VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

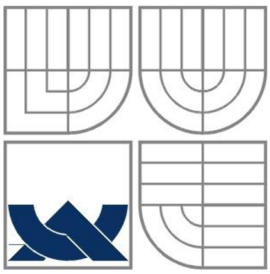
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOŠKOLA – PRAVIDLA SILNIČNÍHO PROVOZU

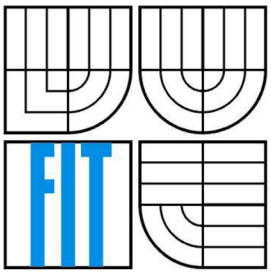
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE Bc. Jiří Porč
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

AUTOŠKOLA – PRAVIDLA SILNIČNÍHO PROVOZU

DRIVING SCHOOL – RULES OF THE ROAD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. Jiří Porč

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Peter Chudý, Ph.D. MBA

BRNO 2010

Zadání diplomové práce/7137/2009/xporcj01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Porč Jiří, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Autoškola - pravidla silničního provozu**
Driving School - Rules of the Road
Kategorie: Počítačová grafika

Pokyny:

1. Nastudujte problematiku simulace silničního provozu a metodiku kontroly dodržování pravidel silničního provozu. Analyzujte platná pravidla systému a vytvořte jejich seznam dle priority implementace.
2. Vytvořte architekturu systému simulátoru a uživatelského rozhraní.
3. Vytvořte (případně zajistěte z vhodných zdrojů) základní 3D modely a funkční 3D engine, které budu použity v simulátoru.
4. V simulátoru implementujte základní fyzikální model, sledování dodržování prioritních pravidel a reprezentaci pohybu účastníků provozu a jejich vzájemné interakce.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu.

Literatura:

- dle pokynů vedoucího

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1, 2 a část 3.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Chudý Peter, Ing., Ph.D. MBA**, UPGM FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2
L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá návrhem simulátoru silničního provozu. Je zde popsána problematika simulačních metod, návrhu modelu města, silniční sítě a jejich využití při implementaci v dopravním simulátoru. Práce dále vysvětluje principy a postupy pro vytvoření modelu města v editoru a analyzuje dopravní pravidla, bez kterých by simulátor tohoto typu nebylo možné vytvořit. Vytvořený simulátor pro svůj běh využívá různé enginy. Bez jejich správného nastavení by nebylo v práci možné pokračovat, proto je část práce věnována právě instalaci použitých enginů. Jsou zde také rozebrány postupy pro použití 3D modelů a textur.

Abstract

This diploma thesis focuses on a traffic simulator design. The matter of simulation methods is described here as well as a project of a town model, road system and its usage in an implementation in a traffic simulator. The work further explains the principles and techniques for creation of the town model in an editor and it analyzes traffic rules that are necessary for the creation of the simulator of this type. The created simulator uses various engines for its functioning. It would not be possible to continue in further work without their proper adjustment. That is why the installation of the used engines is described in the thesis. Principles of 3D model and texture usage are also explained.

Klíčová slova

Autoškola, pravidla silničního provozu, simulátor, simulace provozu, grafický engine, fyzikální engine, detekce kolizí, 3D model, textura.

Keywords

Driving School, Rules of the Road, simulator, traffic simulator, graphics engine, physics engine, collision detection, 3D model, textures.

Citace

Porč Jiří: Autoškola – pravidla silničního provozu, diplomová práce, Brno, FIT VUT v Brně, 2010

Autoškola – pravidla silničního provozu

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Petra Chudého, Ph.D. MBA.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Bc. Jiří Porč
5. 1. 2009

Poděkování

Chtěl bych poděkovat vedoucímu mé diplomové práce panu Ing. Petru Chudému Ph.D. MBA. za jeho vedení a cenné rady, které mi poskytl při mnoha konzultacích. Dále pak své rodině za podporu a zázemí nejen při tvorbě této práce, ale při celém studiu.

Obsah

1 Úvod

Už v minulosti, kdy došlo k rozvoji měst, vznikla potřeba dopravovat se. Z počátku se jednalo zejména o přepravu mezi městy, dopravu zboží na trh, později o přesuny armád a armádního vybavení. Dnešní dobu si bez dopravy a propracované dopravní infrastruktury neumíme představit. S jejich rozvojem se stále zvyšuje důraz na bezpečnost pohybu po komunikacích. Aby ale pohyb po pozemních komunikacích byl možný a bezpečný, je nutné stálé dodržování určitých zásad a pravidel. Bez jejich dodržování by nebylo cestování vůbec možné, docházelo by k neustálým dopravním nehodám, nikdo by nevěděl, jak se na pozemních komunikacích chovat, po které straně vozovky se pohybovat, jak projíždět křižovatky nebo jak reagovat na semafory. Platnost těchto pravidel nemusí být globální, ale mohou platit vždy v určitém místě, nejlépe pro uzavřenou skupinu lidí. Například některé ostrovní státy místo vpravo jezdí po vozovce vlevo, různé státy mají různé maximální rychlosti a různě vypadající dopravní značky. I přes tyto většinou drobné odlišnosti je hlavním cílem dopravních pravidel to, aby se účastníci silničního provozu dokázali domluvit a uspořádat a aby nedocházelo ke zbytečným ztrátám na majetku a lidských životech.

Protože se státní instituce snaží o snížení nehodovosti, dochází občas ke změnám legislativy a úpravám dopravních pravidel. „Zavedení celoročního povinného svícení podle zkušeností z jiných zemí velmi pravděpodobně sníží počet nejvážnějších nehod (čelní nárazy mimo obec, srážky při odbočování vlevo), u řady dalších se mohou zmírnit následky. Na poklesu počtu obětí dopravních nehod v posledních dvou letech se povinné svícení v zimním období již projevilo. Odborníci odhadují, že počet těžkých nehod s nejvážnějšími následky by mohl klesnout dokonce až o 10 až 15 procent. V našich podmínkách to může představovat až 100 zachráněných lidských životů.“ [18]. „Různé průzkumy prokázaly, že telefonování za jízdy zvyšuje riziko nehody obdobně jako například alkohol. Riziko nehody stoupá na čtyřnásobek (!), přičemž nebezpečnější než samotné držení přístroje je ztráta pozornosti a koncentrace na jízdu. Reakce jsou pomalejší až o jednu a půl vteřiny, což představuje prodloužení brzdné dráhy o desítky metrů. Při nejvyšší dovolené rychlosti, například na dálnici 130 km/h, se takto prodlouží brzdná dráha až o 50 metrů! Nesoustředění řidiči podle zjištění psychologů přehlížejí značky a mnohdy nedají přednost v jízdě.“ [19].

„Na českých silnicích zemřelo od počátku roku 1990 do konce roku 2006, tzn. za posledních 17 let, **24312 lidí** (do 30 dnů po dopravní nehodě), tedy v průměru každý rok v tomto období zemřelo na silnicích 1430 lidí. Celospolečenské ztráty v důsledku dopravních nehod se pohybují nyní ročně okolo padesáti miliard korun.“ [13]. V roce 2008 bylo v České republice nahlášeno více než 160 tisíc dopravních nehod, což vychází v průměru na 439 nehod každý den roku [14]. Toto číslo je dosti vysoké, ke snížení počtu nehod a následných materiálních škod by jistě bylo prospěšné, kdyby řidiči měli více zkušeností a lépe ovládali svoje vozidla. I v těchto situacích lze využít simulace a simulační modely. Mezi velké výhody využití simulačních modelů při výuce řidičů patří jejich naprostá bezpečnost. Při havárii v automobilovém simulátoru nedochází k žádným škodám ani na majetku, ani na životech, havárie v simulátoru jsou součástí jejich simulace. Použití simulátoru je tedy naprosto bezpečné

i pro začínající řidiče, kdy na simulátoru mají možnost si vyzkoušet základní vlastnosti automobilu, dodržování dopravních pravidel a chování ostatních řidičů na vozovce před tím, než usednou do opravdového vozidla.

Tato diplomová práce se zabývá problematikou simulace silničního provozu a metodikou kontroly dodržování pravidel silničního provozu. Součástí je také analýza platných pravidel, vytvoření architektury simulátoru s možností kontroly těchto pravidel, reprezentace pohybu účastníků provozu a jejich vzájemná interakce. Vytvářený simulátor jsem se rozhodl založit na práci [6], která se zabývala problematikou simulátoru 3D autoškoly. V této práci byla nastíněna problematika simulací a rozpracován základní koncept simulátoru autoškoly. Součástí aplikačního výstupu bylo vozidlo s možností jízdy po městě a dodržování několika základních pravidel silničního provozu bez dalších účastníků provozu. Simulátor 3D autoškoly [6] bylo potřeba rozšířit o kontrolu dodržování prioritních pravidel a semaforů, dále bylo do této práce nutné přidat další účastníky silničního provozu a umožnit jednodušší vytváření města.

Tato práce rozebírá problematiku simulací silničního provozu, proto se první kapitola věnuje principům modelování a simulací. Jsou v ní vysvětleny základní pojmy a principy z oblasti simulací. Tato kapitola popisuje výhody a nevýhody simulačních metod.

Další kapitola je věnována návrhu modelu města. V původní práci nebyl model města propracován, proto bylo nutné navrhnout a vytvořit novou reprezentaci modelu města. Aby šlo model města jednoduše měnit, je výhodné město modelovat v nějakém vizuálním editoru. Vytvoření takového editoru by ale vydalo na samostatnou diplomovou práci, proto jsem se rozhodl využít externího editoru JOSM [8]. Tato kapitola dále popisuje pravidla pro vytvoření města v tomto editoru tak, aby je bylo možné vytvořeným simulátorem zpracovat a následně zobrazit. Podle těchto pravidel tedy si může každý uživatel vytvořit svoje město a následně se v něm ve vytvořeném simulátoru projet.

Třetí kapitola analyzuje dopravní pravidla potřebná pro vytvoření simulátoru. Bez těchto pravidel by simulátor nemohl vzniknout. Jsou zde popsány a zobrazeny typy dopravních značek, princip fungování semaforů. Kapitola dále popisuje princip průjezdů přes křižovatky všech typů a vysvětluje chování při jejich průjezdu.

Další navazující kapitola rozebírá návrh simulátoru po jednotlivých částech. Je zde vysvětlen princip jízdy vozidel po hraně a dodržování bezpečné vzdálenosti, princip zamykání a odemykání zámků a jsou zde popsána řešení průjezdů křižovatkami. Uvnitř této kapitoly je rozebrán simulátor 3D autoškoly [6], jsou zde popsány jeho vlastnosti, ale také slabá místa a nedostatky, které je nutné odstranit.

Aby simulátor byl co nejreálnější, jsou v něm použity 3D modely a textury. Této problematice se věnuje pátá kapitola. V této kapitole jsou rozepsány některé nástroje potřebné pro tvorbu a editaci modelů i textur a postup k umožnění jejich načtení ze simulátoru. Simulátor 3D modely načítá ve speciálním formátu, proto je před jejich použitím nutné modely do tohoto formátu převést.

Ve výsledném simulátoru jsou použity grafický engine OGRE [16] a fyzikální engine Bullet [23]. Bez správného zprovoznění těchto enginů není možné dále pokračovat v mnou vytvořeném simulátoru. Protože instalace a zprovoznění použitých enginů bylo dost komplikované, je na tento problém zaměřena šestá kapitola. Jsou v ní popsány použité enginy,

včetně jejich verzí. Dále je tu vysvětleno, jak je možné tyto enginy zprovoznit a použít, jak musí být nastaveno vývojové prostředí a kde je možné hledat další informace.

Poslední kapitolou je závěr. Tato kapitola shrnuje celý smysl práce, popisuje význam jednotlivých kapitol a možností rozšíření.

2 Simulace

Kapitola se zabývá vysvětlením základních pojmů z oblasti modelování a simulací. Je zde popsáno základní rozdělení, jejich hlavní výhody a nevýhody a také základní princip realizace.

Simulace je obecně chápána jako napodobování objektů či modelů, kdy se snažíme o zachycení a napodobení hlavních vlastností nebo chování napodobovaného systému. Se simulací se setkáváme i v běžném životě, pro nás je ale více důležitá simulace z hlediska techniky, tedy počítačová simulace. Ve zbytku práce pod pojmem simulace budeme uvažovat počítačovou simulaci. Simulace sice existovaly již před nástupem počítačů, ale jejich možnosti byly dost omezené.

2.1 Počítačová simulace

Pro pochopení principu simulace zavádí literatura [1, 2] několik základních pojmů.

- **Systém** může být nadefinován jako soubor několika částí, které mezi sebou mají určitý vztah. Systém můžeme rozdělit například na reálný (části skutečného světa) a nereálný (například počítačové hry) nebo na statický (nemění své vlastnosti v čase) a dynamický (proměnlivý v čase). Z hlediska simulace jsou nejzajímavější dynamické systémy.
- **Model** se snaží napodobit systém jiným systémem, nejčastěji počítačovým programem. U modelu reálného systému se zpravidla jedná o abstrakci, nově vzniklý model obsahuje jen důležité informace pro účel modelování.
- **Modelování** je proces sestavení nového modelu na základě informace o systému, který chceme namodelovat. Úroveň abstrakce a kvalita nového modelu ovlivní výsledek simulace.
- **Simulace** je metoda pro získávání znalostí o systému experimentováním s jeho modelem. Aby simulace co nejvíce korespondovala se systémem, musíme simulaci několikrát opakovat s různě nastavenými parametry. Cílem simulace je získání informací o zkoumaném systému.
- **Reálný čas** probíhá ve skutečném (reálném) systému.
- **Modelový čas** určuje čas běžící při simulaci modelu. Rychlost plynutí modelového času nemusí být oproti reálnému stejná, může být zrychlen nebo zpomalen.
- **Strojový čas** je čas spotřebovaný na vykonání programu. Závisí na složitosti modelovaného systému a na vlastnostech modelu.

Při počítačové simulací tedy zkoumáme vlastnosti nějakého (většinou reálného) systému na základě experimentů s jeho matematickým modelem. Tyto vlastnosti můžeme využít k vyhodnocení chování zkoumaného systému za určité situace (podle vstupních parametrů).

2.2 Výhody a nevýhody simulačních metod

Složitost simulace, kterou jsme schopni na počítači provést, je limitována pouze jeho výpočetním výkonem. Počítače se stále stávají výkonnějšími, a tudíž se zvětšují možnosti pro využití simulací. Mezi hlavní výhody simulací v porovnání s experimenty v reálném světě jistě patří nižší finanční náklady, možnost změny délky provádění experimentu a bezpečnost. Cena oproti experimentům v reálném světě je v některých případech mnohonásobně nižší. Jedná se zejména o případy, kdy náklady na reálné provedení experimentu jsou značně vysoké. Mohou to být například testy letadel, družic, satelitů, kosmických raket, automobilů, lodí a podobně. V takovém případě lze na provedení testu s využitím simulace ušetřit značné finanční obnosy, které mohou být využity na další vývoj. Změna rychlosti modelového času je výhodou zejména v případech, kdy by délka reálného experimentu byla příliš krátká (experimenty s LHC[1]), nebo naopak příliš dlouhá (evoluce[2]). Pokud je délka experimentu příliš krátká, nemusí být použitý měřicí přístroj schopen tak rychle reagovat na změnu měřené veličiny a může vracet zcela odlišné hodnoty. Pokud je naopak délka experimentu příliš velká, nemusíme být ochotni tak dlouho čekat na výsledek. Pomocí změny rychlosti modelového času můžeme celou simulaci podle potřeby zpomalit nebo naopak zrychlit. Nespornou výhodou simulačních metod je i jejich bezpečnost. V některých případech reálných testů podmínku bezpečnosti nelze dodržet (testování jaderných výbuchů nebo šíření virů), proto je nejen zde nutné využít počítačových simulací. Simulací je také možné využít, když nejsme za žádných okolností schopni reálný experiment provést (pohyby nebo srážky vesmírných těles).

Simulační metody mají ale také řadu nevýhod. Složitost modelu určuje časovou a paměťovou náročnost na provedení simulace, běžné počítače mají omezený výpočetní výkon. Pro extrémně složité simulace mohou být použity superpočítače[3], ale i ty jsou výpočetním výkonem limitovány. Výkon počítačů stále roste, ale i přesto pro některé extrémně složité simulace nemusí být dostatečný.

Při každé simulaci získáme informace o chování simulovaného systému, v reakci na jedny konkrétní vstupní parametry, které simulační model dostal. V případě, že nás zajímá závislost chování na některé vstupní veličině, musíme simulaci opakovat při změnách této vlastnosti. Pokud bude rozsah vstupní veličiny velký, celá simulace může trvat velmi dlouho. Další nevýhodou je jistě nepřesnost numerických metod při špatném použití. Je proto nutné vybrat vhodné výpočetní metody takové, které pro náš případ použití mají dostatečnou přesnost.

---

[1] **Large Hadron Collider** – největší urychlovač částic na světě je umístěn v kruhovém tunelu o obvodu 27 km. Přístroj urychluje dva paprsky částic proti sobě rychlostí větší než 99,99% rychlosti světla. Protonu trvá méně než 90 μs oběhnutí obvodu přístroje [3].

[2] **Evoluce** – v biologii je označována jako proces změn dědičných vlastností mezi organismy. Evoluční změny se měří v řádech milionů let [4].

[3] **Superpočítač** – je označení pro velmi výkonný počítač nebo počítačový systém, který by měl dosahovat alespoň desetinásobného výkonu běžně dostupných počítačů [5].

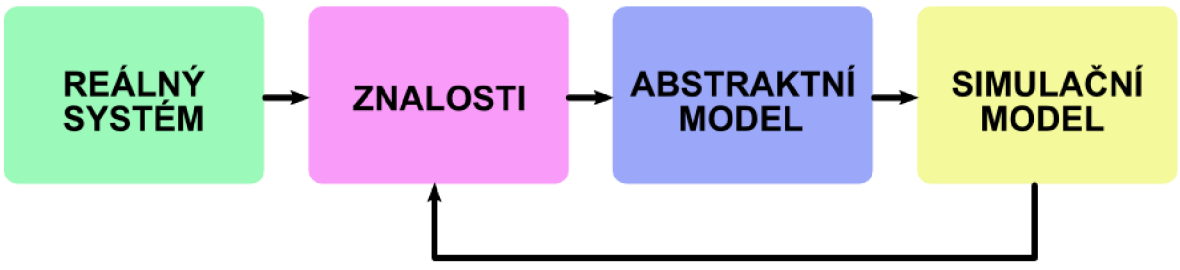
2.3 Modelování systému

Princip modelování systému může být [podle 1, 2] rozdělen do několika kroků. Nejdříve se z reálného systému na základě našich znalostí a informací o systému vytvoří abstraktní model. Protože nedokážeme sledovat celý reálný systém (celý svět do nejmenšího detailu), vybereme si pouze jeho část, kterou budeme simulovat. Dochází tedy k abstrakci, zjednodušení. Tuto část vybíráme podle toho, co je pro nás z hlediska simulace zajímavé a důležité.

Následně můžeme podle abstraktního modelu vytvořit model simulační. Simulační model vůči abstraktnímu už není nijak zjednodušen, musí obsahovat všechny jeho vlastnosti. Simulační model umožňuje, na rozdíl od abstraktního, provádění experimentů, je to například spustitelný počítačový program, který na základě vstupů a nastavených parametrů počítá výstupy simulace.

Nad již vytvořeným simulačním modelem můžeme provést simulaci. Jako simulaci označujeme fázi experimentování se simulačním modelem na základě vstupních parametrů a proměnných. Vlastní fázi simulace by mělo předcházet ověření, zda simulační model koresponduje s modelem abstraktním. Samotná simulace spočívá v opakovaném řešení modelu při změnách vstupních parametrů. Po každém kole řešení je nutné vyhodnotit výsledky, které představují chování systému na stejné vstupní parametry, jako měl simulační model při simulaci. Simulace opakujeme s různými parametry tak dlouho, dokud nemáme postačující informace o chování celého systému. Důležitou složkou simulace je kontrola platnosti modelu, tj. jestli model s aktuálními vstupními parametry odpovídá modelovanému systému. Pokud mu model neodpovídá, musíme jej na základě nově nabytých informací upravit.

Obrázek 2.1 znázorňuje kroky při modelování systému. Z reálného systému získáme pomocí experimentů a pozorování určité znalosti. Na základě těchto znalostí, vytvoříme abstraktní model. Podle abstraktního modelu vytvoříme simulační model, na kterém provedeme jedno kolo simulace. Podle výsledku se rozhodneme, zda jsme s výstupem simulace spokojeni nebo jestli podle nově získaných zkušeností upravíme abstraktní model a simulaci budeme opakovat. Jednotlivá kola simulací opakujeme tak dlouho, dokud výsledek nesplňuje naše očekávání.



Obrázek 2.1 Znázornění celého procesu získávání znalostí s využitím simulace

2.4 Rozdělení simulačních modelů

„Přesné rozdělení modelů do kategorií je poměrně náročné, protože často neexistují jednoznačná klasifikační kritéria. Modely lze dělit například podle jejich matematického popisu, podle úrovně abstrakce, podle metody implementace (paralelní, distribuované) a podle celé řady dalších kritérií.

Tradičně dělíme modely na:

- **Modely spojité** – proměnné modelu mění svůj stav spojitě. Tyto modely jsou popsatelné například diferenciálními rovnicemi.
- **Modely diskrétní** – stav modelu se mění skokově v diskrétních časových okamžicích. Příkladem takového modelu může být konečný automat.
- **Modely kombinované** – obsahují spojité i diskrétní prvky současně v jednom modelu. “[1].

Tato práce se zabývá simulací silničního provozu. Pozice a pohyb automobilu po vozovce jsou definovány spojitě. Světelné signalizační zařízení (semafory) přechází mezi stavy diskrétně. Proto celou simulaci silničního provozu můžeme zařadit do kategorie kombinovaných modelů.

3 Model města

Tato kapitola se zabývá reprezentací modelu města. Jsou zde zmíněna všechna důležitá kritéria pro vytvoření města a také důvody pro výběr již existujícího editoru města a silnic.

Tato práce je zaměřena na simulaci silničního provozu. Proto je nutné nějakým způsobem určit, jak budou silnice a celé město vypadat. Statická reprezentace modelu města je zcela nevhodná, protože by město nebylo možné měnit. Velmi důležitou součástí této práce je tedy i reprezentace celého města, silnic, křižovatek, dopravních značek a semaforů, které dokáže uživatel vytvořit a aplikace následně zobrazit. Je tedy zřejmé, že model města bude umístěn v externím souboru. Musí proto být přesně definováno, jak budou reprezentovány jednotlivé silnice, budovy a dopravní značky.

3.1 Geometrická primitiva

Geometrická primitiva (bod, úsečka) jsou základní stavební prvky pro vektorovou reprezentaci objektů. Pomocí nich lze reprezentovat složitější typy vektorových objektů. Vektorové objekty modelují objekty v mapě znázorněné formou čar (v našem případě zejména silnice) nebo čar ohraničujících nějakou plochu (budovy, parkoviště), případně formou bodů (dopravní značky, semafory) [7].

- **Bod** – je základním geometrickým prvkem vektorových objektů. Je definován souřadnicemi v prostoru, dále může obsahovat informaci o jeho napojení na linii. Z geometrického pohledu nemá žádné rozměry, jeho dimenze je 0. Jeho topologickou reprezentací je **uzel**.
- **Úsečka** – je zpravidla spojnice dvou koncových bodů. Matematicky ji lze popsat pomocí různých rovnic, ale takovéto zpracování všech úseček na mapě by bylo pro získání geometrických informací příliš náročné. Proto se běžně reálná úsečka zobrazuje jako přímá spojnice počátečního a koncového bodu. Ve své podstatě se jedná o diskretizaci reálného liniového objektu. Má konečnou délku a nulovou plochu, její dimenze je 1. Z topologického hlediska se jedná o propojení dvou uzlů, označujeme ji jako **hranu**.
- **Linie** – je složena z více úseček, přičemž první a poslední uzel se neshodují. Každá úsečka je v linii jen jednou. Kromě prvního a posledního uzlu se ostatní uzly v linii vyskytují ve dvou úsečkách zároveň (jako koncový uzel jedné a počáteční uzel druhé úsečky). Znovu je její délka konečná, plocha nulová a dimenze je 1. Linie tedy nejsou uzavřeny do kruhu a netvoří uzavřenou plochu.
- **Polygon** – je tvořen liniemi, které jsou uzavřeny a jednotlivé linie tvoří obvod tohoto polygonu. To znamená, že každý uzel je počátečním nebo koncovým uzlem dvou hran. Plocha polygonu je konečná a jeho dimenze je 2. Z topologického hlediska se jedná o **plochu**.

„Uvedené elementy **uzel** a **hrana** jsou základem pro tvorbu grafů (teorie grafů).
Graf G je dvojice $G = [N, E]$, kde:

- N je množina uzlů (nodes),
- E je množina hran (edges).

V grafu $G = [N, E]$ je každá hrana $e \in E$ spojením dvou uzlů z N. Pokud má smysl mluvit o orientovanosti hran, pak je hrana dána jako uspořádaná dvojice. (Typickým grafem je například systém řek v nějaké oblasti. Pokud nás zajímá i směr toku řek, musíme hrany definovat jako orientované. Orientované hrany mají smysl i v popisu hraničních linií polygonů.)“ [7].

3.2 Topologické vlastnosti

Topologie pomocí matematických pravidel explicitně vyjadřuje prostorové vztahy mezi jednotlivými geoobjekty. „Uložení topologické informace významným způsobem urychluje analytické operace. Mějme například mapu parcel, kde potřebujeme zjistit sousedící parcely daného pozemku. Bez uložení topologie *sousednost* bychom museli projít kompletní seznam parcel a pro každou se ptát: *je sousedící s danou parcelou?*“ [7].
Můžeme definovat několik základních topologických vlastností vektorových objektů.

- **Konektivita (spojitost/connectivity)** – znamená propojení dvou linií v uzlu. Například spojení silnic v křižovatce.
- **Sousednost (přilehlost/adjacency)** – je vlastnost, kdy dvě linie mají určitý směr a udržují informaci o objektech, které jsou napravo a nalevo od nich. Principielně se jedná o podobný princip, jako se používá při využití okřídlené hrany pro popis modelů.
- **Orientace (orientation)** – určuje směr z – do. Definuje počáteční a koncový bod a tím i orientaci celého objektu.
- **Obsahování (containment)** – určuje, že nějaký objekt obsahuje uvnitř sebe jiný objekt. Například zahrada (vymezená polygonem) obsahující uvnitř sebe parcelu domu.

3.3 Vektorové modely

Vektorové modely slouží pro reprezentaci geoobjektů, zejména jejich polohu a topologii. Existuje mnoho modelů, které se liší složitostí struktury, redundancí souřadnic nebo možností zaznamenávání topologických vztahů mezi sousedními objekty.

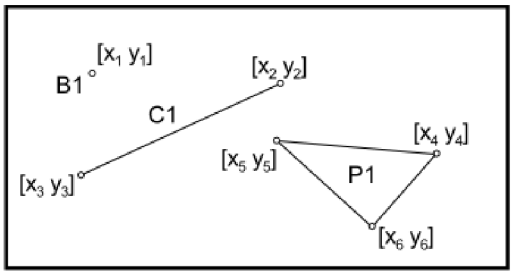
3.3.1 „Špagetový“ model

„Špagetový“ model patří k nejjednodušším modelům, ale neobsahuje žádnou informaci o topologii objektů. V tomto modelu jsou všechny typy vektorových objektů uloženy v jednom heterogenním seznamu, který má vždy dvě položky.

- První položkou je určení **typu** objektu. Jedná se zpravidla o **bod**, **linii**, nebo **polygon**.

- Druhou položkou jsou **souřadnice**. U **bodu** je uložena souřadnice tohoto bodu. U **linie** se jedná o souřadnici počátečního a koncového bodu. U **polygonu** jsou zde uloženy souřadnice všech uzlů, kterými linie polygonu prochází (souřadnice koncových bodů obvodových hran).

Tento model je sice nejjednodušší, ale zjištění nějaké topologické informace (například sousednosti) je velmi obtížné a musí se projít celý seznam. Navíc zde dochází k redundanci dat. Situaci lze připodobnit ke dvěma sousedním pozemkům, kde každý pozemek má svůj vlastní plot (v místě sousedství, jsou pozemky odděleny dvěma ploty).



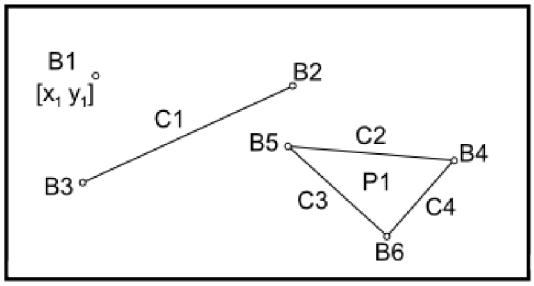
TYP OBJEKTU	ID	POLOHA
Bod	B1	x_1 y_1
Čára	C1	x_2 y_2, x_3 y_3
Polygon	P1	x_4 y_4, x_5 y_5, x_6 y_6

Obrázek 3.1 Ukázka uložení dat ve "špagetovém" modelu

3.3.2 Hierarchický model

Hierarchický model ukládá veškeré informace o objektech na úrovni jejich prostorové dimenze. Vytváří hierarchickou strukturu objektů. Všechny objekty rozděluje do čtyř základních seznamů:

- **Bod** – je základní prvek tohoto modelu. Každý bod v tomto seznamu obsahuje informaci o své poloze.
- **Úsečka** – je seznam pro spojnice dvou bodů. Případně může obsahovat další informace (svoji délku, …).
- **Linie** – je neuzavřená sekvence úseček. Seznam tedy obsahuje výpis úseček patřící do linie (případně opět s dalšími informacemi).
- **Polygon** – se skládá z několika linií, seznam obsahuje jejich výčet. Seznam může navíc obsahovat další informace o obvodu, obsahu, atd.



TABULKA BODŮ	
B1	x_1 y_1
B2	x_2 y_2
B3	x_3 y_3
B4	x_4 y_4
B5	x_5 y_5
B6	x_6 y_6

TABULKA ČAR	
C1	B2 B3
C2	B4 B5
C3	B5 B6
C4	B6 B4

TABULKA POLYGONŮ	
P1	C2 C3 C4

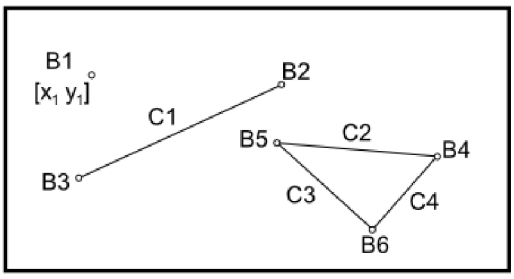
Obrázek 3.2 Ukázka uložení dat v hierarchickém modelu

3.3.3 Topologický model

Topologický model je kompromisem mezi „špagetovým“ a hierarchickým modelem. Je to jeden z nejoblíbenějších modelů pro uchovávání vektorových dat. Tento model udržuje seznam bodů a seznam hran. Každá položka ze seznamu bodů má souřadnice, kde je daný bod umístěn. Každá linie může začínat a končit pouze v uzlu, linie mají uloženy svůj počáteční a svůj koncový uzel (z čehož můžeme zjistit orientaci linie). Každé dvě linie se mohou protínat opět jen v uzlu. Linie má navíc ukazatel na pravý a levý polygon, čímž zachovává základní vztahy pro potřeby topologické analýzy. Výhodou tohoto řešení je to, že souřadnice je uložena pouze v bodech a nedochází tedy k redundanci dat. I přesto má topologický model, stejně jako „špagetový“, velkou nevýhodu v naprosté neuspořádanosti záznamů. K vyhledání všech linií ohraničujících určitý polygon je potřeba projít třeba i všechny záznamy.

TABULKA BODŮ	
B1	x_1 y_1
B2	x_2 y_2
B3	x_3 y_3
B4	x_4 y_4
B5	x_5 y_5
B6	x_6 y_6

TABULKA ČAR	
C1	B2 B3
C2	B4 B5
C3	B5 B6
C4	B6 B4



Obrázek 3.3 Ukázka uložení dat v topologickém modelu

3.4 Java OpenStreetMap Editor

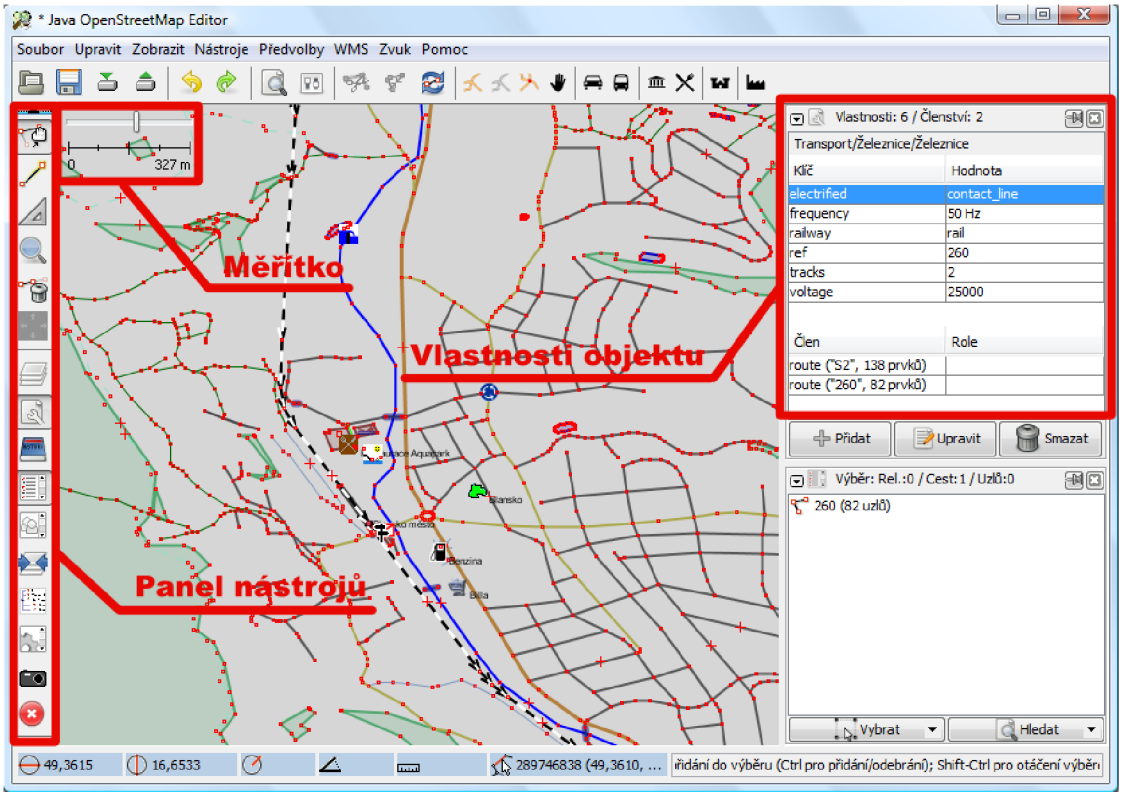
Už při popisu reprezentace vektorových objektů (jako je v našem případě mapa města) je poznat, že se nejedná o zcela jednoduchou záležitost. Vytvořit ručně tímto způsobem mapu celého města, aby ji následně mohla aplikace načíst a zobrazit, by bylo velmi pracné (nehledě na chyby, které by při konstrukci města vznikly). Bylo proto zcela nezbytné, aby existovala nějaká aplikace, která by uměla vytvořit město jednodušším způsobem, nejlépe s nějakým vizuálním prostředím. Vytvoření takového editoru města, by svojí složitostí vydalo na celou diplomovou práci [11]. I z tohoto důvodu jsem se rozhodl využít již existující aplikace.

JOSM – Java OpenStreetMap Editor [8] je editor map vytvořený hlavně pro projekt OSM (OpenStreetMap[4]). Je vytvořen v programovacím jazyce Java, jeho vývoj stále pokračuje a podporuje velké množství zásuvných modulů. **JOSM** je uvolněn pod licencí GNU GPL v2. Mezi jeho základní funkce patří vkládání, editace a značkování geodat. Uživatel má možnost vizuálně interpretovat GPS záznamy nebo družicové snímky a rozšiřovat tak OSM projekt.

Při zohlednění těchto vlastností jsem se rozhodl využít **JOSM** [8] jako externí editor pro generování celého města (silnic, budov, dopravních značek, semaforů). Uživatel má tedy možnost podle určitých pravidel v tomto editoru vytvořit město a vyexportovat do souboru *osm*. Následně může vytvořené město zobrazit v aplikaci a testovat v něm své jízdní schopnosti.

Obrázek 3.4 zobrazuje okno programu **Java OpenStreetMap Editor**, které je rozděleno na několik částí. V horní části je standardní menu aplikace a ikonky často používaných akcí. V levé části může uživatel zvolit některý nástroj z panelu. Jsou to nástroje jak pro kreslení čar nebo bodu, tak pro označování, mazání a otevírání dalších podoken. Ve spodní části se nachází souřadnice aktuální polohy kurzoru a nápověda k právě prováděné akci. Pravá část je věnována vlastnostem právě označeného objektu mapy, v této části může uživatel každému objektu přiřadit nějakou vlastnost a nastavit její hodnotu. Pomocí těchto vlastností lze odlišit silnice, budovy, značky a semafory. V prostřední části je aktuálně vytvořená mapa s měřítkem.

[4] **OSM (OpenStreetMap)** – je projekt, jehož hlavním cílem je tvorba volně dostupných geografických dat a následně jejich vizualizace do podoby topografických dat. Projekt je založen na kolektivní spolupráci a na koncepci Open Source [9, 10].



Obrázek 3.4 Okno aplikace Java OpenStreetMap Editor [8]

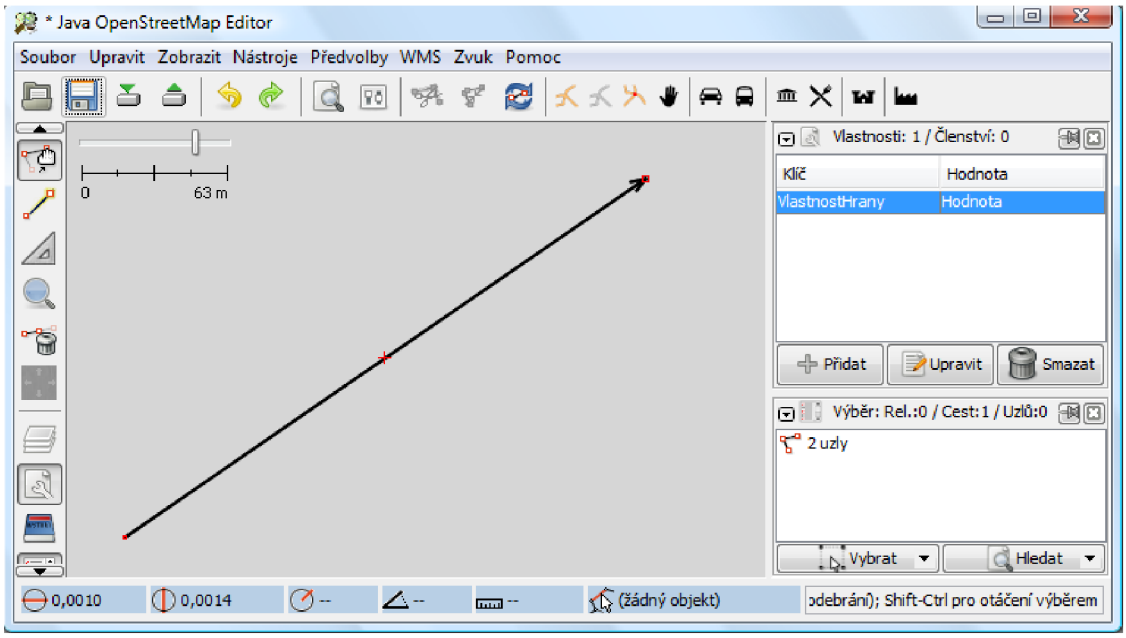
3.4.1 Pravidla pro vytvoření města

Soubor osm je ukládán v textové formě ve formátu XML[5] a lze jej proto snadno zpracovat pomocí nějakého XML parseru. Pro tento účel jsem použil parser TinyXML [20], který je pro zpracování jednoduché struktury souboru naprosto dostačující. JOSM používá topologický model ukládání dat. To znamená, že ve výsledném souboru je nejprve seznam všech uzlů z mapy (*node*). Každý z těchto uzlů má zaznamenanou souřadnici, na které se vyskytuje. Za seznamem uzlů se nachází seznam orientovaných hran (*way*). Každý uzel a hrana mají svůj vlastní identifikátor. Každá hrana navíc obsahuje informaci o tom, kterými uzly prochází a tím je přesně vymezena její poloha. Hrany i uzly mohou navíc obsahovat nějakou proměnnou s určitou hodnotou, čehož je využito při rozlišování silnic, budov, značek, semaforů a dalších objektů.

[5] **XML (Extensible Markup Language)** – je značkovací jazyk. Umožňuje snadné vytváření značkovacích jazyků (tzv. aplikací) pro různé účely a různý typy dat [12].

```
<?xml version='1.0' encoding='UTF-8'?>
<osm version='0.6' generator='JOSM'>
  <node id='-1' visible='true' lat='-6.680845360002487E-4' lon='-7.592513333354021E-4' />
  <node id='-2' action='modify' visible='true' lat='7.57095419755095E-4' lon='0.001265321363105358'>
    <tag k='VlastnostBodu' v='Hodnota' />
  </node>
  <way id='-3' action='modify' visible='true'>
    <nd ref='-1' />
    <nd ref='-2' />
    <tag k='VlastnostHrany' v='Hodnota' />
  </way>
</osm>
```

Zdrojový kód 3.1 Ukázka jednoduchého výstupu z JOSM editoru



Obrázek 3.5 Vizuální zobrazení podle zdrojového kódu 3.1

Celé město, které aplikace dokáže zobrazit, může být vytvořeno podle několika následujících pravidel:

- **Startovní bod** – Bod na mapě označený jako startovní bod je místo, kde se při spuštění aplikace zobrazí automobil uživatele. Je tedy zřejmé, že by neměl být umístěn uvnitř budovy a měl by mít volný přístup k okolním cestám. Pokud tento bod nebude do vytvořené mapy přidán, vytvoří se automobil na souřadnicích [0, 0]. Aby byl vytvořený bod považován za startovní, musí mít nastavenou vlastnost *StartPoint* na hodnotu *yes* (*StartPoint* = *yes*). Startovní bod navíc obsahuje různá nastavení pro celou aplikaci. Celočíselná hodnota vlastnosti *CountOfCars* určuje počet aut zobrazených na mapě (např. *CountOfCars* = *5*). Vzhledem k tomu, že **JOSM** editor ukládá polohu bodů jako jejich zeměpisnou šířku a délku, souřadnice dvou bodů ležících nedaleko sebe se hodnotou moc neliší. Při zobrazování pomocí grafického enginu Ogre [16], je vhodnější

použití souřadnic, které se budou hodnotou více lišit. Proto jsou všechny souřadnice vynásobeny určitým koeficientem, který může být nastaven u startovacího bodu. Celočíselná hodnota vlastnosti *KSize* určuje, kolikanásobně se budou všechny souřadnice zvětšovat (*KSize = 35000*). Pokud startovní bod nemá tuto položku nastavenu, je jako výchozí zvětšení použito 35000x.

- **Budova** – Za budovu je označen každý polygon (nejlépe uzavřený), kterému je nastavena vlastnost *Building* na *yes* (*Building = yes*). Tuto vlastnost není nutné psát ručně, jako budovu lze každý uzavřený polygon označit i pomocí menu editoru **JOSM** (kliknutím na položku menu: *Předvolby->Objekty a další POI->Budovy->Budova)*. Orientace u budov na rozdíl od cest nemá žádný význam.

- **Silnice, cesta** – Silnicí je každý polygon, který není označen jako budova. Silnice má orientaci, která je znázorněna v editoru pomocí šipky. Orientace silnice je rozhodující pro zobrazení značky, určuje, na které straně silnice se značka zobrazí. Rozhodnutí, na které straně silnice se značka zobrazí, je nastaveno vlastností *direction* ve vlastnostech značky. Silnice může mít vlastnost *SignsRef*, která určuje, jaké značky se budou uplatňovat a zobrazí se vedle této cesty. Jedná se o seznam jedinečných identifikátorů značek (*Ref* ve vlastnostech značky) oddělených čárkou. Například pro zobrazení a uplatnění značky s identifikátorem 2 a 3 musí být nastaveno: *SignsRef = 2,3*. Další vlastností, kterou mohou silnice mít je *SignsNonRef*, což je seznam značek, které se na silnici budou uplatňovat, ale nebudou se zobrazovat. Silnicím lze také nastavit vlastnost *TrafficLightChannel*, která je popsána u vytváření semaforů.

- **Značka** – Značku představuje bod v mapě, který má nastavenou vlastnost *Sign* na hodnotu *yes*. Aby bylo možné značku použít, musí také mít nastaveny vlastnosti *Ref* a *Meaning*. Celočíselná vlastnost *Ref* představuje jedinečný identifikátor značky, bez něj by nebylo možné rozpoznat značky od sebe a přiřadit je k silnici. Dále je nutné správně nastavit vlastnost *Meaning*, která určuje význam značky. *Meaning* může nabývat následujících hodnot: *nejvyssi_povolena_rychlost_10, nejvyssi_povolena_rychlost_20, nejvyssi_povolena_rychlost_30, nejvyssi_povolena_rychlost_40, nejvyssi_povolena_rychlost_50, nejvyssi_povolena_rychlost_60, nejvyssi_povolena_rychlost_70, nejvyssi_povolena_rychlost_80, nejvyssi_povolena_rychlost_90, nejvyssi_povolena_rychlost_100, nejvyssi_povolena_rychlost_110 nejvyssi_povolena_rychlost_120, nejvyssi_povolena_rychlost_130, zakaz_zastaveni, zakaz_vezdu_vsech_vozidel_jednosmerny, zakaz_vezdu_vsech_vozidel_v_obou_smerech, hlavni_pozemni_komunikace, konec_hlavni_pozemni_komunikace, dej_prednost_v_jizde, stop_dej_prednost_v_jizde, jine_nebezpeci, krizovatka, prikazany_smer_jizdy_primo, prikazany_smer_jizdy_vpravo,*

prikazany_smer_jizdy_vlevo, jednosmerny_provoz, parkoviste, slepa_pozemni_komunikace a *obytna_zona*.
U každé hodnoty je patrné, jaký význam značka bude mít. Další možná vlastnost značky je *direction*, která může nabývat hodnot *1* nebo *-1* (např. *direction* = *-1*). Touto vlastností značka určuje, ke kterému směru hrany se přidá. Pokud je hodnota nastavena na *1*, je značka umístěna ve směru šipky, pokud na *-1*, je umístěna ve směru proti šipce hrany. Vlastnost *direction* není povinná, pokud není nastavena, uvažuje se umístění ve směru hrany.

- **Semafor** – Pro vytvoření semaforu se bodu představujícímu křižovatku (bod uprostřed křižovatky) nastaví vlastnost *TrafficLight* na hodnotu *yes*. Hrany každého semaforu, na kterých se mají semafory zobrazit, mohou být rozděleny do dvou vln, kanálů. V rámci jednoho kanálu jsou barvy na semaforu stejné a přepínají se zároveň. Pro přepínání barev hrany je nutné hraně přidat vlastnost *TrafficLightChannel* a jako hodnotu vložit číslo požadovaného kanálu. *TrafficLightChannel* může nabývat hodnot *1* a *2*.

- **Tráva** – Ve městě je také možné zobrazit různé traviny. V místě každého bodu v mapě, který bude mít nastavenou položku *Grass* na hodnotu *yes*, bude zobrazen jako některá z dostupných typů trávy.

4 Dopravní pravidla

Podle Českého statistického úřadu [21] a [22] bylo už v roce 2008 v České republice registrováno 4423370 osobních automobilů, které se mohly pohybovat po 54963 km silnic I., II. a III. třídy. Je zřejmé, že pokud se řidiči pomocí automobilů chtějí někam dopravit, jen výjimečně se na vozovkách nepotkají s dalšími automobily. V případě, že by neplatil žádný řád, který by určoval kdo, kdy a kam může jet, jen velmi těžko by se řidiči při jízdě domluvili na „hladkém“ průjezdu. Je tedy jasné, že musí existovat nějaký všeobecně platný a všem známý řád, určující veškerá pravidla pohybu na komunikacích a všichni účastníci provozu na komunikacích tento řád musí respektovat a řídit se podle něj. V České republice se tento řád jmenuje **Zákon o silničním provozu** [23]. Tento zákon upravuje práva a povinnosti účastníků provozu na pozemních komunikacích, pravidla provozu, úpravu a řízení provozu, řidičská oprávnění a řidičské průkazy, vymezuje působnost a pravomoc orgánů státní správy a Policie České republiky. Zákon o silničním provozu je značně rozsáhlý (v aktuálním znění účinném od 1. července 2010, je rozdělen do sedmi částí, má 143 paragrafů a je rozepsán na více než 100 stran).

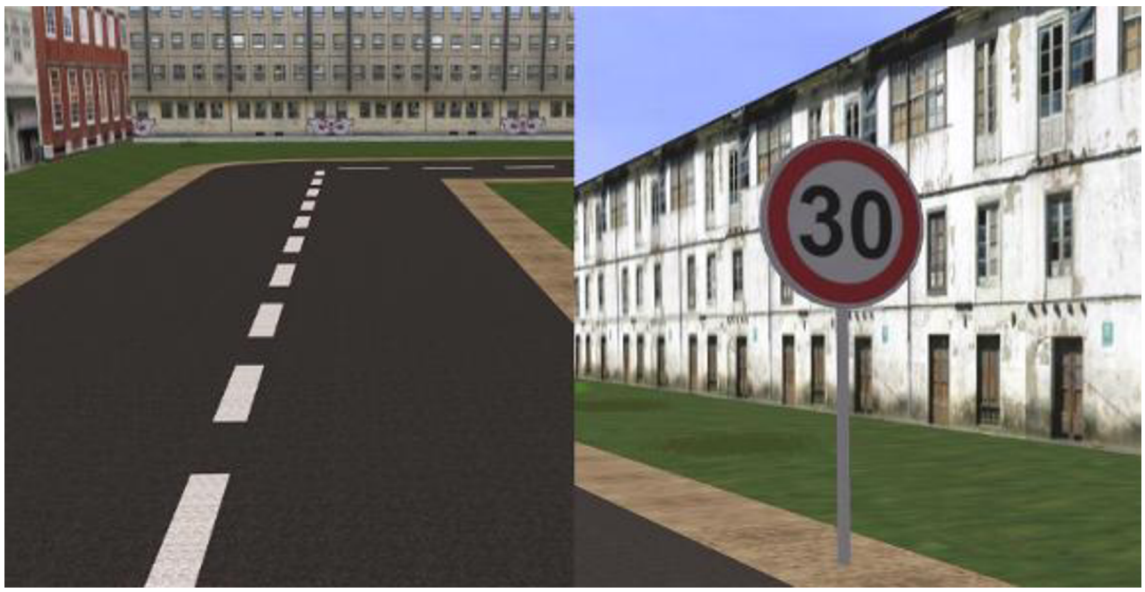
Tato kapitola se věnuje platným dopravním pravidlům. Jsou zde analyzována a popsána základní použitá dopravní pravidla, bez kterých by nebylo možné vytvořit simulátor provozu splňující podmínky platného zákona. Analýza těchto pravidel, je tedy nedílnou součástí této práce.

4.1 Dopravní značky

Dopravní značky nás na pozemních komunikacích provázejí každý den. Mohou mít různé tvary, barvy, formu a každá značka má svůj vlastní význam. Historie dopravních značek sahá až do období antiky, kdy jejich funkci plnily patníky a milníky. Dopravní značky obsahují různé obrázky a piktogramy, které mají za úkol usnadňovat pochopení a zapamatování jejich významu. Značky řídí a regulují silniční provoz, snaží se upozorňovat na nebezpečná místa, ukládají všem účastníkům provozu různé příkazy, zákazy nebo omezení, mohou zpřesňovat nebo upravovat význam jiných dopravních značek. Dopravní značky se dělí na dva základní typy: na značky svislé a značky vodorovné.

- **Svislé dopravní značky** – Tento typ značek je umisťován vedle vozovky nebo nad vozovkou. Značky tohoto typu mohou mít stálou, proměnnou a přenosnou variantu. Stálé značky jsou umisťovány na sloupcích nebo konstrukcích pevně zabudovaných do terénu. Svislé značky se dělí na značky upravující přednost, výstražné, zákazové, příkazové, informativní a dodatkové tabulky. Každý z těchto typů má svá specifika, podle kterých je možné značky rozeznat.

- **Vodorovné dopravní značky** – Značky tohoto typu jsou vyznačeny zpravidla na pozemní komunikaci, nejčastěji barvou, případně jiným vhodným způsobem. Využívají se buď samostatně nebo ve spojení se svislými dopravními značkami. Mezi vodorovné dopravní značení paří podélné a příčné čáry, šipky, označení stání a parkovišť, zastávek, zákazů stání a zastavení a další. Tento typ značek může mít stálou nebo přechodnou variantu, které se odlišují barvami.



Obrázek 4.1 Ukázka zobrazení vodorovné a svislé dopravní značky v aplikaci

4.1.1 Typy značek

Svislé dopravní značky se dělí na několik typů. Každý z těchto typů má jiné vzhledové rysy a slouží k jinému účelu.

- **Značky upravující přednost** – Stanoví přednosti v jízdě v provozu na pozemních komunikacích. Tvar značek tohoto typu je záměrně odlišný od ostatních typů, aby jejich vzhled bylo možné rozeznat i za nepříznivých podmínek.

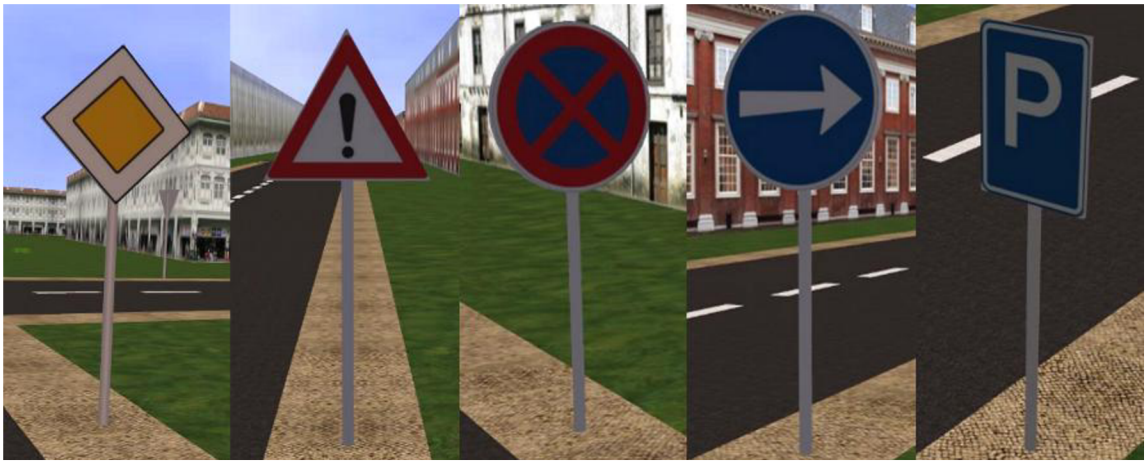
- **Výstražné značky** – Upozorňují na místa, kde účastníkům provozu hrozí případné nebezpečí a kde musí dbát zvýšené opatrnosti. Jejich charakteristickým tvarem je trojúhelník se špičkou směřující vzhůru. Obvod trojúhelníku má červený okraj a symbol určující význam je zobrazen uvnitř.

- **Zákazové značky** – Značky patřící do této skupiny ukládají účastníkům provozu zákazy nebo omezení. Charakteristickým tvarem značek je kruh s červeným okrajem, symbol určující význam je opět uvnitř kruhu.

- **Příkazové značky** – Tyto značky ukládají účastníkům provozu příkazy. Typický tvar je modrý plný kruh, uvnitř kterého je bílou barvou zobrazen symbol určující význam.

- **Informativní značky** – Poskytují účastníkům provozu nutné informace, slouží k orientaci nebo ukládají povinnosti stanovené zákonem. Typickým tvarem je zpravidla čtyřúhelník, mají široké využití, a proto mohou být zobrazeny pomocí mnoha barev.

- **Dodatkové tabulky** – Slouží ke zpřesnění, doplnění nebo omezení významu ostatních dopravních značek, pod kterými jsou umístěny. Bez značky, kterou mají doplňovat, nemají význam. Tvarem je také čtyřúhelník, který má bílou barvu a obsahuje symbol nebo nápis v černé barvě.



Obrázek 4.2 Ukázka typů značek upravujících přednost, výstražných, zákazových, příkazových a informativních zobrazovaných v aplikaci

4.2 Světelné signály

Světelnými signály (běžně nazývané jako semafory) se řídí provoz na pozemních komunikacích nebo se jimi upozorňuje na nutnost dbát zvýšené opatrnosti. Světelné signály mají vyšší prioritu než dopravní značky. To znamená, že v případě, pokud se na křižovatce vyskytnou dopravní značky společně se světelnými signály, musí se řidiči řídit podle světelných signálů (semaforů). Až v případě, že semafory jsou mimo provoz, musí se všichni účastníci provozu řídit podle dopravních značek. U světelných signálů se v podstatě jedná o rozsvěcování světla v různých barvách (v některých případech se pro zdůraznění významu využívá i tvarů světla), přičemž každá barva (barevný kód) má svůj vlastní význam. Důležité je, že tyto barevné kódy jsou neměnné a jejich význam nelze zaměňovat. Je pravidlem, že světlo červené barvy dává signál „**Stůj!**“, určený k úplnému zastavení. Žluté nebo oranžové světlo se používá pro varování, dává nám signál „**Pozor!**“ a světlo zelené barvy zpravidla naznačuje volný průjezd, jedná se o světelný signál „**Volno**“.

Na křižovatkách řízených světelnými signály se nevyužívá pouze jeden význam pro každou barvu, svůj význam mají i kombinace barev. Pokud u semaforu se třemi barvami svítí červené světlo, jedná se o signál „**Stůj!**“ (význam červeného a zeleného světla zůstává vždy stejný). Po červeném světle následuje rozsvícení červeného a žlutého současně, což je signál „**Pozor!**“, značící povinnost připravit se k jízdě. Dále následuje rozsvícení zeleného světla, značící signál „**Volno**“ s nezměněným významem (možnost pokračování v jízdě). Jako další se na semaforech vždy rozsvítí samotné nepřerušované žluté světlo se signálem „**Pozor!**“. Tento signál má (na rozdíl od současně rozsvíceného červeného a žlutého světla) obdobný význam jako signál „**Stůj!**“, ale v případě, že vozidlo je již moc blízko semaforu a řidič by nezvládl zastavit, může pokračovat v jízdě. Po tomto signálu se opět cyklicky rozsvěcuje červené světlo. Je nutné zmínit, že v případě, kdy svítí neustále přerušované žluté světlo (nebo dokonce kdy nesvítí žádné světlo na semaforu), není semafor v chodu (v danou chvíli se nejedná o křižovatku s provozem řízeným světelnými signály), musí se účastníci provozu řídit dopravními značkami.

Obrázek 4.3 Zobrazení posloupnosti přepínání světelných signálů v aplikaci

4.3 Jízda křižovatkou

Pokud se při pohybu po pozemních komunikacích budeme chtít rozhodovat, kam pojedeme, musíme nutně využívat křižovatek. Křižovatky patří k nejnebezpečnějším místům v provozu, protože se zde protínají různé pozemní komunikace a dráhy všech účastníků provozu, kteří se po těchto komunikacích pohybují. Každý den se stává velké množství dopravních nehod právě při průjezdu křižovatkami. Jedná se tedy o propojení více cest, musíme nutně rozhodnout, ze které komunikace pojedou účastníci silničního provozu nejdříve. Pokud některý z účastníků provozu špatně vyhodnotí svoji přednost a najede do křižovatky ve chvíli, kdy nemá, hrozí velké riziko dopravní nehody. Je tedy nutné, aby byly zavedeny stálé principy průjezdů a hlavně, aby všichni účastníci silničního provozu tyto principy znali a respektovali.

Běžně využíváme několik typů označování předností v křižovatkách. Křižovatky, na kterých je přednost upravována pomocí dopravních značek a křižovatky, kde přednost dopravní značky neupravují. Zde jsou popsány základní pravidla a principy, bez kterých by implementace dopravního simulátoru nebyla možná.

Obrázek 4.4 Situace s využitím pravidla o odbočování vlevo

4.3.1 Odbočování vlevo

V případě že na křižovatce řidič odbočuje vlevo, musí dát přednost v jízdě všem protijedoucím účastníkům provozu, kterým kříží dráhu jízdy. Toto pravidlo platí pro všechny typy určování přednosti na křižovatkách, tj. platí v křižovatkách bez dopravních značek, s dopravními značkami i v křižovatkách, kde jsou přednosti v jízdě určeny pomocí semaforů.

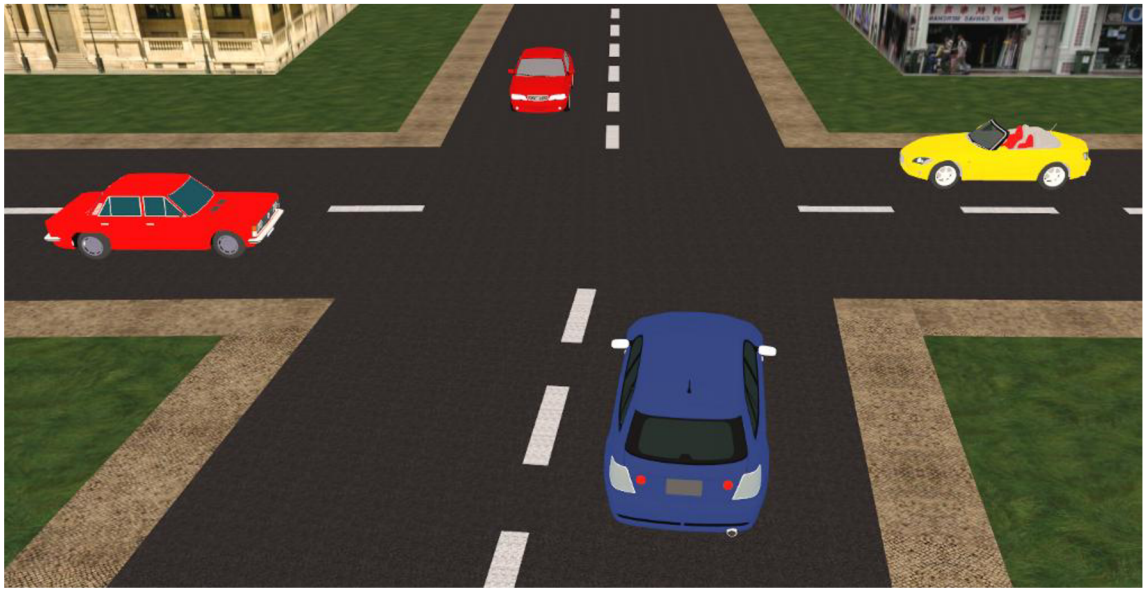
Obrázek 4.4 ukazuje situaci při využití pravidla pro odbočování vlevo. Na obrázku je vidět, že oba automobily jsou sice na hlavní silnici, ale bílý automobil odbočuje vlevo. Proto musí dát přednost protijedoucímu červenému automobilu. Červený automobil tedy projede křižovatkou jako první.

4.3.2 Křižovatky bez dopravních značek

Křižovatky, kde přednost neupravují dopravní značky, nejsou tak časté. Můžeme se s nimi setkat spíše na místech, kde není provoz příliš hustý. O křižovatkách, kde není přednost upravována značkami, hovoří **Zákon o silničním provozu** [23] takto: „Nevyplývá-li přednost v jízdě z dopravních značek, musí dát řidič přednost v jízdě vozidlům nebo jezdcům na zvířatech přijíždějícím **zprava** nebo organizované skupině chodců nebo průvodcům hnaných zvířat se

zvířaty přicházejícím **zprava**.“. Toto základní pravidlo o přednosti zprava platí **ve všech situacích**, kdy přednost není určena dopravními značkami nebo světelnými signály. Důležité je si uvědomit, že pravidlo o přednosti zprava nevylučuje pravidlo o odbočování vlevo.

Křižovatky bez dopravních značek se sice vyskytují spíše na méně frekventovaných místech, ale i přesto mohou nastat situace, kdy průjezd křižovatkou podle pravidla o přednosti zprava není možný. Jedná se zejména o případy, kdy do křižovatky přijíždí automobily ze všech směrů zároveň nebo je-li v křižovatce málo místa pro odbočování rozměrnějšího vozidla. V těchto případech je nutné se s ostatními řidiči zřetelně domluvit, nejčastěji vizuálními posunky a gesty, kdy se některý z řidičů musí vzdát své přednosti v jízdě. Ostatní řidiči mohou potom pokračovat jako by jeho směr byl volný.



Obrázek 4.5 Při použitím samotného pravidla o přednosti zprava dojde k zablokování

Na obrázku 4.5 je vidět situace, při které s využitím pravidla o přednosti zprava dojde k zablokování. Každý z automobilů má příjezdovou komunikaci po své pravé straně obsazenou jiným automobilem, kterému dává přednost.

4.3.3 Křižovatky včetně dopravních značek

Pokud přednosti na křižovatce upravují dopravní značky, je situace přehlednější. Znamená to, že je nastálo určeno, ze které vozovky mají přijíždějící vozidla přednost. Určení přednosti je dáno příslušnými dopravními značkami, vozidla přijíždějící po hlavní pozemní komunikaci mají přednost před vozidly přijíždějícími z vedlejších pozemních komunikací. **Zákon o silničním provozu** [23] tuto situaci popisuje takto: „Řidič přijíždějící na křižovatku po vedlejší pozemní komunikaci označené dopravní značkou *Dej přednost v jízdě!* nebo *Stůj, dej přednost v jízdě!* musí dát přednost v jízdě vozidlům nebo jezdcům na zvířatech přijíždějícím po hlavní pozemní

komunikaci nebo organizované skupině chodců nebo průvodcům hnaných zvířat se zvířaty přicházejícím po hlavní pozemní komunikaci.“. Pokud dvě vozidla přijíždějí po rovnocenných komunikacích (obě po hlavní nebo obě po vedlejší pozemní komunikaci) platí mezi nimi pravidlo o přednosti zprava. Navíc i zde při odbočování musí platit pravidlo o odbočování vlevo.

Obrázek 4.6 Průjezd automobilů křižovatkou s hlavní a vedlejší silnicí

Na obrázku 4.6 jsou zobrazeny dva automobily, modrému automobilu přijíždí zprava automobil bílý. Podle pravidla o přednosti zprava by musel počkat na průjezd bílého automobilu, ale protože se nachází na hlavní silnici (a bílý na vedlejší), má přednost před bílým automobilem. Je důležité si uvědomit, že v případě, kdy by hlavní silnice byla zatočena **doprava** (a tudíž obě vozidla by byla na hlavní) a obě jela rovně, musel by řidič modrého automobilu dát přednost v jízdě automobilu bílému, z důvodu platnosti pravidla o přednosti zprava.

4.3.4 Křižovatky s předností řízenou světelnými signály

Jak již bylo popsáno v kapitole 4.2, řízení přednosti pomocí světelných signálů je nadřazeno řízení přednosti dopravními značkami. Pokud jsou v křižovatce dopravní značky i světelné signály, přednost je určována podle světelných signálů (za předpokladu, že jsou zapnuty), na dopravní značky není brána zřetel. V případě, že semafory nejsou v provozu (nebo přerušovaně svítí žluté světlo), nejedná se o křižovatku s provozem řízeným světelnými signály a v tomto případě je přednost určována podle dopravních značek.

Obrázek 4.7 Křižovatka s provozem řízeným světelnými signály

Obrázek 4.7 zobrazuje situaci na křižovatce s provozem řízeným světelnými signály, kam právě přijely tři automobily. Dopravní značky *„Dej přednost v jízdě“* napovídají, že bílé a červené vozidlo přijíždí po vedlejší komunikaci. Jenže křižovatka je řízena světelnými signály, proto v této křižovatce dopravní značky přednosti neurčují, přednost mají vozidla, jež mají na semaforu zelenou (což jsou právě bílé a červené vozidlo). Protože bílé vozidlo odbočuje vlevo a kříží cestu červenému, musí mu před svým průjezdem dát přednost. Jako první tedy křižovatkou projíždí červené vozidlo, hned za ním bílé. Žluté vozidlo musí počkat na zelenou na svém semaforu a až poté může vjet do křižovatky.

4.4 Předjíždění

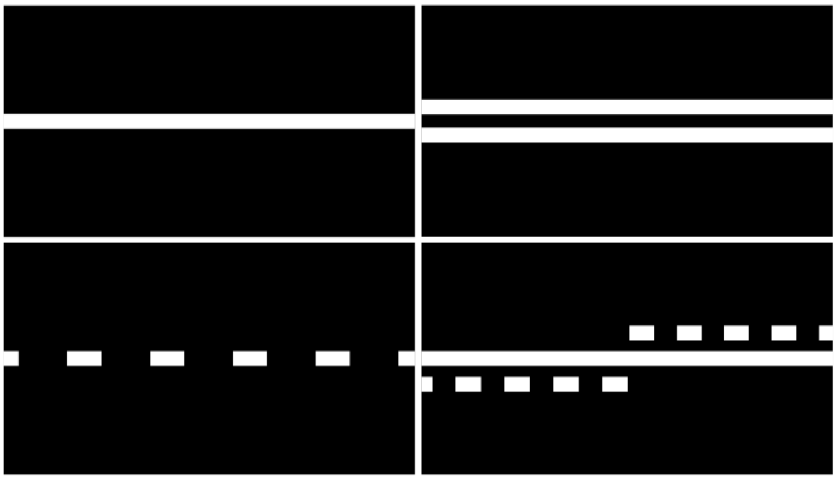
Předjížděním je nazýván manévr, při kterém se rychlejší vozidlo snaží dostat před pomalejší vozidlo jedoucí stejným směrem. Při této akci předjíždějící vozidlo opouští svůj jízdní pruh a dočasně najíždí do pruhu, který vede opačným směrem. Vzniká tedy velké riziko srážky s protijedoucím vozidlem. Takovéto čelní střety mívají velmi vážné následky, rychlosti a síly obou vozidel v momentu nárazu se sečtou, a proto je jejich destruktivní síla daleko větší. Čelní náraz dvou vozidel, kdy obě jedou rychlostí 90 km/h, se přibližně rovná přímému nárazu do pevné překážky (například do zdi) v rychlosti 180 km/h. Při čelních střetech vzniká nejvíce smrtelných úrazů.

Pouze v případě, že se pomalejší vozidlo (to, které chceme předjet) chystá odbočovat vlevo, smí ho rychlejší vozidlo předjíždět zprava. Aby bylo možné vozidlo zprava předjet, nesmí být pochyb o jeho úmyslu odbočovat vlevo. Pokud se ale pomalejší vozidlo odbočovat vlevo nechystá, rychlejší vozidlo může pomalejší předjet pouze zleva. Předjíždění z jiné strany než je

dovoleno, je hrubým porušením dopravních předpisů. Řidič nesmí předjíždět, pokud nemá před sebe rozhled na takovou vzdálenost, která je nutná k bezproblémovému předjetí. Dále pak pokud by se nemohl bezpečně zařadit před předjížděné vozidlo nebo jestliže by ohrozil nebo omezil protijedoucí řidiče nebo jiné účastníky provozu na pozemních komunikacích. Předjíždění dále není dovoleno na přechodech pro chodce, železničních přejezdech a bezprostředně před nimi nebo v případě, že vozidlo vepředu dává znamení o změně směru jízdy vlevo a není možné ho předjet zprava.

Obrázek 4.8 Dopravní značky upravující zákaz předjíždění

Možnost předjíždění je také upravována dopravními značkami a vodorovným dopravním značením. Na obrázku 4.8 jsou vyobrazeny dopravní značky „Zákaz předjíždění“, „Zákaz předjíždění pro nákladní automobily“, „Konec zákazu předjíždění“ a „Konec zákazu předjíždění pro nákladní automobily“. Už z názvu značek je patrné jaký mají význam, v místě platnosti těchto zákazových dopravních značek („Zákaz předjíždění“ a „Zákaz předjíždění pro nákladní automobily“) je zakázáno předjíždění ostatních motorových automobilů vlevo. Výjimkou je motocykl bez postranního vozíku, který předjet lze. Zákazy předjíždění jsou ukončeny značkami „Konec zákazu předjíždění“ a „Konec zákazu předjíždění pro nákladní automobily“. Také vodorovné dopravní značení upravuje předjíždění. Obrázek 4.9 zobrazuje podélné čáry souvislou, dvojitou souvislou, přerušovanou a souvislou doplněnou čárou přerušovanou. Pokud je jízdní pruh oddělen podélnou čárou souvislou nebo dvojitou souvislou čárou, je předjíždění nebo přesahování nákladem přes tuto čáru zakázáno (pokud to není nutné k objetí překážky na pozemní komunikaci). Když je jízdní pruh oddělen čárou přerušovanou, je předjíždění (a tím i najetí do protisměrného pruhu pouze na nezbytně dlouhou dobu) dovoleno. V případě, že je jízdní pruh oddělen čárou souvislou doplněnou čárou přerušovanou, platí pro řidiče význam jen čáry bližší jízdnímu pruhu, ve kterém se nachází. Smí tedy předjíždět jen v místě, kdy je podélná čára z jemu bližší strany přerušovaná.



Obrázek 4.9 Vodorovné silniční značení upravující možnosti předjíždění

Předjíždění na křižovatce a její těsné blízkosti je dovolenou pouze v několika případech. Řidiči smí v křižovatce předjíždět, pouze pokud se jedná o předjíždění zprava nebo jde-li o předjíždění jízdního kola, mopedů a motocyklů bez postranního vozíku. Další situace, kdy je povoleno předjíždění v křižovatce, je, pokud vozidla přijíždí po hlavní pozemní komunikaci nebo pokud se jedná o křižovatku s řízeným provozem. Je důležité počítat s tím, že zdaleka ne všichni řidiči si možnost předjíždění v křižovatce uvědomují. Řidiči, vjíždějící do křižovatky po vedlejší pozemní komunikaci odbočující vpravo na hlavní, se velmi často podívají pouze vlevo. Navíc předjíždění na přechodech pro chodce (které jsou velmi často umístěny v okolí křižovatek) je velmi nebezpečné a striktně zakázáno. Jestliže hrozí sebemenší riziko kolize, je vhodnější nechat předjíždění na bezpečnější místo za křižovatkou. U předjíždění vždy platí: pokud si nejsme jistí, zda můžeme předjet, nepředjíždíme.

5 Návrh simulátoru

Součástí této práce je i návrh dopravního simulátoru. Předchozí kapitola 4 analyzuje a rozebírá základní dopravní pravidla, která jsou nutná pro vytvoření dopravního simulátoru. Cílem této kapitoly je vysvětlit, jak k pravidlům přistupuji ve vytvořeném simulátoru a jakým způsobem jsou tato pravidla použita a naimplementována.

Protože problematika simulátoru provozu je značně rozsáhlá a bylo by velmi komplikované vytvořit simulátor provozu od samého začátku (z časových důvodů a velké složitosti), rozhodl jsem se částečně vyjít z již existujícího simulátoru 3D autoškoly [6].

5.1 Simulátor 3D autoškoly

Simulátor 3D autoškoly [6] vytvořil Ing. Lukáš Pernica. Tento simulátor umožňuje simulaci jízdy vozidlem po městě. Mezi základní vlastnosti vozidla patří možnost výběru manuálního a automatického řazení rychlostních stupňů, samozřejmostí jsou jízda vpřed a couvání. Dále je to možnost výhledu z vozidla vpravo a vlevo pomocí myši. Tato vlastnost je důležitá zejména při jízdě přes křižovatku, protože při pohledu přímo dopředu nejsou vidět žádná potenciální vozidla přijíždějící ze stran. Navíc výhledu do stran musí v reálné situaci využít každý řidič vozidla, proto je dobré si na tuto nutnost zvyknout už při výuce řízení na simulátoru. Uživatel má možnost přepnout si pohled z pozice přímo za volantem (jako v reálném automobilu) do pozice za automobilem, což zvyšuje přehlednost v dopravních situacích, ale snižuje stupeň reálnosti.

Obrázek 5.1 Vzhled původního simulátoru 3D autoškoly – pohled z vozidla. Zdroj [6]

V simulátoru je implementováno mnoho dopravních značek. Při nedodržení některého zákazu (například zákazu vjezdu nebo překročení maximální povolené rychlosti) se uživateli na obrazovce zobrazí varovné hlášení. Textury na domech se vybírají náhodně, takže při opakovaném spuštění vypadají budovy jinak. Vážným nedostatkem této aplikace je neexistence ostatních účastníků dopravního provozu, tzn. uživatel nemá možnost vyzkoušet si vzájemnou interakci s ostatními automobily. Bez dávání přednosti v jízdě se není možné po reálných pozemních komunikacích pohybovat. Dalším vážným nedostatkem je absence semaforů, takže uživatel nemá možnost vyzkoušet si průjezd křižovatkou se semafory.

Tento simulátor je vytvořen v programovacím jazyce C++, který patří mezi nejrozšířenější jazyky. Z důvodu zvýšení efektivity a ušetření práce nebyly použity standardní zobrazovací knihovny jako OpenGL nebo DirectX, místo nich byl použit vhodný engine. Jako grafický engine byl vybrán OGRE [16], což je flexibilní 3D engine napsaný v C++, umožňující programátorovi abstrakci od úplných implementačních detailů, jako jsou například nastavení zobrazování, rozlišení, úroveň antialiasingu, ořezových rovin a mnoha dalších. Při použití tohoto enginu může navíc programátor jednodušeji pracovat s celou zobrazovanou scénou a objekty. Pro zastoupení fyziky v simulátoru byl použit fyzikální engine Bullet [23]. Tento engine je také napsán v jazyce C++ a umožňuje detekovat vzájemné kolize dvou objektů, které jsou pro dopravní simulátor nezbytné. Oba tyto enginy jsou volně použitelné pro nekomerční použití, navíc mají podporu pro Microsoft Visual Studio [24], ve kterém byl celý simulátor vytvořen.

Obrázek 5.2 Vzhled původního simulátoru 3D autoškoly – pohled shora. Zdroj [6]

5.1.1 Slabé stránky

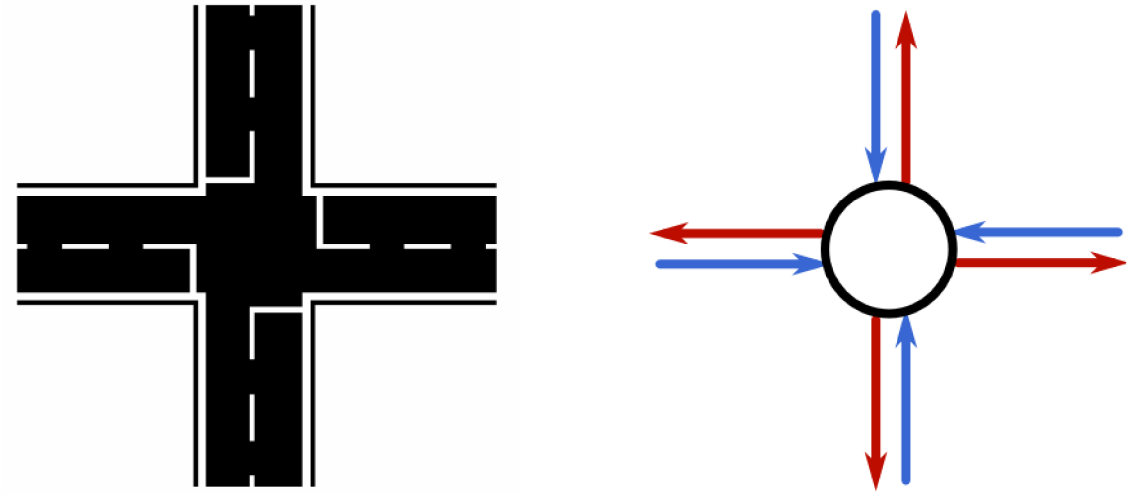
Jak už bylo zmíněno, mezi hlavní slabé stránky původního simulátoru patří neexistence ostatních účastníků provozu. Případným zájemcům o simulaci jízdy ve městě nemůže poskytnout dostatečně reálné prostředí městského provozu, zájemci o autoškolu si nemohou vyzkoušet „nanečisto“ jízdu v provozu a interakci s ostatními vozidly. Tento problém bude patřit mezi hlavní cíle řešené v této práci. Pro vytvoření města chybí vhodný editor, ve kterém by bylo možné jednoduše celé město vytvořit. Ideální by byl nějaký editor, kde by celé město šlo „naklikat“ myší, včetně vytvoření budov, osazení silnic dopravními značkami. Dalším závažným nedostatkem, který je třeba vyřešit, je nemožnost přidání semaforů do mapy města. Schopnost jízdy podle signalizace semaforu je jistě také důležitou vlastností každého řidiče, zejména při průjezdu městem. V původní aplikaci nelze vytvořit parkoviště a některé značky nejsou implementovány.

5.1.2 Rozšíření původního simulátoru

Pro přidání ostatních účastníků provozu bude v původním simulátoru nutné přepracovat reprezentaci křižovatek, zastoupení celé křižovatky jedním uzlem je nedostačující. Ve vytvořené aplikaci bude také nově možnost vytvoření mapy města editorem. O vytváření mapy města se zmiňuje kapitola 3. Po vytvoření rozšířeného grafu již bude možné přidat další účastníky provozu, což bude další krok. Každý z účastníků provozu musí městem projíždět podle určité logiky, aby byl schopen dodržovat pravidla provozu. Úkolem této logiky bude, aby chování vozidla co nejvíce odpovídalo chování skutečného automobilu jedoucímu po vozovce. Ke snadnějším úkolům bude nejspíš patřit pohyb vozidla po rovné vozovce. Ve své podstatě se bude jen jednat o pohyb ve směru hrany a o kontrolu případných okolních překážek (například ostatních automobilů). Již podstatně složitějším problémem bude samotný průjezd křižovatkou, pro jehož vyřešení je nutný dobrý návrh modelu křižovatky. Aby se přidané automobily chovaly a jezdily jako skutečné, musí bezpodmínečně dodržovat dopravní pravidla popsaná v kapitole 4.

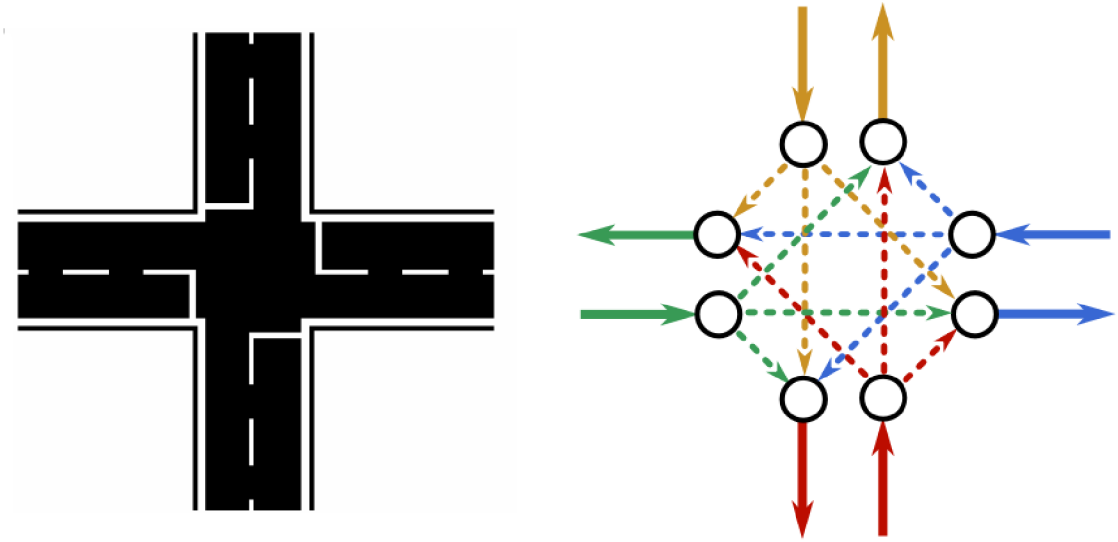
5.2 Model křižovatky

Aby bylo možné v aplikaci využívat silnic města, musí být všechny silnice reprezentovány nějakou vhodnou strukturou, která se bude snadno modelovat, procházet a bude u ní možné zjišťovat a zaznamenávat různé stavy (například obsazenost vozidlem, použití značky, semaforu, atd.). Pro reprezentaci celé silniční sítě byl už i v původní aplikaci zvolen orientovaný graf. Nicméně model křižovatky byl pro více účastníků provozu nedostatečný.



Obrázek 5.3 Model křižovatky v původní aplikaci

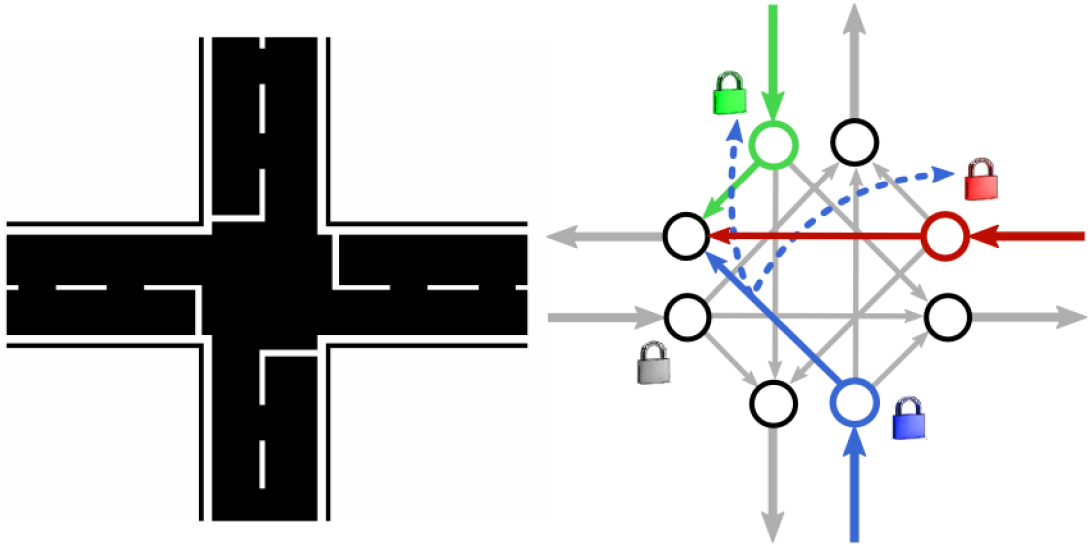
Obrázek 5.3 zobrazuje jeden uzel grafu silniční sítě v původní aplikaci. Každý silniční pruh je reprezentován jednou hranou, každá křižovatka je zastoupena právě jedním uzlem, do kterého vstupují hrany zastupující pruhy vjíždějící do této křižovatky. Pro jeden automobil projíždějící městskou sítí je tato reprezentace dostatečná. Automobil jede po hraně (reprezentující jízdní pruh), při vjezdu do křižovatky (od uzlu) si může vybrat některou z výstupních hran uzlu a tím odjet jízdním pruhem vedoucím z křižovatky pryč. Hrany sítě ale nemají informaci o ostatních hranách do uzlu vstupujících, dávání předností v jízdě by se s tímto modelem křižovatky řešilo dost těžko. Toto řešení je pro průjezd více automobilů křižovatkou nevhodné. Je tedy nutné původní uzel rozšířit, aby byl použitelný i pro průjezd více automobilů.



Obrázek 5.4 Rozšířený model křižovatky

Obrázek 5.4 znázorňuje, jak bude vypadat nová reprezentace křižovatky. Původní jeden uzel je nahrazen několika uzly, pro každou vstupní a výstupní hranu jeden. Z každého uzlu (v rámci jedné křižovatky) vede hrana do uzlu, do kterého může automobil jet. Tímto propojením se navíc vyřešil problém s přikázaným směrem nebo zákazem odbočení. Pokud

automobil nemůže do nějakého pruhu najet (zákaz odbočení), tak v grafu jednoduše není hrana, po které by mohl jet. Na tomto obrázku je tedy vidět model, ze kterého přijíždějící vozidlo „pozná“, kam může jet. Není tu ale žádná informace o dalších projíždějících vozidlech nebo obsazenosti křižovatky. K tomuto modelu je tedy proto nutné připojit další mechanismus, podle kterého budeme schopni rozpoznat, že křižovatkou už projíždí jiné vozidlo.



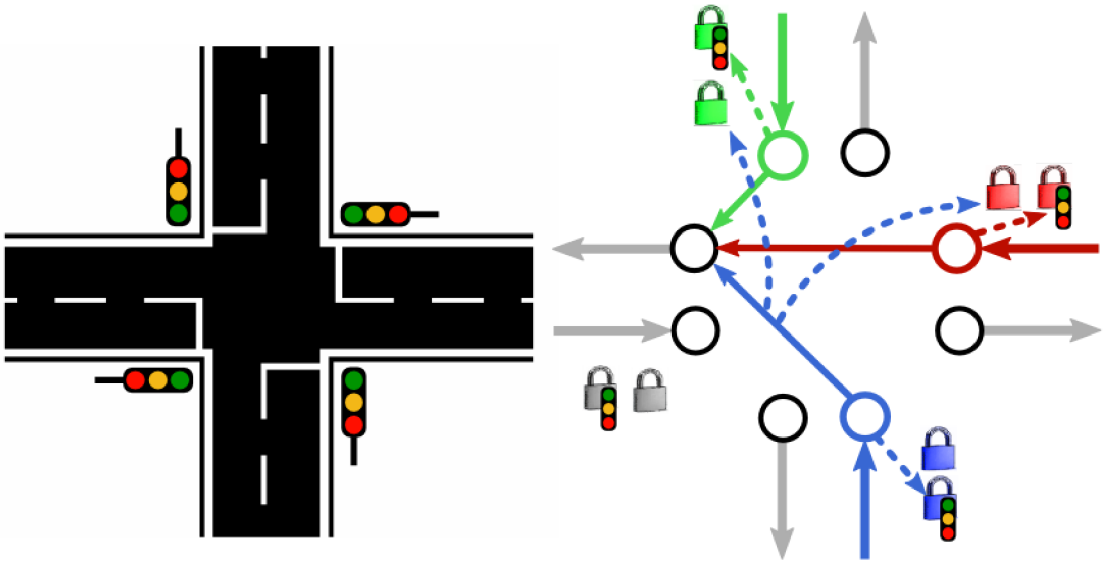
Obrázek 5.5 Model křižovatky se systémem zámků

V publikaci [15] byl popsán způsob řešení průjezdu automobilů křižovatkou pomocí systému zámků. Zámky slouží k určení, zda jízdní pruh (hrana), do které chceme vjet, není obsazen (uzamčen) automobilem, který má při vjezdu do tohoto pruhu přednost.

Obrázek 5.5 zobrazuje použití zámků v křižovatce bez označení (s předností zprava). Pro jednoduchost jsou barevně znázorněny pouze hrany důležité pro průjezd řidiče jedoucího po modré hraně (odbočujícího vlevo). Tento řidič při vjezdu do křižovatky „uzamkne“ modrý zámek a tím signalizuje, že jeho vstupní uzel je obsazen automobilem. Řidič hodlá pokračovat vlevo, musí projet po spojnici uzlů (hraně), ze kterého přijel a do kterého směřuje. Tato hrana má nastavený příznak závislosti na červeném a zeleném zámku. To znamená, aby mohl automobil po této hraně projet, musí být červený a zelený zámek odemčen, jinak dává přednost vozidlům na zelené nebo červené hraně a nikam nejede. Tímto způsobem je možné řešit i křižovatky s označenou předností v jízdě. Je pouze nutné správně postavit systém zámků a jejich závislostí podle konkrétní křižovatky a předností na ní.

Jak již bylo zmíněno, do aplikace bude možné přidat světelná signalizační zařízení (semafory). Aby v křižovatce byl umožněn průjezd podle semaforů, musí se původní model křižovatky ještě trochu přepracovat. Do původního modelu se musí přidat další zámek pro každý pruh, na kterém má být semafor umístěn. Pokud na semaforu naskočí červená, dojde k „uzamčení“ zámku, který k semaforu patří a tím ke znemožnění vjezdu vozidla do křižovatky.

Obrázek 5.6 zobrazuje rozšířený model křižovatky pro přidání semaforů. Pro jednoduchost je zobrazen průjezd řidiče opět pouze po modré hraně. Tento řidič při průjezdu ke křižovatce musí nejdříve zkontrolovat příznak zámku od svého semaforu. Pokud je tento zámek zamčen (na semaforu je červená), čeká na jeho odemčení a do té doby nemůže pokračovat v jízdě. Až je zámek odemčený (na semaforu naskočila zelená) může pokračovat v jízdě stejně jako v situaci bez semaforů. Nejdříve musí obsadit vstupní uzel zamčením modrého zámku. Poté zkontroluje, jestli může najet na spojovací hranu s koncovým uzlem. To znamená, že zkontroluje červený a zelený zámek. Jsou-li odemčeny, může pokračovat v jízdě, pokud je aspoň jeden z nich zamčený, musí dát přednost projíždějícímu vozidlu.



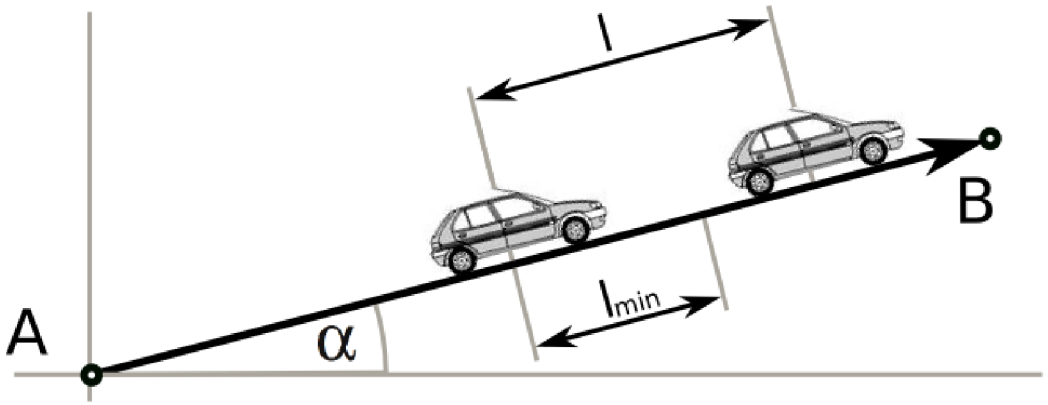
Obrázek 5.6 Rozšířený model křižovatky se systémem zámků pro přidání semaforů

5.3 Jízda po hraně

Kapitola 3 vysvětluje, jak se dá vytvořit město. Po načtení z vytvořeného souboru jsou ulice města reprezentovány vektorovým grafem, automobily mohou cestovat vždy v rámci tohoto grafu. Každý z úseků vozovky (je-li vozovka rozdělena na více úseků) je tedy definován počátečním a koncovým bodem a hranou mezi nimi. Z těchto informací dokážeme jednoduše vypočítat úhel hrany, tj. směr, do kterého se automobil při jízdě natočí. Jízda vozidla po vozovce pak znamená výpočet jeho nových souřadnic při konstantním posunu (podle rychlosti) ve směru hrany. Každý automobil je definován jedním bodem, umístěným ve svém středu. Na nově vypočítanou pozici se tedy posunuje střed vozidla.

Každý automobil si při jízdě po hraně musí kontrolovat mimo jiné i přítomnost dalších vozidel. Všechny hrany mají seznam vozidel, která po ní právě jedou. Každé z vozidel tedy musí kontrolovat tento seznam a v případě, že na své hraně není samo, musí sledovat vzdálenost od ostatních vozidel. Pokud zjistí, že v určité vzdálenosti před ním je další vozidlo, musí

zastavit. V opačném případě by mohlo dojít ke kolizi. Tímto způsobem je kontrolováno dodržování bezpečné vzdálenosti.

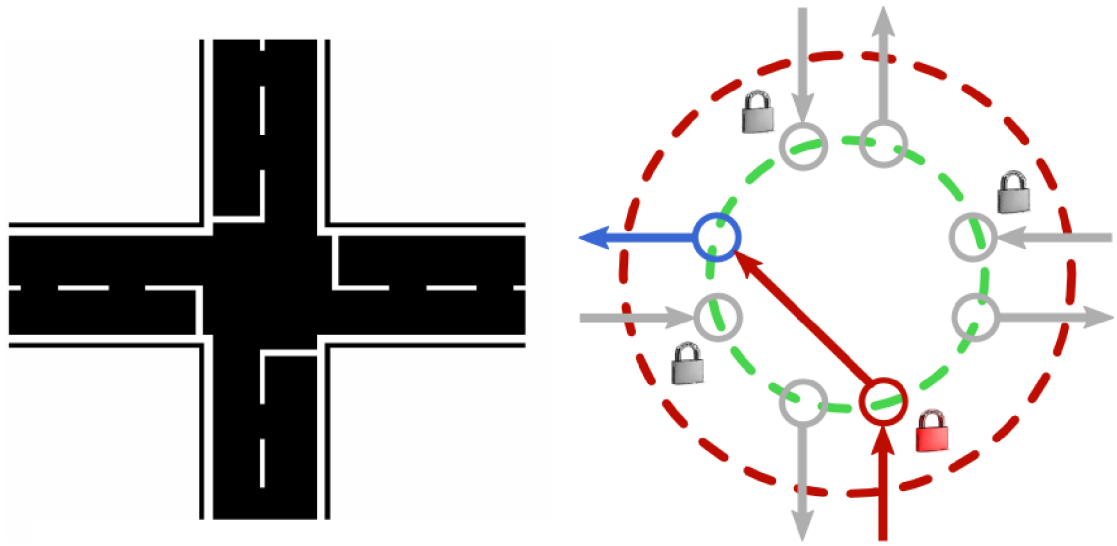


Obrázek 5.7 Princip kontroly bezpečné vzdálenosti mezi vozidly

Obrázek 5.7 zobrazuje jeden úsek cesty v grafu města, po kterém právě jedou dvě vozidla. Hrana vede z bodu ***A*** do bodu ***B***, pod úhlem α. Vozidla jsou od sebe vzdálena na vzdálenost l. Pokud před předním vozidlem vznikne nějaká událost, kvůli které by musel zastavit a nemohl pokračovat v jízdě (například povinnost dát přednost v jízdě jinému vozidlu), může zadní vozidlo dojet pouze na vzdálenost konstanty l_{min}. Tím bude dodržovat bezpečnou vzdálenost a odstup od vozidla před sebou.

5.4 Zamykání zámků

Průjezd křižovatkou a kontrola předností je založena na systému zámků popsaném v předchozích kapitolách. Pro názornost zde byl nakreslen pouze jeden zámek u každé hrany. Tento systém jednoho zámku by dokázal obsloužit pouze jedno vozidlo vstupující do křižovatky. Jenže reálný model města vyžaduje umožnění průjezdu i více vozidel. Proto každá hrana nebude mít pouze jeden zámek, ale seznam zámků všech přijíždějících vozidel. Při kontrole zda je hrana obsazena vozidlem, nekontrolujeme jeden zámek, ale musíme se podívat do seznamu zámků, zda obsahuje odkaz na nějaká přijíždějící vozidla (v implementaci potom stačí kontrolovat velikost tohoto seznamu, kdy nulová velikost je rovna volné, odemčené hraně). Aby vozidla mohla dostat přednost před tím, než do křižovatky najedou, zamykají se zámky už ve vhodné vzdálenosti před křižovatkou. Zamčení zámku tedy znamená přidání zámku vozidla do seznamu na hraně. Jakmile vozidlo opustí křižovatku, odebere svůj zámek ze seznamu a tím uvolní hranu, ze které přijelo (v případě, že do křižovatky po stejné hraně nepřijíždí jiné vozidlo).



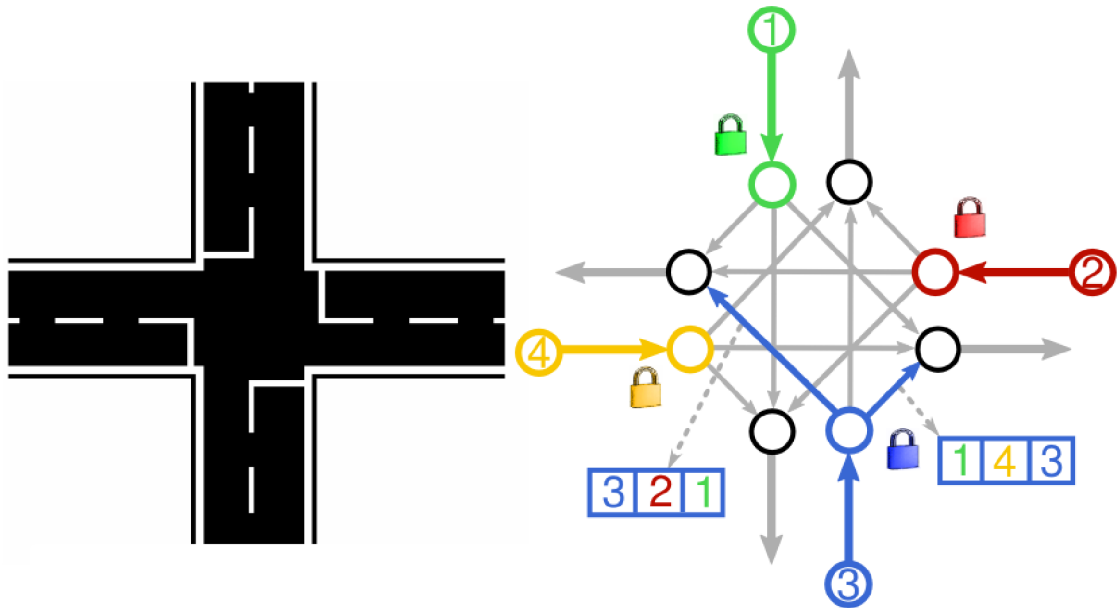
Obrázek 5.8 Znázornění principu zamykání zámků

Na obrázku 5.8 je znázorněn princip zamykání při průjezdu křižovatkou. V momentu, kdy vozidlo vjede do kruhu naznačeného červenou barvou (přiblíží se přibližně ke středu křižovatky na vzdálenost poloměru červeného kruhu), vloží do seznamu u příjezdové hrany ukazatel na své vozidlo a tím poznačí zamčení hrany. V případě, že má volno, najede do křižovatky na hranu, kterou si vybral. Při výjezdu z křižovatky najíždí do uzlu a následně na hranu, oboje naznačené modrou barvou. Opouští také kruh naznačený zelenou barvou. Při opuštění tohoto (zeleného) kruhu odstraní svůj ukazatel ze seznamu zámků u příjezdové hrany a tím uvolní svůj pomyslný zámek. Pokud za tímto vozidlem do křižovatky nenajíždí další vozidlo, je seznam u vstupní hrany prázdný a tedy hrana je odemčená. Tento princip je stejný pro každé vozidlo projíždějící křižovatkou.

Obrázek 5.6 ukazuje rozšířený systém zámků, ve kterém jsou přidány navíc zámky pro semafory. Tyto zámky jsou řízeny pouze podle semaforu. Jakmile je na semaforu červená, zámek se zamkne, když se na semaforu rozsvítí zelená, zámek se odemkne.

5.5 Průjezd křižovatkou

Kapitola 5.2 popisuje význam rozšíření modelu křižovatky. Pro průjezd křižovatkou je ale nutné vědět, která hrana (zámek na hraně) má přednost a jestli je obsazen. Přesně pro tento účel musí každá z hran v křižovatce mít seřazený seznam hran, kterým musí dávat přednost. V podstatě se vždy jedná o seznam hran seřazených zprava, které vedou na stejnou výstupní hranu z křižovatky. Takže při procházení tohoto seznamu dokážeme určit, jak jsou hrany zprava uspořádány. V případě, že jsme schopni rozhodnout, kdy je a není některá hrana obsazena a které hrany vedou zprava, dokážeme také říci, kterým hranám musíme dávat přednost v jízdě.

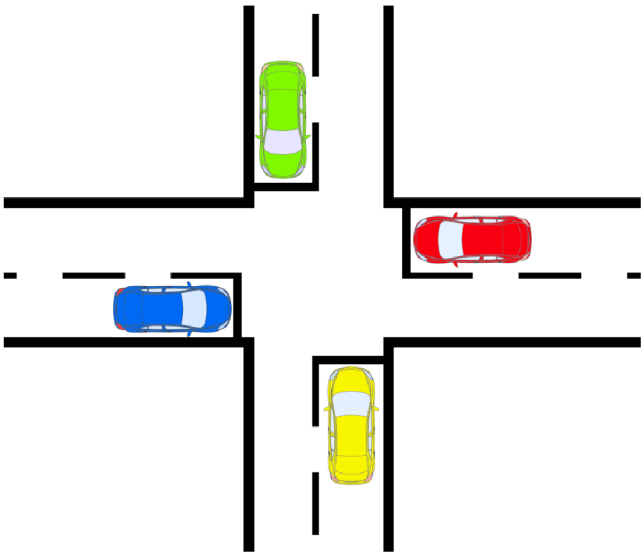


Obrázek 5.9 Ukázka vektoru předností v křižovatce (pro jednoduchost jen u dvou hran)

5.5.1 Křižovatka bez dopravních značek

V křižovatce bez dopravních značek je situace nejjednodušší. V případě, že některý z přidaných účastníků provozu dojede ke křižovatce a vybere si směr, kam pojede (následující hranu), musí procházet postupně zprava seřazený seznam hran, než najde svoji hranu. Pokud je zámek na některé z hran zamčen, znamená to, že hrana je obsazena a musí provést další kontrolu, jestli jej tato hrana blokuje a nesmí do jejího odemčení pokračovat v jízdě. Jestliže při procházení seznamu na žádnou zamčenou hranu nenarazil a zároveň zde objevil svoji příjezdovou hranu, znamená to, že nikomu přednost dávat nemusí a může projet křižovatkou. Pomocí tohoto seznamu lze vyřešit jak platnost dopravního pravidla o přednosti zprava, tak pravidlo o odbočování vlevo.

Na křižovatkách bez dopravních značek a semaforů může nastat situace, kterou nelze vyřešit podle pravidel o přednosti zprava a o odbočování vlevo. Jedná se o situaci, při které do křižovatky přijedou automobily ze všech stran. Pak každý automobil musí dávat přednost zprava a celá křižovatka se zdá zablokovaná. Řešením takovéto reálné situace je vzájemná domluva řidičů, například posunky a gesty, kdy se některý z řidičů vzdá své přednosti v jízdě a ostatní řidiči křižovatkou projedou, jako by jeho příjezdová komunikace nebyla obsazena. Toto ale v počítačovém zpracování není možné, musí tedy být zavedeno nějaké pravidlo, které tuto situaci vždy vyřeší.



Obrázek 5.10 Zablokovaná křižovatka

Ve vytvořeném simulátoru jsem se rozhodl situaci zablokování v křižovatce bez značek vyřešit tak, že v případě, kdy automobily přijíždějící do křižovatky zjistí, že jsou všechny příjezdové komunikace obsazeny a žádný z automobilů nemůže projet, nastaví se časový limit. Po uplynutí časového limitu jeden z automobilů křižovatkou projede i za předpokladu, že by měl dávat přednost jinému vozidlu zprava. Po jeho odjezdu se jeho příjezdová cesta uvolní (odemkne se zámek na jeho příjezdové hraně) a ostatní automobily mohou křižovatkou pokračovat, jako by k žádnému zablokování nedošlo.

5.5.2 Křižovatka označená dopravními značkami

Pokud je v křižovatce přednost řízena pomocí dopravních značek, je situace o trochu složitější. Nejprve je nutné rozhodnout, zda automobil přijíždí po hlavní nebo po vedlejší komunikaci. Hlavní a vedlejší komunikaci je možné rozpoznat podle indikátoru nastaveného hraně při načítání města z mapy. Pokud automobil přijíždí po vedlejší, musí nejprve zkontrolovat, jestli nekříží cestu nějakému vozidlu jedoucímu po hlavní komunikaci. V případě že kříží, nemůže do křižovatky najet a musí počkat, než se tato hrana uvolní a zámek na ní se odemkne. Pokud ale žádné vozidlo jedoucí po hlavní komunikaci při své jízdě neohrožuje ani neomezuje, může zkontrolovat případná jedoucí vozidla z vedlejších komunikací. Nyní už kontrola bude probíhat stejně, jako při průjezdu křižovatkou bez značek, popsané v kapitole 5.5.1. Jakmile automobil do křižovatky přijíždí po hlavní komunikaci, situace se částečně zjednoduší. Vozidla přijíždějící po hlavní komunikaci mají přednost před vozidly z vedlejší pozemní komunikace. V tomto případě tedy není nutné kontrolovat vozidla přijíždějící z vedlejších, stačí opět stejným způsobem provést kontrolu předností (obsazenost příslušných zámků) na hlavních komunikacích. Průběh kontroly tedy bude opět stejný s jedinou výjimkou, že budeme kontrolovat pouze hlavní komunikace.

Výhodou křižovatek označených dopravními značkami je, že zde nemůže dojít ke stejné situaci zablokování jako v případě bez dopravních značek. Pokud je některá z komunikací hlavní a některá vedlejší, nemůže dojít k cyklickému dávání přednosti zprava. Proto lze tento typ křižovatek (na rozdíl od křižovatek bez dopravních značek) lépe používat i v hustějším provozu.

5.5.3 Křižovatka s přednostní řízenou světelnými signály

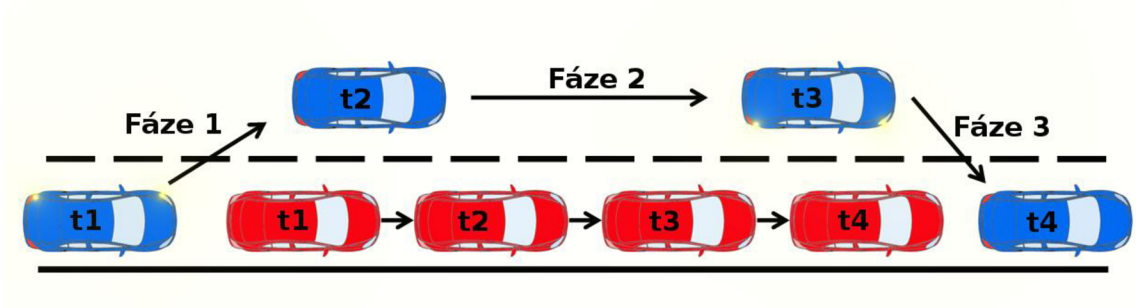
V křižovatkách, kde je přednost řízena světelnými signály, je situace opět docela přehledná. Světelné signály jsou pro každý jízdní pruh reprezentovány zámky, zamčený zámek u hrany symbolizuje červenou na semaforu. Je jasné, že pokud vozidlo přijede ke křižovatce a na semaforu svítí červené světlo (zámek této hrany je zamčený), nemůže pokračovat v jízdě a musí počkat na rozsvícení zelené. Naopak pokud přijede ke křižovatce a na semaforu určeném pro jeho pruh svítí zelená (zámek je odemčený), musí nejprve zkontrolovat, zda nemá nějakému vozidlu dávat přednost. Vzhledem k faktu, že při funkčních semaforech se dopravní značky neuplatňují, jedná se o kontrolu přednosti velmi podobnou jako v křižovatce bez značek. Rozdílem zůstává pouze počet kontrolovaných hran, v tomto případě kontrolujeme pouze přednosti pro hrany, které také mají zelenou na semaforu (tj. hrany, které také mají odemčený zámek semaforu). Hrany, které mají červenou na semaforu, nemohou křižovatkou projet, a proto je vůbec v kontrole přednosti nemusíme uvažovat.

Ani v tomto typu křižovatek nemůže nastat situace zablokování po příjezdu automobilů ze všech příjezdových komunikací. Světelné signály určují, ze které komunikace mohou a nemohou vozidla jet, a proto nehrozí, že by se snažila vjíždět ze všech stran zároveň.

5.6 Předjíždění

Zjišťování informací pro předjíždění v dopravním simulátoru je velmi podobné jako zjišťování informací při předjíždění ve skutečném provozu. Pokud se vozidlo přiblíží k jinému vozidlu, může se rozhodnout jej předjet. Tomuto rozhodnutí musí předcházet vyhodnocení okolní situace. Nejprve si „řidič“ musí zkontrolovat, jestli na vozovce (hraně) kde jede, nebyla značka „Zákaz předjíždění“ (případně značka „Zákaz předjíždění pro nákladní automobily“, jedná-li se o nákladní automobil). Dále musí vědět, že v místě, kde se chystá předjet, není jako vodorovné značení „Podélná čára souvislá“, neboť v opačném případě je předjíždění zákonem zakázáno. Následuje kontrola rychlostí obou vozidel. Rozdíl obou rychlostí musí být dostatečně velký, pokud by předjíždějící vozidlo jelo jen o málo rychleji, celý proces předjetí by trval příliš dlouho a mohlo by dojít ke kolizi s jiným vozidlem v protisměru. Je také nutné si zkontrolovat, jestli je volno v protisměrném pruhu, do kterého by předjíždějící vozidlo muselo najet. Tento pruh musí být volný až do jeho opuštění, je tedy nutné počítat i s tím, že předjíždění trvá určitou dobu a případné vozidlo jedoucí v protisměru se pohybuje nějakou rychlostí. Také musíme kontrolovat, jestli již nezačalo předjíždět nějaké vozidlo za námi, tj. jestli už levý jízdní pruh není obsazen předjíždějícím vozidlem. Dále je nutné sledovat vzdálenost od křižovatky a od konce hrany,

konec hrany může značit zatočení vozovky. Jednou z posledních věcí je kontrola, zda jede více vozidel v koloně, případně kontrola délky vozidla, které chceme předjet. Pokud by jelo za sebou více vozidel, musíme se ujistit, že budeme mít místo pro zařazení zpět do pravého jízdního pruhu. V případě, že se rozhodneme předjíždět větší množství automobilů, je nutné u všech sledovaných hodnot počítat s více vozidly.

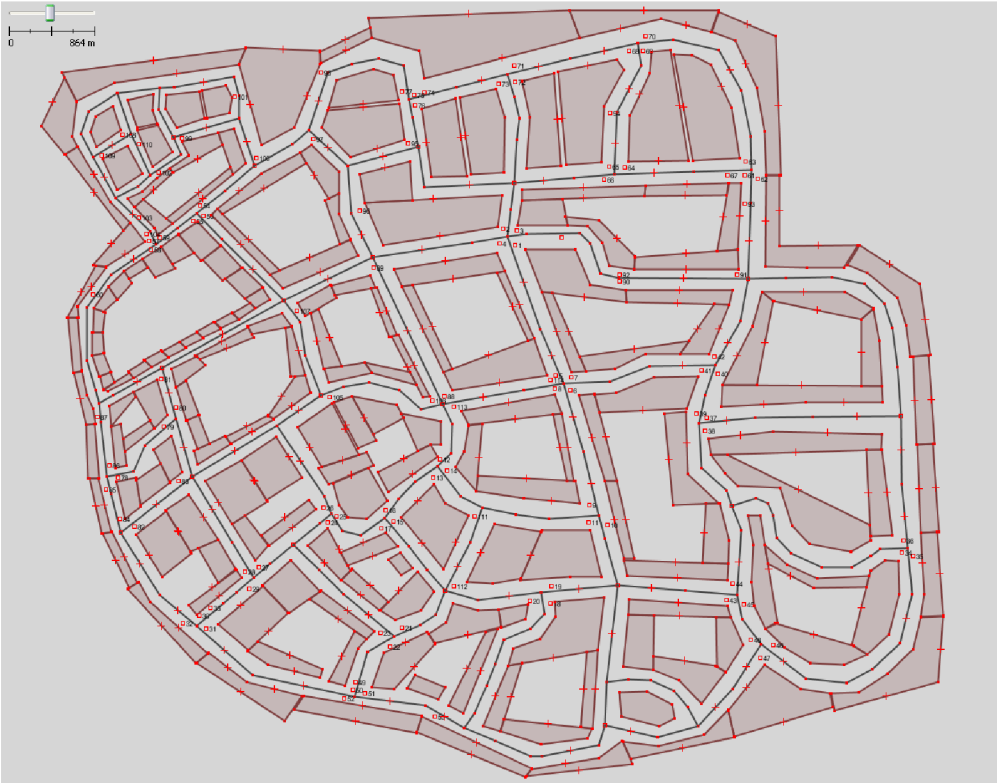


Obrázek 5.11 Znázornění fází předjíždění

Celý proces předjíždění je zobrazen na obrázku 5.11 a můžeme ho obecně rozdělit na tři fáze. Před počátkem samotného předjíždění musí být provedena výše popsaná kontrola, zda je předjetí vůbec možné bez kolize s jiným vozidlem. Po zjištění, že lze předjíždět, dojede modré vozidlo těsně za červené a může začít s procesem předjíždění. V této chvíli jsou časy vozidel označeny jako *t1*. První fází předjetí červeného modrým vozidlem je šikmé najetí do protisměrného pruhu, kde jsou obě vozidla v čase *t2*. Již víme, že protisměrný pruh je volný, a proto nehrozí kolize s vozidlem jedoucím v protisměru. Další fází je jízda protisměrným pruhem, kdy se modré vozidlo potřebuje dostat před předjížděné červené. V případě, že by rychlost předjíždějícího vozidla nebyla dostatečně velká, trvala by tato fáze příliš dlouho. Z tohoto důvodu je předjíždění možné jen v případě, že rychlost předjíždějícího vozidla je výrazně vyšší než rychlost vozidla předjížděného. Tato fáze končí, když se dostaneme dostatečně daleko před předjížděné červené vozidlo a můžeme se bezpečně zařadit zpět do pravého jízdního pruhu. Je důležité si uvědomit, že pokud předjíždíme více pomalejších vozidel, musíme dojet až před první z nich. Předjíždění celé kolony vozidel také může trvat velmi dlouho, proto takovéto předjetí patří k nebezpečným úkonům. Ukončení druhé fáze je označeno časem *t3*. Poslední fází je zařazení modrého vozidla zpět do pravého jízdního pruhu před vozidlo červené. Když modré vozidlo najelo dostatečně daleko před červené, již mu nic nebrání v nájezdu zpět. Tato fáze je tedy jen šikmé najetí do pruhu na pravé straně. Po ukončení této fáze, označené časem *t4*, modré vozidlo úspěšně předjelo pomalejší červené vozidlo a dostalo se před něj. Je tedy jasné, že nejdůležitější na předjíždění je rozpoznat, zda a kdy můžeme proces předjetí vykonat.

5.7 Vytvořený model města

Součástí hotové aplikace musí také být model města, po kterém se může uživatel projet. Rozhodl jsem se tedy v editoru JOSM [8] vytvořit dostatečně velké město, aby si uživatel mohl v simulátoru ověřit své jízdní vlastnosti. Dále by bylo vhodné, aby hustota provozu v tomto města byla vyšší, v opačném případě by si uživatel nemohl osvojit interakci s ostatními účastníky provozu. Po vytvoření relativně rozlehlého města jsem ale zjistil, že pro dosažení velké hustoty provozu by se v tomto městě muselo pohybovat velké množství automobilů. Pro ovládání tolika automobilů jsou ale kladeny velké nároky na hardware počítače, na počítacích, které nejsou dostatečně výkonné, by pohyb po tomto městě nemusel být plynulý. Z tohoto důvodu jsem se rozhodl vytvořit dvě města.

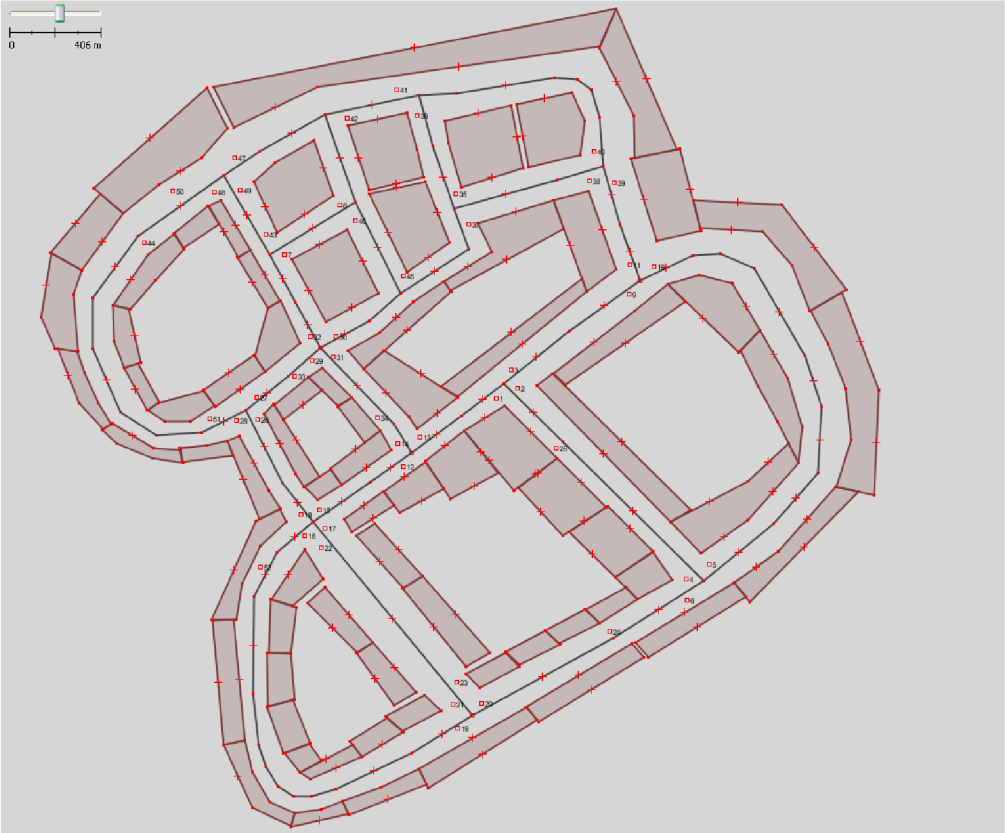


Obrázek 5.12 Půdorys většího města

Na obrázku 5.12 je vidět, jak vypadá větší město. V tomto městě je celkem 216 dopravních značek, 57 křižovatek, z toho 18 křižovatek se semafory. Jsou zde 4 dopravní značky „Zákaz zastavení“, 5 značek „Zákazů vjezdu“ a v celém městě je velké množství budov. Toto město je dostatečně velké na to, aby si uživatel vyzkoušel průjezdy všemi typy křižovatek, včetně řízených světelným signalizačním zařízením. Aby uživatel mohl na každé křižovatce potkávat několik jiných vozidel, muselo by se jich po celém městě pohybovat velmi mnoho. Z výše popsaných důvodů jsem vytvořil menší město s menším počtem vozidel.

Obrázek 5.13 zobrazuje půdorys menšího města. Druhé město je daleko menší, je v něm jen 16 křižovatek a 53 dopravních značek. Protože je toto město menší, stačí zde daleko méně

vozidel k vyšší hustotě provozu. Při projíždění se tedy uživatel bude často setkávat s ostatními účastníky provozu. Průjezd není tak náročný na výkon počítače, nenastává zde problém jako při průjezdu větším městem.



Obrázek 5.13 Půdorys menšího města

6 Zpracování modelů a textur

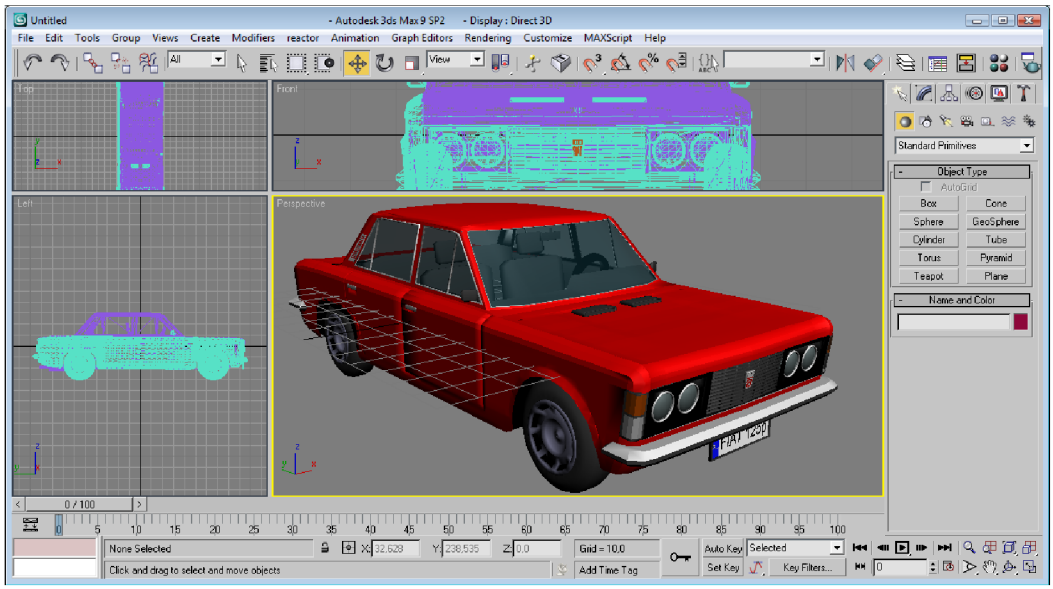
Cílem práce bylo vytvořit co nejvíce realistický simulátor dopravního provozu. Aby simulátor vypadal co nejreálněji, jsou v práci použity 3D modely skutečných automobilů. Na domech a okolí jsou umístěny různé, vhodné textury.

6.1 3D modely

Grafický engine OGRE [16] (ve kterém je celý simulátor vytvořen) umí načítat 3D modely ze svého souborového formátu *.mesh* a přiřazuje k nim materiály ze souborů ve formátu *.material*. Popis těchto souborů lze nalézt na manuálových stránkách enginu OGRE [36]. Při vytváření nějakého modelu je nutné, aby pro použitý modelovací software existoval exportér do těchto formátů.

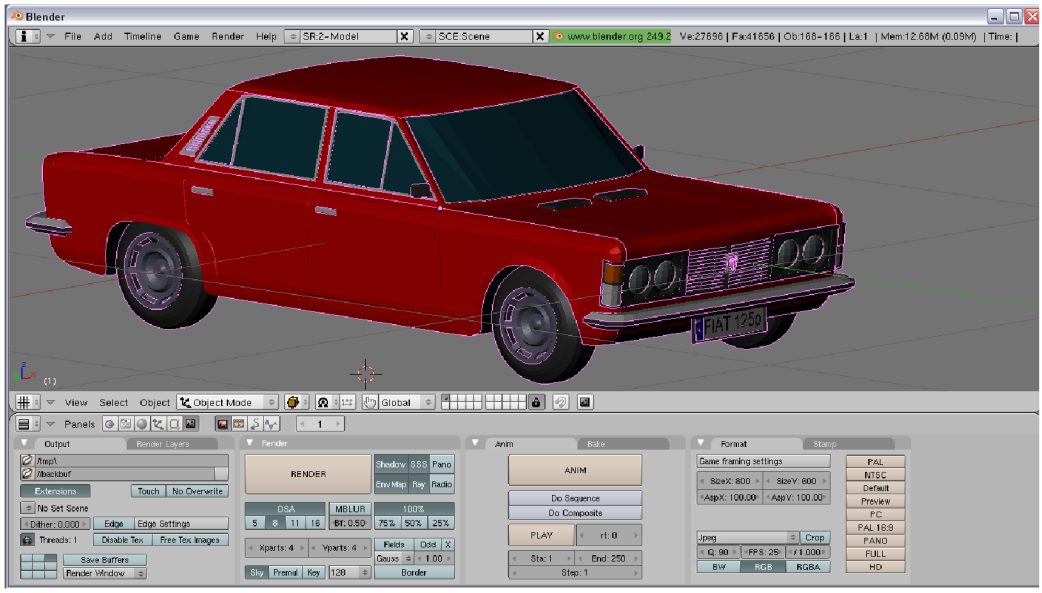
Vytvoření 3D modelu není vůbec jednoduchá záležitost, jsou na to potřeba předlohy modelů, modelovací prostředí a také velké množství zkušeností s modelováním. Modelovacích nástrojů, ze kterých je možné exportovat, je velké množství. Některé nástroje jsou zpoplatněny, některé se mohou využívat zcela zdarma. Výpis exportérů a postupů pro export je na webových stránkách s návody a tutoriály enginu OGRE [25].

- **3D Studio Max [26]** – je známý komerční, profesionální software pro tvorbu 3D grafiky, animací a vizualizací, ale i her. Má mnoho funkcí, možností a zásuvných modulů. Vyvíjí jej společnost Autodesk, podporuje velké množství souborových formátů (*max, 3ds, drf, chr, lay, dwg, dxf, ...*). Ze stránek společnosti lze stáhnout 30-ti denní trial verzi. Aktuální verze 2010 stojí v ČR zhruba 113400 Kč. Na práci s tímto softwarem je napsáno mnoho postupů, tutoriálů a knih. Tento nástroj je určen především profesionálům, kteří využijí velkou zásobu funkcí a jsou za něj ochotni zaplatit značnou finanční částku. Protože je tento nástroj tak všestranný, je pro naprosté začátečníky práce s ním dost složitá.



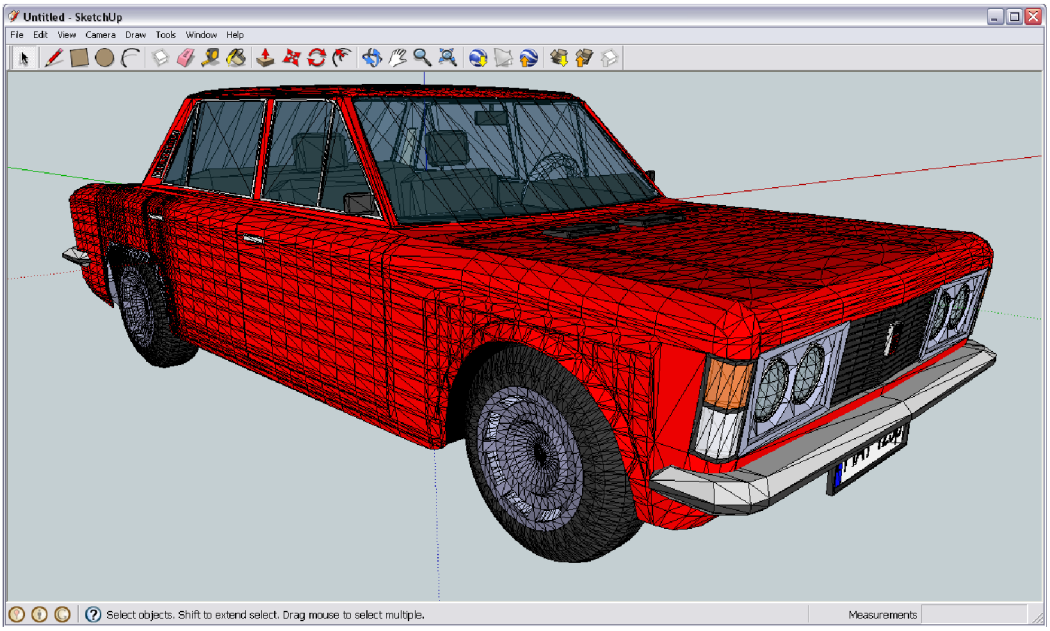
Obrázek 6.1 Prostředí aplikace 3D Studio Max

- **Blender [27]** – je také známý software pro modelování a vykreslování 3D grafiky. Tento open-source software je zdarma a lze jej používat na mnoha platformách (samozřejmostí jsou Windows a Linux). Pro tento software je také napsáno mnoho tutoriálů a návodů. Blender umožňuje tvorbu animací, má svůj vestavěný herní engine, kterým je možné vytvářet různé 3D aplikace (počítačové hry, vizualizace, …). Od verze 2.40 používá jako fyzikální engine Bullet [23], který je také použit pro tvorbu fyziky ve vytvořeném dopravním simulátoru. Vzhledem k faktu, že je tento software zdarma, je svými možnostmi velmi zajímavý. Jako začátečník jsem se v jeho ovládání trochu ztrácel, vzhledem ke složitosti uživatelského rozhraní na mě subjektivně působil velmi nepřehledně.



Obrázek 6.2 Prostředí aplikace Blender

- **Google SketchUp [28]** – je další z řady 3D modelovacích nástrojů. Tento software je uvolněn i jako freeware (placená verze přidává další možnosti pro tisk a export), je vyvíjen společností Google. Export pro engine OGRE je popsán na webových stránkách [29] a [25]. Tento nástroj se snaží prosadit svojí jednoduchostí, tvrdí o sobě, že se jej kdokoli naučí používat za několik desítek minut. Klade důraz na volnost a kreativitu uživatelů, není nutné zadávat žádné úhly a souřadnice. Už při pohledu na uživatelské rozhraní je jasné, že nebude nabízet tak velké množství funkcí a nástrojů, jako konkurenční produkty. Tento software je určen spíše pro rychlé vytvoření náčrtů a jednoduchých modelů. V neplacené verzi lze scénu exportovat pouze jako obrázek nebo video. Placená verze pak umožňuje exportování scén do souborových formátů *.dwg*, *.dxf*, *.3ds*, *.fbx*, *.obj*, *.xsi*, *.vrml*.



Obrázek 6.3 Prostředí aplikace Google SketchUp

- **Cinema 4D [30]** – je další komerční 3D modelovací software od společnosti Maxon Computer. Díky snadnému ovládání a široké paletě možností se stal velmi populární. Umí používat velké množství modulů a pluginů. Opět umí načítat a ukládat do mnoha souborových formátů (*.3ds*, *.dxf*, *.fbx*, *.vrml, .mov, .avi, …*). I pro tento nástroj je napsáno velmi mnoho návodů a lze z něj modely exportovat pro použití v enginu OGRE. Z webových stránek výrobce lze stáhnout zdarma demoverzi, která umožňuje ukládání modelů po dobu 42 dní. Placenou studentskou licenci lze (v období duben 2010) zakoupit od 2564 Kč.

Softwaru pro tvorbu 3D modelů, ze kterých je možný export do OGRE je velmi mnoho. Vytvoření několika reálně vypadajících modelů automobilů ale časovou náročností přesahuje rámec této diplomové práce. Proto jsem se rozhodl využít již existujících modelů, které mohou

být volně použity pro nekomerční účely. Takovéto modely je možné najít například na webových stránkách [31], [32] a [33]. Po stažení již vytvořených modelů byla nutná úprava v některém z výše uvedených nástrojů. U všech stažených modelů bylo zapotřebí upravit velikost a směr natočení modelu. Poté následoval převod s využitím vhodného exportéru do souborových formátů *.mesh* a *.material*.

6.2 Textury

Pro dosažení reálně vypadajícího prostředí bylo v aplikaci použito mnoho textur. Všechny textury, zobrazované na domech, cestách a okolí, byly vytvořeny z obrázků stažených z webových stránek [34] a [35]. Z těchto serverů je možno volně stáhnout velké množství obrázků pro nekomerční použití. Pro vytvoření použitelné textury bylo zapotřebí každý obrázek upravit. U všech bylo nutné odstranit okolní prostředí, upravit perspektivu, oříznout, změnit velikost a upravit tak, aby se při nanesení na objekt textura opakovala (pravá strana navazovala na levou). Ke všem těmto úpravám jsem použil rastrový editor GIMP. Tento editor je distribuován jako open-source a je dostupný na webových stránkách [37]. Je zřejmé, že čím větší počet textur domů ve výsledném simulátoru bude, tím bude simulátor různorodější.

Obrázek 6.4 Textura a jejího použití v aplikaci

V levé části obrázku 6.4 je zobrazena jedna z hotových, použitých textur. V pravé části stejného obrázku je vidět, jak vypadá tato textura nanesená na model domu z pohledu řidiče.

7 Instalace vývojového prostředí

Jak již bylo popsáno výše, vytvořený simulátor pro zdokonalení grafiky a fyziky využívá různé enginy. Jako grafický engine byl použit OGRE [16], jako fyzikální Bullet [23]. Vzhledem k faktu, že výběr a hlavně následná instalace a zprovoznění obou enginů nebyla vůbec jednoduchá a vyžadovala velmi mnoho úsilí, rozhodl jsem se popsat postup na zprovoznění těchto enginů a celého vývojového prostředí do zvláštní kapitoly. Tato kapitola vysvětluje, podle čeho byly vybrány použité enginy a také je zde popsán postup instalace těchto enginů. Součástí odevzdané práce je i CD se zdrojovými kódy. Bez správného nainstalování těchto enginů není možné pokračovat ve vývoji a rozšiřování již vytvořeného simulátoru.

7.1 Výběr enginu

Prvním z řady kritérií výběru byla možnost vytvářet aplikace s použitím enginu v programovacím jazyce *C* nebo *C++*. Dalším důležitým kritériem byla nulová cena a volné použití pro nekomerční aplikace. Dále byly důležité vlastnosti tohoto enginu, jak se s ním pracuje, zda podporuje objektově orientovaný návrh aplikace, zda umí načítat 3D modely, pracovat s fyzikou těles, zda má propracované manuálové stránky a mnoho dalších. Při hledání jsem použil databázi herních a grafických enginů dostupnou na [38]. Tato databáze obsahuje více než 300 enginů. Databázi je možné procházet a hledat v ní podle mnoha různých parametrů a vlastností, velkou výhodou je možnost zobrazení hodnocení a komentářů od ostatních uživatelů enginu.

- **Crystal Space 3D [39]** – Jako první jsem se snažil použít Crystal Space 3D. Podle databáze enginů dosáhl vysokého hodnocení, komentáře uživatelů jsou vesměs kladné a splňuje výše uvedené požadavky. Crystal Space je balík knihoven určených pro 3D grafické aplikace jako jsou hry nebo virtuální realita. Je volně použitelný a šířen pod licencí *LGPL*. Lze ho použít na běžných platformách (Windows, GNU/Linux, Mac OS X), je vytvořen v jazyce *C++*, k renderování využívá OpenGL. Protože se jedná o balík knihoven, je k jeho překladu potřeba překladač jazyka *C++*. Může implementovat mnoho modulů, podporuje objektově orientovaný návrh, zvládá základní fyziku, detekce kolizí objektů, umí pracovat se světly, stíny, texturami, umožňuje zpracovávat animace, zvuky a některé další efekty. Pro dynamické chování a detekce kolizí může používat engine ODE [17] nebo Bullet [23]. Pro tvorbu dopravního simulátoru vypadal jako ideální kandidát. Nicméně po stažení zdrojových kódů a ani po jejich bližším zkoumání se mi Crystal Space nepodařilo zprovoznit. Při kompilaci stále vznikaly chyby, které jsem nedokázal odstranit ani s pomocí z příslušného diskusního fóra. Bylo tedy nutné hledat jiný engine, který by mohl být použitelný.

- **OGRE [16]** – OGRE (Object-Oriented Graphics Rendering Engine) je objektově orientovaný, široce použitelný 3D herní engine. Je vytvořený v *C++* a podle databáze enginů také vypadá velmi slibně. Autoři o něm tvrdí, že je navržen s maximálním ohledem na jednoduchost a intuitivnost pro vývojáře aplikací využívající 3D akcelerace. Je multiplatformní, lze ho provozovat na Windows, Linux a Mac OS X, pro renderování umí používat OpenGL nebo Direct3D (pouze na Windows). Také je distribuován pod licencí LGPL jako volně šiřitelný, jeho třídy abstrahují všechny detaily používaných systémových knihoven, o které se tudíž vývojář nepotřebuje starat. OGRE není jen herní engine, je možné do něj přidat mnoho dalších knihoven. OGRE umí pracovat se světly, stíny, texturami, modely, umí zpracovávat animace, v aplikacích pracovat se zvuky, sítěmi, je možné přidat podporu základní fyziky a detekce kolizí (například pomocí fyzikálních enginů ODE [17] nebo Bullet [23]), má modely pro návrh grafického uživatelského rozhraní a mnoho dalších. Navíc na oficiálních webových stránkách je mnoho tutoriálů od základního sestavení scény, až po složitě propracované scény se světly a animacemi. Součástí webu je i živé diskusní fórum, kde je možné se zeptat na případné chyby nebo najít spoustu dalších informací. Tento engine se zdál velmi přívětivý a splňující všechny požadavky, a proto jsem se rozhodl ho vyzkoušet. Bylo tedy nutné najít vhodný fyzikální engine pro podporu fyziky v simulátoru.

- **ODE [17]** – Open Dynamics Engine je engine pro simulaci dynamiky pevných těles, umí detekovat kolize a chování objektů včetně tření. ODE je použitelný pro simulace automobilů, objektů v prostředí virtuální reality a virtuálních tvorů. Může být využit v počítačových hrách, 3D výukových nebo simulačních nástrojích. ODE je vytvořen v jazyce *C++* a je šířen jako svobodný software pod licencí LGPL a BSD. ODE lze používat v mnoha programovacích jazycích (*Delphi, Java, Python, .NET, FreeBASIC, JavaScript*, …). Pro propojení OGRE a ODE je potřeba použít modul nazvaný OgreODE. Na oficiálních stránkách enginu OGRE jsou napsány návody a postupy na propojení těchto enginů. Nicméně ani s těmito návody se mě nepodařilo zajistit bezproblémovou spolupráci obou enginů. Byl jsem tedy nucen vyzkoušet fyzikální knihovnu Bullet.

- **Bullet [23]** – Bullet je balík knihoven pro detekci kolizí a dynamické chování pružných i pevných těles. Nejvíce je používaný v počítačových hrách a ve filmech při tvorbě speciálních efektů. Je distribuován jako otevřený, svobodný software, je volně použitelný pro komerční i nekomerční použití a je uvolněn pod licencí ZLib. Bullet je také vytvořen v programovacím jazyce *C++*. Oficiální stránky obsahují uživatelský manuál a dokumentaci, je zde také možné nalézt mnoho tutoriálů a živé diskusní fórum. Pro spojení knihoven Bullet a enginu OGRE je zapotřebí modulu nazvaného OgreBullet. OgreBullet je modul, který umožňuje jednodušší integraci fyzikálních knihoven Bullet do projektu používající engine OGRE.

Kombinace OGRE a Bullet s využitím modulu OgreBullet se jeví jako vyhovující a proto byla v simulátoru použita. Přesnější popis verzí, celé instalace a nastavení popisuje následující kapitola.

7.2 Instalace prostředí

Simulátor je tedy vytvořen pomocí enginu OGRE [17] ve spolupráci s fyzikálním enginem Bullet [23]. Pro jejich propojení je zapotřebí použít modul OgreBullet [40]. Celý simulátor byl vytvářen ve vývojovém prostředí Microsoft Visual Studio [24], instalované enginy bylo potřeba zaintegrovat do tohoto prostředí. Postupů i návodů k jejich instalaci je jistě mnoho. Tato kapitola popisuje, jak jsem při jejich instalaci a nastavení postupoval já.

1. **Microsoft Visual Studio [24]** – Visual Studio je vývojové prostředí od společnosti Microsoft. Podporuje více programovacích jazyků, mezi vestavěné patří *C/C++ (použitím Visual C++), VB.NET, C#*. Nejdříve bylo potřeba nainstalovat toto vývojové prostředí. V mém případě se jednalo o Microsoft Visual 2005, označované jako verzi *8.0*.

2. **Bullet [23]** – Dalším krokem byla instalace enginu Bullet. Tento engine je možné stáhnout z oficiální stránky ve formě zdrojových kódů, já jsem použil verzi *2.70*. Stažený balíček jsem rozbalil do složky *C:\bullet*, složka následně obsahuje zdrojové kódy a také *Solution*[6], které je zapotřebí ve Visual Studiu otevřít a zkompilovat jako *release* verzi. Důležité je kompilovat správnou verzi *Solution*. Tuto verzi lze poznat podle názvu složky, ve které je umístěn. V mém případě jsem tedy otevřel a kompiloval soubor ze složky *.\msvc\8\wksbullet.sln* (neboť verze mnou použitého Visual Studia je *8.0*). Po bezchybném zkompilování se musí nastavit systémová proměnná BULLET_HOME na domovský adresář enginu Bullet (V mém případě tedy na hodnotu *C:\bullet*).

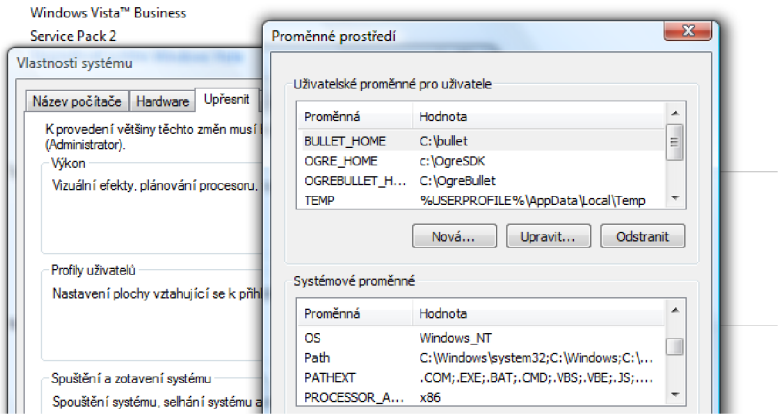
3. **OGRE [17]** – Z oficiálních stránek tohoto enginu lze stáhnout zdrojové kódy nebo SDK (Software Development Kit), který je již předkompilovaný a není tedy nutná další kompilace (stažený balík stačí pouze nainstalovat). Po mnoha experimentech s různými verzemi jsem zjistil, že lze použít SDK verzi *1.6.0*. Jedná se tedy o stažení a nainstalování souboru *OgreSDKSetup1.6.0_VC80.exe*, přičemž číslice 8 na konci opět určuje verzi Visual Studia. Na webových stránkách [41] jsou popsány návody a případné problémy vzniklé při instalaci. Stáhl jsem tedy a nainstaloval příslušný soubor do složky *C:\OgreSDK* (podle diskusních fór je vhodná například tato složka, naopak nevhodné

[6] **Solution** – Microsoft Visual Studio používá označení *Solution* pro strukturu obsahující jeden nebo více projektů, které jsou pomocí příslušné *Solution* organizovány. Soubor pro *Solution* typicky končí koncovkou *.sln*.

jsou názvy složek obsahující mezeru). Následně jsem nastavil systémovou proměnnou OGRE_HOME na domovský adresář tohoto enginu (v mém případě tedy na *C:\OgreSDK*). Změny systémových proměnných se projeví až po restartu počítače.

4. **Ogre SDK Application Wizard [42]** – Instalací tohoto skriptu se do nabídky Visual Studia přidá možnost vytvořit nový projekt jako *OGRE SDK Aplication* s pomocí průvodce. Výhodou použití průvodce je automatické nastavení potřebných proměnných a cest ke knihovnám už při vytvoření projektu. Popis instalace tohoto skriptu je popsán na webových stránkách [43]. Podmínkou bezproblémové instalace je správné nastavení systémové proměnné OGRE_HOME.

5. **OgreBullet [40]** – Dalším krokem je propojení enginu OGRE a Bullet instalací a zkompilováním modulu OgreBullet. Tento modul lze opět stáhnout ve formě zdrojových kódů nebo binárního balíčku (určeného pro OgreSDK) z oficiálních stránek. Protože jsem použil OgreSDK, pro instalaci tohoto modulu jsem stáhl soubor *OgreBullet_SourceSDK_Setup.exe*. Spuštěním tohoto souboru dojde k rozbalení zdrojových kódů, v mém případě do složky *C:\OgreBullet* (důležité je vyvarovat se opět mezer v názvu složky). Potom je vhodné nastavit další systémovou proměnnou OGREBULLET_HOME na hodnotu, ve které je tento modul nainstalován (v mém případě tedy *C:\OgreBullet*). Následně je nutné zkompilovat tyto zdrojové kódy otevřením a zkompilováním souboru *OgreBullet_SDK.sln* (který obsahuje Solution) ve Visual Studiu. Po úspěšném zkompilování se ve složce *%OGRE_HOME%\bin\release* vytvoří spustitelná ukázková aplikace, nazvaná *OgreBulletDemos.exe*. Pokud tento spustitelný soubor existuje, enginy a jejich knihovny jsou pravděpodobně správně zkompilovány a je tedy již možné začít s vývojem aplikací využívajících tyto enginy.



Obrázek 7.1 Příklad nastavených systémových proměnných

7.2.1 Nastavení aplikace a odstranění chyb

Pokud při kompilaci modulu OgreBullet vypsal kompilátor nějaké chyby, je zapotřebí zkontrolovat správné nastavení projektu. Visual Studio umožňuje různá nastavení každého projektu, jakými jsou nastavení knihoven a cest pro jejich hledání. O správné nastavení nového projektu by se měl postarat nainstalovaný Ogre SDK Application Wizard. Okno pro nastavení projektu je ve Visual Studiu možné najít v menu pod položkou *Project–>Properties* nebo při stisku klávesové zkratky *Alt+F7*. Každý vytvořený projekt by měl na záložce vlastností mít nastavené následující parametry.

Položka *Configuration Properties–>C/C++–>General–>Additional Include Directories* by měla obsahovat:

```
$(BULLET_HOME)\src
$(OGRE_HOME)\include
$(OGRE_HOME)\samples\include
$(OGREBULLET_HOME)\Demos\include
$(OGREBULLET_HOME)\Dynamics\include
$(OGREBULLET_HOME)\Collisions\include
..\include
```

Zdrojový kód 7.1 Výpis nastavení pro Additional Include Directories

Položka *Configuration Properties–>Linker–>General–>Additional Library Dependencies*:

```
$(BULLET_HOME)\lib\Release\
$(OGRE_HOME)\lib
$(BULLET_HOME)\out\release8\libs\
```

Zdrojový kód 7.2 Výpis nastavení pro Additional Library Dependencies

Položka *Configuration Properties–>Linker–>Input–>Additional Dependencies*:

```
OgreMain.lib
OIS.lib
OgreBulletCollisions.lib
OgreBulletDynamics.lib
libbulletcollision.lib
libbulletdynamics.lib
libbulletmath.lib
```

Zdrojový kód 7.3 Výpis pro Additional Dependencies

Toto nastavení je určeno pro verzi *Release*. Verze *Debug* by se v některých detailech lišila, zejména by se změnila jména linkovaných knihoven.

Pokud kompilace proběhla bez problémů a chyba vznikla až při spuštění aplikace, nejedná se o špatné nastavení projektu, spíše jde o problém s nastavením konfiguračních souborů aplikace. Například chybové hlášení o chybějícím souboru *d3dx9_37.dll*, značí nepřítomnost balíku DirectX. Na webových stránkách [41] je mnoho návodů pro instalaci, jsou zde i popsány případné chyby a problémy.

8 Závěr

Tato diplomová práce se zabývá problematikou simulace silničního provozu a metodikou kontroly dodržování pravidel provozu. Úkolem bylo analyzovat tato pravidla, navrhnout architekturu simulátoru a implementovat dopravní pravidla v simulátoru. Dopravní simulátor by měl sledovat dodržování prioritních pravidel, reprezentovat ostatní účastníky provozu a umožnit jejich vzájemnou interakci. Implementační částí jsem se rozhodl navázat na simulátor 3D autoškoly [6]. V simulátoru [6] bylo navrženo základní rozhraní autoškoly, kdy má uživatel možnost vyzkoušet si jízdu vozidlem po městě, dále je zde implementována kontrola základních dopravních předpisů, jako je překročení maximální dovolené rychlosti, zastavení v zákazu zastavení, najetí do zákazu vjezdu. Tato práce popisuje návrh modelu města, možnost vytvoření města externím editorem, jeho následné načtení a zobrazení v simulátoru, simulaci silničního provozu i s ostatními účastníky, metodiku kontroly dodržování pravidel. Použitý model křižovatky využívá principu zámků, podle kterých lze rozhodnout o dávání přednosti v jízdě v rámci křižovatky. Pomocí tohoto systému je možné implementovat křižovatky bez značek, se značkami, ale i se světelným signalizačním zařízením. Smyslem této diplomové práce je navrhnout a vytvořit architekturu dopravního 3D simulátoru, který by umožňoval uživateli průjezd po městě z pohledu řidiče, čímž může zdokonalovat svoje základní řidičské dovednosti. Takovýto simulátor se mi podařilo navrhnout a vytvořit. Uživatel může projíždět městem přes křižovatky s ostatními účastníky provozu, vyhýbat jim a narážet do nich. Ostatní vozidla dodržují platné dopravní předpisy a projíždí městem podle dopravních pravidel.

I přestože implementační část byla časově velmi náročná, jistě by se dala dále rozšiřovat. V aplikaci není implementováno předjíždění pomalejších automobilů. Pokud jedoucí automobil dojede vozidlo před sebou, pokračuje v bezpečné vzdálenosti za ním, ale nepředjede ho. Zajisté by se také dalo do aplikace přidat větší množství 3D modelů automobilů a textur domů. Aplikace by potom byla atraktivnější a vypadala by různoroději. Dále by bylo přínosné přidat chodce a přechody pro ně, uživatel by musel dávat pozor nejen na vozovku, ale i na její okolí, což by také zvýšilo reálnost simulátoru. V simulátoru zatím nejsou chodci implementováni. Další možností pro rozšiřování je přidání povětrnostních vlivů. Zatím může uživatel projíždět městem pouze za slunného dne, bylo by vhodné, kdyby si mohl zvolit mezi slunným dnem, mlhou, deštěm, nocí nebo třeba sněžením.

Tato diplomová práce se týká dopravního simulátoru, proto první kapitola vysvětluje význam a základní principy simulací a simulačních metod, jejich výhody a nevýhody. Je zde popsán postup pro vytvoření simulačního modelu. Hlavním cílem kapitoly je pochopení důležitosti využití simulací.

Další kapitola se věnuje návrhu modelu města. Tato kapitola popisuje různé vektorové modely, které je možné pro návrh města využít. Aby bylo možné do výsledného dopravního simulátoru jednoduše přidat město, je vhodné pro jeho vytvoření použít nějaký editor. Protože vytvoření editoru města by bylo časově velmi náročné, rozhodl jsem se pro tento účel využít již existujícího editoru. Použil jsem *Java OpenStreetMap Editor* [8]. Práce s ním, pravidla a postupy pro vytvoření města jsou také popsány v této kapitole. Editor umí vytvořený model

města vyexportovat do souboru, tento soubor dokáže vytvořený dopravní simulátor načíst, zpracovat a následně zobrazit jako město, ve kterém se uživatel pohybuje.

Další kapitola analyzuje dopravní pravidla. Jsou tu rozebrány typy a vzhled dopravních značek, princip světelného signalizačního zařízení. Dále jsou v této kapitole popsány typy křižovatek a u každého z těchto typů jsou vysvětleny priority kontroly pravidel.

V kapitole 5 je návrh dopravního simulátoru. Tato kapitola rozebírá vlastnosti simulátoru 3D autoškoly [6], na který implementační část navazuje. Jsou tu také popsány slabé stránky tohoto simulátoru, které bylo nutné odstranit. Je zde navržen rozšířený model křižovatky se systémem zámků, který je možné použít pro průjezd více automobilů na všech typech křižovatek. Kapitola také popisuje chování automobilů při jízdě po hraně a při příjezdu do křižovatky. Je tu také popsáno chování automobilů ve vytvořeném simulátoru při průjezdu křižovatek různých typů. Společně se simulátorem jsou vytvořena dvě města, uživatel má možnost volby mezi větším a menším městem. Aby se uživatel často setkával s ostatními účastníky provozu, musí být ve městě dostatečný počet vozidel. Pokud ke zvýšení hustoty provozu ve větším městě použijeme velké množství vozidel, nastává problém s nároky na výkon počítače. Z tohoto důvodu je vytvořeno ještě menší město, které pro dosažení vysoké hustoty provozu potřebuje vozidel daleko méně, a proto nároky na výkon nejsou tak vysoké.

Celá práce se týká dopravního simulátoru. Při tvorbě simulátoru jsem se snažil dosáhnout co nejrealističtějšího vzhledu výsledné aplikace. Aby simulátor vypadal realisticky, bylo nezbytné použít 3D modely automobilů a mnoho dalších textur (například na okolní budovy, silnice a okolí). Problematice využití 3D modelů a textur se věnuje kapitola 6. V této kapitole jsou popsány některé nástroje na tvorbu modelů, jsou zde také jejich výhody a nevýhody. Použité 3D modely a textury byly použity z různých zdrojů, tyto zdroje jsou také popsány v této kapitole.

Na vytvoření celého simulátoru byly zkoušeny a použity různé enginy. Zjištění jejich vlastností a následné zprovoznění bylo dosti komplikované. Některé ze zkoušených enginů se mě nepodařilo zkompilovat a použít. Proto instalaci celého vývojového prostředí je věnována poslední kapitola. Výsledný dopravní simulátor byl vytvořen s pomocí enginu OGRE [16] a fyzikálního enginu Bullet [23].

Literatura

[1] Peringer Petr: *Modelování a simulace*: Studijní opora. Brno : VUT FIT, 2006.

[2] Rabova Z. a kol: *Modelovani a simulace:* skriptum VUT, 1992

[3] *LHC – The Large Hadron Collider* [online]. 2009 [cit. 2009-12-14].
Dostupný z WWW: <http://lhc.web.cern.ch/lhc/>.

[4] *Understanding Evolution* [online]. 2009 [cit. 2009-12-14].
Dostupný z WWW: <http://evolution.berkeley.edu/>.

[5] *Supercomputer* [online]. 2008 [cit. 2009-12-15]. Dostupný z WWW:
<http://en.wikipedia.org/wiki/Supercomputer>.

[6] Pernica Lukáš: *3D Autoškola.* Brno, 2009, diplomová práce, FIT VUT v Brně.

[7] Hrubý Martin: *Geografické informační systémy (GIS*): Studijní opora. Brno: VUT FIT, 2006.

[8] JOSM: *Java OpenStreetMapEditor* [online]. [cit. 2009-12-16].
Dostupný z WWW: <http://josm.openstreetmap.de/>

[9] OSM: *OpenStreetMap* [online]. [cit. 2009-12-16].
Dostupný z WWW: <http://www.openstreetmap.org/>

[10] OSM: *OpenStreetMap* [online]. [cit. 2009-12-16].
Dostupný z WWW: <http://gama.fsv.cvut.cz/wiki/index.php/OpenStreetMap>

[11] Nosál Jaroslav: *Generátor krajiny pro simulátor jízdy*. Praha, 2007, diplomová práce, ČVUT.

[12] *Extensible markup language (XML)* [online]. 2007 [cit. 2009-12-20].
Dostupný z WWW: <http://www.w3.org/XML/>.

[13] *Hloubková analýza mezinárodního srovnání dopravní nehodovosti v ČR* [online]. 2008 [cit. 2009-12-21]. Dostupný z WWW:<http://www.czrso.cz/index.php?id=402>.

[14] *Nehody v dopravě: časové řady* [online]. 2009 [cit. 2009-12-21]. Dostupný z WWW:
<http://www.czso.cz/csu/redakce.nsf/i/nehody_v_doprave_casove_rady>.

[15] Hák Roman: *Dopravní simulátor*. Praha, 2007, bakalářská práce, ČVUT.

[16] *OGRE : Open Source 3D Graphics Engine* [online]. 2000 [cit. 2010-01-03].
Dostupný z WWW: <http://www.ogre3d.org/>.

[17] *Open Dinamics Engine* [online]. 2000 [cit. 2010-01-03].
Dostupný z WWW: <http://www.ode.org/>.

[18] *Nová pravidla: Hlavní změny v pravidlech* [online]. 2006 [cit. 2010-01-04]. Dostupný z WWW: <http://www.novapravidla.cz/HLAVNI-ZMENY-V-PRAVIDLECH/CELOROCNI-SVICENI>.

[19] *Nová pravidla: Hlavní změny v pravidlech* [online]. 2006 [cit. 2010-01-04]. Dostupný z WWW:<http://www.novapravidla.cz/HLAVNI-ZMENY-V-PRAVIDLECH/TELEFONOVANI-ZA-JIZDY>.

[20] TinyXML: [online]. [cit. 2010-04-04].
Dostupný z WWW: <http://sourceforge.net/projects/tinyxml/>

[21] *Dopravní park: časové řady* [online]. 2009 [cit. 2010-04-06].
Dostupný z WWW:<http://www.czso.cz/csu/redakce.nsf/i/dopravni_park_casove_rady>.

[22] *Dopravní infrastruktura: časové řady* [online]. 2009 [cit. 2010-04-06].
Dostupný z WWW:<http://www.czso.cz/csu/redakce.nsf/i/dopravni_infrastruktura_casove_rady>.

[22] *Zákon o silničním provozu* [online]. 2009 [cit. 2010-04-06].
Dostupný z WWW:<http://www.novapravidla.cz/ZAKON-O-SILNICNIM-PROVOZU>.

[23] Bullet: *Game Physics Simulator* [online]. 2008 [cit. 2010-04-14].
Dostupný z WWW: <http://bulletphysics.org/>

[24] Microsoft Visual Studio [online]. 2009 Microsoft Corporation, c2009 [cit. 2010-04-14].
Dostupný z WWW: <http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx>

[25] *Ogre Exporters: Ogre Wiki* [online]. 2009 [cit. 2010-04-20]. Dostupný z WWW: <http://www.ogre3d.org/wiki/index.php/OGRE_Exporters>
[26] Autodesk: *3ds Max* [online]. 2010. [cit. 2010-04-20]. Dostupný z WWW: <http://www.autodesk.com/3dsmax>
[27] *Blender* [online]. 2010 [cit. 2010-04-20]. Dostupný z WWW: <http://www.blender.org/>
[28] Google: *Google SketchUp* [online]. 2010. [cit. 2010-04-20]. Dostupný z WWW: <http://sketchup.google.com/>
[29] *SketchUp to Ogre Exporter* [online]. 2008 [cit. 2010-04-20]. Dostupný z WWW: <http://www.di.unito.it/~nunnarif/sketchup_ogre_export/>
[30] Maxon Computer: *CINEMA 4D* [online]. 2010. [cit. 2010-04-20]. Dostupný z WWW: <http://www.maxon.net/products/cinema-4d.html/>
[31] *ArchiDOM free 3D models* [online]. 2008 [cit. 2010-04-21]. Dostupný z WWW: <http://www.archidom.net/ddd/model_F.htm>
[32] *Free 3D models* [online]. 2009. [cit. 2010-04-21]. 3DM3.com. Dostupný z WWW: <http://www.3dm3.com/modelsbank/>
[33] PlanIt 3D: *free poser 3D models* [online]. 2009. [cit. 2010-04-21]. planit3d.com. Dostupný z WWW: <http://www.planit3d.com/source/meshes_files/vehicles01.html>
[34] *CG Textures: The worlds largest free texture site* [online]. 2008 [cit. 2010-04-21]. Dostupný z WWW: <http://www.cgtextures.com/>
[35] *Mayang's Free Texture Library* [online]. 2009 [cit. 2010-04-21]. Dostupný z WWW: <http://mayang.com/textures/>
[36] *OGRE Manual v1.7: Material Script* [online]. 2009 [cit. 2010-04-21]. Dostupný z WWW: <http://www.ogre3d.org/docs/manual/manual_14.html#SEC23>
[37] *GIMP: The GNU Image Manipulation Program* [online]. 2010. [cit. 2010-04-22]. GIMP. Dostupný z WWW: <http://www.gimp.org/>
[38] *DevMaster.net: The 3D Game and Graphics Engines Database* [online]. 2010. [cit. 2010-04-23]. Dostupný z WWW: <http://www.devmaster.net/engines/>
[39] *Crystal Space 3D: Main Page* [online]. 2010. [cit. 2010-04-23]. Dostupný z WWW: <www.crystalspace3d.org/>
[40] *Ogre Wiki: OgreBullet* [online]. 2009. [cit. 2010-04-27]. Dostupný z WWW: <http://www.ogre3d.org/wiki/index.php/OgreBullet>
[41] *Ogre Wiki: Instalation the Ogre SDK for Visual C++* [online]. 2009. [cit. 2010-04-27]. Dostupný z WWW: <http://www.ogre3d.org/wiki/index.php/Installing_the_Ogre_SDK_for_Visual_C%2B%2B>
[42] *Ogre AppWizards:* [online]. 2010. [cit. 2010-04-27]. Dostupný z WWW: <http://code.google.com/p/ogreappwizards/>
[43] *The Complete Blank Guide To Using The Ogre AppWizard* [online]. 2010. [cit. 2010-04-27]. Dostupný z WWW: <http://www.ogre3d.org/wiki/index.php/The_Complete_Blanks_Guide_To_Using_The_OGRE_SDK_AppWizard>

Seznam obrázků