VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

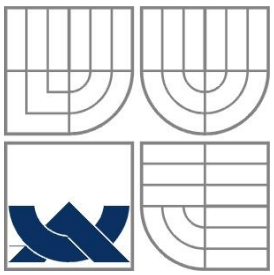
EDITOR PRO SIMULÁTOR DOPRAVY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

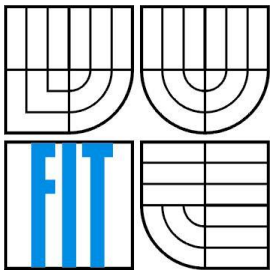
AUTOR PRÁCE
AUTHOR

MICHAL MALANÍK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

EDITOR PRO SIMULÁTOR DOPRAVY

EDITOR FOR TRAFFIC SIMULATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL MALANÍK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JAROSLAV ROZMAN, PH.D.

BRNO 2014

Abstrakt

Cílem této práce je vytvořit editor křižovatek s grafickým uživatelským rozhraním pro simulátor brněnské dopravy. Jako vstup editoru slouží soubor geografických dat stažený z volně dostupné databáze serveru OpenStreetMap. Ve stažených geografických datech jsou nejprve detekovány křižovatky. Detekce probíhá za pomoci detektoru, který byl také vytvořen v rámci této práce. Editor poté umožňuje editovat křižovatky tak, aby co nejrealističtěji odpovídaly skutečnosti. Tento proces zahrnuje modelování počtu jízdních pruhů, rozmístění různých typů semaforů, také dopravních detektorů a mnoha dalších objektů reálného světa. Editovaná data jsou následně k dispozici ve formě výstupního souboru ve formátu XML a jsou takto poskytnuta simulátoru dopravy.

Abstract

The aim of this work is to create an editor of intersections with a graphical user interface for Brno traffic simulator. As an input of the editor is used file of geographic data, which is downloaded from the freely available database of server OpenStreetMap. At first are in downloaded geographic data detected intersections. The detection is performed using a detector, which was also created within this work. Editor then allows to edit the intersections to be as much as possible accurate. This process involves modeling the number of lanes, layout and types of traffic lights and also traffic detectors and many other real-world objects. The edited data are then made available in the form of an output file in XML format and are hereby provided for the traffic simulator.

Klíčová slova

Simulace dopravy, OpenStreetMap, křižovatky, editor křižovatek, dopravní detektory

Keywords

Traffic simulation, OpenStreetMap, intersections, editor of intersections, traffic detectors

Citace

Malaník Michal: Editor pro simulátor dopravy, bakalářská práce, Brno, FIT VUT v Brně, 2014

Editor pro simulátor dopravy

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jaroslava Rozmana, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Malaník
20. 5. 2014

Poděkování

Rád bych tímto poděkoval vedoucímu mé bakalářské práce Ing. Jaroslavu Rozmanovi, Ph.D. za poskytnuté vedení a odborné rady při tvorbě této práce.

© Michal Malaník, 2014.
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů.

Obsah

Obsah 1
1 Úvod 2
2 Simulace dopravy 3
2.1 Makro simulace dopravy 3
2.2 Mikro simulace dopravy 4
3 Projekt OpenStreetMap 5
3.1 Historie OpenStreetMap 5
3.2 Struktura dat 5
3.2.1 Uzel (Node) 5
3.2.2 Cesta (Way) 6
3.2.3 Relace (Relation) 7
3.2.4 Značka (Tag) 7
3.3 Fyzické objekty 7
4 Struktura výstupního XML souboru 9
4.1 Struktura dat pozemních komunikací 9
4.2 Detekce křižovatek 9
4.3 Struktura dat křižovatek 10
4.3.1 Struktura jízdních pruhů 11
4.3.2 Struktura semaforů 12
4.3.3 Struktura dopravních značek 13
4.3.4 Struktura přechodů pro chodce 13
4.3.5 Ostatní údaje křižovatky 14
4.3.6 Struktura dopravních detektorů 14
4.3.7 Druhy dopravních detektorů 15
5 Implementace detektoru křižovatek 17
5.1 Použité knihovny 17
5.1.1 RapidXml 17
5.2 Funkčnost detektoru křižovatek 18
5.2.1 Vstupní soubor 18
5.2.2 Detekce křižovatek 18
5.2.3 Kontrola validity nalezeného křížení cest 19
5.2.4 Uložení nalezených křižovatek 22
5.2.5 Nastavení detektoru křižovatek 23
6 Editor křižovatek 24
6.1 Použité knihovny 24
6.2 Implementace editoru křižovatek 24
6.2.1 Hlavní prvky 25
6.2.2 Změna pozice grafických objektů 25
6.2.3 Popis činností po spuštění aplikace 25
6.2.4 Nabídky editačních nástrojů 26
6.2.5 Označení grafických objektů 27
6.2.6 Uložení naeditované křižovatky 27
7 Závěr 28
Bibliografie 29
Seznam příloh 31

1 Úvod

V České republice je silniční doprava nejoblíbenějším a také nejpoužívanějším druhem dopravy. Tato skutečnost vychází z dnešního životního stylu. Důraz je kladen především na rychlost a vzhledem k tomu je motorové vozidlo pro většinu lidí neodmyslitelnou nutností. Počet motorových vozidel na silnicích neustále roste, neboť je již zcela běžné, že jedna rodina vlastní více než jedno motorové vozidlo. Silniční doprava je také ve velké míře využívána k nákladní přepravě. Především kamionová doprava působí značné problémy, zejména pak v České republice, neboť pro evropské společnosti je finančně mnohem výhodnější vozit své zboží přes naši dopravní síť než po silnicích našich sousedů. Naneštěstí stav českých pozemních komunikací není nikterak uspokojivý a řešení dopravní infrastruktury taktéž není v některých případech zcela ideální.

Ze všech těchto skutečnostní vyplývá potřeba počítačových softwarů, které lidem pomohou se správným rozvržením současné nebo i budoucí dopravní infrastruktury. Software, který je používán odborníky pro zjišťování plynulosti dopravy na základě její simulace se obecně nazývá dopravní simulační nástroj. Hlavní směr, kterým se dnes snažíme ubírat je odklonění velkých dopravních tahů a především kamionové dopravy mimo centra měst na obchvaty. A to nejen kvůli plynulosti městské dopravy, ale také kvůli dopadu na životní prostředí. Nejen s analýzou a návrhem řešení těchto problémů jsou nápomocny právě dopravní simulační nástroje.

Cílem této práce je vytvořit editor pro simulátor dopravy, jehož úkolem je zpracovat geografická data stažená z databáze serveru OpenStreetMap a upravit je tak, aby mohla sloužit jako vstupní data simulačního nástroje brněnské dopravy. Úprava geografických dat zahrnuje především odfiltrování nepotřebných objektů, detekování křižovatek v dané mapové oblasti a následnou editaci těchto křižovatek za pomoci grafického uživatelského rozhraní. Výstupní data editoru pro simulátor dopravy jsou poté uložena do souboru ve formátu XML a jsou tak k dispozici k jejich dalšímu využití.

V kapitole 2 se nachází stručný popis teorie dopravní simulace. Tato část je zaměřena především na rozdíly mezi makro simulačními a mikro simulačními nástroji.

Vzhledem k tomu, že jsou v této práci využívána geografická data z databáze serveru OpenStreetMap, jsou v kapitole 3 uvedeny informace o projektu OpenStreetMap a popsány základní prvky tvořící jeho datový model.

Kapitola 4 se věnuje návrhu XML struktury pro uložení křižovatek. Jsou zde popsány jednotlivé úseky struktury sloužící pro uložení všech důležitých prvků křižovatky a jejich konkrétní příklad je předveden na ukázkové křižovatce.

Kapitola 5 je zaměřena na popis implementace a funkčnosti programu pro detekci křižovatek. Jsou zde také popsány příklady nevalidních křížení cest, které jsou v rámci detektoru křižovatek ošetřeny.

Popis jednotlivých prvků uživatelského rozhraní společně s popisem implementace editoru křižovatek je uveden v kapitole 6.

2 Simulace dopravy

Simulace dopravy je proces získávání nových znalostí o systému dopravní infrastruktury experimentováním s jeho simulačním modelem za pomoci počítačového softwaru, sloužící k lepšímu plánování, navrhování a samotnému fungování dopravní infrastruktury. Simulace v dopravě [1] je velmi důležitá, protože jsme díky ní schopni studovat modely, které jsou příliš komplikované pro analytické nebo numerické vyjádření, může být použita pro experimentální studie, a navíc nám může poskytnout vizuální výstup současného a budoucího scénáře. [2]

Dopravní simulační nástroje slouží k zjišťování plynulosti toku dopravy. Hlavním důvodem pro zjišťování těchto informací je předvídání tvorby dopravní zácpy a pochopení interakce jednotlivých vozidel na vozovce. V dnešní době, kdy populace naší planety neustále roste, se zvyšuje také počet motorových vozidel a hustota dopravy. Tato skutečnost vede k tomu, že v určitých denních hodinách je na pozemních komunikacích příliš mnoho motorových vozidel, a výsledkem jsou dopravní zácpy. Tento fakt se neprojevuje jen na snížení efektivity dopravní infrastruktury, ale taktéž má dopad na životní prostředí, neboť je do ovzduší vypouštěno větší množství výfukových plynů a také stoupá spotřeba motorového paliva. V rámci moderních trendů se odborníci snaží tuto situaci vyřešit a právě k tomu jim jsou nápomocny nástroje pro dopravní simulaci.

Při plánování nové pozemní komunikace je velice důležité znát budoucí zatížení dané komunikace, neboť dodatečné změny ve vybudovaném řešení jsou většinou buď zcela nemožné, nebo nesmírně nákladné. Zjišťování takových informací se souhrnně nazývá dopravní analýza. Její přesnost závisí na použitém modelu, tedy čím přesnější model máme, tím přesněji jsme schopni analyzovat budoucí zatížení pozemní komunikace. Principem dopravních modelů je pro dané účely co nejvěrněji modelovat pohyby vozidel a jejich vzájemné ovlivňování. Nelze však vytvořit jeden univerzální model, který by byl použitelný pro modelování všech situací. Hlavními kritérii jsou rozsah modelované sítě a míra přiblížení se reálnému světu. V zobrazení reálného světa jednoznačně dominují mikro simulační nástroje. Pro analýzy rozsáhlých sítí je však tvorba modelů těmito nástroji neúměrně náročná a zbytečně detailní. Zde je tedy vhodné využít větší míry abstrakce, což znamená použít makro simulační nástroje. Typickým příkladem abstrakce u makro simulačních nástrojů jsou křižovatky, které jsou zde vnímány jen jako uzly, bez většího vlivu jejich uspořádání. Naopak u mikro simulačních modelů je možné křižovatky modelovat do nejmenších detailů. Požadovaná úroveň detailnosti simulace je tedy úměrná rozsahu zkoumané sítě. [3]

2.1 Makro simulace dopravy

Makro simulace dopravy hodnotí dopravu jako celek, používá se při analýze v rámci větších oblastí, aniž by brala v úvahu charakteristiky a vlastnosti individuálních vozidel v provozu. Makro simulace dopravy vycházejí z matematických modelů, které popisují pohyb a chování všech vozidel [4]. Nevýhodou těchto modelů je to, že nerozlišují jednotlivé typy vozidel, ani neberou v úvahu rozdílné chování řidičů za volantem.

V reálném světě však existuje mnoho druhů vozidel, které se liší v mnoha parametrech, většina vozidel má rozdílnou maximální rychlost, velikost, váhu a brzdnou dráhu. Také reakce jednotlivých řidičů motorových vozidel se liší. Je prokázáno, že doba, která uplyne, než řidič zareaguje na nenadálou situaci v provozu, se zvyšuje s věkem řidiče. Také je třeba brát v potaz rozdílné styly řízení u jednotlivých řidičů, někdo preferuje opatrný a jistý styl jízdy, někdo jiný zase rychlý a agresivní styl jízdy. Při simulaci dopravy je třeba brát v potaz také chodce na přechodech. Tady se opět dostáváme k nedokonalosti v makro simulaci dopravy. Nedá se totiž předpokládat, že by

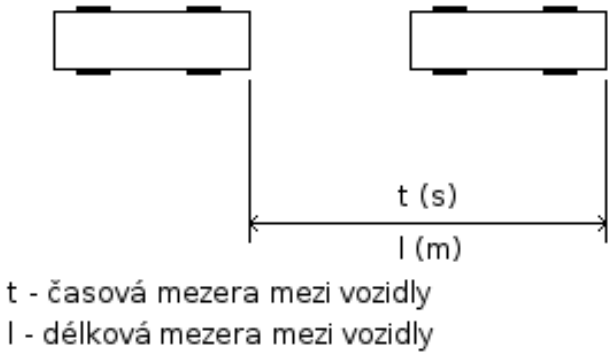
v reálném silničním provozu všichni chodci přešli přes přechod stejnou a konstantní rychlostí. Pokud tedy chceme simulovat dopravu opravdu reálně, musíme využít tzv. mikro simulace dopravy.

2.2 Mikro simulace dopravy

Mikro simulace dopravy bere v potaz modelování individuálních vlastností a charakteristik motorových vozidel v silničním provozu a využívá algoritmy k předpovídání a modelování pohybu každého vozidla v provozu. Jsou při tom zohledňovány všechny parametry dopravní infrastruktury i dopravních prostředků, a to včetně chování řidičů. Mikro simulace dopravy nám umožňuje detailně popisovat jednotlivé prvky dopravní sítě a je prováděna na základě přímé interakce mezi těmito prvky. Využívá se spíše pro simulaci menších částí dopravní sítě, často pouze jen jedné křižovatky.

Pro potřeby mikro simulace se zadávají obecně známé parametry, základem je kvalitní digitální podklad, který slouží k detailní geometrické definici posuzovaného objektu. Tímto objektem je myšlena buď určitá část pozemní komunikace, nebo křižovatka. V rámci této práce slouží jako zdroj pro definici geometrických vlastností objektů geografická data ze serveru OpenStreetMap. Samotnému projektu OpenStreetMap je věnována kapitola 3.

U silničních vozidel se zadávají údaje jako rozměry, maximální dosahovaná rychlost, brzdná dráha, výkon motoru, zrychlení a hmotnost vozidla. Součástí vstupů simulace jsou dále údaje o charakteristikách řidičů, cyklistů, chodců a dalších účastníků silničního provozu. Také je důležité zadat informace o vytížení komunikací a podílu nákladních vozidel, osobních automobilů nebo informace týkající se městské hromadné dopravy. Dalšími důležitými parametry souvisejícími přímo s mikro simulací jsou časová a délková mezera. Časová mezera je doba mezi průjezdy dvou po sobě jedoucích vozidel měřená v daném měřícím místě při průjezdu nárazníků v sekundách [5]. Délková mezera je vzdálenost mezi dvěma po sobě jedoucími vozidly měřená při průjezdu nárazníků.



Obrázek 2.1 - grafické znázornění měření časové a délkové mezery

Výstupem mikro simulace dopravy jsou poté údaje jako kapacita komunikací, nebo křižovatek, tvoření front vozidel a jejich délka, průměrná rychlost v daném úseku komunikace atp. Některé pokročilejší softwarové nástroje umožňují simulovat i náročnější odvětví, jako je například dopad pozemní komunikace na životní prostředí, využívají přitom údaje o emisích vypouštěných do ovzduší a také o průměrné spotřebě paliva motorových vozidel. V rámci mikro simulace může dále simulovat například vytížení parkovišť, provoz v místech dálnice, na kterých se platí mýtné a hlavně chování městské hromadné dopravy v silničním provozu.

3 Projekt OpenStreetMap

OpenStreetMap je projekt, který vznikl za účelem vytvoření podrobných geografických dat pod svobodnou licencí. Projekt je založený na kolektivní spolupráci, uživatelé po celém světě mohou vkládat informace do mapové databáze. U většiny ostatních dostupných map je jejich používání podmíněno, nebo alespoň omezeno zákonem. Oproti tomu OpenStreetMap umožňuje lidem po celém světě volně nakládat s jeho geografickými daty a využívat je tak k mnoha různorodým účelům. Díky této skutečnosti dal OpenStreetMap za vznik již mnoha projektům, které by se bez takto dostupných geografických dat neobešly. Důkazem toho je i vznik této práce. Vznik OpenStreetMap byl inspirován projekty, jako je například Wikipedie, což znamená, že data mohou být uživateli vkládána i editována a uchovává se kompletní historie provedených změn. Výsledek tohoto snažení je volně dostupný veřejnosti. [6][7][8]

3.1 Historie OpenStreetMap

V roce 2004 Steve Coast zakládá open-geodat projekt s příznačným názvem OpenStreetMap. O dva roky později vzniká v Anglii stejnojmenná nadace, která si klade za úkol shromáždění finančních prostředků pro podporu a rozvoj projektu OpenStreetMap. V prosinci roku 2006 se přidává firma Yahoo, která nabízí k dispozici své letecké snímky pro podporu tvorby map. Zatímco v roce 2006 bylo v projektu zaregistrováno cca 3000 uživatelů, začátkem roku 2010 již jejich počet přesáhl dvě stě tisíc a v lednu roku 2013 oznámila společnost OpenStreetMap první milion zaregistrovaných uživatelů. Od roku 2007 se v rámci OpenStreetMap každoročně pořádá konference State of the Map. [9][10]

3.2 Struktura dat

Samotná mapa je reprezentována mapovými daty, která se nacházejí v datovém souboru OpenStreetMap. Mapová data jsou tvořena několika základními prvky datového modelu OpenStreetMap a jejich vzájemnými vazbami.
Základními prvky datového modelu OpenStreetMap jsou:

- Uzel (Node)
- Cesta (Way)
- Relace (Relation)

Způsoby, jakými jsou mapová data pomocí těchto prvků tvořena, jsou vysvětleny v následujícím textu.

3.2.1 Uzel (Node)

Uzel definuje jeden specifický bod na zemském povrchu na základě jeho zeměpisné šířky a délky. Každý uzel musí obsahovat alespoň svůj unikátní identifikátor (id) a dvojici zeměpisných souřadnic.

Uzel může být použit k definování jedno bodového objektu na mapě, například požárního hydrantu nebo telefonní budky. Mnohem důležitějším použitím uzlu v rámci této práce je definování tvaru cest. Cesta je tvořena skupinou na sebe navazujících uzlů. [9]

Příklad uzlu v datovém souboru OpenStreetMap:

```
<node id="623899884" lat="49.2212172" lon="16.5844366" version="1"
timestamp="2010-01-31T10:43:41Z" changeset="3757768" uid="172938"
user="chimp"/>
```

Popis atributů používaných při definici vlastností uzlu:

- `id`: hodnota datového typu integer. Slouží jako jednoznačný identifikátor uzlu v rámci celé databáze OpenStreetMap.
- `lat`: hodnota určuje zeměpisnou šířku daného bodu.
- `lon`: hodnota určuje zeměpisnou délku daného bodu, společně s atributem `lat` tak tvoří kompletní zeměpisné souřadnice.
- `version`: hodnota datového typu integer, která určuje, o kolikátou modifikaci daného uzlu se jedná. U nově vytvořeného uzlu je automaticky přiřazena atributu `version` hodnota 1 a při každé modifikaci daného uzlu server automaticky inkrementuje tuto hodnotu o jedna.
- `timestamp`: hodnota datového typu W3C určující čas poslední modifikace uzlu.
- `changeset`: hodnota datového typu integer. Slouží jako jednoznačný identifikátor skupiny změn, které byly provedeny jedním uživatelem v rámci jednoho uložení těchto změn do databáze OpenStreetMap
- `uid`: hodnota datového typu integer sloužící jako jednoznačný identifikátor uživatele, který vytvořil nebo modifikoval daný uzel.
- `user`: hodnota datového typu string uchovávající jméno uživatele, který naposledy daný uzel modifikoval. Toto jméno si každý uživatel volí sám a vkládá pod ním veškeré informace do databáze OpenStreetMap.

3.2.2 Cesta (Way)

Cesta je seřazený seznam minimálně dvou a maximálně dvou tisíc uzlů, které spolu dohromady definují křivku. Cesta není chápána pouze jako pozemní komunikace, ale je pomocí ní definován jakýkoliv přímočarý objekt na mapě, například řeka. Pokud je první a poslední bod cesty shodný, což znamená, že cesta tvoří uzavřený útvar, mohou být pomocí cesty definovány i plochy na mapě. [10]

Příklad cesty v datovém souboru OpenStreetMap:

```
<way id="37667360" version="11" timestamp="2012-10-09T19:29:59Z"
changeset="13431980" uid="718054" user="Piskvor">
     <nd ref="88940708"/>
     <nd ref="617309053"/>
     <nd ref="623899884"/>
     <nd ref="431622104"/>
     <tag k="highway" v="secondary"/>
     <tag k="maxspeed" v="50"/>
     <tag k="name" v="Královopolská"/>
     <tag k="trolley_wire" v="yes"/>
</way>
```

Atributy `id`, `version`, `timestamp`, `changeset`, `uid` a `user` mají u cesty stejné vlastnosti, jako tomu bylo u uzlu. Velice důležité jsou zde elementy `nd` obsahující atribut `ref`. Hodnota atribut `ref` slouží jako odkaz na uzel tvořený na základě hodnoty atributu `id` odkazovaného uzlu. Tímto způsobem je v datovém modelu OpenStreetMap určeno, jaké uzly leží na jaké cestě a

v podstatě je tak definován tvar daného úseku cesty. Na příkladu můžeme vidět, že hodnota atributu `id` elementu `node` z předcházejícího textu týkajícího se uzlu je shodná s hodnotou atributu `ref` třetího elementu `nd` ve zde uvedené cestě. Takto můžeme snadno poznat, že uvedený uzel leží na zde uvedené cestě.

Zbývající elementy, které můžeme vidět v rámci definice cesty, jsou tzv. značky (tags), které blíže specifikují vlastnosti cesty, více si o nich řekneme níže.

3.2.3 Relace (Relation)

Relace určuje vztah mezi dvěma nebo více datovými elementy (uzel, cesta, nebo další relace) a definuje, jak spolu jiné datové elementy souvisí.

3.2.4 Značka (Tag)

Značka popisuje funkci elementu, ve kterém je uvedena. Značku mohou obsahovat všechny výše zmiňované datové elementy. Značka přesněji popisuje vlastnosti daného elementu, nebo určuje, jaký objekt z reálného světa je pomocí datového elementu definován, a to ve tvaru ' klíč ' = ' hodnota'. Například `<tag k="highway" v="motorway"\>` určuje, že se jedná o pozemní komunikaci (highway), která je zároveň dálnice (motorway). Jako příklad popisu vlastností za pomoci značky je vhodné uvést značku `<tag k="maxspeed" v="50"/>`, která určuje maximální povolenou rychlost silničního vozidla v kilometrech za hodinu. [11]

3.3 Fyzické objekty

V následujícím textu jsou popsány objekty, které jsou v datovém modelu OpenStreetMap definovány pomocí značky (tag). Jelikož je však takových objektů nepřeberné množství, jsou zde uvedeny pouze ty, které souvisí s touto prací, tedy objekty týkající se silniční dopravy. Neboť ostatní objekty jsou v rámci této práce z datového souboru OpenStreetMap odfiltrovány.

- **Dálnice** `<tag k="highway" v="motorway"\>`:
  Dálnice vyhrazená motorovým vozidlům, má dvě oddělené směrové části, přičemž každá z nich musí obsahovat alespoň dva jízdní pruhy, případně odstavný pruh.
- **Rychlostní komunikace** `<tag k="highway" v="trunk"\>`:
  Rychlostní komunikace, nebo jiná významná komunikace, která svými stavebními parametry nesplňuje předpoklady pro to, aby byla nazývána dálnice. Má nižší maximální povolenou rychlost než dálnice.
- **Hlavní silniční tah** `<tag k="highway" v="primary"\>`:
  Hlavní silniční tah, v České republice je administrativně začleněn jako silnice první třídy s čísly 1 až 70.
- **Silniční tah** `<tag k="highway" v="secondary"\>`:
  Silniční tah, v České republice je administrativně začleněn jako silnice druhé třídy s třícifernými čísli.
- **Silnice třetí třídy:** `<tag k="highway" v="tertiary"\>`:
  V ČR jsou administrativně začleněny jako silniční tahy s čtyřcifernými nebo pěticifernými označeními. Tato označení se odvíjí od silnic druhé třídy, popřípadě první třídy, na které je daná silnice napojena.
- **Lokální pozemní komunikace** `<tag k="highway" v="unclassified"\>`:
  Pozemní komunikace mimo administrativní zařazení první, druhé, nebo třetí třídy. Bývá

vedena mimo zastavěné území obce a umožňuje obousměrný provoz motorových vozidel po pevném povrchu vozovky. Nejčastěji se jedná o silniční spojení na lokální úrovni mezi místními částmi obce, nebo mezi obcemi.

- **Pozemní komunikace pro přístup k nemovitostem**
  `<tag k="highway" v="residential"\>`:
  Komunikace nacházející se v obci, nezařazená do vyšší administrativní třídy silnic a sloužící primárně pro přístup k nemovitostem.
- **Účelová pozemní komunikace** `<tag k="highway" v="service"\>`:
  Je určená pro přístup motorových vozidel a technického vybavení k průmyslovým objektům, nákupním centrům nebo různým areálům. Často se jedná o pozemní komunikaci s omezeným přístupem například pro zásobování, zákazníky, nebo dopravní obsluhu. [12]

4 Struktura výstupního XML souboru

Jako výstup editoru pro simulátor dopravy slouží soubor ve formátu XML. Tento soubor musí obsahovat data, která jsou nezbytná pro samotnou simulaci dopravy, neboť slouží jako vstupní soubor pro simulátor a na základě dat, která obsahuje, bude celá simulace probíhat. Musí v něm tedy být obsaženy informace o všech pozemních komunikacích, zastávkách městské hromadné dopravy a také o všech křižovatkách v rámci simulované oblasti. Návrh struktury výstupního XML souboru by se dal pomyslně rozdělit na dvě části. V první části jsou uložena data pozemních komunikací, která jsou získána z databáze serveru OpenStreetMap. V druhé části jsou uloženy informace o všech křižovatkách. Křižovatky je třeba nejprve detekovat, jejich existence totiž není v databázi OpenStreetMap explicitně uvedena. Následně je třeba veškeré informace o křižovatce, jako jsou např. počty a rozmístění jízdních pruhů, semaforů, dopravních značek a přechodů pro chodce, uložit do vhodné XML struktury. Všechny tyto části definující celkovou strukturu výstupního XML souboru budou podrobně popsány v následujícím textu.

4.1 Struktura dat pozemních komunikací

Data pozemních komunikací jsou získána z databáze serveru OpenStreetMap. Protože se však při stažení mapových dat simulované oblasti v datovém souboru nachází nepřeberné množství objektů, které nejsou pro simulaci dopravy vůbec potřebné, je nutné tato data z datového souboru odfiltrovat. Jedná se o data reprezentující například obytné domy a jejich adresy, ale také stromy v městském parku a mnoho dalších objektů. Tato přebytečná data způsobují, že vyhledávání dat důležitých pro simulaci dopravy je zbytečně náročné a neefektivní. K odfiltrování přebytečných dat je použit nástroj Osmfilter. Tento nástroj byl vyvinut přímo k filtrování dat datového souboru OpenStreetMap na základě jeho XML značek. Jedná se o nástroj, který je volně dostupný, včetně svého zdrojového kódu, na oficiálních webových stránkách podpory serveru OpenStreetMap [14]. Program funguje tak, že je mu na vstup předán vstupní soubor a filtrační pravidla, na základě kterých chceme data filtrovat. Program obstará filtraci dat a vrátí požadovaný výstup. Pro účely simulace dopravy je validní ponechat pouze data týkající se pozemních komunikací a dopravní infrastruktury. Za tímto účelem byla zvolena následující filtrační pravidla *keep="highway=" drop="highway=footway" drop="highway=steps" drop="highway=cycleway"*. Díky těmto pravidlům Osmfilter ponechá v datovém souboru pouze pozemní komunikace a zahodí všechny chodníky, schody a cyklostezky, které jsou ve struktuře dat OpenStreetMap také definovány jako cesty. Do první části struktury výsledného XML souboru jsou poté vložena odfiltrovaná data o pozemních komunikacích ze serveru OpenStreetMap, jejich samotnou syntaxi není nutné pro účely simulace měnit.

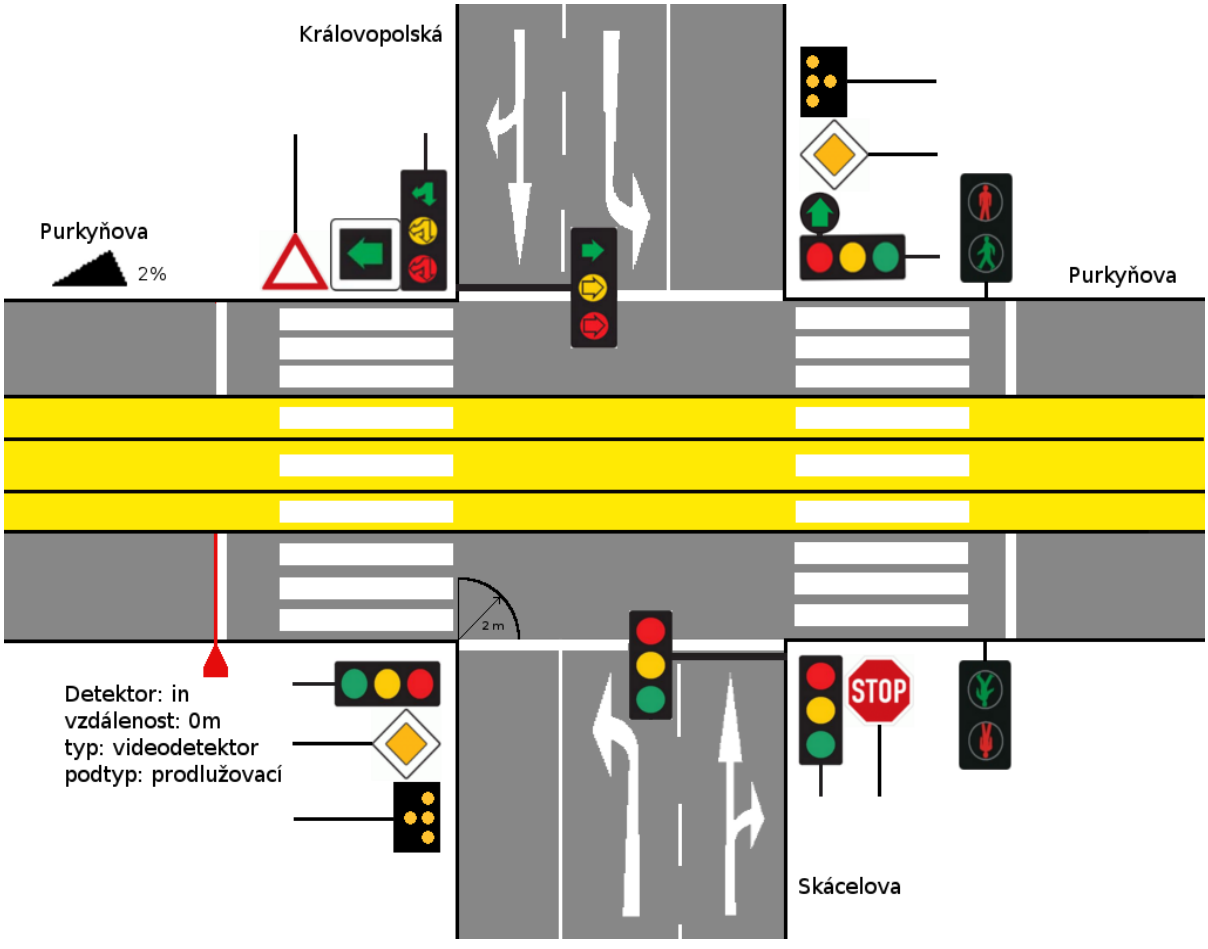
4.2 Detekce křižovatek

Velmi důležitou částí této práce bylo nalézt způsob, jak v datovém modelu OpenStreetMap detekovat místo, ve kterém se protínají dvě nebo více pozemní komunikací, neboť taková informace se sama o sobě v datovém souboru nenachází. Studií struktury dat obsažených v datovém souboru bylo zjištěno, že místo, ve kterém se pozemní komunikace protínají, je v datovém souboru označeno pomocí jednoho konkrétního uzlu. Protože je však každá cesta utvářena velkým množstvím uzlů a uzel, který označuje křížení cest, není nijak odlišený od ostatních, nelze křížení rozpoznat přímo. Proto je třeba křížení cest detekovat tak, že u každého uzlu zkontrolujeme, jestli neleží na dvou, nebo více cestách zároveň. Pokud takový uzel nalezneme, jedná se o křižovatku.

Za tímto účel byl implementován program, který systematicky projde všechny uzly na mapě a určí, zda se jedná o křižovatku. Pokud program křižovatku nalezne, zjistí její zeměpisné souřadnice, jména ulic, které se v křižovatce setkávají a také informaci o existenci světelné signalizace. Všechny tyto informace nakonec program uloží do struktury výstupního XML souboru. Další podrobné informace o každé křižovatce se již v datovém souboru mapy nenacházejí, a tudíž musejí být do výstupního XML souboru vloženy za pomoci editoru tak, aby co nejpřesněji odpovídaly skutečnosti. Editor křižovatek je taktéž součástí této práce a více podrobností o něm nalezneme v kapitole 6.

4.3 Struktura dat křižovatek

V rámci návrhu XML struktury pro uložení veškerých informací o křižovatkách byla vytvořena následující ukázková křižovatka a k ní odpovídající kód v jazyce XML. Tato křižovatka je svou geometrií inspirována křižovatkou, ve které se protínají Brněnské ulice Purkyňova, Královopolská a Skácelova. Z prezentačních důvodů a z hlediska přehlednosti návrhu byla však tato křižovatka upravena, jedná se zejména o rozdílné typy semaforů a dopravních značek proto, aby mohly být předvedeny různé varianty výstupních dat.



obr. 4.1: Návrh ukázkové křižovatky

V jazyce XML jsou všechny údaje o křižovatce obaleny elementem `intersection`. Příklad na ukázkové křižovatce:

```
<intersetcion id="1" lat="49.221502" lon="16.5852758">
    …
</intersection>
```

Atribut `id` zde obsahuje jedinečný identifikátor křižovatky, atributy `lat` a `lon` slouží pro určení zeměpisné šířky a délky dané křižovatky a tvoří tak spolu zeměpisné souřadnice.

Uvnitř elementu `intersection` je vnořen element `ways`, který obsahuje element `way` pro každou pozemní komunikaci, která do křižovatky vstupuje. Křižovatka je tedy definována tak, že se ukládají informace o každé pozemní komunikaci vstupující do křižovatky zvlášť a všechny pozemní komunikace jsou provázány právě pomocí elementu `ways`. Příklad na ukázkové křižovatce:

```
<ways>
      <way id="43530902" name="Purkyňova"> … </way>
      <way id="4823244" name="Skácelova"> … </way>
      <way id="37361258" name="Purkyňova"> … </way>
      <way id="37667360" name="Královopolská"> … </way>
</ways>
```

Atributy elementu `way` jsou `id` a `name`. `Id` slouží jako jedinečný identifikátor pozemní komunikace a `name` obsahuje název ulice, po které daná pozemní komunikace vede. Oba tyto identifikační údaje mají stejné hodnoty jako identifikační údaje té samé pozemní komunikace v databázi OpenStreetMap. Díky této skutečnosti je zajištěno provázání údajů o křižovatce s údaji o pozemní komunikaci v databázi OpenStreetMap, což je pro účely simulace zcela nezbytné. Zejména proto, aby při načtení křižovatky mohly být v simulátoru zjištěny údaje definující tvar a vlastnosti pozemní komunikace, která je její součástí.

Každý element `way` poté již přímo obsahuje informace o křižovatce, které lze vidět na obr. 4.1, tzn. počty jízdních pruhů, typy a rozmístění semaforů atp. V následujícím textu si popíšeme XML kód definující pozemní komunikaci Purkyňova, která se na obrázku nachází vlevo.

4.3.1 Struktura jízdních pruhů

Začneme údaji o jízdních pruzích, které jsou obsaženy v elementu `lines`.

```
<way id="43530902" name="Purkyňova">
      …
      <lines count="2">
            <lines_in>
                  <line id="1">
                        <dir>all</dir>
                  </line>
            </lines_in>
            <lines_out>
                  <line id="2"/>
            </lines_out>
      </lines>
      …
</way>
```

Atribut `count` elementu `lines` určuje celkový počet jízdních pruhů pozemní komunikace na hranici křižovatky. Element `lines` obsahuje vnořené elementy `lines_in` a `lines_out` pro rozlišení jízdních pruhů vstupujících a vystupujících z křižovatky z hlediska směru jízdy. Každý jízdní pruh má svůj jedinečný identifikátor v rámci dané pozemní komunikace (atribut `id` elementu

line). U vstupních jízdních pruhů je navíc třeba definovat, jakým směrem z něj lze pokračovat v jízdě křižovatkou. Tato informace je určena pomocí elementu dir. Ve zde uvedeném příkladu lze z jediného vstupního pruhu pokračovat všemi možnými směry, tzn. lze z něj odbočit doleva i doprava a taktéž je možné projet křižovatku rovně. Proto element dir obsahuje textovou hodnotu all označující všechny směry. Další přípustné hodnoty jsou straight, left a right. Pokud by jízdní pruh obsahoval kombinaci některých předešlých uvedených hodnot, například rovně a zároveň doprava, byl by element dir uveden dvakrát. U výstupních jízdních pruhů není třeba jejich směr definovat.

4.3.2 Struktura semaforů

Dále se zaměříme na údaje o semaforech, které jsou vnořeny v elementu traffic_lights.

```
<way id="43530902" name="Purkyňova">
      …
      <traffic_lights>
            <traffic_light>
                  <location>next_to</location>
                  <type>all_diriction</type>
            </traffic_light>
            <traffic_light>
                  <location>next_to</location>
                  <type>tram_light</type>
            </traffic_light>
      </traffic_lights>
      …
</way>
```

Každý semafor je definován elementem traffic_light, který obsahuje vnořené elementy location a type. Element location určuje, kde u silnice je daný semafor fyzicky umístěn, textová hodnota elementu location může nabývat hodnot over a next_to. Hodnota over znamená, že je semafor umístěn na dopravním sloupu nad vozovkou u hranice křižovatky. Hodnota next_to znamená, že je semafor umístěn na dopravním sloupu, který se nachází vedle vozovky taktéž u hranice křižovatky. Element type poté určuje, o jaký typ semaforu se jedná, ve zde uvedeném příkladu je první semafor všesměrový (hodnota all_direction elementu type) a druhý semafor je tramvajový (hodnota tram_light elementu type). Textová hodnota elementu type může dále nabývat hodnot:

- straight_right (semafor rovně a doprava)
- straight_left (semafor rovně a doleva)
- right (semafor jen doprava)
- left (semafor jen doleva)
- right_arrow značící šipku pro odbočení doprava
- exit_arrrow značící šipku pro opuštění křižovatky

Posledním volitelným elementem pro definici semaforu může být element over_line, pomocí kterého se dá přesně definovat, nad kterým jízdním pruhem se daný semafor nachází, na základě id daného jízdního pruhu. Tento element je přípustný pouze tehdy, když se semafor nachází nad vozovkou (hodnota over elementu location). Tento element by mohl vypadat následovně.

```
<over_line id="1">
```

4.3.3 Struktura dopravních značek

Údaje o dopravních značkách jsou obaleny elementem `traffic_signs`.

```
<way id="43530902" name="Purkyňova">
      …
      <traffic_signs>
            <sign>hlavni silnice</sign>
      </traffic_signs>
      …
</way>
```

Každá dopravní značka je definována elementem `sign`. Element `sign` obsahuje textovou hodnotu, která určuje, o jakou dopravní značku se jedná. Ve zde uvedeném příkladu se jedná o dopravní značku označující hlavní silnici.

4.3.4 Struktura přechodů pro chodce

Údaje o přechodech pro chodce umístěných na popisované pozemní komunikaci jsou vnořeny v elementu `zebra_cross`.

```
<way id="43530902" name="Purkyňova">
      …
      <zebra_cross>
            <distance>0</distance>
            <type>free</type>
      </zebra_cross>
      …
</way>
```

Textová hodnota elementu `distance` zde označuje vzdálenost přechodu od hranice křižovatky. Jednotkou této hodnoty jsou metry. Pokud je textová hodnota elementu `distance` rovna nule, znamená to, že přechod vede podél hranice křižovatky. Textová hodnota elementu `type` určuje, zdali se jedná o přechod pro chodce se světelnou signalizací, nebo bez světelné signalizace. Hodnota free označuje přechod pro chodce bez světelné signalizace, druhá varianta se světelnou signalizací je vyjádřena pomocí hodnoty traffic_signal.

4.3.5 Ostatní údaje křižovatky

V neposlední řadě je třeba definovat sklon silnice, poloměry zatáček a tramvajové koleje vedoucí po pozemní komunikaci.

```
<way id="43530902" name="Purkyňova">
      …
      <railway>tram</railway>
      <turning_radius>
            <right>2</right>
            <left>5</left>
      </turning_radius>
      <slope>2%</slope>
      …
</way>
```

Element `railway` s textovou hodnotou tram určuje, že po popisované pozemní komunikaci vede tramvajový pás. Element `turning_radius` slouží k definování poloměrů zatáček, textové hodnoty jeho vnořených elementů `right` a `left` určují poloměr zatáček doprava a doleva z dané pozemní komunikace. Jednotkou této hodnoty jsou metry. Textová hodnota elementu `slope` určuje sklon silnice směrem do křižovatky. Pokud tedy přijíždíme ke křižovatce do kopce, bude tato hodnota kladná, pokud z kopce, hodnota bude záporná.

4.3.6 Struktura dopravních detektorů

Na základě analýzy řešené problematiky bylo zjištěno, že pro úspěšnou simulaci dopravy ve městě je třeba zahrnout do systému také dopravní detektory. Tyto detektory se používají k zjišťování informací o obsazenosti zadaných míst vozidly, počtu dopravních prostředků projíždějících zadaným úsekem cesty nebo vybraným jízdním pruhem. Dále je lze použít k zjištění informací, jako je směr pohybu a druh dopravních prostředků, jejich aktuální rychlost nebo místa s jejich zvýšenou koncentrací. [13]

Ve struktuře jazyka XML jsou dopravní detektory obaleny elementem `detectors`.

```
<way id="43530902" name="Purkyňova">
      …
      <detectors>
            <detector>
                  <dir>in</dir>
                  <distance>0</distance>
                  <type>videodetektor</type>
                  <subtype>prodlužovací</subtype>
            </detector>
      </detectors>
      …
</way>
```

Údaje o každém jednotlivém detektoru jsou vnořeny v elementu `detector`. Element `dir` určuje, zdali se jedná o detektor umístěný na vjezdu, nebo na výjezdu z křižovatky. Textová hodnota

tohoto elementu může nabývat hodnot in, nebo out. Element `distance` určuje, v jaké vzdálenosti od hranice křižovatky se daný detektor nachází. Jeho textová hodnota udává počet metrů určující tuto vzdálenost. Elementy `type` a `subtype` vyjadřují typ a podtyp daného detektoru podle způsobu použití a podle použité technologie detektoru. Rozdělení dopravních detektorů je popsáno v následujícím textu.

4.3.7 Druhy dopravních detektorů

V závislosti na způsobu získávání údajů o dopravních prostředcích a použité technologii se detektory dělí na indukční, ultrazvukové, pneumatické, elektrostatické, mikrovlnné a optické.

Podle účelu použití se detektory dělí na:

- **Trolejové detektory**, které se používají pro detekci tramvají. Patří sem například pružinové detektory fungující na principu mechanického kontaktu, dále pak infračervené a indukční detektory umístěné těsně nad trolejovým vedením.
- **Výstupy z elektrického ovládání výhybek**, používají se pro směrovou detekci tramvají, tedy pro rozlišení směru jízdy tramvaje dle postavení výhybek. Systém je ovládaný dálkově z přijíždějících tramvají pomocí rádiového přijímače umístěného v kolejišti.
- **Vzdálená rádia**, primárně se používají pro směrovou detekci tramvají. Největší výhodou oproti detekci výstupu z elektrického ovládání výhybek je zjištění směru jízdy tramvaje na větší vzdálenost, což znamená, že je tramvaj dříve detekována a přihlášena k světelnému signalizačnímu zařízení. Díky tomu je zajištěn plynulejší průjezd tramvaje křižovatkou. Přijímač je umístěn v kolejišti.
- **Indukční detektory**, skládají se z vlastního detektoru, indukční smyčky a analytické jednotky. Pod povrchem vozovky se nachází vodič, tvořící indukční smyčku, která je součástí nízkofrekvenčního generátoru, jehož frekvence se mění v závislosti na přítomnosti předmětu na vozovce. Indukční detektor je díky své jednoduché konstrukci a vysoké spolehlivosti jedním z nejpoužívanějších silničních detektorů.
- **Infračervené detektory**, fungují na základě zjišťování pohybu ve vyzařovacím prostoru detektoru pomocí infračerveného světla. Infračervené detektory jsou značně nepřesné, neboť detekují každý pohyb a nejsou schopny rozeznat, zdali se jedná o silniční vozidlo. Jejich použití je vhodné tam, kde je z nějakého důvodu nemožné umístit indukční detektor do povrchu silnice, nebo tam, kde se detektor umísťuje jen dočasně.
- **Video detekce**, jejíž hlavní součástí je kamera, která bývá umístěna např. na sloupu veřejného osvětlení a snímá určitou oblast na dopravní komunikaci. Dá se použít pro detekci všech typů vozidel. Její hlavní výhodou je však to, že na rozdíl od infračervených detektorů je dobře softwarově řiditelná a proto se dá použít k detekci jen určitého typu vozidla. Tato skutečnost je důležitá pro preferenci MHD, neboť se dá kamera nastavit například tak, aby snímala jen vozidla, která jsou značně podobné autobusu, či trolejbusu. Nevýhoda video detekce spočívá v náchylnosti na povětrnostní podmínky. Vlivy jako mlha, silný déšť, husté sněžení nebo slunce svítící přímo do objektivu kamery silně ovlivňují přesnost detekce vozidel.
- **Radary**, které na rozdíl od ostatních druhů detektorů jsou schopny detekovat také informace jako rychlost vozidla, objem vozidla, zařazení v jízdním pruhu, či detekci kolony stojících vozidel. Bývají umístěny poblíž dopravní komunikace například na sloupech s veřejným osvětlením.

- **Datové zprávy**, zasílající se z vozidla vybaveného mobilním vysílačem do řadiče světelných signalizačních zařízení. Způsob přenosu se odvíjí od technologie, na základě které funguje mobilní vysílač. Může to být např. způsob šíření pomocí radiového signálu nebo přenos pomocí bezdrátové sítě. Tento typ detekce je uplatněný hlavně u preference vybraného typu vozidel, z praktického hlediska tedy hlavně pro detekci autobusů a trolejbusů MHD. Tyto silniční vozidla jsou totiž pomocí pasivní detekce (např. pomocí indukčních smyček) stěží rozpoznatelné, a tak je aktivní zasílání datových zpráv přímo z vozidla velice vhodným způsobem detekce. [13]

5 Implementace detektoru křižovatek

5.1 Použité knihovny

Většina implementace detektoru křižovatek souvisí se zpracováváním XML dat, neboť jak vstup, tak i výstup tohoto programu jsou soubory ve formátu XML. Kromě standardních knihoven programovacího jazyka C++ bylo tedy třeba zvolit správnou knihovnu pro práci se značkovacím jazykem XML. Bylo třeba, aby tato knihovna umožňovala nejen parserování a vyhledávání elementů v již existujícím XML stromu, ale také tvorbu a úpravu nového XML stromu. Zároveň však byly kladeny velké požadavky na rychlost se kterou knihovna umí zpracovávat XML data, neboť mapová data jsou velmi obsáhlá (mnohdy až statisíce řádků XML kódu) a pomalu pracující knihovna by celý proces detekce křižovatek výrazně zpomalila. Nakonec byla vybrána knihovna RapidXml [15] s volně dostupným zdrojovým kódem, která splňuje výše uvedené požadavky.

5.1.1 RapidXml

Jedná se o XML parser napsaný v programovacím jazyce C++. Jeho hlavní výhoda oproti ostatním knihovnám je skutečnost, že se jedná o in-situ parser. Tato vlastnost umožňuje dosahovat velkých parserovacích rychlostí, neboť in-situ parser nevytváří kopie textových dat, jako jsou např. jména a hodnoty elementů, ale namísto toho umísťuje ukazatele do zdrojového kódu v DOM hierarchii.

Další nespornou výhodou této knihovny je fakt, že její zakomponování do existujícího projektu není nikterak náročné. Celá knihovna se nachází v jednom hlavičkovém souboru rapixml.h. Další hlavičkový soubor určený pro tisknutí XML na výstup se nazývá rapidxml_print.h a je taktéž v detektoru křižovatek využit.

V tabulce 5.1 je možné vidět srovnání rychlosti RapidXml s dalšími XML parsery a s funkcí strlen(). Veškeré hodnoty jsou v jednotkách počet procesorových cyklů na jeden znak zdrojového textu.

Tabulka 5.1: Srovnání rychlosti RapidXml s jinými XML parsery

Platforma	Překladač	strlen()	RapidXml	pugixml 0.3	pugxml	TinyXml
Pentium 4	MSVC 8.0	2,5	5,4	7	61,7	298,8
Pentium 4	gcc 4.1.1	0,8	6,1	9,5	67	413,2
Core 2	MSVC 8.0	1	4,5	5	24,6	154,8
Core 2	gcc 4.1.1	0,6	4,6	5,4	28,3	229,3
Athlon XP	MSVC 8.0	3,1	7,7	8	25,5	182,6
Athlon XP	gcc 4.1.1	0,9	8,2	9,2	33,7	265,2
Pentium 3	MSVC 8.0	2	6,3	7	30,9	211,9
Pentium 3	gcc 4.1.1	1	6,7	8,9	35,3	316

Z tabulky vyplývá, že rychlost zpracování jednoho znaku zdrojového textu je u RapidXml několikanásobně větší, než u ostatních XML parserů a blíží se rychlosti zpracování funkce strlen().

Jediný parser, který je schopen držet s RapidXml krok je pugixml 0.3, právě tímto parserem byl inspirován vznik RapidXml.

5.2 Funkčnost detektoru křižovatek

Struktura programu sloužícího k detekování křižovatek na mapě je rozdělena do několika fází. Jako první se provede načtení vstupního souboru. Následuje odfiltrování přebytečných dat tak, jak bylo popsáno v kapitole 4.1. A to z toho důvodu, aby se následující operace prováděly jen nad potřebnými daty a tím se výrazně zvýšila efektivita celého programu. V dalším kroku se provádí samotná detekce křižovatek a s tím spojená kontrola validity křížení cest. A v poslední fázi se nalezené křižovatky uloží do XML souboru. Všechny fáze funkčnosti detektoru křižovatek budou nyní podrobně rozebrány.

5.2.1 Vstupní soubor

Jako vstup programu pro detekci křižovatek slouží soubor s příponou .osm. Jedná se o soubor obsahující mapová data oblasti, jejíž polohu a velikost si uživatel zvolí při stažení tohoto souboru. Od běžného XML souboru se liší pouze příponou, struktura dat je nezměněná, a tudíž je možné s ním pracovat jako s jakýmkoliv jiným XML souborem.

Samotné stažení vstupního souboru je možné přímo z oficiálních stránek OpenStreetMap [16] přes k tomu určené rozhraní, avšak jen pro malé velikosti souboru (řádově jednotky MB). Pro větší mapové oblasti, tudíž i pro objemnější datové soubory je nutné využít některý ze sekundárních serverů OpenStreetMap [17]. Tyto servery jsou v pravidelných intervalech synchronizovány s hlavní databází na primárním serveru a jsou určeny mimo jiné právě pro stahování obsáhlých mapových dat za účelem nepřetěžování primárního serveru. Jen pro představu, soubor obsahující mapová data Brněnské městské části Žabovřesky (cca 2 MB) bylo možné stáhnout přes rozhraní na primárním serveru. Soubor s mapovými daty centra Brna již však obsahoval příliš mnoho informací o objektech nacházejících se v této oblasti, a tudíž musel být stažen ze sekundárního serveru.

Z hlediska implementace detektoru křižovatek je tento vstupní soubor, stažený uživatelem, předán na vstup programu pomocí vstupních parametrů a načten do proměnné datového typu `xml_document<>` z knihovny RapidXml. Dále se pracuje již jen s touto proměnnou, nikoliv se vstupním souborem, ten zůstává nedotčen.

5.2.2 Detekce křižovatek

Po načtení vstupního souboru je zavolána funkce `find_intersection()` zajišťující samotnou detekci křižovatek. Tato funkce postupně načítá ze vstupního XML souboru jednotlivé mapové uzly (viz. 3.2.1) a pro každý z nich zkontroluje, zdali se v tomto bodě neprotínají dvě nebo více cest (viz. 3.2.2). Pokud je takový uzel nalezen, jsou o něm zjištěny další informace, jako je jeho unikátní identifikační číslo, zeměpisné souřadnice pro určení polohy a informace o všech cestách, které se v uzlu setkávají. Z informací o cestách je zjištěno jejich unikátní identifikační číslo a jméno. Všechny tyto informace jsou poté uloženy do proměnné datového typu vector, jehož prvkem je struktura, která je i s naplněnými daty zobrazená v tabulce 5.2.

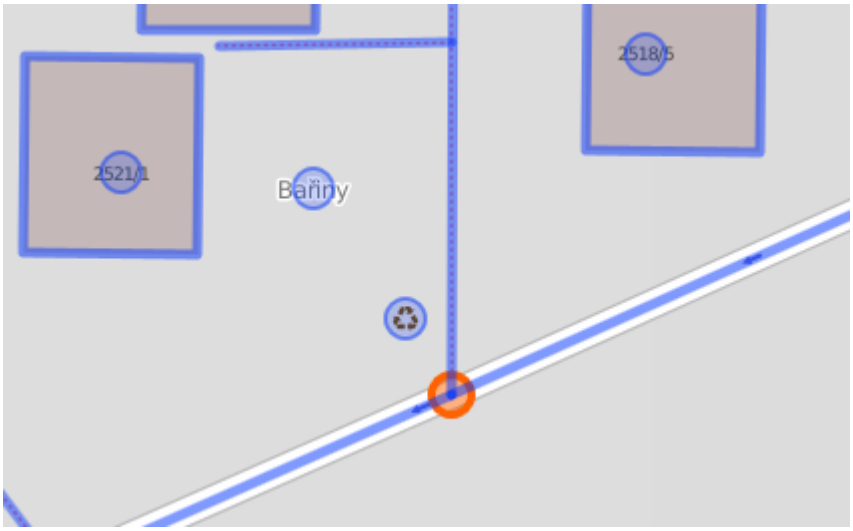
Tabulka 5.2: Struktura pro uložení křižovatky s příkladem konkrétních dat

id	"27605198"			
lat	"49.217713"			
lon	"16.5692978"			
ways_id	"22726558"	"43587807"	"170244863"	"172810281"
ways_name	"Vrázova"	"Colova"	"Strmá"	"Vrázova"

Pokud je nalezen uzel, ve kterém se setkává více cest, nemusí to ještě nutně znamenat, že se zde setkává více pozemních komunikací. Jak již bylo vysvětleno, cesta v datovém modelu OpenStreetMap definuje kromě pozemních komunikací také chodníky, cyklostezky a další objekty z reálného světa a právě tyto objekty se mohou také křížit s pozemními komunikacemi. V takovém případě se ovšem nejedná o křižovatku takovou, jakou hledáme. Pro účely této práce je třeba detekovat pouze křížení pozemních komunikací, a proto bylo nezbytné do programu přidat kontrolu validity nalezeného křížení.

5.2.3 Kontrola validity nalezeného křížení cest

Existuje několik výjimek nalezeného křížení cest. Tyto výjimky je třeba rozpoznat a nebrat je v potaz jako nalezenou křižovatku. První takový případ je vyobrazen na obr. 5.1.



obr. 5.1: Příklad nevalidního křížení cest způsobeného chodníkem

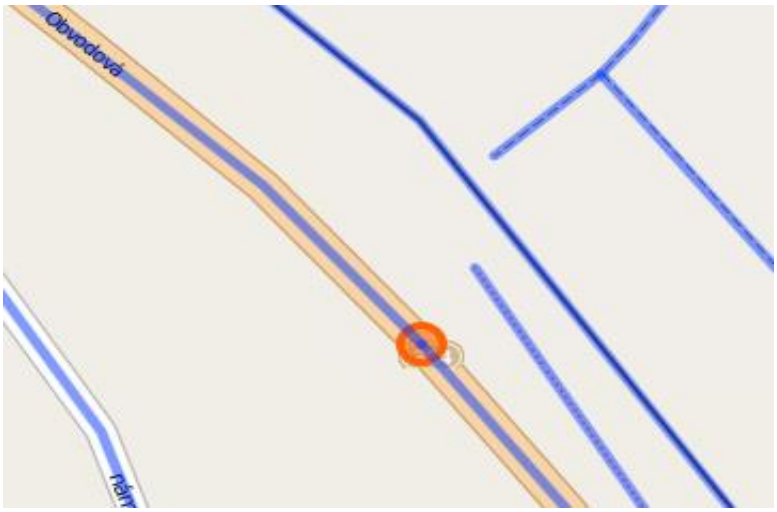
V zobrazené situaci se protíná pozemní komunikace vedoucí po ulici Gabriely Preissové s chodníkem vedoucím od obytných domů. Problém je však v tom, že daný chodník byl z datového souboru již odstraněn při fázi odfiltrování dat nepotřebných pro simulaci dopravy a tudíž se zde již dále nenachází. Zkusíme-li si daný chodník z obrázku odmyslet, zůstane nám pouze uzel, ve kterém se spojují dvě cesty: část pozemní komunikace před uzlem a část pozemní komunikace za uzlem. Vzhledem k tomu bude daný uzel označen funkcí pro detekci křižovatek jako křížení cest, avšak o křižovatku se v tomto případě rozhodně nejedná.

Při řešení daného problému byl zjištěn způsob, jakým lze zde vyobrazenou situaci rozpoznat. Za tímto účelem byla implementována funkce `intersection_is_ok()`, které je předáno každé nalezené křížení cest. Funkce `intersection_is_ok()` nejprve zkontroluje, jestli se jedná o křížení právě dvou cest a jestli obě cesty mají shodná jména. Pokud ano, je dále kontrolováno, zdali

uzel křížení není posledním uzlem na jedné cestě a zároveň prvním uzlem na druhé cestě, případně naopak. Pokud jsou všechny podmínky splněny, logicky to znamená, že se nejedná o křižovatku, nýbrž pouze o uzel, ve kterém se napojuje jedna pozemní komunikace na druhou. V takovém případě vrací funkce `intersection_is_ok()` nepravdivou návratovou hodnotu, křížení cest se neuloží mezi nalezené křižovatky a program pokračuje hledáním dalšího křížení cest. Ve struktuře XML kódu je tato situace rozeznatelná z umístění uzlu křížení na daných cestách, což lze vidět na následující ukázce.

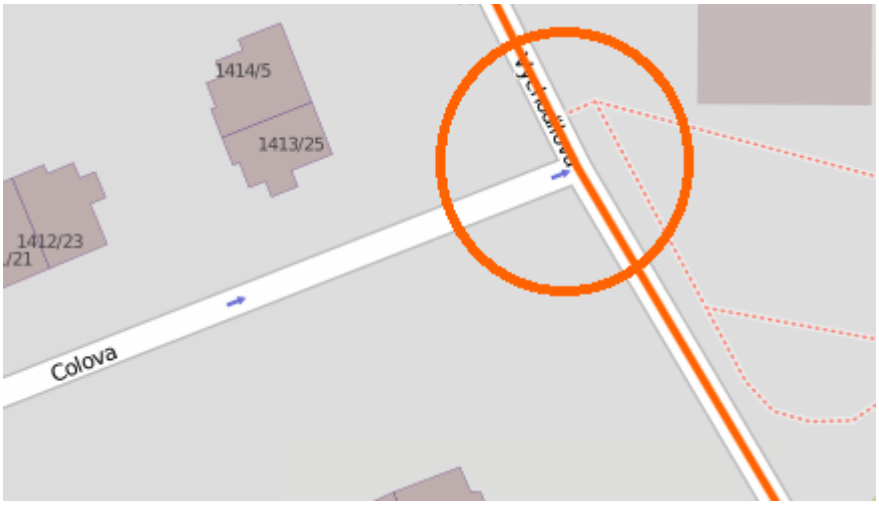
```
<!-- Uzel křížení -->
<node id="550472230" lat="49.2200187" lon="16.5739789"/>
<way id="144174604" version="1">
      <nd ref="550472230"/> <!-- První uzel na dané cestě -->
      <nd ref="550472229"/>
      <nd ref="289580195"/>
      <tag k="highway" v="residential"/>
      <tag k="name" v="Gabriely Preissové"/>
</way>
<way id="109698241" version="3">
      <nd ref="289580193"/>
      <nd ref="654449219"/>
      <nd ref="654203793"/>
      <nd ref="452700921"/>
      <nd ref="289580194"/>
      <nd ref="550472230"/> <!-- Poslední uzel na dané cestě -->
      <tag k="highway" v="residential"/>
      <tag k="name" v="Gabriely Preissové"/>
</way>
```

Další případ nevalidního křížení cest můžeme vidět na obr. 5.2. Jedná se o velice podobnou situaci jako v prvním případě, avšak s tím rozdílem, že v této situaci není problém způsoben odstraněným chodníkem. Problém je zde, na rozdíl od předešlé situace, způsoben tím, že v označeném bodě daná cesta končí a zároveň v tom samém bodě začíná cesta nová se shodnými vlastnostmi i názvem, avšak s rozdílným unikátním identifikačním číslem. Tedy místo toho, aby označený uzel byl jen jedním z mnoha uzlů uvnitř cesty definující její tvar, stává se z něj opět uzel, ve kterém se napojuje jedna pozemní komunikace na druhou. Ale jak je patrné, také se zde nejedná o křižovatku, ale pouze o chybu v mapových datech. Řešení tohoto problému je stejné jako v prvním případě a funkce `intersection_is_ok()` takto nevalidní křížení také rozpozná.



obr. 5.2: Příklad nevalidního křížení cest způsobeného chybou v mapových datech

Zde se nabízí otázka, proč nevyřešit všechny předchozí problémy tak, že by program vyhledával pouze křížení tří a více cest? Nenastaly by tak problémy s uzly, ve kterých se napojují pouze dvě cesty a tudíž se nejedná o křižovatku. Odpověď na tuto otázku je objasněna v následujícím příkladu na obr. 5.3.



obr. 5.3: Příklad křížení cest s průběžnou pozemní komunikací

Ve zde zobrazené situaci není ulice Vychodilova (označena oranžovou čarou) rozdělena na úsek před křižovatkou a úsek za křižovatkou, jako je tomu u většiny křižovatek, nýbrž je průběžná a ulice Colova se na ni napojuje. Co to znamená z hlediska mapových dat? Neznamená to nic jiného, než že se v tomto uzlu nachází křížení pouze dvou pozemních komunikací a přesto se jedná o křižovatku. Z toho důvodu nemůže být program pro detekci křižovatek vytvořen tak, aby vyhledával pouze křížení tří a více cest.

Ve většině nalezených křižovatek jsou setkávající se pozemní komunikace správně rozděleny na úsek před bodem křížení a úsek za bodem křížení. Najde se však i množství křižovatek podobných jako křižovatka popsaná v předcházejícím příkladu, a to především na velkých mapových oblastech. Tato skutečnost je zapříčiněna způsobem, jakým byly tyto pozemní komunikace do mapy přidány, tedy lidským faktorem. Pro účely editace křižovatek je však nutné, aby byly všechny pozemní komunikace rozděleny na úsek před nalezenou křižovatkou a úsek za nalezenou křižovatkou. V editoru křižovatek se totiž edituje každé rameno křižovatky zvlášť.

Za tímto účelem byla vytvořena funkce `extend_intersection()`, které je předána ke kontrole každá nalezená křižovatka. Pokud funkce zjistí, že se jedná o křižovatku s průběžnou pozemní komunikací, rozdělí tuto pozemní komunikaci na dvě části. Do křižovatky je přidáno nové rameno se stejným jménem a unikátním identifikačním číslem, jako měla průběžná pozemní komunikace. Způsob, jakým jsou křižovatky s průběžnou pozemní komunikací rozpoznány, opět vychází z uspořádání uzlů v XML struktuře dané pozemní komunikace. Jak lze vidět v následující ukázce, pokud se uzel křížení nachází na dané cestě kdekoliv mezi jejími krajními body, tato cesta v daném uzlu nezačíná, ani nekončí, a tudíž se jedná o křižovatku s průběžnou pozemní komunikací. Příklad korektní cesty tvořící jedno rameno křižovatky lze taktéž vidět v následující ukázce XML kódu.

```
<!-- Uzel křížení -->
<node id="27605199" lat="49.2187269" lon="16.5732757" version="2">
<way id="8423633" version="8">
      <nd ref="21639259"/>
      <nd ref="27605199"/> <!-- Uzel křížení na průběžné cestě -->
      <nd ref="1577468352"/>
      <nd ref="570085507"/>
      <tag k="highway" v="residential"/>
      <tag k="name" v="Vychodilova"/>
</way>
<way id="43587807" version="3">
      <nd ref="1836938984"/>
      <nd ref="570088284"/>
      <nd ref="27605199"/> <!-- Uzel křížení na korektní cestě -->
      <tag k="highway" v="residential"/>
</way>
```

5.2.4 Uložení nalezených křižovatek

Uložení nalezených křižovatek do výstupního XML souboru zajišťuje funkce `append_intersection()`. Veškeré křižovatky jsou ukládány do již existující proměnné datového typu `xml_document<>`. Do této proměnné jsou nejprve uložena mapová data týkající se simulace dopravy a poté je vytvořen element `intersections` pro uložení jednotlivých křižovatek. V těle funkce `append_intersection()` je dále obsažena iterace nad proměnnou datového typu vector, v níž jsou uloženy informace o všech nalezených křižovatkách. Každá křižovatka je poté přidána do elementu `intersections` v podobě, která odpovídá návrhu struktury výstupního XML souboru pro uložení křižovatek. Struktura uložené křižovatky je již tedy nachytána tak, aby do ní mohly být uloženy podrobné informace za pomoci editoru křižovatek. Konkrétní případ výstupu lze vidět v následujícím příkladu.

```
<intersection id="21289272" lat="49.1982198" lon="16.6139353">
      <ways>
            <way id="84802829" name="Jezuitská"/>
            <way id="81147895" name="Koliště"/>
            <way id="4844437" name="Bratislavská"/>
            <way id="239879489" name="Koliště"/>
      </ways>
</intersection>
```

5.2.5 Nastavení detektoru křižovatek

Program lze nastavit tak, aby vyhledával křižovatky jen na určitých typech pozemních komunikací. Chceme-li totiž simulovat například dopravní provoz na největších pozemních komunikacích města a jejich křižovatkách, je zbytečné, aby se do výstupního XML souboru vkládaly údaje o křižovatkách nacházejících se na parkovištích v obytných oblastech města. Nastavení programu probíhá za pomoci jeho vstupních parametrů.

Prvním vstupním parametrem je soubor s mapovými daty. Pomocí druhého vstupního parametru lze nastavit nejnižší úroveň pozemní komunikace, na které chceme křižovatky vyhledat. Každá úroveň automaticky zahrnuje všechny vyšší úrovně. Pokud budeme chtít vyhledávat například křižovatky na pozemních komunikacích první a druhé třídy, zadáme přepínač *secondary*. Přípustné hodnoty přepínače jsou shodné s hodnotami značek pozemních komunikací z datového souboru OpenStreetMap viz. 3.3. Při zadání přepínače *traffic_signals* se budou vyhledávat pouze světelné křižovatky. Tento přepínač je možné zadat buďto samostatně, anebo v kombinaci s předchozím přepínačem. Pokud není zadán přepínač žádný, vyhledávají se všechny křižovatky v dané oblasti. S nastavením detektoru křižovatek přichází uživatel do styku jen prostřednictvím uživatelského rozhraní v editoru křižovatek, přípustné vstupní parametry jsou zde popsány jen pro vysvětlení způsobu komunikace mezi editorem a detektorem.

6 Editor křižovatek

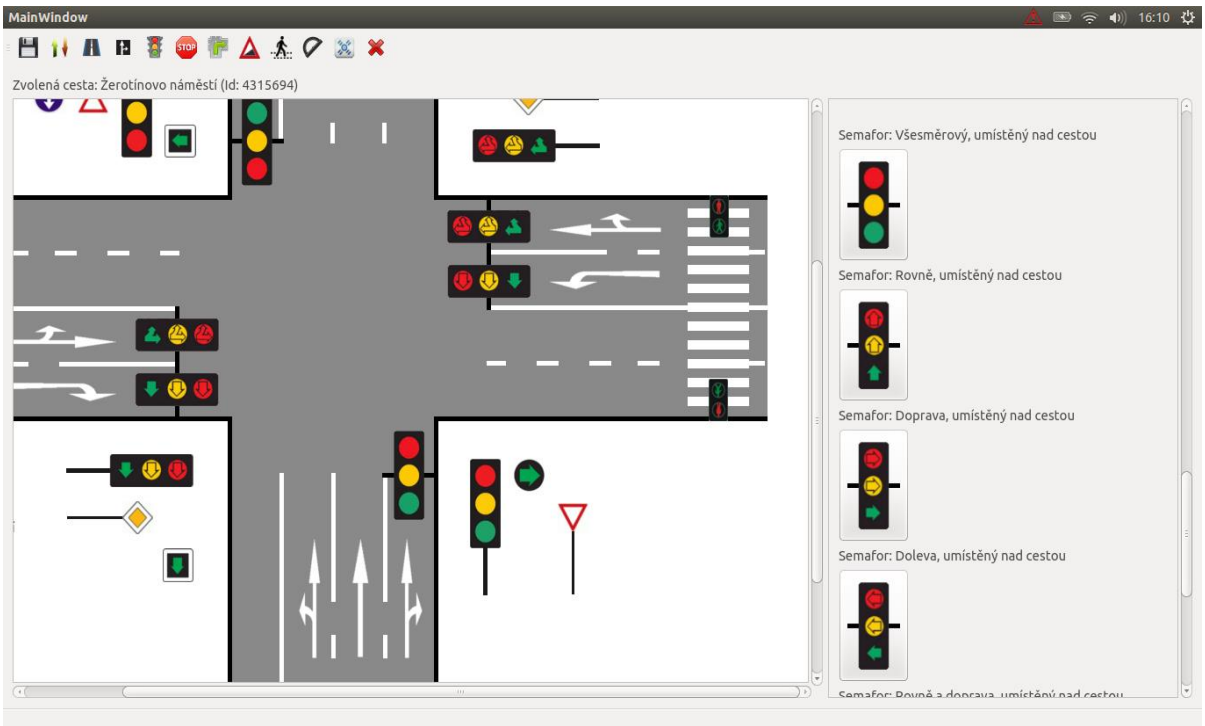
Samotný editor křižovatek pro simulátor dopravy obsahuje grafické uživatelské rozhraní umožňující velmi jednoduše editovat křižovatky tak, aby co nejpřesněji odpovídaly skutečnosti. Editor je implementován v programovacím jazyce C++ s využitím knihovny Qt, která je zaměřena právě na vytváření uživatelských rozhraní. Mezi hlavní výhody knihovny Qt patří multiplatformnost, vývojové prostředí Qt Creator a velmi přehledně a komplexně zpracovaná dokumentace. Hlavní předností této dokumentace je jistě její přímá integrace do vývojového prostředí Qt Creator a s tím související možnost zobrazení nápovědy při tvorbě zdrojového kódu.

6.1 Použité knihovny

Kromě knihovny Qt a standardních knihoven jazyka C++ je v editoru použita knihovna RapidXml viz. 5.1.1. Tato knihovna je využívána pro načítání XML souboru od detektoru křižovatek a také pro ukládání naeditovaných podrobností o křižovatkách do XML struktury dané křižovatky.

6.2 Implementace editoru křižovatek

V této kapitole jsou popsány jednotlivé kroky celkové funkčnosti editoru křižovatek a jejich implementace. Dále jsou zde popsány jednotlivé prvky, kterými je aplikace tvořena. Na obr. 6.1 lze vidět ukázku samotné aplikace, aby bylo možné si tyto prvky lépe představit. Pokud není uvedeno jinak, veškeré metody jmenované v této kapitole jsou metody třídy `MainWindow`, tedy hlavního okna aplikace.



obr. 6.1: Ukázka editoru křižovatek

6.2.1 Hlavní prvky

V horní části hlavního okna se nachází objekt třídy `QToolBar`. Tento objekt slouží jako menu pro výběr editačních nástrojů. Za pomoci signálů jsou jednotlivá tlačítka v nástrojovém menu propojena s příslušnými sloty, v nichž je poté implementována funkčnost každého editačního nástroje.

V pravé části hlavního okna se nachází objekt třídy `QScrollArea`, jenž slouží k zobrazení nabídky možností editace. Tato nabídka odpovídá zvolenému editačnímu nástroji, na obr. 6.1 je zobrazena nabídka editace semaforů.

Jedním z nejdůležitějších prvků hlavního okna editoru křižovatek je objekt třídy `QGraphicsView` sloužící k zobrazení grafické scény. Tento prvek totiž tvoří grafický zobrazovač právě editované křižovatky. Grafická scéna je utvářena pomocí objektů třídy `QGraphicsPixmapItem`, které jsou umístěny do jejího souřadnicového systému. Těmto objektům je na základě akce uživatele nastavena pixelová mapa dle předem vytvořených grafických prvků, a tím je určen jejich tvar a vzhled. Tímto způsobem jsou v grafické scéně zobrazeny jednotlivé pozemní komunikace, semafory, dopravní značky, ale také přechody pro chodce a směrové šipky jízdních pruhů. Velmi důležité je, aby na sebe jednotlivé grafické objekty neustále navazovaly a tvořily tak celistvou strukturu křižovatky a jejich okolních objektů. Tuto funkčnost zajišťují metody pro změnu pozice objektů.

6.2.2 Změna pozice grafických objektů

Představme si situaci, kdy například přidáme do jedné z pozemních komunikací jízdní pruh. Změní se velikost dané pozemní komunikace, a proto je třeba také změnit velikost středového prvku křižovatky. Pokud by se tak nestalo, pozemní komunikace by byla širší než středový prvek a společně by tak netvořili kompaktní celek.

Za tímto účelem byla implementována metoda `update_mid()`, která je zavolána při každé změně struktury křižovatky, způsobené akcí uživatele. Tato metoda nastavuje velikost středového prvku podle pozemní komunikace s největším počtem jízdních pruhů v křižovatce. Pokud se změní velikost středového prvku, je třeba také změnit pozici všech ostatních fyzických objektů v křižovatce, jako jsou semafory, dopravní značky, směrové šipky a další. Také je nutné posunout pozemní komunikace dále od středu souřadného systému. Tuto činnost taktéž zajišťuje metoda `update_mid()`. Podle aktuální velikosti středového prvku jsou upraveny souřadnice všech ostatních grafických objektů v křižovatce tak, aby křižovatka z vizuálního hlediska opět tvořila kompaktní celek.

Dále je nutné, aby se změnou rozměrů křižovatky byly změněny také souřadnice jednotlivých jízdních pruhů a semaforů umístěných nad pozemní komunikací. Jelikož tyto činnosti nejsou zcela triviální, jsou umístěny do samostatných metod `update_edges()` a `update_over_sems()`, které jsou taktéž volány při změnách struktury křižovatky.

6.2.3 Popis činností po spuštění aplikace

Po spuštění aplikace je zavolána metoda `application_begin()`, v rámci které je uživatel dotázán, zdali chce zahájit nové hledání křižovatek z datového souboru OpenStreetMap, nebo zdali chce pokračovat v editaci již existujících křižovatek. Po zvolení jedné z možností je za pomoci slotu `button_begin_clicked()` zobrazena výzva k zadání vstupního souboru. Tato výzva je implementována pomocí standardních dialogových oken knihovny Qt, konkrétně voláním metody `getOpenFileName()` třídy `QFileDialog`. V případě, že uživatel zvolí první možnost, je vyzván, aby zadal vstupní soubor s příponou .osm, tedy datový soubor stažený z databáze

OpenStreetMap. Poté je zavolána metoda `finding_options()`, v rámci které je zobrazena výzva ke zvolení parametrů vyhledávání křižovatek a po jejich zadání je zavolán program k detekci křižovatek s příslušnými vstupními parametry. Když jsou křižovatky nalezeny, zavolá se metoda `begin_intersection()`, která zobrazí výzvu k výběru konkrétní křižovatky. V případě, že uživatel po startu aplikace zvolí druhou možnost, tedy editovat již existující křižovatky, je jako vstupní soubor očekáván ten, který byl výstupem předcházející editace, tedy soubor s příponou .xml. Po jeho zadání je rovnou zavolána metoda `begin_intersection()`. Po zvolení konkrétní křižovatky je zavolána metoda `start_vizualization()`, v rámci které jsou inicializovány grafické objekty, následně jsou vloženy do grafické scény a ta je poté zobrazena.

Dále je v rámci této metody zavolána metoda load_from_xml(), která ve vstupním XML souboru zkontroluje, zdali zvolené křižovatka nebyla již dříve editována. Pakliže je struktura křižovatky v XML souboru již uložena, jsou na jejím základě zavolány příslušné metody pro přidávání jednotlivých objektů do křižovatky a ta je tak vizualizována ve stejném stavu, v jakém byla při předcházející editaci uložena. Pokud struktura křižovatky ještě není v XML souboru uložena, znamená to, že se jedná o její první editaci a tudíž je vizualizována jen základní prázdná křižovatka s jedním vstupním a jedním výstupním jízdním pruhem na každém rameni.

6.2.4 Nabídky editačních nástrojů

Po výběru některého z editačních nástrojů je zobrazena příslušná nabídka obsahující možnosti úpravy křižovatky daným nástrojem. Příkladem může být nabídka pro změnu počtu vstupních a výstupních jízdních pruhů na označené pozemní komunikaci, viz. obr. 6.2. Pokud uživatel zvolí nástroj pro úpravu počtu jízdních pruhů, je zavolána metoda `show_edit_lines()`, která zobrazí příslušnou nabídku. V rámci této nabídky jsou zobrazeny objekty třídy `QLabel` pro zobrazení textových popisků a dále objekty třídy `QPushButton`, jejichž signály `clicked()` jsou propojeny s příslušnými sloty. Pokud uživatel zvolí typ pozemní komunikace, je vyvolán slot na základě stisknutého tlačítka. Každé tlačítko má takto definovánu svou vlastní funkčnost. Po kliknutí na tlačítko je příslušné pozemní komunikaci nastaven požadovaný počet jízdních pruhů, na základě toho je správně posunuta v souřadnicovém systému grafické scény a jsou zavolány metody pro změny pozic ostatních grafických objektů v křižovatce.

obr. 6.2: Ukázka nabídky pro změnu počtu jízdních pruhů

6.2.5 Označení grafických objektů

Označení jednotlivých pozemních komunikací a jízdních pruhů v grafické scéně se provádí dvojklikem. K tomuto účelu je použita metoda `mouseDoubleClickEvent()` třídy `QWidget`. Protože však původní chování metody neumožňovalo předání ukazatele na označený grafický objekt do hlavního okna aplikace, bylo chování této metody reimplementováno. Metoda `mouseDoubleClickEvent()` vygeneruje signál `clicked()`, v rámci kterého je předán ukazatel na označený objekt v grafické scéně. Signál `clicked()` je poté propojen se slotem `click_accepted()` v objektu hlavního okna aplikace, kde je následně změněn aktuální aktivní objekt grafické scény.

6.2.6 Uložení naeditované křižovatky

Uložení křižovatky do výstupního XML souboru zajišťuje metoda `append_to_xml()`. V průběhu editace je aktuální struktura křižovatky zaznamenávána do proměnných datového typu vector pro každý výstupní XML element zvlášť. Ve chvíli, kdy se uživatel rozhodne křižovatku uložit, je zavolána metoda `append_to_xml()`, která na základě aktuálního stavu těchto proměnných převede křižovatku do výstupní XML struktury a celou naráz ji uloží do výstupního souboru.

7 Závěr

V rámci této práce byl vytvořen editor dopravních křižovatek v programovacím jazyce C++ s využitím knihovny Qt. Dále byl v programovacím jazyce C++ implementován program sloužící k detekci křižovatek z mapových podkladů stažených ze serveru OpenStreetMap.

Prezentovaná práce se z velké části zabývá strukturou geografických dat z databáze serveru OpenStreetMap, neboť právě analýza těchto dat byla klíčová při tvorbě programu pro detekci křižovatek. Jako stěžejní bod této práce lze označit nalezení způsobu, jakým se dají jednotlivé křižovatky z geografických dat detekovat. Následně bylo třeba zjistit, jakým způsobem rozpoznat nevalidní křížení cest, které netvoří dopravní křižovatky. Program se podařilo úspěšně implementovat a poté ho využít pro nalezení křižovatek v libovolné oblasti města Brna.

Při tvorbě editoru křižovatek bylo nejsložitější nalézt způsob, jakým vizualizovat právě vytvářenou křižovatku. K tomuto účelu byla zvolena třída `QGraphics` knihovny Qt, umožňující tvorbu grafické scény za pomoci jednotlivých grafických objektů. Prvek sloužící k zobrazení křižovatek se podařilo úspěšně implementovat a integrovat do prostředí editoru.

Program pro detekci křižovatek tvoří spolu s editorem křižovatek komplexní nástroj umožňující editovat nalezené křižovatky podle skutečné předlohy. Nástroj byl otestován na mapové oblasti města Brna a jeho křižovatkách, ale vzhledem k jednotnému modelu geografických dat ze serveru OpenStreetMap by jeho použití bylo možné i pro jiné oblasti. V rámci této práce byla vybrána skupina reálných křižovatek, které byly naeditovány podle skutečnosti a uloženy do přiloženého souboru. Jedná se o křižovatky nacházející se na prvním brněnském okruhu.

V práci by bylo možné dále pokračovat zdokonalením způsobu výběru křižovatek, které si uživatel přeje editovat. Křižovatka by se mohla volit například ze zobrazené mapy. Dále by se dal zdokonalit systém editace tramvajových kolejí, zejména jejich vzájemného propojení uvnitř křižovatky. K docílení realističtějšího vzhledu by bylo možné posunout přechody pro chodce blíže hranici křižovatky. Možným budoucím zdokonalením této práce by také mohlo být rozšíření editoru křižovatek o možnost editovat víceramenné křižovatky a kruhové objezdy.

Bibliografie

[1] WIKIPEDIA, *Traffic simulation* [online]. Poslední změna 28. prosince 2013 [cit. 2. ledna 2014]. Dostupné na: <http://en.wikipedia.org/wiki/Traffic_simulation>.

[2] PURSULA, M. Simulation of Traffic Systems – An Overview. *Journal of Geographic Information and Decision Analysis*, 1999, č. 3. S. 1-8.

[3] KŘIVDA, V. a ŠKVAIN, V. *Modelování a simulace dopravního proudu* [online]. Poslední změna 2013 [cit. 3. ledna 2014]. Dostupné na: <http://kds.vsb.cz/mkk/modelovani-05.htm>.

[4] BARCELÓ, J. *Fundamentals of Traffic Simulation*. 1. vyd. New York: Springer New York, 2010. ISBN 978-1-4419-6141-9.

[5] KŘIVDA, V. a ŠKVAIN, V. *Mikroskopické simulační modely* [online]. Poslední změna 2013 [cit. 4. ledna 2014]. Dostupné na: <http://kds.vsb.cz/mkk/modelovani-08.htm>.

[6] WIKI OPENSTREETMAP, *About* [online]. Poslední změna 2014 [cit. 10. ledna 2014]. Dostupné na: <http://wiki.openstreetmap.org/wiki/About>.

[7] COAST, S. How OpenStreetMap Is Changing the World. *Web and Wireless Geographical Information Systems*, 2011, č. 1. S. 4.

[8] GRAF, F., KRIEGEL, H.-P., RENZ, M. et al. MARiO: Multi-Attribute Routing in Open Street Map. *Web and Wireless Geographical Information Systems*, 2011, č. 1. S. 486-490.

[9] WIKIPEDIA, *OpenStreetMap* [online]. Poslední změna 10. ledna 2014 [cit. 11. ledna 2014]. Dostupné na: < http://cs.wikipedia.org/wiki/OpenStreetMap>.

[10] BÁRTA, D. *Projekt OpenStreetMap z pohledu geoinformatika* [online]. Poslední změna 2014 [cit. 11. ledna 2014]. Dostupné na: <http://geoinformatics.fsv.cvut.cz/gwiki/Projekt_OpenStreetMap_z_pohledu_geoinformatika>.

[11] WIKI OPENSTREETMAP, *Elements* [online]. Poslední změna 2014 [cit. 12. ledna 2014]. Dostupné na: <http://wiki.openstreetmap.org/wiki/Elements>.

[12] WIKI OPENSTREETMAP, *Map Features* [online]. Poslední změna 2014 [cit. 12. ledna 2014]. Dostupné na: <http://wiki.openstreetmap.org/wiki/Map_Features>.

[13] GROSSMANN, M. *Preference pražských tramvají - technika* [online]. Poslední změna 2013 [cit. 18. ledna 2014]. Dostupné na: <http://preference.prazsketramvaje.cz/showpage.php?name=technika>.

[14] WIKI OPENSTREETMAP. Osmfilter [online]. [cit. 4. února 2014]. Dostupné na: <http://wiki.openstreetmap.org/wiki/Osmfilter>.

[15] WEBOVÁ STRÁNKA RAPIDXML. RapidXML [online]. [cit. 14. března 2014]. Dostupné na: <http://rapidxml.sourceforge.net>.

[16] WEBOVÁ STRÁNKA OPENSTREETMAP. OpenStreetMap [online]. [cit. 20. dubna 2014]. Dostupné na: <http://www.openstreetmap.org/export>.

[17] WEBOVÁ STRÁNKA OVERPASS API. Overpass API [online]. [cit. 20. dubna 2014]. Dostupné na: <http://overpass.osm.rambler.ru/>.

Seznam příloh

Příloha 1: CD

Obsah CD:

- src/ - zdrojové kódy
- readme.txt – návod k instalaci
- xmalan01_bp.docx – upravitelná verze písemné práce
- xmalan01_bp.pdf – elektronická verze písemné práce