruhů. Výhodou první možnosti je jednoduchá orientace a jednoznačná přehlednost u složitějších situací, jako jsou dálniční nájezdy, sjezdy, křižovatky, sjednocování a rozdělování pruhů. Druhá možnost je ovšem paměťově jednodušší a bude pravděpodobně méně výpočetně náročná. Grafem silnic se tato práce zaobírá detailněji v sekci 4.5 Implementace.

3.2. Navádění vozu po vozovce

Vedle geometrie trati máme již zadefinován náš graf silnic, díky kterému jsme schopni určit, kudy může vozidlo jet, který směr je povolen, jakou rychlostí jím můžeme projíždět a zdali je dovoleno předjíždět. Zbývá otázka, jak autonomně řízené vozidlo udržet na tomto grafu. Jak již bylo zmíněno, je tvořen uzly a orientovanými hranami, resp. body v prostoru, kde každý bod má seznam svých sousedů, do kterých je povoleno cestovat. Vozidlo si tedy bude udržovat seznam uzlů seřazený v pořadí, v jakém je chce projíždět, o délce nejméně jedna.



Obrázek 12. Ilustrace znázorňuje způsob, jakým je zjištěna odchylka směru vozidla od směru požadovaného. W je bod destinace, do kterého se vůz snaží dostat. $\vec{O}$ je vektor orientace vozu. $\vec{D}$ je vektor požadovaného směru a α je úhel definující odchylku.

Na začátku simulace je pro každý vůz graf procházen a je nalezen bod vozidlu nejvohdnější pro iniciální navedení na vozovku, resp. graf. Tento bod je nalezen na základě několika faktorů:

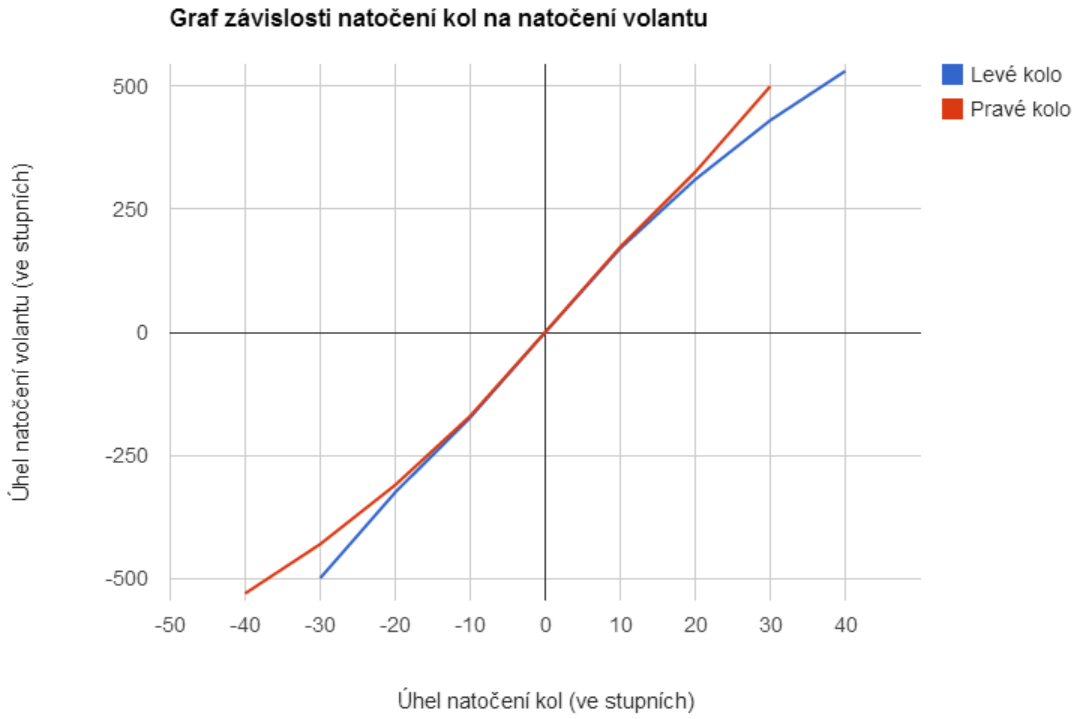
- Bod by měl být umístěn co nejblíže aktuálnímu vozidlu
- Část grafu jejíž součástí je daný bod by měla být orientována podobně, jako je momentální orientace vozidla

Tím se zabrání, aby vybraný bod nebyl příliš daleko od vozidla, a aby se vozidlo mířící jedním směrem nenavedlo na pruh v protisměru. Vybraný bod je následně vložen do seznamu průjezdových uzlů vozidla.

Při průjezdu bodem je tento bod ze seznamu odstraněn a je vybrán bod následující (viz obrázek 14). Pokud je seznam průjezdových bodů vozidla prázdný, je naplněn náhodně vybraným sousedem posledního projížděného uzlu grafu. Tímto je zajištěno, že vůz má k dispozici minimálně jeden uzel ve svém seznamu, ke kterému se má dostat.

Jednotlivé uzly grafu jsou ovšem umístěny v trojrozměrném prostoru, a tak je potřeba vozidlo naučit korigovat svůj směr, aby body projíždělo. Vůz o sobě nese mimo jiné i informaci o poloze, směru a rychlosti. Spolu s informací o umístění následujícího projížděného uzlu máme vše, co potřebujeme ke korekci směru vozu:

Obrázek 12 naznačuje mechanizmus řízení vozu. Bod P (point) je bod definující polohu vozu, W (waypoint) je bod definující projížděný uzel. Dále $\vec{O}$ (orientation) je vektor definující současnou orientaci vozidla. Pak $\vec{D} = W - P$ (desired orientation) je vektor definující požadovanou orientaci vozidla a konečně úhel α mezi vektory $\vec{D}$ a $\vec{O}$ je odchylka mezi aktuálním a požadovaným směrem. Tuto odchylku nyní musíme přepočítat do hodnoty nasměrování volantu a tím pádem natočení kol. Natočení volantu je definováno parametrem o hodnotě v rozsahu $< -1; 1 >$, kde -1 znamená úplné natočení kol vlevo, 0 reprezentuje vycentrovaná kola a 1 maximální natočení kol vpravo. Není nám ovšem znám způsob přepočtu této hodnoty do úhlu natočení kol, resp. do obvodové/úhlové rychlosti vozidla. Z obrázku 13 dodaného firmou Škoda Auto je znát, že při maximálním natočení volantu svírají kola s osou vozidla úhel asi 40°, což je jen velmi nepřesný odhad, zejména z důvodu, že vztah otáčení volantu a natáčení kol není lineární, a také, že kola jsou jemně vyosena z technických důvodů, vylepšujících jízdní vlastnosti vozu. Vzhledem k tomu, že náš modul má za úkol simulovat pouze okolní dopravu a vyplnit vozovky silničním ruchem, nám stačí počítat s odhadem, že konverze otočení volantu na natočení kol je lineární a že kola jsou rovnoběžná. Takto vzniklá



Obrázek 13. Graf znázorňující rozdíl natočení kol způsobeno Ackermanovou geometrií.

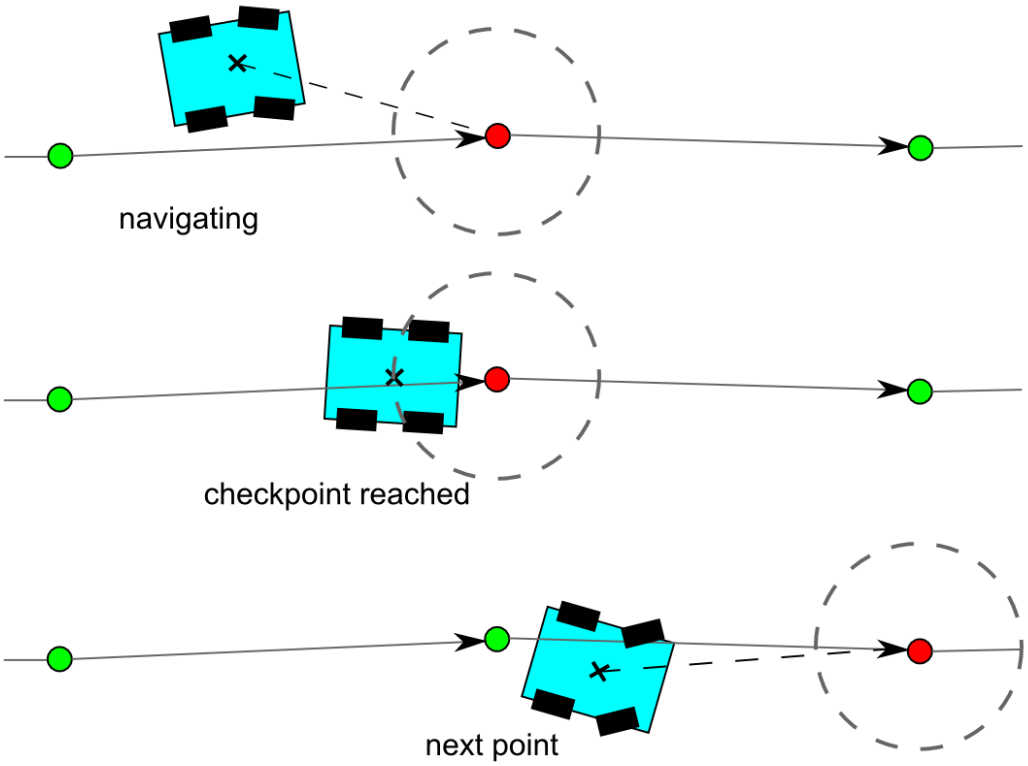
chyba nám způsobí jemné nepřesnosti korekcí při simulaci a tím pádem zvýší dojem chybovosti autonomních řidičů a realističnosti simulace. Vůz se totiž pouze snaží otáčet kola tak, aby svírala s vektorem požadované orientace minimální úhel. Pro dojem plynulého natáčení kol a zatáčení je vůz dovolen v jedné iteraci simulační smyčky natočit kola pouze o omezenou hodnotu z daného intervalu <-1;1>. Tato hodnota je dána konfigurovatelným parametrem (viz sekce 4.8 Konfigurovatelné parametry). To zajistí odezvu mezi momentem zjištění potřeby zatočit a skutečným odbočením, což opět přidá na dojmu nedokonalosti a realističnosti simulace.

Tímto je zajištěno navádění vozidla po silnici. Vůz zatáčí tak, aby se nasměroval na projížděný uzel a při jeho projetí je vybrán uzel následující.

3.3. Dodržování rychlosti

Předpokladem simulace je, aby vozidla dodržovala danou rychlost. Na to abychom mohli mít kontrolu nad rychlostí vozu Simulátor ovšem nezná příslušné jízdní vlastnosti vozidla, v tomto případě konkrétně výkon a tah motoru. Zná jen aktuální rychlost vozidla a může simulátoru posílat hodnotu plynového a brzdového pedálu v rozsahu <0;1>, kde krajní hodnoty reprezentují pedál minimálně a maximálně sešlápnutý. Nemáme tedy dostačující informace k tomu, abychom mohli přesně vypočítávat pedálové hodnoty k přesnému dodržení rychlosti. Nám dostupné prostředky ovšem k jednoduché simulaci dodržování rychlosti stačí.

Dovolená rychlost na trati je definována v každém uzlu grafu silnic jako jednoduchý parametr. Při projíždění daného bodu si vůz přečte hodnotu tohoto parametru a porovnává ji se svou aktuální rychlostí, je-li nižší, přidá hodnoty na plynovém pedálu. Je-li rychlost vyšší, ubere. Překročí-li rozdíl dovolené a aktuální rychlosti danou hodnotu,



Obrázek 14. Ilustrace znázorňuje průjezd vozidla grafem. Červeně je označen bod, na který je vůz aktuálně naváděn. Kolem něj je vyznačen práh tolerance. Při dosažení tohoto prahu je vozidlu změněn průjezdový bod na následující uzel v grafu a simulace pokračuje.

nastaví hodnotu pedálu na 0 a začne přidávat na hodnotě brzdového pedálu. Parametr ovlivňující hodnotu kroku plynového a brzdového pedálu je spolu s ostatními parametry popsán v sekci 4.8 Konfigurovatelné parametry.

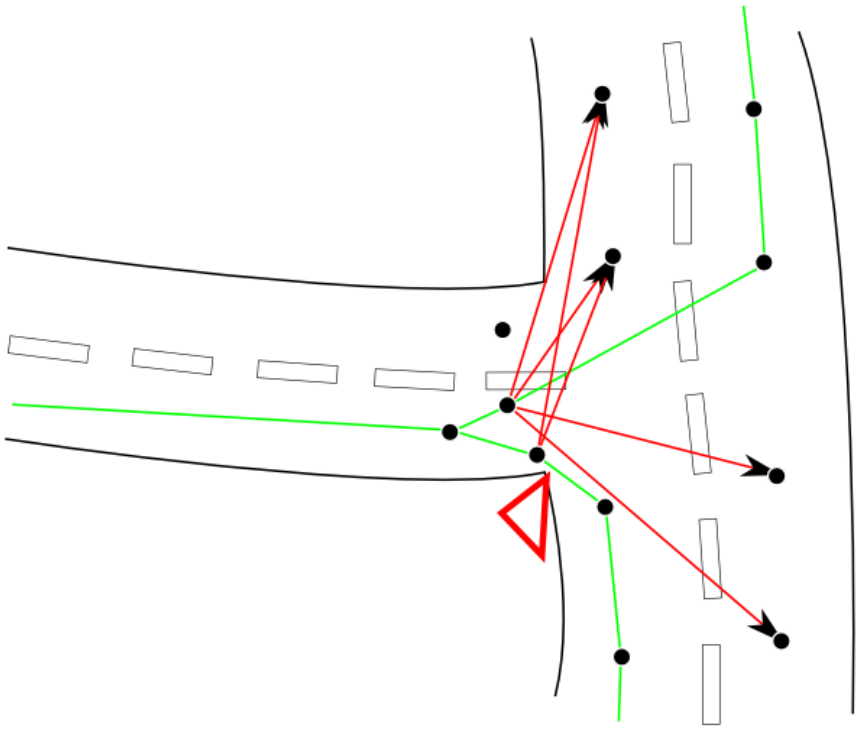
Aby simulace dosáhla jisté úrovně rozmanitosti, model definuje u každého autonomně řízeného vozidla atribut *willingness*, emulující ochotu autonomních řidičů. Cílem tohoto atributu je zajistit rozmanitost v rychlostech, které jsou vozy ochotny dosahovat. Tím vzniknou podmínky potřebné ke spuštění předjížděcího manévru. K tomu totiž dojde pouze pokud jedno vozidlo jede rychleji než jiné.

Atribut *willingness* nabývá hodnot v rozsahu $< 0.5; 1.0 >$. Vynásobí-li se touto hodnotou rychlostní omezení, definované v projížděném uzlu, získáme hodnotu blízkou omezení rychlost. Vozidlo se pak snaží udržet rychlost ekvivalentní této hodnotě. Je-li například omezení rychlosti na dané trati $50km/h$ a vozidlo má hodnotu *willingness* 0.7, vozidlo se ve výsledku pokouší udržet rychlost $50 * 0.7 = 35km/h$.

3.4. Křižovatky

Jedním ze základních stavebních prvků silniční dopravy jsou křižovatky. Je kladen důraz na to, aby náš modul uměl simulovat dopravní situaci na křižovatkách, aby si vozidla dávala korektně přednost podle dopravních předpisů a značení a aby byla simulace jízdy autonomního vozidla křižovatkou co nejautentičtější.

Křižovatka je v kontextu grafu silnic místo, kde se protínají dvě a více větví grafu. V místě protnutí je umístěn speciální uzel, který nese informaci o tom, ze kterého



Obrázek 15. Schéma znázorňující definici křižovatky včetně předností. Z důvodu přehlednosti jsou hrany vyobrazeny pouze pro jeden jízdní pruh. Zelené hrany značí průjezdové možnosti, červené pak přednost. Uzel do kterého červená hrana vstupuje má přednost před uzlem ze kterého hrana vystupuje. Projíždějící vozidlo tedy jede po zelených hranách a kontroluje si přítomnost jiných vozidel v uzlech, spojených červenou hranou.

jízdního pruhu se dá dostat do jiného a o prioritě silnic (hlavní silnice vs. vedlejší).

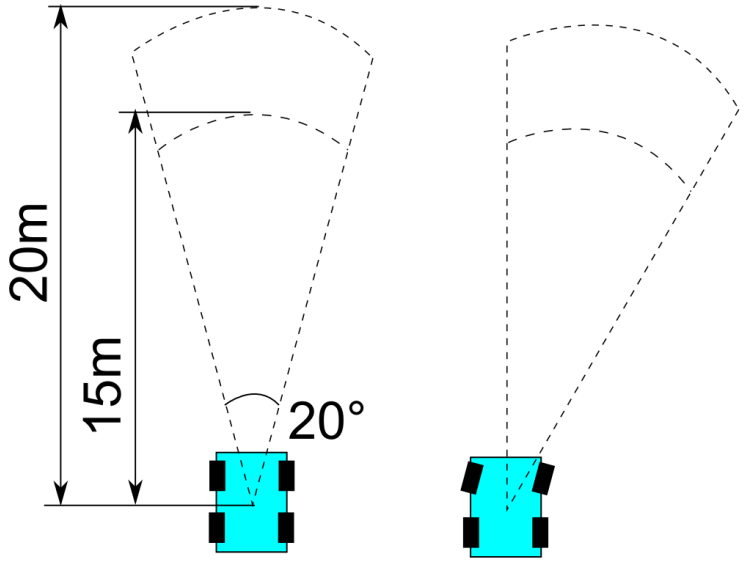
Při projíždění křižovatkou v reálném světě je řidič nucen brát ohled na několik faktorů:

- je na hlavní nebo vedlejší ulici
- jsou v jiných pruzích vozidla
- odbočuje
- má/dává přednost "z prava"

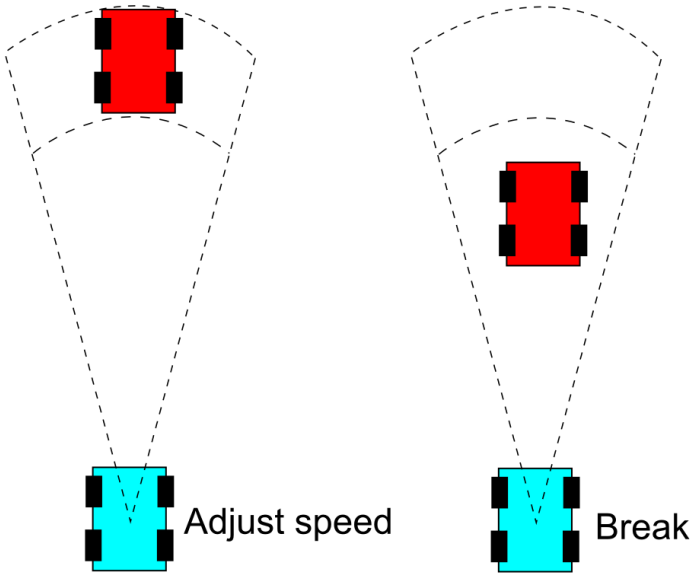
Vzhledem k tomu, že simulace disponuje grafem silnic, jsme schopni tyto faktory vyřešit pouhým vhodným zadefinováním vlastností příslušných uzlů (viz obrázek 15). Z uzlu do kterého vozidlo vjíždí jsme schopni vyčíst všechny potřebné informace. Víme, které jízdní pruhy (uzly grafu) mají před daným uzlem přednost, a víme, zdali těmito uzly projíždí nějaké vozidlo. Pruhy, resp. uzly grafu, před kterými má naše vozidlo přednost nijak neřešíme. Vozidlo je nutno zastavit pouze v případě, že některým z přednostních uzlů projíždí jiný vůz.

3.5. Stíhání vozidel (jízda v koloně)

V reálné situaci na vozovce se často střetáváme s kolonami, dopravní zácpou, nebo obecně situací, kdy dvě a více za sebou jedoucích vozidel musejí dávat pozor na subjekt před sebou, aby nedošlo ke kolizi. Požadavkem na modul Traffic je, aby tento jev dokázal simulovat.



Obrázek 16. Prostor který si vozidlo hlídá je reprezentován jako výseč, která se příslušně naklání při natočení kol. Ta se dělí na dvě části a vozidlo reaguje různě podle toho, ve které části se nachází vozidlo před ním.



Obrázek 17. Hlídaný prostor před vozidlem je rozdělen na dvě části. V levé části obrázku je stíhané (červené) vozidlo ve vzdálenější zóně. Stíhací (modré) vozidlo se tedy přizpůsobuje rychlosti stíhaného vozu. Objeví-li se stíhané vozidlo v bližší zóně (viz pravý obrázek), situace je interpretována jako kritický stav a stíhací vozidlo začne brzdit, dokud nezastaví, nebo dokud stíhané vozidlo nevyjede z bližší zóny, aby nedošlo ke kolizi.

Předpokladem pro simulaci tohoto jevu tedy je, aby měla vozidla informaci o vzájemné poloze, směru a rychlosti. Obrázek 16 naznačuje, jaký prostor před sebou si vozidlo hlídá. Pokud se v tomto prostoru objeví jiné vozidlo, příslušně zareaguje. Obrázek 17 zobrazuje rozdělení zóny na dvě. Je-li stíhané vozidlo dostatečně daleko, napodobuje pouze jeho rychlost a drží se tím pádem za ním. Dostane-li se ale vozidlo příliš blízko, je nuceno přibrzdit, aby zabránilo případné kolizi.

Tímto jednoduchým mechanizmem dokážeme zabránit vzájemným kolizím a vozidla jsou schopna jet v koloně.

3.6. Předjíždění

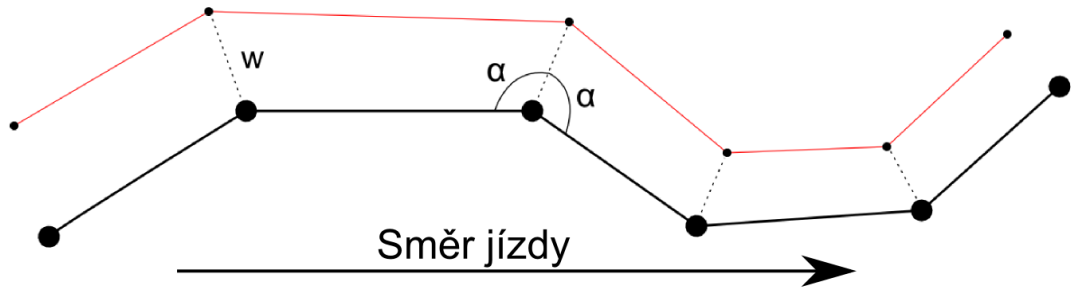
Předjíždění je mánévr, při němž rychlejší vozidlo objede pomaleji jedoucí vozidlo a zařadí se před něj. K jeho simulaci je potřeba si uvědomit několik základních kritérií. Článek [1] popisuje simulací předjížděcího manévru následujícími podmínkami ovlivňujících předjíždění:

- V daném dosahu silnice nesmí být žádné omezení předjíždění. To znamená, že v daném jízdním pruhu musí být dovoleno předjíždět. Zákazy mimo tento dosah neovlivní rozhodování.
- Vzdálenost uražená při předjížděcím manévru musí být menší, než je dostupný prostor, daný buď protijedoucím autem, nebo dohledem řidiče.
- Vůz musí být schopen předjížděcího manévru. Je potřeba vhodným parametrem omezit maximální vzdálenost předjíždění k zamezení extrémně dlouhých manévrů. Vůz musí být také schopen jet o danou hodnotu rychleji, než předjížděný vůz.
- Ochota předjíždět (willingness).

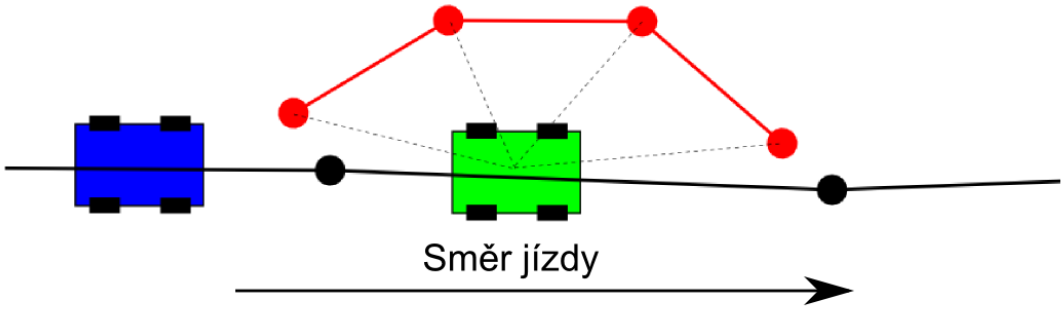
Dojde-li ke splnění těchto podmínek, spustí se manévr. Intuitivně se nabízí otázka, kudy má předjížděný vůz jet. Nabízí se možnost vedle série aktuálně projížděných uzlů grafu vytvořit jakýsi imaginární předjížděcí pruh ze seukndárního seznamu tak, jako je znázorněno na obrázku 18. Při spuštění předjížděcího manévru se přestane předjíždějící vůz navigovat pomocí klasických hran, ale těchto předjíždějících. Tento způsob ale nese spoustu nevýhod. Musíme v grafu udržovat o tuto informaci navíc pro každý uzel, ve kterém je možno předjíždět. Dále bychom v průběhu manévru museli zjišťovat polohu předjížděného vozu a detekovat v jaké pozici vůči němu se nachází předjíždějící vůz. To by znamenalo obtížný způsob detekce ukončení předjíždějícího manévru.

Lepší způsob, který se při testování (viz kapitola 5) osvědčil je ten, že se předjíždecí uzly vygenerují až v momentě předjíždění, a to dynamicky, na základě polohy předjížděného vozidla. Obrázek 19 znázorňuje, jak jsou předjíždecí uzly vygenerovány v daných polohách okolo vozu. Tyto uzly jsou v průběhu simulace aktualizovány a pohybují se zároveň s předjížděným vozidlem. Pro předjíždějící vůz to potom jen znamená, že se odpojí od hlavních uzlů a začne projíždět předjíždějící uzly. Faktem že se tyto uzly se pohybují zároveň s předjížděným vozem je zajištěno, že vůz svého kolegu bezpečně předjede. Aby k manévru došlo co nejrychleji a nejplynuleji, je předjíždějícímu dovoleno zvýšit rychlost a předjížděný vůz je vynucen zpomalit. Při průjezdu posledním předjíždějícím uzlem je vozidlo napojeno zpět na hlavní hrany vozovky a tímto je předjíždějící manévr ukončen.

V kontextu grafu silnic je předjíždecí manévr oproti obyčejné jízdě odlišný v tom, že se vůz nesnaží držet grafu silnice v daném místě, ale vygeneruje si svoje vlastní uzly na základě dané situace, které definují jízdní dráhu předjížděcího manévru.



Obrázek 18. Možnost způsobu vygenerování předjíždějících uzlů (spojeny červenými hranami). Jsou vytvořeny ve vzdálenosti w od hlavního jízdního pruhu.



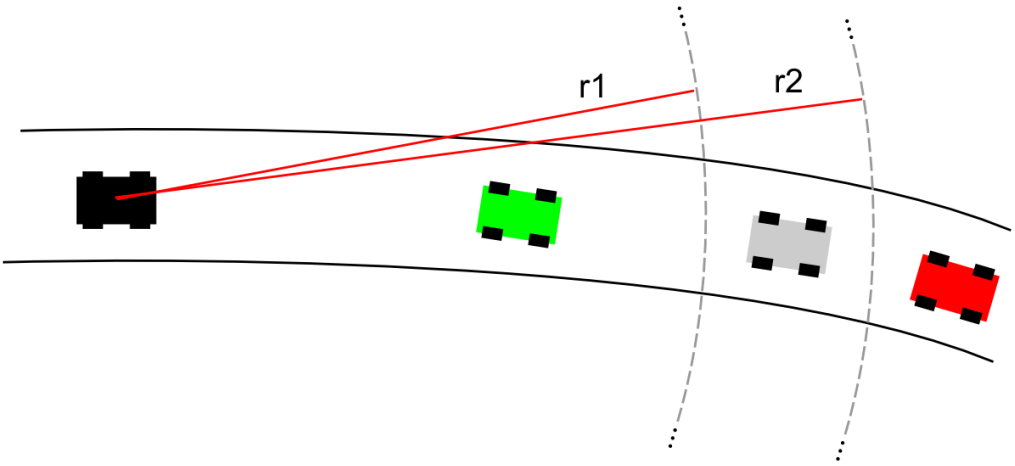
Obrázek 19. Lepší způsob generování předjíždějících uzlů. Jsou umístěny v daných polohách a vzdálenostech od předjížděného vozu a v průběhu simulace je jejich poloha aktualizována na základě pohybu předjížděného vozu. Modrý vůz pak při spuštění manévru přejde na průjezd těchto (červených) hran. Předjížděný (zelený) vůz pokračuje v průjezdu hlavních (černých) hran.

3.7. Změna jízdního pruhu

V kontextu grafu silnic je změna jízdního pruhu téměř totožný dopravní jev, jako jsou křižovatky. Jedná se o větvení grafu v místě, kde se větví jízdní pruhy, nebo kdekoliv, kde je možno jízdní pruh změnit. Uzly, ze kterých je možno vjet do jiného pruhu si musí udržovat informaci o části grafu, respektive uzlech, ve kterých si hlídá přítomnost jiných vozidel, které mají přednost.

3.8. Simulační okénko

Jelikož v simulaci může být jen jedno manuálně řízené vozidlo, nemá význam plýtvat výpočetním výkonem simulováním autonomních vozidel mimo dosah řidičova vnímání. Cílem práce je simulovat okolní provoz. Inspiroval jsem se tedy v článku [1], kde simulaci rozdělili na dva přístupy podle toho, jak daleko je simulované vozidlo od řízeného vozidla. Vytvořil jsem model, který zajistí, že modul bude simulovat pouze vozidla v určitém dosahu od řízeného vozu. Model nese název *Simulační okénko* a napodobuje chování, které je zmíněno v sekci 2.1 Simulační modely.



Obrázek 20. Schéma, znázorňující simulační okénko. Černě je označen manuálně řízený vůz, kolem kterého jsou přerušovanými čarami vyznačeny hranice prstence. $r1$ a $r2$ jsou vzdálenosti hranic, definující prstenec, ve kterém jsou generovány nové vozy. Zelený vůz je uvnitř simulačního okénka a zůstává v simulaci i příští iteraci simulační smyčky. Šedou barvou je označen vůz, který se nachází v prostoru prstence. Pokud vůz existoval v předchozí iteraci smyčky, v simulaci zůstává. Pokud ne, a jsou splněny předpoklady k vygenerování nového vozu, je přidán do simulace. Červený vůz je mimo simulační okénko a bude odstraněn ze simulace.

Vozidla mimo tento dosah jsou skryta, případně odstraněna. Oblast okolí simulátoru a hustota výskytu vozidel se pohybuje zároveň se řízeným vozidlem a je ovlivněna třemi parametry:

- Maximální počet aktivních vozidel
- Vnitřní poloměr simulační zóny
- Vnější poloměr simulační zóny

Těmito parametry jsem schopen řídit, v jakém dosahu budou vozidla simulována a jsem tak vlastně schopen řídit hustotu provozu. Hustota provozu je dána oblastí, ve které jsou vozidla simulovány a maximálním počtem vozidel, které může v simulaci najednou být. Vyjedou-li auta z této oblasi, ze simulace jsou odstraněny. Zbývá určit, kdy a kde vozidla do simulace vkládat. Simulační okénko je rozděleno na dvě zóny (viz obrázek 3.8). Vnější zóna je zónou pro generování nových vozidel. Nová vozidla mohou vzninkout v definovaných uzlech grafu silnic, které mají vytváření vozidel povoleno. Vygenerování nového vozidla v generační zóně simulačního okénka je ovlivněno následujícími podmínkami:

- Vyskytne-li se uzel grafu silnic v generační zóně
- Má-li tento uzel dovolen generovat nová vozidla
- Nenachází se v daném dosahu žádné jiné vozidlo
- Aktuální počet simulovaných vozidel je menší, než maximální možný počet.

Při splnění těchto podmínek je v daném uzlu vytvořeno nové vozidlo a je nasměrováno směrem, jakým vede příslušný segment grafu. Tyto podmínky jsou kontrolovány v každém kroku simulace a tím je zajištěno, aby, pokud je to možné, byl v simulaci požadovaný počet vozidel.

Jak je vidět, většina dopravních jevů a způsoby průjezdu vozidel tratí jsou definovány grafem silnic. Je tedy kladen důraz na to, aby byl deklarován co možná nejpřesněji, protože vozidla na něj spoléhají jak při normálním průjezdu tratí, tak při rozhodování ve speciálních případech.

3.9. Couvání

Jako zajímavost jsem do modelu přidal schopnost vozidel couvat. V případě, že došlo ke kolizi, pokusí se vozidla vycouvat opačným směrem, než je jejich požadovaná orientace. Detekce kolize je provedena jednoduchou podmínkou a tou je, že pokud se vozidlo nepohybuje a mělo by, předpokládá se, že došlo ke kolizi. Vozidlo tedy natočí kola opačným směrem a couvá, dokud není nasměrováno požadovaným směrem. Maximálně však couvá 5 vteřin, aby se v případě vážné kolize simulace nezastavila a neskončila pokusem couvat všech účastníků kolize. Tímto způsobem vozy zkoušejí opětovně couvat a popojíždět dopředu, dokud se jim nepodaří se z kolize zotavit.

4. Implementace

Jelikož se práce zabývá modulem do již existujícího rozsáhlého projektu, je předpokládáno dodržení již zadaných konvencí a způsobů přístupu k jednotlivým problémům a psaní kódu. Následující kapitola popisuje, jak je modul implementován.

4.1. VRUT

Katedra počítačové grafiky a interakce ČVUT FEL vytvořila ve spolupráci s firmou Škoda Auto projekt s názvem VRUT - "Virtual Reality Universal Toolkit". Jedná se o modulární software, schopný zobrazování grafických dat. Ačkoliv nosí v názvu termín "universal", firma Škoda Auto jej plánuje využívat k simulacím jízd, například k testování jízdních vlastností vozidel. Díky tomu, že je aplikace modulární, je usnadněna práce při rozšiřování, v případě projektu VRUT mohou tedy například studenti ČVUT vytvářet moduly pro své potřeby.

Software již má za sebou několik let vývoje, a tak již disponuje mnoha moduly, sloužícími zejména k načítání modelů v různých formátech, jejich zobrazování na výstupní zařízení, simulaci fyzikálních a jízdních vlastností vozidel, zpracování vstupu různých vstupních zařízení k ovládání vozidel, a jiné. Autorovi nového modulu tedy dopadá nutnost implementovat části, jako je vstup, výstup, grafický výstup, graf scény a mnohé další, které se přímo netýkají úlohy.

4.2. Technologie

VRUT je multiplatformní (Windows, Linux), modulání aplikace s podporou vícevláknového zpracování napsaná v C++, využívající wxWidgets a OpenGL jako hlavní aplikační a grafické rozhraní.

Díky požadavku na zachování modularity projektu je komunikace mezi jednotlivými moduly a částmi aplikace zajištěna Event-driven architekturou. Existuje tedy sada událostí (viz dokumentace VRUT [6]), kterými se mezi moduly předávají informace a data. Modul tedy bude taktéž napsaný v C++ a bude sestávat ze dvou částí. Tou první je parser geometrického modelu virtuálního světa, který jednorázově vygeneruje trasy, resp. průjezdové body pro autonomní vozy (více viz následující odstavec). Druhá část bude samotný životní cyklus ovládání autonomních vozů v samostatném vlákně, běžícím paralelně se zbytkem aplikace.

4.3. Traffic jako modul

Hlavní komponentou aplikace VRUT je jádro, které řídí registraci modulů, událostí, interpretuje příkazy ze skriptů nebo konzole. Spravuje scénu a všechny moduly mohou do scény zasahovat pouze přes něj.

Jádro je také spojovacím článkem všech modulů. Ty mohou komunikovat pomocí přijímačů a vysílačů datových balíčků - událostí. K implementaci komunikačního rozhraní používá knihovnu *wxWidgets*. Každý modul si může zaregistrovat, které události chce

Obrázek 21. Náhled VRUT aplikace

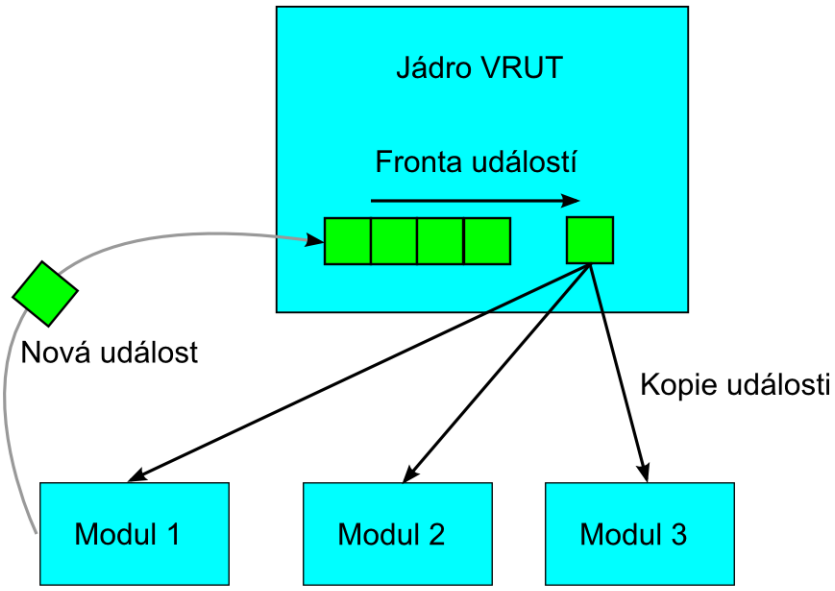
přijímat, a při odeslání této události jádru je pak distribuována příslušným modulům, viz obrázek 22.

Pro nás nejvíce důležitou komponentou aplikace je modul VehicleSimulator. Ten má ve spolupráci s dalšími podpůrnými moduly na starosti simulaci jízdních vlastností, ovládání a vizualizaci vozidel. Modul poslouchá na daném portu vstupní zařízení (klávesnice, volant) a jednotlivé vstupy předává simulačnímu modulu, který na ně reaguje a adekvátně mění příslušné atributy vozidla (plyn, brzda, otáčky motoru, natočení volantu, ...) přiřazeného portu vstupního zařízení. Vedle toho běží nekonečná smyčka simulující fyzikální model. Ta aplikuje dané atributy a příslušně rozpohybovává vozidla.

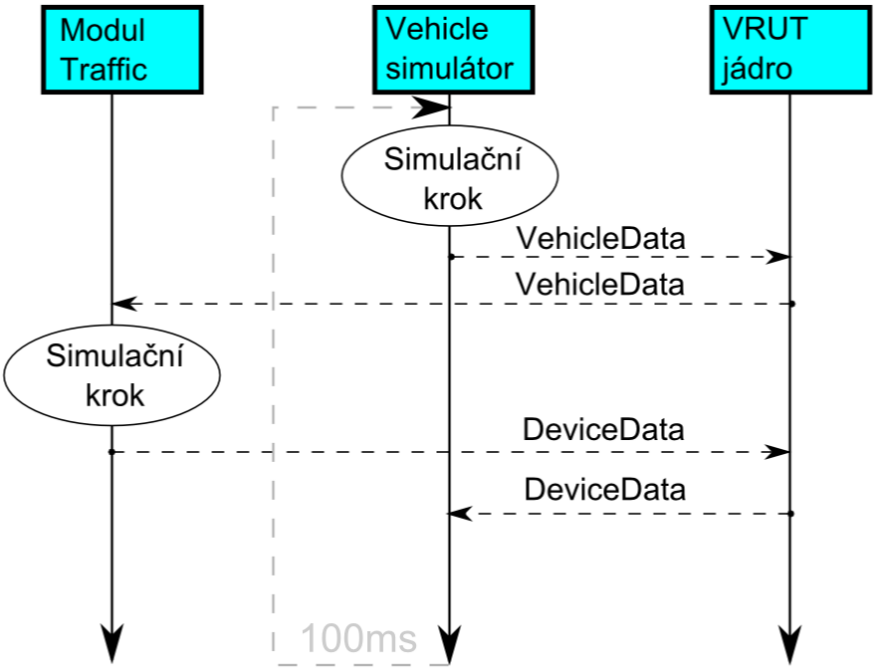
Modul VehicleSimulator podporuje simulaci více vozidel. Udržuje si tabulku, ve které přiřazuje každému vozidlu atribut *controllerID*, který identifikuje port, na kterém vozidlo naslouchá vstupním zařízením. Modul Traffic může vstupy ze vstupních zařízení emulovat pomocí události *GET EVT INPUT TRACKING*, která nese informaci o stavu plynového pedálu, brzdového pedálu a natočení volantu, zapouzdřenou v objektu *DeviceData*.

Modul VS v pravidelných interval emituje událost *EVT SCENE VEHICLE SIMULATION DATA* s objektem *VehicleData*, který nese informaci o aktuální poloze, orientaci, rychlosti a identifikátor příslušného vozidla. Interval je natvrdo v kódu modulu nastaven na 100 milisekund. V případě bezchybné komunikace mezi moduly a plynulé simulace bez zpoždění má tedy modul Traffic informaci o stavu každého vozidla každých 100 milisekund. Obrázek 4.4 znázorňuje diagram, který naznačuje, jak modul zpracovává tuto informaci.

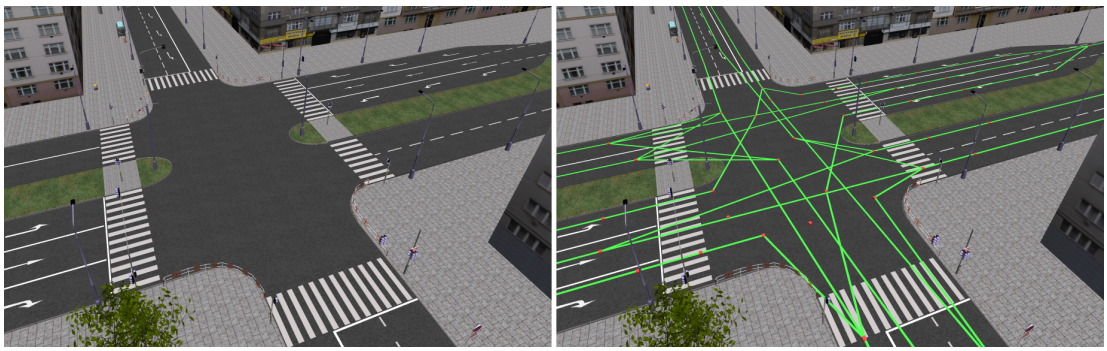
Ve výsledku má tedy modul přístup k informaci o poloze, směru a orientaci každého vozidla desetkrát za sekundu. V těchto intervalech má možnost data zpracovat a zareagovat odpovědí, nesoucí informaci o korekci natočení kol a stavů pedálů.



Obrázek 22. Schéma komunikace událostí a jejich distribuce jádrem. Moduly přijímají jen předem zaregistrované události. Při zpracování události jádrem je pak tato zpráva odeslána jen zaregistrovaným modulům.



Obrázek 23. Diagram znázorňuje komunikaci modulu Traffic s modulem VehicleSimulator pomocí událostí *VehicleData* a *DeviceData* přes jádro aplikace. Tento obrázek ilustruje pohled na jeden simulační krok v kontextu zpracování jednoho konkrétního vozidla.



Obrázek 24. Náhled na geometrii křižovatky bez zobrazení grafu silnic (vlevo). Křižovatka s vykreslenými hranami a uzly grafu silnic (vpravo). Uzly (červené body na silnici) slouží jako průjezdové body pro autonomně řízená vozidla. Hrany (zelené čáry) představují pouze logické spojení (sousedství) uzlů.

4.4. Simulační smyčka

Modul běží na vlastním vlákně, které v cyklu sériově zpracová zprávy přicházející ze všech vozidel. Jedna iterace životníhi cyklu modulu v při samotné simulaci totiž sestává z několika základních činností:

- zjistit polohu, rychlost a směr vozidel
- zkontrolovat vozidla v simulačním okénku (viz 3.8 Simulační okénko) a odstranit ta, která vyjela ven
- pokud to dovolují podmínky (viz 3.8 Simulační okénko), přidat nové vozy do simulace
- zjistit následující uzel (nebo i více) pro každé vozidlo
- pokud je následující uzel křižovatka, rozhodnout se, kam odbočit
- zjistit případnou přítomnost dalších vozidel (viz sekce 3.5 Stíhání)
- detekovat speciální případy (zastavení, předjíždění, couvání)
- vypočítat korekci směru a rychlosti
- odeslat na VehicleSimulator

V každé iteraci je spočtena odchylka od vektoru směru k následujícímu uzlu a vypočítána korekce, která je následně odeslána příslušné instanci VehicleSimulatoru konkrétního vozu. Tento modul zpracovává informace o vstupu, jako je natočení volantu, sešlápnutí pedálu, atp. Tyto operace je potřeba provést pro všechny autonomní vozy, takže je důležité dbát na optimalizaci a výpočetní nároky.

4.5. Graf silnic

Virtuální svět je sestaven pouze z geometrického modelu s texturami a kolizním modelem pro základní simulaci jízdy. Neobsahuje tedy žádné záchytné nebo orientační body pro simulaci jízdy autonomního vozidla. Je tedy nutno vytvořit datovou strukturu, pomocí které budou autonomní vozy navigovány. Tuto strukturu nazývám grafem silnic a detailněji ji popisuji v sekci 3.1 Rozbor řešení. Následující sekce popisuje tuto strukturu z implementačního a praktického hlediska.

Abychom nemuseli graf silnic při každém spuštění simulace generovat a ručně upravovat, je exportován do XML souboru, který je následně načítán při inicializaci. Specifikace grafu silnic si vyžádala celkem 3 typy XML elementů: uzly (*nodes*), hrany (*connections*) a přednosti (*priorities*).

Uzel (node) je základní stavební prvek grafu silnic. Reprezentuje jeden uzel grafu a jako hlavní atributy nese identifikátor ID, použitý k identifikaci uzlu ve scéně, a polohu v prostoru, definouvanou třemi souřadnicemi x, y, z (v milimetrech). Dále nese doplňující atribut *o*, který může mít hodnotu 1 nebo 0 a určuje, zdali je v uzlu dovoleno předjíždět. Pokud je v uzlu dovoleno generovat vozidla do simulace, nabývá atribut *sp* hodnoty 1, v opačeném případě 0. Posledním atributem je *sl* (speed limit) definující omezení rychlosti v daném uzlu.

```
<nodes>
  <node id="1" x="123" y="456" z="0" o="0" sp="1" sl="30"/>
  ...
</nodes>
```

Abychom měli v XML souboru uloženou informaci o tom, které uzly na sebe navazují, je potřeba zavést element reprezentující hranu (*connection*). Tento element obsahuje pouze dva identifikátory *from*, nesoucí ID uzlu, ze kterého hrana vystupuje, a *to*, jakožto ID uzlu, do kterého hrana vstupuje. Tím je jednoznačně dána orientace hrany. Uzel může mít libovolný počet vstupních a výstupních hran.

```
<connections>
  <connection from="1" to="2"/>
  ...
</connections>
```

Při předjíždějícím manévru je potřeba ohlídat jízdní pruh vlevo od pruhu, ve kterém jede vůz, který se chystá předjíždět. K tomu slouží element *overtaking*, který nese informaci o tom, ze kterého uzlu (*id*) je potřeba si dát pozor na vozidla v uzlech (*check*) které ohrožují předjížděcí manévr. Pokud se v těchto uzlech nacházejí vozy, nemůže předjížděcí manévr proběhnout.

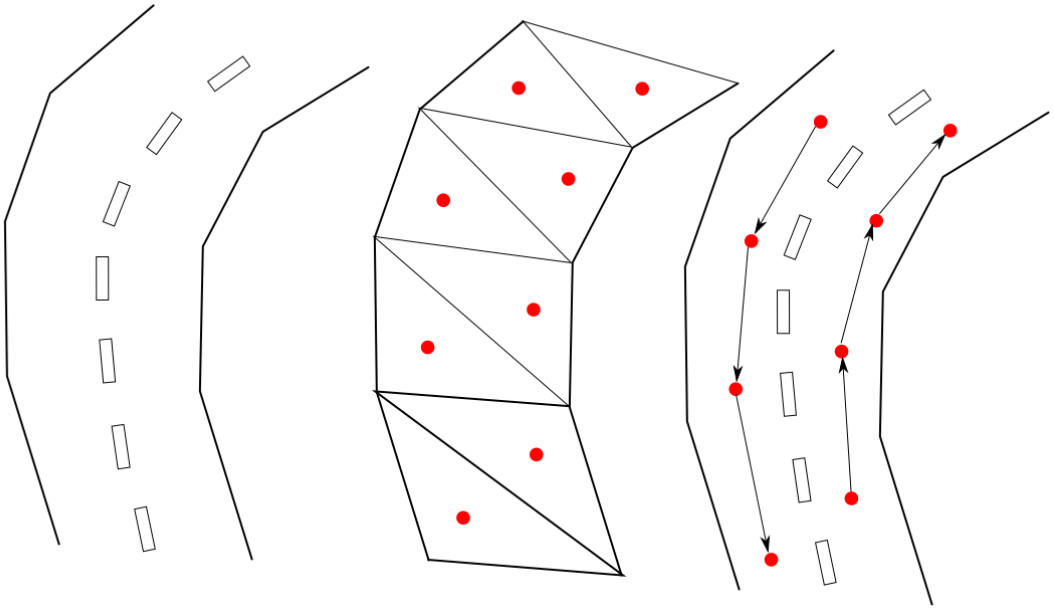
```
<overtakings>
  <overtaking id="4" check="2" />
  ...
</overtaking>
```

Posledním typem elementu je element *priority* - přednost. Nese informaci, který uzel má přednost před kterým, respektive která vozidla. Následující ukázka definuje, že vozy na uzlu s ID 3 musejí dát přednost vozidly jedoucí do uzlu s ID 5.

```
<priorities>
  <priority id="3" priority="5" />
  ...
</priorities>
```

Touto formou jsme schopni plně reprezentovat graf silnic a mít jej uložený v souboru pro opakované použití. Modul avšak momentálně nabízí jen velmi omezené rozhraní pro tvorbu a úpravu grafu silnic.

V případě, že soubor s reprezentací grafu neexistuje, spustí se proces, který vytvoří základní množinu průjezdových uzlů. Modul požádá manažer scény o geometrii vozovek, která je specifikována identifikátorem v souboru *worldFile*, definované v sekci 4.8 Konfigurovatelné parametry. Tím modul dostane seznam trojúhelníků, tvořící síť geometrie vozovky. Nakonec vytvoří naivně průjezdový uzel ve středu každého trojúhelníku a výslednou množinu uzlů vyexportuje do XML souboru *traffic*GraphFile. Dále už je na uživateli, aby přebytečné body ručně odstranil, navazující body propojil a případné



Obrázek 25. Způsob generování grafu silnic. Na prvním obrázku je znázorněna geometrie vozovky, jak vypadá spolu s texturou. Na druhém obrázku je vyobrazena trojúhelníková síť, reprezentující geometrii vozovky, spolu se středy jednotlivých trojúhelníku. Z těchto středů jsou vytvořeny průjezdové body. Toto je plně automatizovaný krok a modul disponuje touto funkcionalitou. Na uživateli pak je, aby příslušné body vhodně spojil a vytvořil tak síť uzlů, které reprezentují jednotlivé jízdní pruhy tak, jako je znázorněno v třetím kroku.

nedokonalosti v umístění bodů upravil v onom souboru. Modul ještě disponuje funkcí, která aktuálně existující uzly opakovaně přeexportuje do již existujícího souboru, který se tím přepíše. Tato funkce je vhodná v případě, že uživatel ručně odmaže nepotřebné uzly v průběhu simulace pomocí manažeru scén, který disponuje uživatelským rozhraním k manipulaci geometrie, a umožnuje tak mazání.

4.6. Simulační oknénko

Modul *Traffic* dědí od předka *SceneModule* který má prostředky k manipulací se scénou a objekty v ní. Jsem tedy schopen geometrii vozů přesouvat, skrývat a znovu ji odkrývat. Vozidla které tedy nechci v simulaci mohu skrýt a ušetřím tak na výpočetním výkonu při počítání kolizí. V modulu je dané vozidlo vyřazeno ze seznamu simulovaných a odpadává tak pro něj potřeba výpočtu celé simulační smyčky. Tím je ušetřno spoustu procesorového času v případě, že trať, na které simulace probíhá, je veliká a k jejímu zaplnění by bylo potřeba veliké množství vozidel.

Implementace tedy disponuje seznamem vozidel k dispozici pro simulaci a vybírá si, které vozy použije. Apliakce tedy musí před spuštěním nainicializovat daný počet vozů, jejich geometrii a simulátor fyzikálního modelu, aby byli k dispozici po dobu běhu.

4.7. Třídy

Projekt si klade důraz na využití principů objektového programování. To má za následek fakt, že při vývoji modulu vzniklo několik tříd, reprezentujících logické celky práce.

Následuje seznam hlavních tříd a struktur modulu:

4.7.1. Traffic

Hlavní třída, tvořící zásadní část modulu. Dědí od třídy *SceneModule* a nabývá tak schopnosti být zavolána jako modul pro jádro VRUT. Také má díky tomu přístup ke správci scén a zaregistrován parametr *sceneID*. Modul má tedy po přiřazení správné hodnoty *sceneID* identifikátor scény a případně přímý přístup k ní. Dokáže tedy manipulovat s objekty ve scéně, přidávat je, nebo mazat. Toho využívá k zobrazení/skrytí vizuální reprezentace grafu silnic a geometrie vozidel.

Předek již zajišťuje inicializaci, schopnost přijímat a zpracovávat události, nebo posílat události vlastní. Třída si při inicializaci v konstruktoru zaregistruje odposlouchávání událostí *EVT SCENE VEHICLE SIMULATION DATA*, které přicházejí v krátkých opakovaných intervalech a nesou informaci o poloze, rychlosti a směru každého vozidla v simulaci. Každou tuto událost nechá zpracovat příslušnou instancí třídy *AICar* pro konkrétní vozidlo a jako odpověď posílá do jádra seznam událostí, které promítají změnu polohy, rychlosti a orientace vozidla do nastavení pedálů a volantu.

4.7.2. AICar

Třída AICar reprezentuje jedno konkrétní autonomně řízené vozidlo. Jeden speciální případ instance je také udržován pro ručně řízené vozidlo a je zařazen do seznamu vozidel, který je procházen a kontrolován při předjíždění, dávání předností a jízdě v koloně.

Třída dále obsahuje stavový automat, který reprezentuje aktuální stav vozidla a upravuje tak jeho chování v každém konkrétním stavu. Následuje seznam možných stavů vozidla:

NORMAL - Normální stav. Vůz se snaží projíždět grafem silnic, uzel za uzlem v předepsané orientaci. Při tom se snaží dodržovat rychlost a kontrolovat své okolí kvůli možným kolizím, přednostem a předjíždění.

STOP - Vůz je zastaven. Tento stav je použit, když je vozidlo mimo dosah řidiče a je deaktivováno. Vůz má také nulovou hodnotu na plynovém pedálu a maximální hodnotu na brzdovém. Tím je zajištěno, že vůz zastaví a zůstane stát.

OVERTAKING - Vozidlo předjíždí. Při vstupu do tohoto stavu je vozidlu dána reference na předjížděný vůz, aby mělo informaci o poloze předjížděcích uzlů, kterých se po dobu předjíždění drží namísto hlavních uzlů grafu silnic. Zároveň je vozidlu nastaveno neomezené omezení rychlosti k urychlení předjíždějícího manévru. Na konci manévru je vůz uveden do stavu *NORMAL*.

OVERTAKEN - Vozidlo je předjížděno. Je mu nastavena nulová hodnota na plynovém pedálu, aby bylo urychleno předjíždění a vozidla se mohla vrátit do normálního stavu co nejdříve .Na konci manévru je vůz uveden do stavu *NORMAL*.

REVERSING - Nedaří-li se vozidlu pohybovat vpřed, je uvedeno do tohoto stavu - stavu couvání. Tento stav může nastat v případě kolize s jiným vozidlem, nebo s překážkou. Vůz vytočí volant na opačnou stranu, než je mu dáno, a je zařazen zpětný chod. Vozidlo se pokouší couvat, dokud není nasměrováno na následující průjezdový uzel, maximálně však po dobu 5 sekund.

WAIT - Vozidlo dává přednost jinému. Tento stav může nastat v případě, nachází-li se vůz na křižovatce a v jízdním pruhu který má přednost se nachází jiné vozidlo. Po uvolnění jízdních pruhů s předností je vůz uveden do stavu *NORMAL*.

Hlavní částí této třídy je metoda *doJob()*, která je zavolána modulem při obdržení události *EVT SCENE VEHICLE SIMULATION DATA*, nesoucí informaci o poloze,

orientaci a rychlosti příslušného vozidla. Metoda má přístup ke grafu silnic a informaci o všech ostatních vozidlech v simulaci. Díky těmto datům má tu moc reagovat na dopravní situace, volit vhodnou úroveň sešlápnutí pedálů a zatáčet příslušně volantem, aby se vozidlo dostalo do požadovaného směru. Vyhodnocuje situaci a na základě toho příslušně přepíná stavy ve stavovém automatu.

4.7.3. RoadGraph

Třída, reprezentující graf silnic (viz sekce 3.1). Jedná se o kořenový, cyklický, orientovaný graf, kde uzly jsou reprezentovány třídou *RoadGraphNode* a informace o hranách je uložena v těchti uzlech.

Pro iniciální navedení vozidla na graf disponuje metodou *findClosest*. Ta najde a vrátí jeden uzel grafu, jakožto nejvhodnějšího kandidáta pro navedení vozu. Projde graf hloubkovým průchodem a zaznamenává si potenciální kandidáty, které se nacházejí před vozidlem (viz obrázek). Z těch vybere nejbližší uzel, ten je zvolem vhodným kandidátem a je předán vozidlu.

Ke generování nových vozidel v prstencové zóně simulačního okénka ještě obsahuje metodu *getSpawnNodes*, která najde uzly, jakožto vhodné kandidáty pro umístění vytvořených vozidel. Opět projde graf do hloubky a vrátí seznam uzlů, které se nacházejí v daném prstenci.

4.7.4. RoadGraphNode

Základní stavební jednotka grafu silnic. Jedná se o třídu reprezentující jeden uzel grafu. Nese v sobě následující informace:

- ID uzlu.
- 3D pozici v prostoru.
- Seznam sousedních uzlů - protože se jedná o orientovaný graf, obsahuje jen ty uzly, do kterých spojující hrana vstupuje.
- Seznam uzlů, ke kontrole přednosti. Vozidlo, které se nachází v aktuálním uzlu pak prochází tento seznam a kontroluje si, zdali se v některém z těchto uzlů nenachází jiné vozidlo - tomu pak musí dát přednost a zastavit.
- Seznam uzlů ke kontrole při předjíždění. Chce-li vozidlo přijíždějící do příslušného uzlu předjíždět, zkontroluje si, zdali se v některém z těchto uzlů nenachází jiné vozidlo, aby nedošlo ke kolizi při předjíždění.
- Seznam vozidel, aktuálně přijíždějících do aktuálního uzlu.
- Hodnotu omezení rychlosti.
- Parametr určující, zdali je dovoleno v tomto uzlu předjíždět.

4.7.5. Zone

Třída reprezentující simulační okénko. Jedná se o jednoduchou strukturu, která reprezentuje prstenec pomocí dvou kružnic, respektive koulí, reprezentovaných polohou v prostoru a poloměry. Toto okno je v každém okamžiku simulace ve stejné poloze, jako je řízené vozidlo.

4.8. Konfigurovatelné parametry

Požadavkem projektu VRUT je, aby byly jednotlivé moduly ovladatelné a konfigurovatelné jednak z grafického rozhraní, které je implicitně dodáno knihovnou *wxWidgets* při

podědění generického modulu *Module*, a také z Javascriptových konfiguračních souborů. Modul *Traffic* není vývjimkou. V průběhu vývoje modulu se objevilo větší množství parametrů, konstant a atributů, které ovlivňují chování autonomně řízených vozidel. Také je zapotřebí dát cestu k souborům popisující vlastnosti trati a definující graf silnic.

Následuje seznam kategorií a parametrů, které jdou v GUI, nebo v Javascriptovém konfiguračním souboru nastavit.

Základní parametry modulu

sceneID - Parametr definující ID scény, ve které má probíhat simulace. V drtivé většině případů probíhá simulace pouze v jedné scéně, která je zvolena inicializačním modulem. Díky tomuto parametru jsme ovšem v modulu schopni získat instanci scény a manipulovat tak s jejími prvky.

trafficGraphFile - Definuje cestu k souboru nesoucí data o grafu silnic ve specifikovaném XML formátu.

worldFile - Definuje cestu k souboru definující parametry světa. Z tohoto souboru si modul zjistí informaci o tom, pod jakým identifikátorem je uložena geometrie silnice k vygenerování uzlů grafu silnic.

enabled - Přepínač, kterým je možno modul vypnout/zapnout.

displayGraph - Přepínač, kterým je možno skrýt/zobrazit jednoduchý geometrický model grafu silnic k ladění modulu a simulace.

controlledVehicle - Identifikátor vozidla, které je řízeno člověkem. Toto vozidlo není simulováno modulem. Jsou ovšem zpracovávány informace o jeho poloze a orientaci, aby byl modul schopen brát vpotaz i toto vozidlo, vyhýbat se mu, předjíždět jej a dát mu přednost na křižovatkách.

Parametry simulačního okénka

maxActiveCars - Číslo definující maximální počet autonomně řízených vozidel, které jsou schopny objevit se v simulaci.

spawnDistance - Vzdálenost v milimetrech definující minimální okruh, ve kterém nesmí být žádné vozidlo, aby bylo možno na dané místo vložit vozidlo nové.

localSpaceInnerRadius - Poloměr vnitřní koule simulačního *okénka* v milimetrech. V této oblasti jsou autonomně řízená vozidla normálně simulována. Žádné nové vozidlo však do této oblasti nemůže být vloženo.

localSpaceOuterRadius - Poloměr vnější koule simulačního *okénka* v milimetrech. V oblasti mezi vnitřní a vnější koulí jsou hledány body k umístění nového vozidla, splňuje-li situace podmínky, viz 3.8.

Parametry chování vozidla

wheelTurnStep - Maximální hodnota natočení volantu za jednu iteraci simulační smyčky. Hodnota v rozsahu $< 0; 1 >$.

throttleStep - Maximální hodnota změny intenzity sešlápnutí plynového pedáluza jednu iteraci simulační smyčky. Hodnota v rozsahu $< 0; 1 >$.

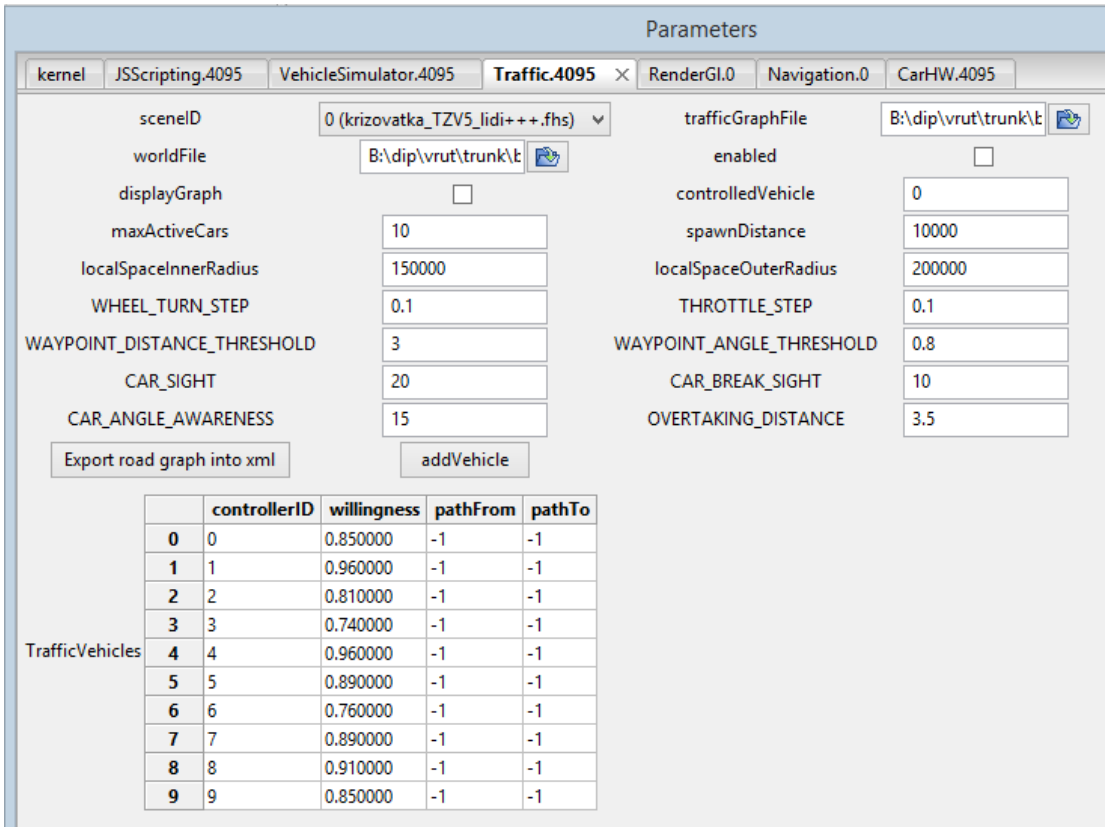
waypointDistanceThreshold - Vzdálenost, do jaké se musí vozidlo dostat k průjezdovému bodu, aby byl simulací považován za dosažený. V metrech.

waypointAngleThreshold - Nejmenší vzdálenost mezi průjezdovým bodem a vektorem orientace vozidla.

carSight - Délka kontrolní zóny *Adjust zone* vozidla, umístěné před vozidlem, v metrech. Je-li jiné vozidlo v tomto dosahu, vůz se snaží napodobit jeho rychlost.

carBreakSight - Délka kontrolní zóny *Slow down zone* vozidla, umístěné před vozidlem, v metrech. Je-li jiné vozidlo v tomto dosahu, vůz brzdí, aby zabránil kolizi.

carAngleAwareness - Úhel výseče kontrolní zóny. Viz obrázek 17.



Obrázek 26. Náhled grafického rozhraní modulu Traffic.

Tlačítka

Export road graph into xml - Tlačítko, které vyexportuje aktuální průjezdové body na trati do souboru, definovaného v parametru *trafficGraphFile*). Pokud soubor existuje, je přepsán.

Modul kromě těchto parametrů disponuje tabulkou (viz obrázek 26), ve které udržuje seznam vozidel k dispozici pro simulaci. Přes tuto tabulku lze nastavit konkrétním vozidlům parametr *willingness* a taky cestu pomocí počátečního a koncového průjezdového bodu, kterou mají v daném scénáři projíždět. Pokud cestu zadanou nemá, nebo ji již projel, jezdí po síti náhodně. Tabulka s vozidly nese tedy tyto hodnoty:

controllerID - unikátní identifikátor vozidla
willingness - Ochota, s jakou vůz dodržuje rychlost (viz sekce 3.3 Dodržování rychlosti)
pathFrom - startovní uzel definované cesty
pathTo - koncový uzel definované cesty

Ačkoliv jsou všechny parametry viditelné a upravitelné v grafickém rozhraní, nedoporučuje se je měnit po spuštění a za běhu simulace. Změna by mohla způsobit neočekávané chování a v určitých případech i pád modulu, nebo celé aplikace. Definice scénáře a nastavení všech atributů by mělo být provedeno před spuštěním modulu, ideálně pomocí Javascriptovým konfiguračním souborem, který je načten prostřednictvím VRUT modulu *JSScripting* při inicializaci aplikace a načítání ostatních modulů. Důvodem, proč jsou všechny parametry zpřístupněny uživatelským rozhraním je způsob, jakým je komunikace modulů VRUT a jejich možnost konfigurace přes konfigurační soubory implementována. Javascript s jádrem a moduly komunikuje pomocí událostí, vázaných na tyto GUI prvky pomocí regist

4.9. Tvorba scénáře

Implementace uživateli nabízí vytvořit si simulační scénář před samotnou simulací. Může si nadefinovat iniciální polohy vozidel, jejich ochotu (*willingness*) a cesty, kudy mají scénu projíždět. Nakonec modulu pomocí parametru *controlledVehicle* nastaví vozidlo které je řízeno ručně a simulace může začít.

Cesta je vozidlům zadání pouze počátečním a koncovým projížděcím uzlem. Modul pak prochodem grafu silnic pomocí Dijkstra algoritmu najde nejkratší cestu mezi těmito uzly a uloží ji do vozidla.

Výsledný scénář je přehrán, respektive simulován při spuštění simulace pomocí tlačítka (události) *enabled*. Vozidla po splnění své cesty, tedy dokončení scénáře pokračují v jízdě. Tam, kde je třeba se rozhodnout, například na rozcestích a křizovatkách, je cesta vybrána náhodně.

4.10. Propojení simulace a simulátoru

Při implementaci jsem narazil na závažný problém. Aktuální datová struktura reprezentující síť silnic ani simulátor nemají vhodný prostředek ke sledování manuálně řízeného vozu. Je známá pouze jeho poloha a orientace a pouze řidič sám ví, kam má jet a kam skutečně jede. Simulovaná vozidla mají oproti tomu tu výhodu, že mají trasu buď danou, nebo se pro ni rozhodnou v příslušný okamžik a tato informace je přístupná všem okolním vozům. Okolní doprava je řízena hlavně grafem silnic a plně spoléhá na korektnost jeho definice.

V Klasické jízdě v koloně, nebo při předjíždění je možno sledovat polohu řízeného vozu stejně, jako je tomu u autonomních vozidel, protože se jedná o jízdu po triviální cestě, reprezentované jedinou sérií uzlů. Podle směru a rychlosti se tak dá snadno odhadnout, kam řidič jede. To už není možné na křižovatkách. Struktura a rozložení uzlů v prostoru je tak komplexní, že autonomní vozidla mohou jen hádat, kam řidič se svým vozem pojede a mohou se mu pokusit dát přednost.

5. Testování a výsledky

K demonstraci a testování modulu Traffic bylo vytvořeno několik simulační scénářů, které využívají veškerou naimplementovanou funkcionalitu modulu. Testování proběhlo na stolním počítači s procesorem Intel Core i5-3570K, 8GB RAM 1600MHz a grafickou kartou NVIDIA GeForce GTX 660. Projekt je zkompilován pod operačním systémem Microsoft Windows 8.1, programem Visual Studio Professional 2012. K automatizaci překladu programu byl použit CMake, verze 2.8.11.

Testy byly zaměřeny na spolehlivost simulace, průjezd autonomně řízených vozidel křižovatkami, jízdu v koloně a předjíždění. Zátěžová zkouška, která byla provedena v rámci testování, měla určit počet vozidel, který je možné simulovat v reálném čase. Dále se testovala schopnost okolních vozidel brát ohled na řízené vozidlo, což byl v průběhu vývoje nejvážnější problém implementace.

Následující sekce popisují jednotlivé testovací scénáře a zjištěné výsledky. V průběhu testování se objevilo několik chyb a problémů. Tato kapitola mimo jiné popisuje pokusy je vyřešit a jejich úspěšnost.

5.1. Jízda v koloně

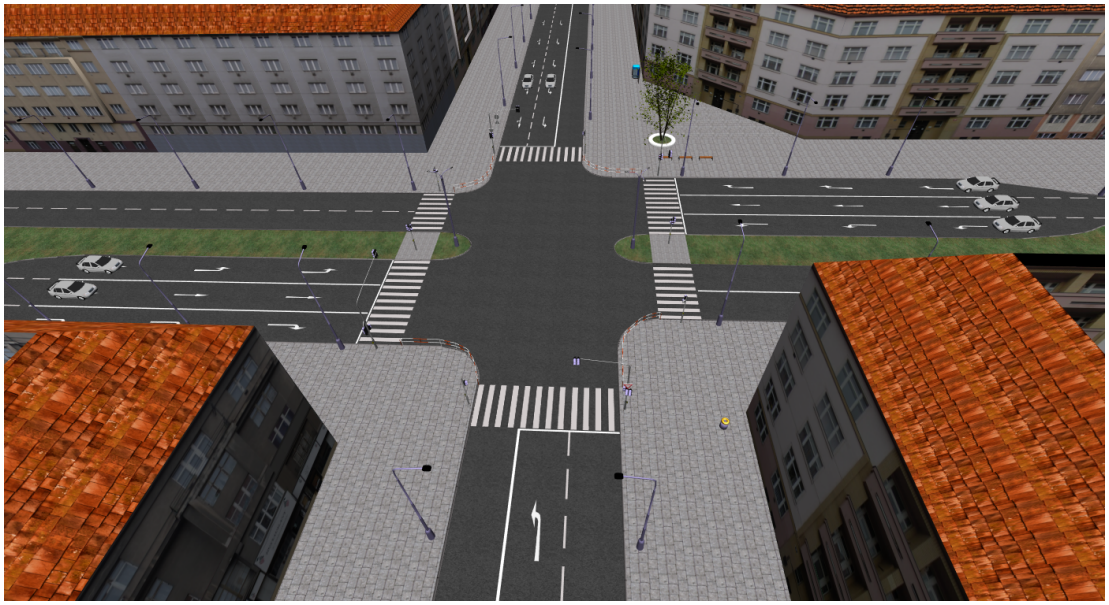
Tento scénář je nejjednodušší k realizaci. Vozidla byla nastavena do iniciální polohy tak, aby stály za sebou na rovné cestě. K testování byla použita trať alpine-1, získané z open-source závodního simulátoru. Jedná se o okružní trať, kde já jsem ale použil jen krátký úsek, na kterém je vozovka rovná a široká a vytvořil na něm malý okruh k testování předjíždění a jízdy v koloně.

Test ukázal, že vozidla dokážou jet plynule za sebou, dodržovat rozestupy (dané parametry *carSight* a *carBreakSight*, viz sekce 4.8) a napodobovoat rychlost vozu před sebou.

5.2. Křižovatka

Firma Škoda Auto poskytla k vývoji a testování trať obsahující komplexní křižovatku (náhled viz obrázek 27). Na této mapě prošel modul těžkou zatěžkávací zkouškou v oblasti řešení předností. Ukázalo se, že velmi záleží na správném pozicování průjezdových bodů, protože vozy tyto dodržují jen s určitou odchylkou a tak se zdá samotný průjezd vozidla uzlem být "odbytý". Také je důležité korektně zadefinovat přednosti v grafu silnic, protože vozidlo dává přednost jen a pouze na základě této informace.

Ukázalo se, že autonomně řízená vozidla mají značný problém s hlídáním pozice a přednosti manuálně řízeného vozu, což bylo zmíněno v sekci 4.10 Propojení simulace a simulátoru. Problém částečně vyřešila následující úprava: Využil jsem funkce, která hledá nejvhodnější kandidáty z úzlů grafu silnic pro autonomní vozy pro navedení na trať. Ta najde bod, který je před vozidlem, dostatečně blízko a s co nejmenší výchylkou od směru vozidla. Takto jsem v některých případech schopen určit, kam chce řidič se svým vozem pravděpodobně jet a dávat na něj pozor při simulaci okolních vozidel.



Obrázek 27. Náhled křižovatky, dodané firmou Škoda Auto. Tato trať byla použita při testování průjezdu křižovatkou. Copyright Škoda Auto.

5.3. Předjíždění

Podobně jako všechny ostatní řešené jízdní úkony je i předjíždění striktně závislé na korektně definovaném grafu silnic. Uzly musí mít zadáno, zdali se na nich dá předjíždět. Dále musí mít správně zadefinovaný seznam uzlů, ve kterých nesmí být jiné vozidlo, aby mohl předjížděcí manévr proběhnout.

Pokud jsou tyto podmínky splněny, předjížděcí manévr proběhne správně. Testování ukázalo, že simulace má problém s manévrem, pokud je rozdíl rychlostí mezi předjížděným a předjížděcím vozidlem příliš vysoká. Předjíždějící vůz jede vůči předjíždějícímu příliš rychle a nestihá dostatečně rychle zatáčet k příslušným předjížděcím uzlům. Problém částečně vyřešila podmínka, která omezuje maxímální rychlost při předjíždění maximální dovolenou rychlostí na daném úseku cesty.

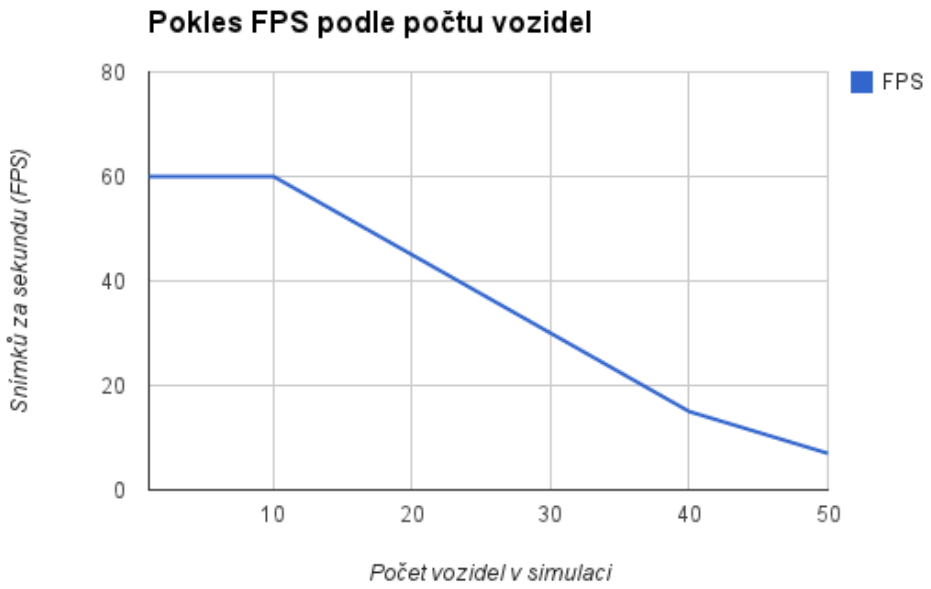
5.4. Simulační okénko

Hlavním optimalizačním faktorem simulace je simulační okénko (viz sekce 4.6). Rozsah simulace je dán třemi parametry:

- Maximální počet aktivních vozidel
- Vnitřní poloměr simulační zóny
- Vnější poloměr simulační zóny

Manipulace s těmito třemi parametry ukázala, že hlavní výpočetní zátěž nespočívá v ovládání okolní dopravy, nýbrž ve fyzikální simulaci vozidel samotných. Modul ovšem nemá možnost, jak tuto simulaci vypnout pro neaktivní vozy. Dokáže jen skrýt jejich geometrii.

Ačkoliv se tedy implementace snaží optimalizovat běh aplikace pomocí simulačního okénka, které zajistí, aby nebyly simulovány vozidla mimo dosah řidiče a jejich geometrie byla skryta, modul VehicleSimulator vozy i nadále bere v potaz a počítá fyzikální model a kolizní detekce. Procesorové zátěži to ulehčí jen málo. Zatěžkávací testy ukázaly, že je na dané platformě možno simulovat cca 30 vozidel, aby mohla simulace probíhat



Obrázek 28. Výsledek zatěžkávacího testu reprezentovaný poklesem FPS v závislosti na počtu vozidel v simulaci.

v reálném čase (viz obrázky 28 a 29). Větší množství vozidel má za následek pokles počtu snímků za sekundu, tedy trhanější obraz, ale hlavně nárůst časových prodlev mezi příchozími událostmi. S příliš dlouhými intervaly nemá modul kontrolu nad vozidlem, a to je pak naváděno nepřesně. S tímto zpožďováním souvisí distribuce událostí v jádře a jejich zpracování v příslušných modulech. Fronta událostí se zahltí, protože události přicházejí častěji, než je aplikace stíhá zpracovávat.

30 simulovaných vozidel je však dostačující počet s ohledem na to, že tato vozidla se budou nacházet v blízkém okolí řízeného vozu. Cílem simulačního okénka tedy nakonec nebyla ani tak úspora výpočetního výkonu, jako spíš koncentrace maximálního možného počtu vozidel do zóny zájmu, tedy okolí řízeného vozu.

5.5. Kolize

Model ovšem není dokonalý a občas dochází ke kolizím. Buď z důvodu špatného navedení na referenční bod a následný náraz do překážky, nebo špatným vyhodnocením předností na křižovatce. Překvapivě efektivně funguje řešení kolizí popsané v sekci 3.9 Couvání. Vozidla skutečně zhodnotí příapdnou situaci jako kolizi a doslova se pokoušejí ze situace vycouvat. Ve většině případů se jim to podaří.

Obrázek 29. Výsledek zatěžkávacího testu reprezentovaný nárůstem prodlevy mezi příchozími událostmi pro jedno vozidlo v závislosti na počtu vozidel v simulaci. Požadovaný interval je 100ms.

6. Závěr

Cílem této práce bylo prostudovat a analyzovat základní metody simulace dopravy ve virtuálním prostředí. Na základě této analýzy jsem měl za úkol implemenotovat zjištěné metody pro simulaci výše uvedených dějů do existujícího software pro simulaci jízdy VRUT (Virtual Reality Universal Toolkit) od společnosti Škoda Auto.

Podařilo se vytvořit pokročilý model k simulaci okolních dopravních dějů. Tento model jsem jako modul naimplementoval do aplikace VRUT. Požadavky ze zadání se podařilo splnit do takové míry, jakou nabízí rozhraní VRUT. Jelikož se jedná o rozsáhlý projekt, na kterém v průběhu několika let pracovalo velké množství vývojařů, systém a vnitřní mechanizmy simulace a řízení jsou velmi komplexní.

Samotný projekt VRUT byl původně diplomovou prací studenta Fakulty elektrotechnické na Českém vysokém učení technickém v Praze. Software ale postupem času nabral na takové komplexnosti, že vývoj samotných modulů do něj se stal netriviálním, a katedra Počítačové grafiky a interakce nabízí studentům implementace modulů jako témata diplomových prací.

Cením si však příležitosti pracovat na projektu, který se dnes používá v oddělení virtuální techniky firmy Škoda Auto. V rámci práce jsem měl navíc možnost navštívit sídlo firmy Škoda Auto a prohlédnout si jejich prostředí.

6.1. Možnosti rozšíření modulu a plány do budoucna

Jelikož se jedná o nezávislý modul projektu VRUT, existuje řada možností, jak funkcionalitu rozšířit jak na úrovni modulu, tak na úrovni celého projektu. V případě, že by jiný student chtěl pokračovat na této práci, nabízím zde několik nápadů, jak modul dále rozšiřovat.

6.1.1. Světelná signalizace

Modul umí simulovat průjezd křižovatkou, simulátor ovšem nepodporuje světelnou signalizaci, tedy semafory. Do budoucna by tak stálo za to prozkoumat metody řízení provozu pomocí semaforů. V samotné implementaci by tak vozidlům přibyla další omezující podmínka při průjezdu křižovatkou - stav příslušného semaforu.

6.1.2. Směrová světla

Simulátor nedisponuje funkcionalitou směrových světel (blinkrů) na vozidlech. Proto modul při simulaci okolní dopravy neumí brát ohled na řidiče na křižovatkách. Okolní vozidla mají vzájemné povědomí o tom, kam které vozidlo jede, protože tuto informaci mají uloženou ve své instanci a je zpřístupněna ostatním. Problém nastává s manuálně řízeným vozidlem. Simulované vozy tedy mohou jen hádat, kam na křižovatce řidič pojede. Tento problém by vyřešila implementace směrových světel.

6.1.3. Makroskopický model dopravy

Modul Traffic simuluje vozidla pouze v určitém okruhu okolo řízeného vozu. Tento dosah je dán simulačním okénkem. Do něj jsou vozy generovány pouze na základě parametru hustoty provozu. Simulátor simulaci vozidel mimo tuto zónu vůbec neřeší. To může způsobit problém v případě, že by vznikl požadavek na VRUT, aby podporoval možnost jízdy více řidičů. Musel by se změnit přístup modulu Traffic. Řešením by bylo naimplementovat model, který je popsán v článku [1], který kombinuje mikro- a makroskopický model simulace dopravy a popisuje, jakým způsobem vyřešit přechod mezi nimi.

6.1.4. Interaktivní editor grafu silnic

Nejvíce by si však modul zasloužil uživatelsky přívětivý grafický editor grafu silnic. Nyní se cesty musejí definovat prakticky ruční úpravou příslušného XML souboru a opakovaným spouštěním simulace. Interaktivní editor by mohl být nástroj k vytváření, úpravě a exportu grafu silnic v průběhu simulace.

Příloha A.

Přílohy

A.1. Obsah přiloženého DVD

Práce obsahuje DVD se zdrojovými kódy, připravenými scénáři a s tímto textem v digitální podobě, které je přiloženo na vnitřní straně zadní desky. Umístění důležitých souborů je popsán zde:

```
/
├── text/ ..... Text diplomové práce v digitální podobě
├── vrut/ ..... Adresář implementace
│   ├── modules/
│   │   └── traffic/ ..... Zdrojové kódy k této diplomové práci
│   ├── bin/ ..... Přeložená aplikace VRUT
│   │   ├── vrut.exe ..... Spustitelná aplikace
│   │   ├── traffic_crossroad_scenario.bat ..... Scénář
│   │   ├── traffic_overtake_scenario.bat ..... Scénář
│   │   ├── traffic_no_overtake_scenario.bat ..... Scénář
│   │   └── traffic_heavy_load.bat ..... Scénář
│   └── data/ ..... Data potřebná k simulaci
└── videos/ ..... Videa s ukázkou některých scénářů
```

Binární soubory u sebe mají vytvořené scénáře. Tyto scénáře lze spouštět jednoduchými dávkovými soubory .bat. Simulace se následně spustí kliknutím na "runScript"a v záložce zaškrtnutím parametru "enabled". Následuje seznam a popis přiložených scénářů:

Traffic overtake scenario - Jednoduchý scénář, demonstrující předjíždění na volné jednosměrné trati a jízdu v koloně. Obsahuje 4 vozidla umístěná za sebou. Po spuštění objíždí krátký okruh a rychlejší vozy se pokoušejí předjíždět pomalejší

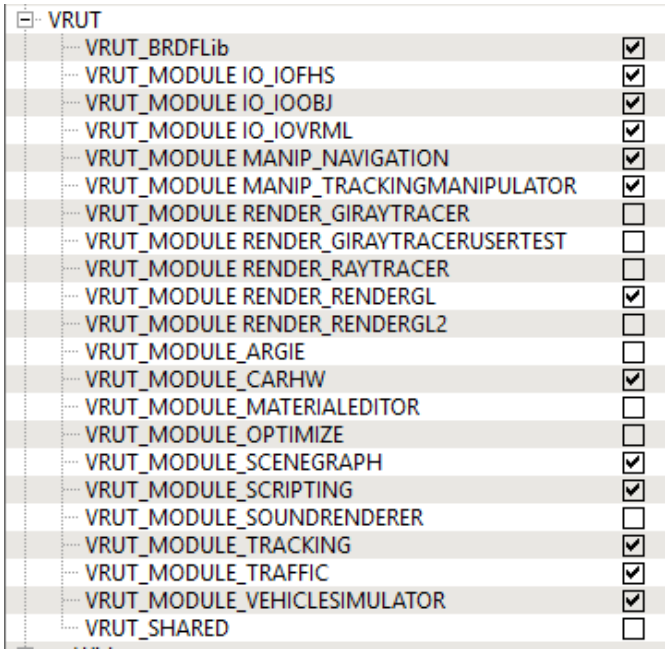
Traffic no-overtake scenario - Scénář, kde vozidlo nepředjede z důvodu blokování vedlejšího levého pruhu jiným vozidlem.

Traffic crossroad scenario - Demonstrace průjezdu křižovatkou, obsahující vůz v každém jízdním pruhu k důkladnému otestování.

Traffic heavy load - Zátěžový test. Simulace inicializuje 30 vozidel do scény a pokouší se je řídit.

A.2. Uživatelská příručka

Projekt obsahuje klasické *makefile* soubory, nebo disponuje *CMake* konfigurací. Detailní popis jednotlivých přístupů je popsán v [7] Dokumentaci aplikace VRUT v. Já jsem vyvíjel na platformě Windows a při práci použil CMake soubory. K sestavení a kompilaci projektu je potřeba mít nainstalován program CMake, Visual Studio. V příkazové řádce jsou zadány následující příkazy:



Obrázek 30. Doporučené nastavení modulů v CMake.

```
cd vrut
mkdir BUILD
cd BUILD
cmake-gui ..
```

Tím se spustí grafické rozhraní CMake, ve kterém je potřeba nakonfigurovat cesty k příslušným knihovnám, jako jsou ODE, GLUT atd. a vybrat moduly, které se budou kompilovat. Obrázek 30 naznačuje kombinaci modulů, potřebných k běhu aplikace spolu s modulem Traffic.

Nakonec je potřeba nechat CMake vygenerovat soubory projektu. Ty jsou následně nalezeny ve vytvořené složce *BUILD* a pomocí souboru *VRUT.sln* je možno spustit projekt ve Visual Studiu.

A.3. Návod ke konfiguraci scénáře

Veškeré možnosti nastavení scénáře je možno ovlivnit v javascriptové konfiguraci. Několik ukázkových příkladů scénářů je uvedeno v adresáři *vrut/bin/* na přiloženém DVD:

- traffic_overtake_scenario.js
- traffic_no_overtake_scenario.js
- traffic_crossroad_scenario.js
- traffic_heavy_load.js

K nastavení konfigurovatelných parametrů modulu Trafic se používá funkce *updateParamValue*, kde první parametr je "Traffic", druhý je příslušný parametr konfigurace a třetím parametrem je hodnota:

```
var trackCFGFile = "krizovatka_TZV5_lidi2.xml";
var graphFile = "trafficGraph.xml";
updateParamValue("Traffic", "WHEEL_TURN_STEP", "0.1");
updateParamValue("Traffic", "THROTTLE_STEP", "0.1");
```

```
updateParamValue("Traffic", "WAYPOINT_DISTANCE_THRESHOLD", "3");
updateParamValue("Traffic", "WAYPOINT_ANGLE_THRESHOLD", "0.8");
updateParamValue("Traffic", "CAR_WIDTH", "2");
updateParamValue("Traffic", "CAR_SIGHT", "20");
updateParamValue("Traffic", "CAR_BREAK_SIGHT", "10");
updateParamValue("Traffic", "CAR_ANGLE_AWARENESS", "15");
updateParamValue("Traffic", "OVERTAKING_DISTANCE", "3.5");
updateParamValue("Traffic", "trafficGraphFile", graphFile);
updateParamValue("Traffic", "worldFile", trackCFGFile);
updateParamValue("Traffic", "sceneID", sceneID);
updateParamValue("Traffic", "maxActiveCars", 10);
updateParamValue("Traffic", "controlledVehicle", 0);
updateParamValue("Traffic", "spawnDistance", 10000);
updateParamValue("Traffic", "localSpaceInnerRadius", 150000);
updateParamValue("Traffic", "localSpaceOuterRadius", 200000);
updateParamValue("Traffic", "displayGraph", "0");
```

K inicializaci atributů jednotlivých vozidel je potřeba zavolat stejnou funkci, která jako druhý parametr nese název tabulky *vehicles* a třetí obsahuje kromě hodnoty i informaci o sloupci a řádku tabulky:

```
var car_id = 0;
// prida vozidlo
updateParamValue("Traffic", "addVehicle", "1");

// nastavi vozidlu 0 prislusnou willingness
var w = 0.7;
updateParamValue("Traffic", "TrafficVehicles", "0,"+car_id+","+ w);


// nastavi vozidlu 0 cestu pro scenar.
// pocatecni bod ma ID 315 a koncovy 270
updateParamValue("Traffic", "TrafficVehicles", "2,"+car_id+","+315);
updateParamValue("Traffic", "TrafficVehicles", "3,"+car_id+","+270);
```

Literatura

[1] Mikael Adlers Pontus Matstoms Johan Janson Olstam Jan Lundgren. "A Framework for Simulation of Surrounding Vehicles in Driving Simulators". In: *ACM Trans. Model. Comput. Simul.* (2008).

[2] K. A. Hawick G. Kotushevski. *A Review of Traffic Simulation Software*. Tech. zpr. CSTN-095. Albany, North Shore 102-904, Auckland, New Zealand: Computer Science, Massey University, 2009. URL: `http://www.massey.ac.nz/~kahawick/cstn/095/cstn-095.pdf`.

[3] David Tichý. "Simulátor jízdy městem". Dipl. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2006.

[4] Espié Stéphane Olstam Johan. "Combination of autonomous and controlled vehicles in driving simulator scenarios". In: *WORKSHOP ON TRAFFIC MODELING: TRAFFIC BEHAVIOR AND SIMULATION,2008*. TU Graz, 2008.

[5] Huihuan Qian Tin Lun Lam a Yangsheng Xu. "Behavior-based Steering Control for Four Wheel Independent Steering Vehicle". In: *International Conference on Robotics and Biomimetics* (2009).

[6] Václav Kýba. "Modulární 3D prohlížeč". Dipl. České vysoké učení technické v Praze, Fakulta elektrotechnická, 2008.

[7] *Dokumentace aplikace VRUT*. České vysoké učení technické v Praze, Fakulta elektrotechnická, Škoda-auto a.s.

[8] Biagio Ciuffo Vincenzo Punzo. "Integration of Driving and Traffic Simulation: Issues and First Solutions". In: *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS* 12.2 (červ. 2011).

[9] Centre for Applied Informatics (ZAIK) a the Institute of Transport Research at the German Aerospace Centre: Simulation of urban mobility - sumo website. Available at http://sumo.sourceforge.net/ (2014).

[10] Quadstone Paramics: Quadstone paramics website. Available at http://www.paramics-online.com/ (2014).

[11] Treiber M.: Microsimulation of road traffic applet. Available at http://www.traffic-simulation.de/ (2014).

[12] TSS - Traffc Simulation Systems: Aimsun website. Available at http://www.aimsun.com/site/ (2014).

[13] Trafficware: Trafficware website. Available at http://www.trafficware.com/ (2014).

[14] James Cremer Peter Willemsen Hongling Wang Joseph K. Kearney. "Steering Behaviors for Autonomous Vehicles in Virtual Environments". In: *Proceedings of the IEEE Virtual Reality 2005 (VR'05)* (2005).

[15] Omar Ahmad Yiannis Papelis. *A Comprehensive Microscopic Autonomous Driver Model for Use in High-Fidelity Driving Simulation Environments*.