Editor silničního grafu v systému VRUT

Vojtěch Kolínský

May 21, 2020

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

Vedoucí práce: Ing. Jiří Bittner Ph.D

Poděkování

Děkuji panu Ing.Jiřímu Bittnerovi, Ph.D. za pomoc při vypracování této práce a za velikou trpělivost, kterou se mnou měl.

Čestné prohlášení

Čestně prohlašuji, že jsem na této práci pracoval samostatně. Zároveň prohlašuji, že jsem uvedl veškeré použité zdroje v souladu s Metodickým pokynem o dodržování etických pokynů při přípravě vysokoškolských závěrečných prací.

Abstrakt

Tato práce má za cíl implementovat editor silničního grafu do již existujícího systému VRUT. Diskutuje oddělení simulační a editační vrstvy.
Cílem této práce je navrhnout vhodný modul do modulárního systému VRUT, který vznikl (a stále se rozvíjí) ve spolupráci s Katedrou počítačové grafiky a interakce na Fakultě elektrotechnické ČVUT.
Systém VRUT nebo-li "Virtual Reality Universal Toolkit" dominuje simulační jízdou po silničním grafu, který je nutné dále editovat.
Modul, který by měl vzniknout na konci této práce, by měl přinést do systému VRUT nový, uživatelsky přívětivý editor silničního grafu.

Klíčová slova

Vozidlo, autonomní, editor, silniční graf

Abstract

This work has one objective. That is implementing functional editor of road graph to an existing system called VRUT. It is discussing the separation of the simulating and editing layers.
The goal of this work is to propose a new module into a modular system called VRUT, which has been developed in cooperation with Cathedra of Computer Graphics of Faculty of Electrotechnical Engineering (CTU).
System VRUT (otherwise "Virtual Reality Universal Toolkit") contains simulating ride over the road graph, which must be still edited.
The module that should be created at the end of this work, should bring into system VRUT new, user-friendly editor of road graph.

Key words

Vehicle, autonomous, editor, road graph

Contents

1 Úvod **7**

2 Motivace k výběru tohoto tématu **8**

3 Cíle práce **8**

4 Rozbor řešení **8**

5 Projekt OpenStreeetMap **8**
5.1 Geodata 9
5.1.1 Kompozice dat 9
5.1.2 Typy tvorby dat 10
5.1.3 Způsob uložení dat 10
5.2 Editory grafu v systému OSM 11
5.2.1 iD editor 11
5.2.2 Potlach 2 12
5.2.3 Merkaator 13
5.2.4 JOSM 13

6 Implementace v systému VRUT **14**
6.1 Stručný popis systému VRUT 14
6.2 Graf silnic 16
6.2.1 Popis grafu silnic 16
6.2.2 Implementace grafu silnic 18
6.3 Editor grafu v systému VRUT 20
6.3.1 Dosavadní řešení 20
6.3.2 Modul Traffic a editor grafu 20

7 Popis vlastní implementace **21**
7.1 Vytvoření modulu *RoadGraphEditor* 21
7.1.1 Získání přístupu k silničnímu grafu 21
7.1.2 Zobrazení grafu silnice do scény 22
7.1.3 Bezpečnostní opatření modulu 22
7.2 Označení uzlu 23
7.3 Odznačení uzlu 23
7.4 Úprava atributů uzlu 24
7.4.1 Spuštění editačních módů 24
7.4.2 Editačních mód- "Node mode" 25
7.5 Vymazání uzlu 25

7.6 Označení hrany 26
7.7 Odznačení hrany 26
7.8 Vymazání hrany 27
7.8.1 Editačních mód- "Edge mode" 27
7.8.2 Funkce Undo 27
7.8.3 Funkce SaveTheGraphFile() 28

8 Rozdíly oproti návrhu řešení a návrh úprav stávající implementace 28
8.1 Hlavní rozdíly oproti návrhu řešení 28
8.2 Návrh úprav stávající implementace 28

9 Přínos modulu RoadGraphEditor 29
9.1 Původní možnost editace 29
9.2 Současná možnost editace 29
9.3 Test na vytvoření kompletně nového silničního grafu 37

10 Závěr 39

1 Úvod

V dnešní době již nejsou autonomní vozidla pouze diskutovanou tématikou, nýbrž stává se z nich téma, které je již nyní testováno v praxi, jak tvrdí článek z roku 2018 [5]. Důvodem pro toto testování je dle slov pana profesora Jiřího Matase z ČVUT fakt, že autonomní vozidla se budou na silnicích objevovat již v příštích deseti letech. Co se týče samotné technologie pro funkční jízdu těchto vozidel, profesor Jiří Matas věří, že již není třeba vice inovovat. To znamená, že veškeré technologie, které budou v této oblasti využívány, již existují. Jediné, co je potřeba, je tyto technologie rozšířit o velké množství dat. Mezi tato data patří mimo jiné informace o cestě, jako je vedení cesty a silnice, ale i záznam veškerých překážek (domy, stromy, lavičky či krajnice). Dále bude třeba, aby byla auta schopna reagovat na neočekávané situace, např. na cyklistu či jiné vozidlo, na chodce, ale i na poletující předměty. Dle slov profesora Matase bude vstup autonomních vozidel na dnešní silnice postupný, tzn. že nejdříve budou pravděpodobně jezdit pouze nákladní auta po fixních trasách. Každopádně autonomní vozidla přinesou do moderní doby spousty benefitů, jmenovitě např. vyšší bezpečnost a plynulost provozu.

I přes to, že již máme většinu potřebných technologií, autonomní jízda stále musí být testována. Důvod je ten, aby funkčnost technologií po vstupu na silnice byla co možná nejvyšší. K tomu mohou sloužit reálné situace, různé testovací tratě či jízdní simulátory. Nejlepší by samozřejmě bylo testovat vozidla v reálných situacích, ale to by bylo příliš nákladné a nevypočitatelné (není např. možné si určit, kdy budeme testovat jízdu v dešti, poněvadž to se nedá příliš naplánovat). Testovací jízdy jsou již izolovanější na určitý druh testů, stále tu však narážíme na jistá omezení (viz. příklad s počasím). Proto může být za jistých podmínek nejvýhodnější použít jízdní simulátor. Jízdní simulátor je v podstatě nástroj, s jehož pomocí můžeme řídit vozidlo ve virtuálním prostředí. V takovémto prostředí lze simulovat různé silniční situace včetně běžné jízdy či riskantního předjíždění. Celý systém silnic je reprezentován grafem a to je ta část, které se věnuje tato bakalářská práce. Z předem uvedených důvodů vyplývá, že pro dostatečné testování potřebujeme velké množství testovacích prostředí. Tato prostředí někdo musí naprogramovat a vytvořit. V této práci budu vytvářet editor silničního grafu pro tato testovací prostředí. Důvod k existenci lepšího editoru grafu je hlavně rychlejší a uživatelsky přívětivější tvorba dalších silnic.

2 Motivace k výběru tohoto tématu

Hlavním důvodem pro výběr tohoto tématu bakalářské práce je jednoznačně fakt, že se rád podílím na vývoji inovativních technologií. Autonomní vozidla (jak již bylo zmíněno) jsou velmi atraktivní pro budoucnost. Věřím, že až těmito auty budeme běžně cestovat, bude mě hřát u srdce, že jsem alespoň částečně přispěl k tomuto posunu v lidské historii. Díky tomu, že Katedra počítačové grafiky a interakce úzce spolupracuje s firmou Škoda Auto právě na tomto projektu pomocí systému VRUT (Virtual Reality Universal Toolkit), budu dokonce vědět, kterou značku aut jsem pomohl takto rozšířit.
Navíc sám osobně někdy zažívám ten pocit, kdy práce s jistým softwarem je mnohem náročnější z důvodu špatného uživatelského rozhraní. Rád bych zajistil, aby se od teď se silničním grafem v systému VRUT pracovalo jednodušeji než kdy dříve.

3 Cíle práce

Hlavním cílem této práce je naimplementovat funkční editor silničního grafu do systému VRUT. K tomu je zapotřebí nejdříve pochopit a zkompilovat systém VRUT. To v sobě také zahrnuje studium materiálů z předešlých bakalářských prací až po studium projektu OpenStreetMap.

4 Rozbor řešení

U editoru grafu silnic je hlavní, aby byl dostatečně jednoduchý pro snadné pochopení toho, jak funguje. To je důležité hlavně proto, že chceme, aby byl téměř každý schopen rychle editovat daný graf. Do toho spadá hlavně intuitivnost a uživatelská přívětivost ovládání editoru. Troufám si tvrdit, že zářný příklad takovéhoto editoru se nachází v již zmíněném projektu OpenStreetMap. V následujících kapitolách budu vysvětlovat, proč jsem došel k tomuto názoru. Tento popis je zčásti založen na vlastním zkoumání projektu a zčásti na diplomové práci Bc. Chrudoše Vorlíčka [3].

5 Projekt OpenStreeetMap

Tento projekt funguje na podobném principu jako Wikipedia.org. Byl založen v roce 2004 Stevem Coastem. Dva roky poté vznikla tzv. OpenStreetMap Foundation, což je iniciativa, která tento projekt podporuje.Vzhledem k

tomu, že projekt se stal poměrně rychle užívaným, společnost Yahoo v roce 2006 poskytla své mapové podklady, které zefektivnily práci na mapách. O rok později se k ní přidala společnost Automotive Navigation Data, která poskytla mapy silniční sítě Nizozemska, Indie a Číny. Hlavním cílem projektu je vložit do rukou obyčejných lidí příležitost ke tvorbě geografických dat, která jsou následně vizualizována do podoby topografických map. To znamená, že do tohoto projektu se může zapojit úplně každý, kdo se zadarmo zaregistruje na jejich stránkách. Počet takovýchto registrovaných členů přesáhl 6. ledna 2013 1 milion. To je poměrně velký nárůst vzhledem k tomu, že v srpnu 2008 bylo registrováno 50 000 uživatelů. Toto číslo i nadále roste. Statistika nám říká, že kolem 30% uživatelů nějakým způsobem přispělo k rozšíření topografických map (alespoň tvorbou jedné sady změn)[2].

Projekt dále disponuje možností stáhnout si data tvořící mapu a v rámci licence ODbL s nimi nakládat dle vlastního uvážení. To je asi hlavní výhoda oproti jiným mapovým aplikacím (např. Seznam mapy). Jeden z následků této možnosti je vznik různých mapových aplikací reflektující potřeby uživatelů.

5.1 Geodata

Geodata jako taková vytváří samotní uživatelé, což vysvětluje jejich rozmanitost. Někteří uživatelé využívají pouze základních prvků jako jsou silnice, stromy, atd. Avšak právě díky otevřenosti projektu je možné zde nalézt i neobvyklé prvky (oproti konkurenčním mapám) jako záznamy radarů na silnicích, či bezpečnostních kamer.

5.1.1 Kompozice dat

Mezi základní prvky patří tyto tři:

- uzly (nodes) - body se souřadnicemi a tagy
  Každý uzel obsahuje 2 povinné atributy- zeměpisné souřadnice a identifikátor. Dále mohou obsahovat volitelné atributy (jako třeba výšku). Uzly mohou reprezentovat samostatné prvky- např. budovy. Další rolí uzlů je reprezentace spojnice více cest.

- cesty (ways) - linie a hranice polygonů
  Cesty tvoří minimálně dva uzly, přičemž maximální množství uzlů je omezeno na 2000. Může nastat situace, že počáteční a koncový uzel se shodují. V takovém případě se jedná o cestu uzavřenou. V opačném

případě se jedná o otevřenou cestu. U uzavřené cesty lze pomocí nastavení tagu "area" na hodnotu "yes" označit tento útvar za plošný prvek.

- relace (relations) - vztah mezi jednotlivými prvky

Vlastnosti těchto prvků se mohou zaznamenat do jejich atributů.

5.1.2 Typy tvorby dat

Existují celkem tři obecné způsoby, jakými lze data vytvořit.

- Prvním a zároveň nejpřímějším způsobem je přímé sbírání dat procházením (či projížděním) terénu a zaznamenávání dat pomocí GPS přístroje. Takto sesbíraná data se pak nahrají do databáze ve formátu GPX.
- Druhou cestou k získání dat je odvozování z jichž existujících projektů. Zde je ovšem důležité mít na paměti, aby nedošlo k plagiátorství. To znamená, že licence "inspirovaného" projektu musí být kompatibilní s licencí projektu OpenStreetMap.
- Poslední možností je využít data projektů firem, které udělily oprávnění k jejich použití. Toto oprávnění musí být uděleno výslovně pro projekt OpenStreetMap.

5.1.3 Způsob uložení dat

Veškerá data jsou ukládána v tabulkách (rozdělena podle prvků) a ty jsou uloženy v databázi, což je velmi důležitý prvek projektu. Databáze obsahuje tabulky k výše popsaným třem základním prvkům, ale zároveň i tabulky k dalším prvkům jako změnové sady, GPX soubory a seznam uživatelů. Databáze si zároveň ukládá jednotlivé verze v závislosti na čase, což umožňuje nahlédnutí do historie databáze či jejího zotavení.

Hlavní soubor, v němž jsou uložena data, se nazývá Planet.osm, což je XML soubor. Ten je vytvářen v týdenních intervalech. Tento soubor je obrovský (dosahuje stovek gigabajtů). Proto se dělají i jeho výběry (navíc ne vždy je potřeba mít k zobrazení celý svět). Tyto výběry pokrývají běžně kontinenty, země či pouze města. Vzhledem k jejich menší velikosti jsou tyto soubory vytvářeny častěji (přibližně jednou denně).
Dalším významným souborem je soubor planet.gpx, který obsahuje údaje z nahraných GPX souborů a historií všech provedených změn. Tyto změny jsou ukládány pro každý objekt. Proto je velmi jednoduché vrátit změny do

původního stavu. Tento soubor je o něco větší než soubor Planet.osm.
Data jsou ukládána na serveru s dostatečnou kapacitou. V současné době je pro tento účel používán server Ramoth. Tento systém řídí operační systém Ubuntu 12.04 LTS Server amd64. Kapacita jeho disků je přibližně 15 TB a operační paměť dosáhla hodnoty 256 GB. Jako databázový systém slouží PostgreSQL 9.1.
Databáze jako taková je velmi komplexní a tudíž složitá. Z toho důvodu existují na wiki projektu stránky, které popisují způsob uložení jednotlivých dat. To je důležité například proto, že každý záznam v databázi obsahuje popisná data a každý typ prvku reprezentuje tabulka. Ovšem ne každý záznam obsahuje všechny atributy, proto jsou některé sloupce prázdné, což může nezkušenému vývojáři způsobit zmatek.

5.2 Editory grafu v systému OSM

Je zřejmé, že tento projekt musí obsahovat jednoduchý a intuitivní editor dat. Samozřejmě obsahuje i editor dat, která nejsou v systému VRUT třeba editovat (jako třeba lesy), avšak obsahuje skvělý editor silnic, kterým se budu velmi inspirovat při mé implementaci.

5.2.1 iD editor

Toto je v současnosti nejnovější editor (spuštěn roku 2013). Licence, kterou používá, se nazývá "Do What The Fuck You Want To Publish" (WTFPL). Výhodou této licence je absolutní volnost při vytváření obsahu. Editor iD je javascriptová aplikace, která užívá k vykreslení knihovnu d3js. S tímto editorem se pracuje online přímo na portálu openstreetmap.org. Je vhodný pro začátečníky, poněvadž je velmi jednoduchý a obsahuje pouze pár základních funkcí. Po kliknutí na tlačítko "Upravit pomocí iD(editor v prohlížeči)" či po kliknutí na tlačítko "Upravit" (neboť toto tlačítko defaultně vybere tuto první možnost) se (po přihlášení) překryje mapa další vrstvou, která obsahuje editační prvky. To je velmi důležité, poněvadž to umožňuje editovat mapu za pochodu (tzn. bez nutnosti restartování celého projektu za účelem vizualizace provedených změn). V horní liště uživatelského rozhraní se nachází prvky "Uzel", "Linie" a "Plocha". Mě zajímá hlavně ten uzel. Kliknutím na tento prvek je možné do mapy vkládat různé uzly, což je jedna z hlavních komponent mého editoru. Dále je důležité, aby šlo jednotlivé uzly spojit a tím vytvořit hranu. K tomu slouží postranní panel, který se objeví po kliknutí na libovolný uzel. V této sekci jsou úplně dole tzv. "Relace",

které se používají právě ke spojování uzlů. Ke smazání uzlu či hrany slouží výběr z možností po kliku pravým tlačítkem myši na daný objekt. Dále je možné editovat polohu uzlu. To se dělá velmi jednoduše. Stačí kliknout levým tlačítkem myši na daný uzel, držet tlačítko stisknuté a poté pohybem myši posouvat uzel po mapě.

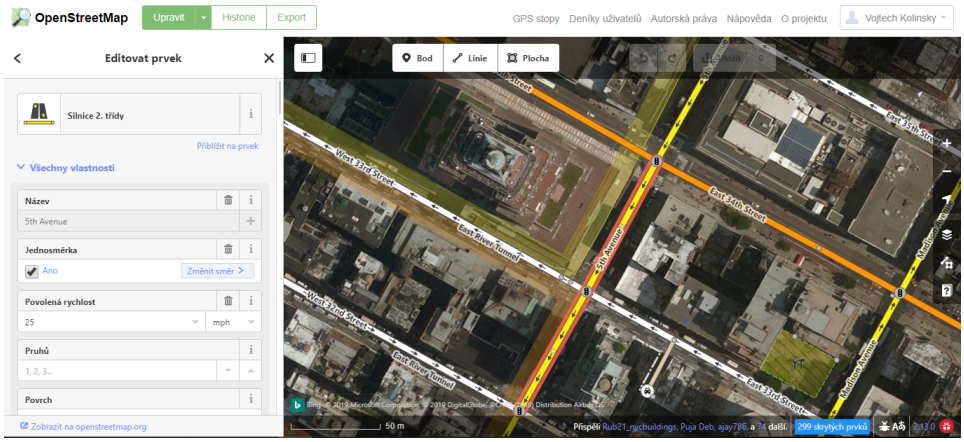

Figure 1: Na obrázku vidíme ukázku editoru iD z projektu OpenStreetMap. Nacházíme se zde právě v editačním módu, proto můžeme pozorovat označenou linii. O ní se nám v levé části objevují různé informace, které je možné editovat.

5.2.2 Potlach 2

Toto je druhý online editor, který nabízí projekt OSM. Také pracuje na bázi licence WTFPL. Ke spuštění tohoto editoru je potřeba mít nainstalovaný a povolený Adobe Flash, neboť právě v něm je editor naimplementovaný. Číslovka 2 v názvu značí, že toto je druhá verze tohoto editoru. Hlavním rozdílem oproti první verzi je přidání zobrazení What You See Is What You Get (WYSIWYG). Dále bylo zjednodušeno tagování a autentizace přes OAth. Avšak vzhledem k existenci editoru iD není tento editor v současnosti dále vyvíjen. Funkčnost tohoto editoru je mírně složitější a komplexnější, proto ji ve své implementaci nebudu využívat.

Figure 2: Na obrázku vidíme ukázku editoru Potlach 2 z projektu OpenStreetMap.

5.2.3 Merkaator

Tento editor je stále ve stádiu vývoje. Je určen pro Unix, Mac Os i Windows a jeho distribuce probíhá pod licencí GNU GPL v2. Obsahuje některé funkce, které nejsou obsaženy v ostatních editorech (např. průhledné zobrazení vrstev či přímé připojení k GPS příjimači). Avšak žádná z těchto "nadřazených" možností nebude v mém editoru přítomna, tudíž není třeba se tímto editorem více zabývat.

5.2.4 JOSM

JOSM je javaskriptová aplikace. Je založena na principu dálkového nahrávání změn do hlavní databáze projektu OSM. K tomu slouží mimo jiné licence General Publish Licence (GPL). Oproti iD editoru obsahuje mnohem více funkcí pro manipulaci s daty. K tomuto editoru lze navíc připojit další rozšíření, čímž množství funkcí rapidně roste. Avšak jeho hlavní výhodou oproti již zmíněným editorům je možnost užívat jej v offline módu.

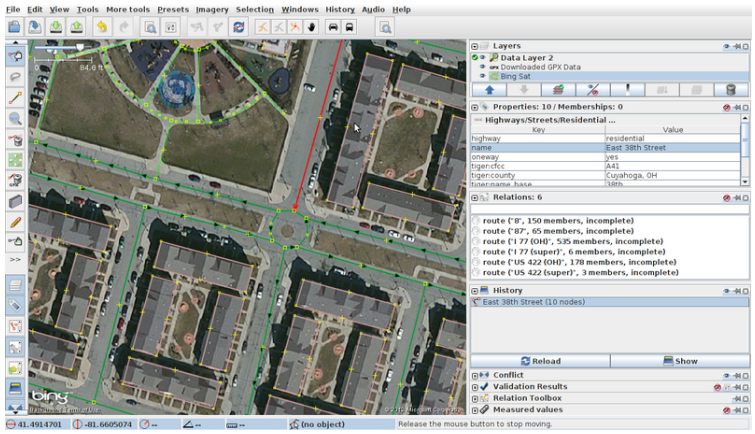

Figure 3: Na obrázku vidíme ukázku editoru JOSM z projektu OpenStreetMap. (Zdroj: OSM Wiki[4])

6 Implementace v systému VRUT

Jak již bylo výše zmíněno, cílem této práce je implementovat editor grafu silnic do již existujícího rozsáhlého projektu. Proto si ve stručnosti pojďme vysvětlit, co to vlastně systém VRUT je a jaká má pravidla, která je třeba při implementaci dodržet. Následující popis systému VRUT je založen na vlastním zkoumání a na diplomové práci BC. Jaroslava Minaříka [1].

6.1 Stručný popis systému VRUT

VRUT (Virtual Reality Universal Toolkit) je především modulární systém. To znamená, že se skládá z jednotlivých modulů, na kterých pracují (či pracovali) různí programátoři (včetně studentů ČVUT). To je jeho obrovská výhoda, poněvadž práce jednotlivých programátorů neovlivní práci těch ostatních v negativním slova smyslu (tedy pokud se nějaký modul nevydaří a nebude fungovat, stačí ho nezařadit do kompilace a vše funguje v pořádku). Z tohoto důvodu je perfektní pro bakalářské práce jako je tato. Komunikace mezi jednotlivými moduly je zajištěna tzv. Event-driven architekturou. To znamená, že se v systému nachází řada událostí, kterými se mezi moduly mohou předávat informace a data.

Figure 4: Ukázka spuštěného systému VRUT. Na obrázku lze spatřit jedno z počítačem řízených vozidel.

Hlavním modulem je tzv. core, což je vlastně jádro celého systému. Řídí registraci modulů, interpretaci příkazů a událostí. Díky tomuto modulu jsou spolu schopny všechny moduly komunikovat, protože tento modul je takovým spojovacím článkem. Komunikace mezi moduly probíhá pomocí událostí (dalo by se říci pomocí přijímačů a vysílačů datových balíčků). Toto komunikační rozhraní je implementováno pomocí knihovny wxWidgets. Pro každý modul je zaznamenána informace o tom, jaké události chce ten daný modul přijímat, a při odeslání této události jádru je pak předána příslušným modulům. Dalším velmi zásadním modulem je modul Traffic, který slouží jako jádro simulace dopravy. To znamená, že zajišťuje načítání grafu silnic a řídí simulovanou dopravu. Tento modul mi bude při mé implementaci sice velkou inspirací, ale nebudu ho zde dále podrobněji vysvětlovat, protože jeho funkce jsou rozdílné oproti cílovým funkcím tohoto projektu. S tím zároveň souvisí modul VehicleSimulator, který je samozřejmě také velmi důležitý, protože má na starosti simulaci jízdních vlastností. Tím se budu také inspirovat, poněvadž obsahuje jednoduchý editor grafu. Ten rozhodně bude využit. Nejdříve si však rozebereme implementaci samotného grafu silnic.

6.2 Graf silnic

Sytém VRUT již disponuje implementací Grafu silnic. Protože se tato práce bude točit hlavně okolo této komponenty, rozebereme si ji podrobněji.

6.2.1 Popis grafu silnic

Autonomní vozidla se smějí pohybovat pouze po silnicích. Tedy je důležité zařídit, aby nesjížděla do příkopů či nenarážela do semaforů. Proto je nezbytné jim předat informaci, kde se nacházejí silnice, semafory, přechody pro chodce, cyklostezky, křižovatky, ale i další objekty, které musí být zohledněny pro plynulou jízdu vozidla. K tomu potřebujeme vhodnou datovou strukturu, která bude schopna udržet tento komplexní soubor informací a navíc bude dobře editovatelná.
V systému VRUT je na to použita pravděpodobně nejintuitivnější možnost- silniční graf. Konkrétně se jedná o orientovaný graf, poněvadž je pro nás důležité se ve struktuře rychle pohybovat mezi sousedními uzly. Každá silnice je tedy reprezentována uzly a orientovanými hranami. Vozidlo se bude pohybovat po jízdních pruzích (souhrn uzlů a hran), přičemž je důležité držet si informaci o sousedních uzlech (reprezentovanou jednoduchým seznamem), aby vozidlo vědělo, kam se má snažit dostat. Za účelem možnosti předjíždět či měnit jízdní pruhy je nezbytné udržovat si informace o sousedních pruzích a nějaké vazby mezi nimi. Nesmíme zapomenout na orientaci vedlejšího pruhu, abychom byli schopni rozlišit, zda se vozidlo právě nachází v levém pruhu dálnice či na normální okresní silnici. To je totiž důležité při výpočtu pro předjíždění a bude velmi důležité pro přidávání nájezdů a sjezdů z dálnice.

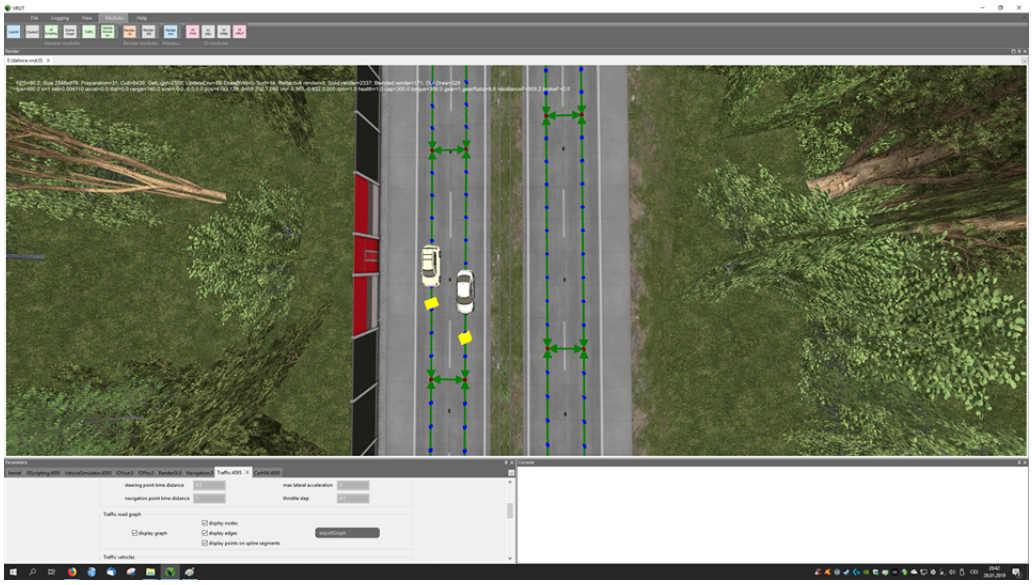

Figure 5: Na obrázku vidíme silniční graf vložený scény systému VRUT. Modré tečky značí uzly grafu a zelené čáry označují hrany grafu (dalo by se říci- možné cesty vozidla).

Je nutné v grafu zaznamenávat i neobvyklé situace. Například křižovatky jsou takovým speciálním případem, který je potřeba také v grafu zaznamenat (totéž platí pro další případy jako třeba spojování pruhů či naopak větvení do více pruhů). Rozeberme si více případ křižovatky. V kontextu grafu křižovatka vlastně znamená, že uzel bude mít více výstupních a vstupních hran. Avšak to neznamená, že daný uzel obsahuje všechny hrany, kam se křižovatka větví, nýbrž pouze ty, do kterých může (vzhledem k předpisům) vozidlo vjet. Aby nedocházelo ke kolizím, na těchto uzlech musíme udržovat informaci o tom, který uzel má před kterým uzlem přednost. Každý uzel si tedy drží seznam uzlů, které mají před tímto daným uzlem přednost, což si budou vozidla díky tomu moci ohlídat. Naneštěstí to nestačí, protože vozidlu nestačí informace, že pro tyto uzly má dát přednost. Musí taktéž vědět, zda je do nějakého z těchto uzlů právě naváděné jiné vozidlo, aby se dle toho mohlo rozhodnout, zda je potřeba zastavit či nikoliv. Proto je potřeba mít v každém uzlu zároveň seznam vozidel, která jsou do něj právě naváděna. Poté si jednoduše vozidlo zjistí, kterým uzlům má dát přednost a zjistí si, jestli do nich zrovna něco nejede a když ne, dá se na cestu.

Dále musí každý uzel obsahovat různé informace, které nejsme schopni nijak vyčíst ze silnice nebo z jejího okolí. Každý uzel proto drží i informaci o povolené rychlosti, druhu silnice, možnosti v daném uzlu předjíždět atd. Každý uzel tedy nese tyto informace:
• Souřadnice (3D) definující polohu uzlu
• Seznam uzlů, které navazují na tento
• Omezení rychlosti
• Druh silnice
• Zda je povoleno předjíždění
• Seznam uzlů, které mají před tímto uzlem přednost
• Seznam uzlů, které do daného uzlu směřují
• Parametry daného uzlu
• A další...

6.2.2 Implementace grafu silnic

Graf silnic je již v systému VRUT implementován, není tedy třeba jej nijak vylepšovat. Je naprogramován optimálně a pro účely této práce je jako stvořený. Mezi jeho hlavní výhody patří fakt, že je exportován do XML souboru. To nám umožňuje jej načítat pouze při inicializaci a nemusíme jej při každém spuštění simulace znovu generovat. Rozlišujeme 4 typy struktur ukládaných do XML souboru, které jsou vyžadovány za účelem efektivní implementace, která splňuje všechny výše uvedené požadavky. Jsou to uzly (*nodes*), hrany (*connections*), předjíždění (*overtaking*) a přednosti (*priorities*). Rozebereme si je postupně, přičemž začneme uzlem.
Uzel je nejzákladnější a nejjednodušší prvek celého grafu. Skládá se z několika atributů, přičemž ty nejdůležitější jsou jeho ID identifikátor a poloha v prostoru (jinak řečeno 3 souřadnice *x,y,z* v milimetrech). Dále obsahuje dva binární atributy o a *sp*. Pokud atribut o nabývá hodnoty 1, znamená to, že v daném uzlu je dovoleno předjíždět. Nula tedy logicky znamená, že předjíždění je zakázáno. Jedničková hodnota atributu *sp* znamená možnost generovat na daném uzlu vozidla pro simulaci. To se hodí, protože jsou místa (například uprostřed křižovatky), kde není úplně dobrý nápad generovat vozidla. Zbývající atribut *sl* označuje omezení rychlosti v daném uzlu. Zde je příklad definovaného uzlu:
<nodes>
<node id=”3” x=”234” y=”543” z=”1” o=”0” sp=”0” sl=”130”/>
...
</nodes>

Pravděpodobně by bylo nemoudré, náročné a implementačně neefektivní snažit se vyjádřit hrany mezi uzly pouze pomocí struktury pro uzly. Z toho důvodu se ve VRUTu nachází i struktura pro hrany (*connection*). Atributy tohoto elementu jsou jen dva- identifikátor s názvem *from*, který nese ID uzlu, ve kterém hrana začíná, a identifikátor *to*, který nese ID uzlu, ve kterém hrana končí. To naprosto jednoznačně vyjadřuje orientaci dané hrany. Co se týče počtu vstupních a výstupních hran pro daný uzel, není nijak omezen. Zde příklad definice hrany:
<connections>
<connection from="3" to="4"/>
...
</connections>
Je zajímavé, že VRUT obsahuje i samostatnou ukládanou strukturu pro předjíždění. Element předjíždění (*overtaking*) slouží k zachování informace o tom, ze kterého uzlu je potřeba si dávat pozor na vozidla v daných uzlech, která by mohla ohrozit bezpečnost při předjíždění. Je totiž podstatné si při předjíždění hlídat jízdní pruh vlevo od našeho, aby nedošlo ke kolizi. Abychom dosáhli bezpečného předjetí, má element předjíždění uchován vždy ID daného uzlu (ten reprezentuje atribut *id*) a ID uzlu, na který je třeba si dát pozor (reprezentován atributem *check*). To tedy znamená, že každý prvek předjíždění má schopnost ohlídat pouze jeden nebezpečný uzel. Proto je běžné, že každý uzel je jako *id* definován ve více prvcích elementu předjíždění. Proto je nutné se při předjíždění z daného uzlu podívat do všech prvků předjíždění a zkontrolovat všechny uzly v *check*. Teprve pokud se v žádném nenacházejí vozidla, můžeme zahájit předjíždění. Příklad:
<overtakings>
<overtaking id="5" check="3" />
...
</overtaking>
Na křižovatkách je nutné si ověřovat, zda nemá před námi jiné vozidlo přednost. K tomu slouží poslední element- přednost (*priority*). Určuje, která vozidla na kterém uzlu mají přednost před vozidly na jiných uzlech. Atribut *id* reprezentuje uzel, který má dát někomu přednost, a atribut *priority* reprezentuje uzel, který má před daným uzlem přednost. Ukázka na závěr:
<priorities>
<priority id="6" priority="9" />
...
</priorities>

Pomocí těchto čtyř elementů jsme schopni plně reprezentovat graf a zároveň ho mít uložen pro veškeré simulace či jiné potřeby. Nevýhodou ovšem doposud byla právě náročná editace tohoto grafu, to se ale díky mé práci brzy změní.
Co nyní je implementováno, je automatické generování grafu, pokud žádný neexistuje. Vytvoří se tím pouze základní množina uzlů, kterou je nadále potřeba ručně zkontrolovat a upravit. Funguje to tak, že modul požádá scénu o geometrii vozovek (specifikovanou identifikátorem v souboru *worldFile*). Takže jakmile má modul seznam trojúhelníků od scény, která tvoří síť vozovky, vytvoří uprostřed každého trojúhelníku uzel. Takto vygenerovanou množinu uzlů uloží do XML souboru *trafficGrapfhFile*. Poté už jen zbývá, aby uživatel ručně v tomto souboru uzly propojil, případně je odstranil tak, aby zmizely všechny nedokonalosti. Je zřejmé, že to není ten nejefektivnější a nejrychlejší možný způsob. Další funkce modulu je také velmi zajímavá a velmi pravděpodobně bude využita v mé implementaci. Je to funkce aktuálně upravené uzly přeexportovat do XML souboru, který se tím pádem přepíše. Ve VRUTu je totiž již implementován velmi jednoduchý manažer scén, který umožňuje uživateli vymazat uzly ze scény za běhu simulace. Využívá k tomu uživatelské rozhraní, které slouží právě k manipulaci geometrie.

6.3 Editor grafu v systému VRUT

Nyní se podívejme na to, jak je vyřešen editor tohoto grafu.

6.3.1 Dosavadní řešení

Dosavadní řešení má bohužel pár much. Mezi hlavní nedostatek patří fakt, že modul Traffic, který řídí prakticky celou simulaci vozidel, také řídí editor grafu. Tento modul umí zobrazit ve scéně graf silnic a ten lze následně upravovat pouze přes XML soubor, který je potřeba ručně upravit a poté spustit znovu celou simulaci. Je zřejmé, že to není zrovna nejrychlejší a nejefektivnější způsob.

6.3.2 Modul Traffic a editor grafu

Implementace z předchozího odstavce je provedena pomocí modulu Traffic. Traffic obsahuje funkce na zobrazení grafu silnice. Ten je uložen pomocí *RoadGraph.cpp* do příslušného XML souboru. Traffic umí přečíst uložená

data a vložit je přímo do simulace jako další vrstvu, přičemž hrany jsou zobrazeny zelenou šipkou (všechny hrany jsou orientované) a uzly červenými krychličkami. Po kliknutí na daný uzel či hranu se objeví nové okno s informacemi o daném prvku (např. třída silnice, povolená rychlost, atd.). Editace jednotlivých uzlů probíhá přes editaci XML souboru, jak již bylo zmíněno.

7 Popis vlastní implementace

Mým prvním krokem bylo vytvoření nového modulu s názvem *RoadGraphEditor*.

7.1 Vytvoření modulu *RoadGraphEditor*

Ve složce "VRUT/modules" jsem vytvořil novou složku s názvem *RoadGraphEditor*. Do ní jsem vložil soubory jménem *RoadGraphEditor.cpp* (tento soubor představuje hlavní třídu modulu), *RoadGraphEditor.h* (v tomto souboru jsou uloženy veškeré deklarace všech struktur a proměnných), *CMakeLists.txt* (tento soubor slouží ke kompilaci modulu). Poslední zmíněný soubor jsem upravil dle příkladu jiného již fungujícího modulu do podoby, aby zahrnoval do kompilace předešlé soubory.

7.1.1 Získání přístupu k silničnímu grafu

Následujícím krokem bylo zkopírování souborů *RoadGraph.h* a *RoadGraph.cpp* z modulu Traffic a přidání těchto souborů do kompilace pomocí souboru *CMakeLists.txt*. Poté jsem pomocí funkce *ChangeScene* z modulu Traffic vytáhnul graf ze souboru a uložil ho do paměti. Toho jsem docílil pomocí metody *Load()* třídy *RoadGraph*. Této metodě bylo třeba předat název souboru, ze kterého má být načten graf. Proto jsem zaregistroval parametr *trafficGraphFileParamID* pomocí *REGISTERPARAMGUIFILECONTROL*.
Po zapnutí scény se vyvolá event *UPDATEPARAMFROMEVENTSTRING* (také přidán do hlavní třídy) a ten zapíše do tohoto parametru informaci o názvu souboru, ve kterém je uložen daný silniční graf pro danou scénu. Tento parametr a event jsem opět zkopíroval z modulu Traffic. Tímto procesem získal náš nový modul přístup k silničnímu grafu.

7.1.2 Zobrazení grafu silnice do scény

Dalším nezbytným krokem k interaktivnímu editoru byla možnost zobrazení grafu do scény. Touto možností opět již disponuje modul Traffic, tudíž jsme stále čerpali z tohoto zdroje. V tomto modulu se graf do scény zobrazí po kliknutí na checkbox "Display graph". V našem modulu se graf do scény zobrazí úplně stejným způsobem. Proto jsem z modulu Traffic zkopíroval jeho rozhraní a to pomocí souborů *TrafficLayout.xrc* a *TrafficLayout.fbp*. Tyto soubory jsem vložil do mého modulu a přejmenoval je na *RoadGraphEditorLayout.xrc* a *RoadGraphEditorLayout.fbp*. Následně jsem je přidal do kompilace stejně jako předchozí soubory. Kliknutí na checkbox v původním rozhraní změnilo hodnotu parametru *displayRoadGraphParamID*, což vyvolalo event, který způsobil zobrazení grafu. Můj modul tímto způsobem pracuje také. Po kliknutí na tento checkbox se vyvolá funkce *UpdateSceneGraphNodes*, která způsobí zobrazení grafu do scény. Tu bylo také potřeba zkopírovat z modulu Traffic.

Figure 6: Checkbox Display Road Graph je zaškrtnut- graf je vykreslen.

7.1.3 Bezpečnostní opatření modulu

Co se týče bezpečnostních návrhů na můj modul, tak ty nebyly naplněny a můj modul je třeba pouštět pouze samostatně jen s moduly, které nespouští žádné simulace. Tudíž pokud chce uživatel editovat silniční graf, musí k tomu přizpůsobit spuštění VRUTu.

7.2 Označení uzlu

Jeden z prvních úkolů mé implementace bylo zařídit jednoduché označení uzlu, abychom si mohli zobrazit jeho parametry. Aby nedocházelo k nemíněným označením uzlů, je třeba před označením uzlu podržet klávesu Control a pak již je možné levým kliknutím vybrat daný uzel. Zde si ukládám informaci o právě zvoleném uzlu a informaci o právě zvolené geometrii. Označený uzel změní barvu na modrou. Následně se do mého stvořeného rozhraní překopírují všechny parametry daného uzlu z grafu a ve scéně se objeví dva malé sloupečky kolem vybraného uzlu určující šířku silnice. Z pozdějších kapitol vyplyne, že je možné i označování hran. Platí, že v jeden okamžik je možné mít označený jen jeden prvek (buď hranu nebo uzel). Proto označení uzlu odznačí hranu a obráceně.

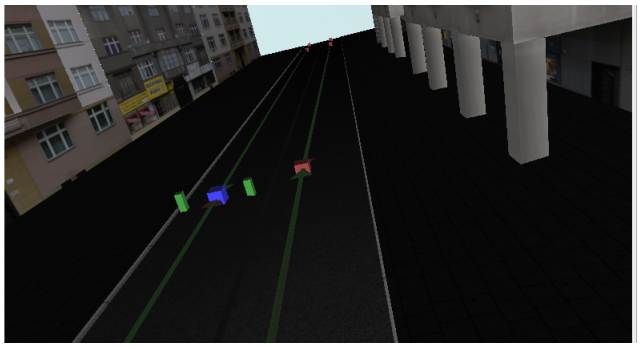

Figure 7: Zde je vidět označený uzel (modrá barva).

7.3 Odznačení uzlu

V jistých situacích může být nutné uzel odznačit. Odznačení uzlu probíhá zčásti automaticky. Uzel se automaticky odznačí po označení jiného uzlu (pokud není záměr mít více označených uzlů, viz níže- "Vložení hrany"). Dále se uzel automaticky odznačí, pokud spustím tzv. "Edge mode" či "Node mode", o nichž bude řeč v dalších kapitolách. Další možností je záměrné odznačení uzlu. To funguje stejným postupem jako označení uzlu. Podržení Control a kliknutí levého tlačítka myši na již označený uzel způsobí odznačení uzlu (změna barvy, změna informace o zvoleném uzlu, smazáni ukazatele šířky silnice).

7.4 Úprava atributů uzlu

Poté co máme označený nějaký uzel, je možné měnit jeho parametry. Změna parametru funguje jednoduše tak, že příslušné pole parametru, které chci změnit, přepíši a zmáčknu tlačítko Enter. To způsobí vyvolání Eventu, který ve struktuře grafu v daném uzlu změní daný parametr. Zatím nedochází k žádnému ukládání do souboru. Změna parametru "width" zároveň způsobí automatické překreslení ukazatelů šířky silnice. Tímto způsobem se zároveň dá měnit poloha uzlu ve scéně. Tedy změnou parametrů: "x" (x-ová souřadnice uzlu), "y" (y-ová souřadnice uzlu), "z" (z-ová souřadnice uzlu). Také se po zmáčknutí tlačítka Enter daný uzel ve scéně překreslí. Spolu s ním se překreslí i všechny hrany, které s ním souvisejí. To probíhá pomocí procházení všech hran v grafu. Veškeré změny jsou ukládány do datové struktury grafu. .

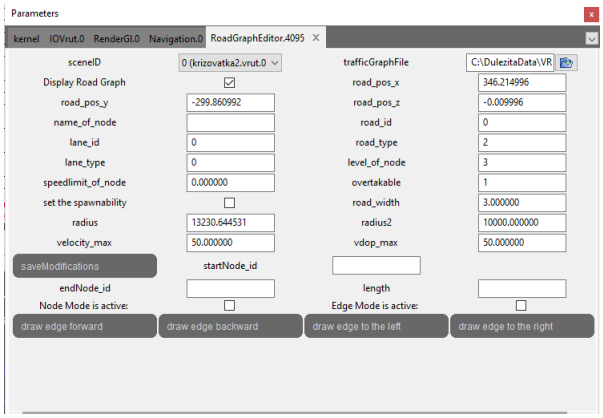

Figure 8: Přepsáním každého atributu ho lze snadno změnit.

7.4.1 Spuštění editačních módů

Dalším krokem bylo vkládání uzlů do scény, avšak pro vkládání uzlů do scény je třeba mít spuštěný tzv. "Node mode". Ten se dá aktivovat v jakýkoliv okamžik stisknutím tlačítka Shift nebo zaškrtnutím daného checkboxu. V mém modulu budeme pracovat ještě s tzv. módem "Edge mode". Ten lze spustit stisknutím tlačítka E, nebo zaškrtnutím daného checkboxu. Platí, že v jeden okamžik smí být aktivní pouze jeden mód, proto spuštění jednoho módu automaticky ukončí ten druhý. Zároveň spuštění libovolného módu vynuluje aktuálně vybraný uzel či hranu. Mezi módy lze rychle přepínat.

To znamená, že pokud spustím nějaký mód, přestože je aktivní již jiný, tak se ten původní ukončí. Ukončení daného módu probíhá stejně jako jeho spuštění- příslušným tlačítkem, nebo příslušným checkboxem.

Figure 9: Editační módy lze spustit kliknutím na příslušný checkbox.

7.4.2 Editačních mód- "Node mode"

Tento mód slouží ke vkládání nových uzlů do scény. Po jeho spuštění již stačí kliknout na libovolné místo ve scéně, a pokud jsme klikli na neobsazené místo, tak se tam vytvoří nový uzel a uloží se do struktury grafu. Pokud budeme pokračovat (opět kliknutím na neobsazenou pozici) přidáním druhého uzlu, tak se zároveň vytvoří v paměti těchto uzlů hrany ForwardEdge a BackwardEdge. Ty se také vykreslí do scény. K tomuto módu jsem implementoval ještě jednu zajímavou funkci a to možnost navázat na rozdělanou silnici novou. Například pro přidání křižovatky je třeba opustit Node mode, spojit hrany a pak pokračovat v původní silnici. K tomu stačí pouze kliknout na uzel, kde chceme navázat, ten se označí žlutě a my můžeme pokračovat v silnici. K ničemu jinému tento mód neslouží.

7.5 Vymazání uzlu

K úspěšnému smazání uzlu je třeba, abychom neměli spuštěný ani jeden editační mód a aby byl daný uzel zvolený pomocí klávesy Control a levým tlačítkem myši. Poté stisknutí tlačítka X způsobí smazání uzlu. To proběhne tak, že se nejdříve podíváme na veškeré hrany, které jsou s tímto uzlem spojené. Pokud není spojený hranou s žádným jiným uzlem, jednoduše se daný uzel odstraní ze scény a ve struktuře grafu se nastaví na hodnotu NULL. Pokud však má nějaké spojení s jinými uzly, nejdříve se hrany mezi nimi odstraní ze scény (v tomto případě hrany záměrně zůstávají ve struktuře grafu) a až po té se odstraní daný uzel.

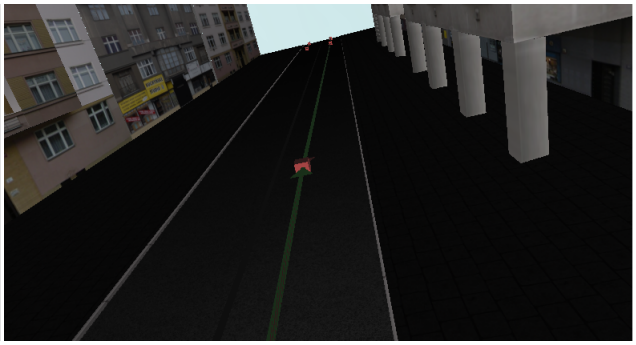

Figure 10: Uzel byl smazán- s ním u příslušné hrany.

7.6 Označení hrany

Označení hrany probíhá stejným způsobem jako označení uzlu. Tato operace jednoduše vytáhne informace o dané hraně z grafu a zobrazí je.

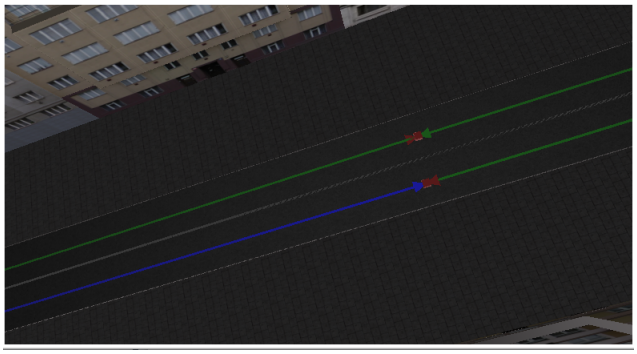

Figure 11: Zde je vidět označená hrana (modře).

7.7 Odznačení hrany

Odznačení hrany funguje stejným způsobem jako odznačení uzlu, s tím rozdílem, že hrany nemají žádný ukazatel šířky silnice, tudíž není třeba jej odstraňovat.

7.8 Vymazání hrany

Vymazání hrany je vlastně jednodušší forma mazání uzlu, poněvadž mazání uzlu obsahuje mazání hran. Hrana se také musí označit a smazat tlačítkem X. Tím je vymazána ze scény a z datové struktury jejích bývalých sousedů.

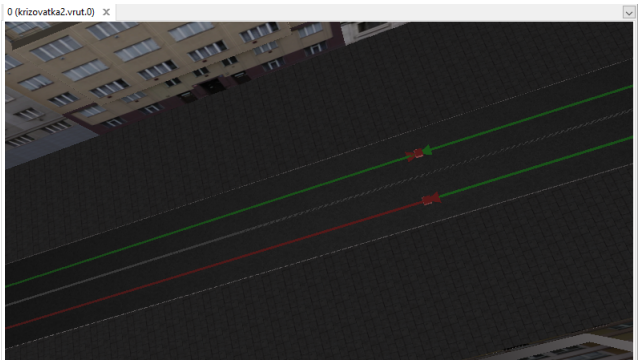

Figure 12: Hrana byla smazána, červeně je nyní vidět hrana vedoucí opačným směrem.

7.8.1 Editačních mód- "Edge mode"

Tento mód slouží ke vkládání hran. Po spuštění tohoto módu je třeba si pomocí myši zvolit (jednoduchým kliknutím) dva uzly, které budu chtít spojit. První zvolený uzel má barvu modrou, druhý žlutou. Opětovné kliknutí na uzel ho odznačí a pokud to byl uzel první a zároveň uzel druhý již byl také zvolen, stává se z druhého uzlu první. Jakmile mám označené dva uzly, hrany mezi ně přidám pomocí tlačítek "Draw Forward Edge", "Draw Backward Edge", "Draw Left Adjacent Edge", "Draw Right Adjacent Edge". Zvolení jednoho z prvních dvou tlačítek vykreslí danou hranu a uloží ji do struktury grafu i do struktur sousedních uzlů. Každý uzel má rozlišeno, o jakou hranu se jedná. Například uzel na křižovatce má sousedící 4 uzly a každý je uložen jinak, aby bylo možné je snadno identifikovat. Maximální počet sousedících uzlů je pro každý uzel stanoven na 4.

7.8.2 Funkce Undo

Uživatel si může někdy přát některé své změny vrátit. Od toho jsem implementoval fukci Undo(). Ta je schopna vrátit zpátky všechny editační prvky, které jsem tu popsal. Slouží k tomu moje speciální struktura EditEvent.

Funguje to tak, že pokaždé, když se spustí validní editace, zároveň se vytvoří jeden prvek struktury EditEvent, kam se uloží všechny potřebné informace k obnově dané změny, např. pro vrácení smazání nějakého uzlu se tam vloží instance tohoto uzlu a instance jeho hran. Takto vytvořený EditEvent se vloží do vektoru editEvents. Po zmáčknutí tlačítka Z (na anglické klávesnici), se vezme poslední prvek z tohoto vektoru a provede se vrácení dané změny. Každý EditEvent prvek si zároveň uchovává informaci o typu změny, která byla provedena. To slouží ke snadnější identifikaci, co s těmi uloženými daty dělat. Ukázka funkčnosti bude v kapitole 9.

7.8.3 Funkce SaveTheGraphFile()

Pro uložení všech změn do XML souboru je třeba zmáčknout tlačítko "Save Graph File", které spustí stejnojmennou funkci. Tato funkce již byla implementována v souboru RoadGraph.cpp. Já ji pouze mírně upravil, aby odpovídala zmíněným editacím. Ukázka funkčnosti bude taktéž v kapitole 9.

8 Rozdíly oproti návrhu řešení a návrh úprav stávající implementace

8.1 Hlavní rozdíly oproti návrhu řešení

Původní záměr začátku editace byl ten, že půjde vyvolat již během probíhající simulace dopravy. Mělo to fungovat tak, že kliknutím na tlačítko "Edit" se spustí editační mód a vypne se simulace. Poté může uživatel upravit silniční graf a po ukončení modulu (opětovné stisknutí Edit) se nový graf uloží a budeme moci pokračovat v aktuální verzi grafu. Nakonec modul nic takového neumožňuje a uživatel si musí před spuštěním VRUTu říci, že chce editovat silniční graf. Ten se ukládá podle přání uživatele samostatným tlačítkem a po vypnutí VRUTu se již dá daný graf používat. Dále hlavní změnou oproti návrhu je použití již integrovaného uživatelského rozhraní pro výčet parametrů uzlu, protože systém VRUT je nekompatibilní s knihovnu wxWidgets.

8.2 Návrh úprav stávající implementace

Jak již bylo řečeno, modul není možné spustit během simulace dopravy, což by však mohlo vynést editace na nový level. Další věc, která by mohla

zpříjemnit uživatelskou práci s editorem, by byl funkční manipulátor geometrie uzlu s funkcí Drag and Drop, aby nebylo třeba odhadovat parametry souřadnic pozice, kde chci daný uzel mít.

9 Přínos modulu RoadGraphEditor

9.1 Původní možnost editace

Jak je nám již známo, k editaci uzlů a hran bylo potřeba až doteď psát přímo do daného XML souboru. Každý uzel musel dostat ručně svoji polohu(která se dala zjistit pouze ve spuštěném VRUTu), všechny své atributy a zde byla také jediná možnost je změnit. To znamenalo, že pokud jste napsali špatnou polohu daného uzlu, bylo třeba celý systém vypnout, danou pozici přepsat a spustit systém znovu. Také bylo třeba ručně vložit jednotlivé hrany a každá chyba udělaná v tomto souboru měla za výsledek nemožnost spuštění systému VRUT. Tímto způsobem tvorba pouze jednoho uzlu zabrala téměř 5 minut, přičemž došlo na straně uživatele k chybnému zadání souřadnic a uzel se vykreslil úplně jinde, než bylo plánováno.

9.2 Současná možnost editace

Modul RoadGraphEditor mnohonásobně zrychluje editaci silničního grafu a zajišťuje snadné a rychlé opravy neúmyslných chyb. Z testování vyplývá, že přidání jednoho uzlu je otázka vteřin a vytvoření sjezdu z dálnice a následného překlenutí dálnice s pomocí mostu, je záležitostí na pár minut.

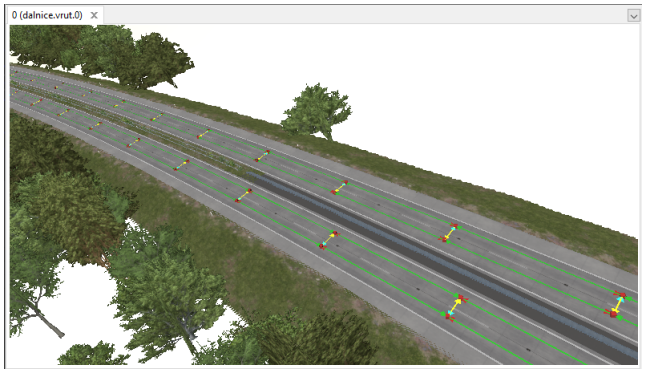

Figure 13: Zde vidíme dálnici před editací.

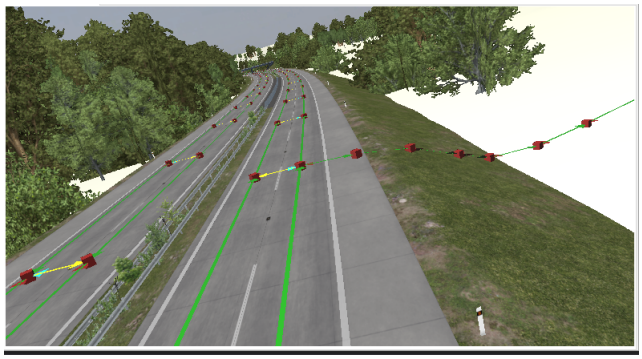

Figure 14: Zde vidíme dálnici po editaci a její sjezd.

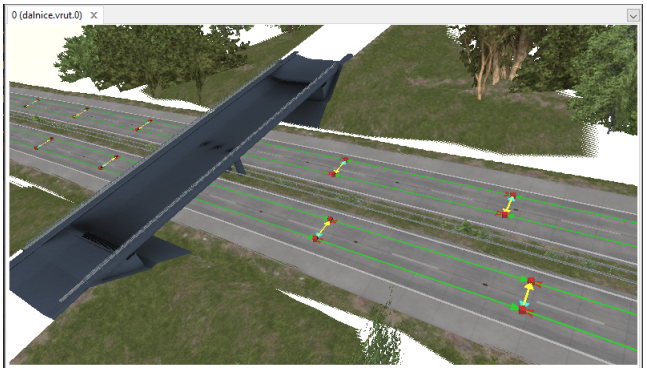

Figure 15: Zde vidíme most před editací.

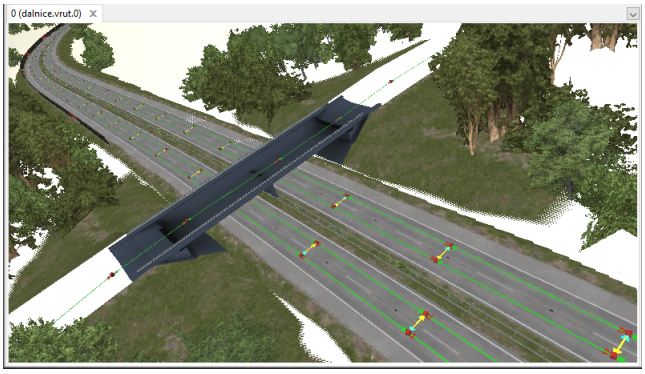

Figure 16: Zde vidíme most po editaci.

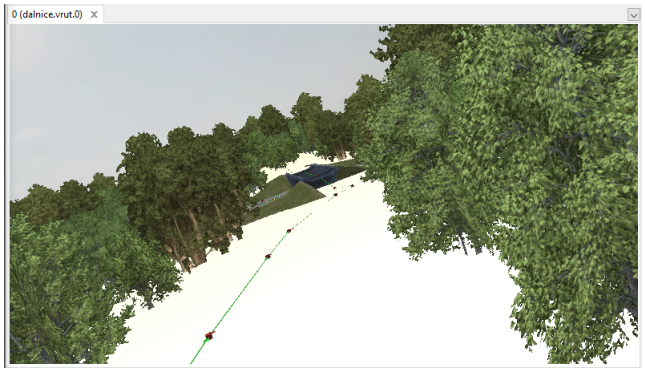

Figure 17: Souhrnný obrázek- celá tato editace trvala přibližně 8 minut.

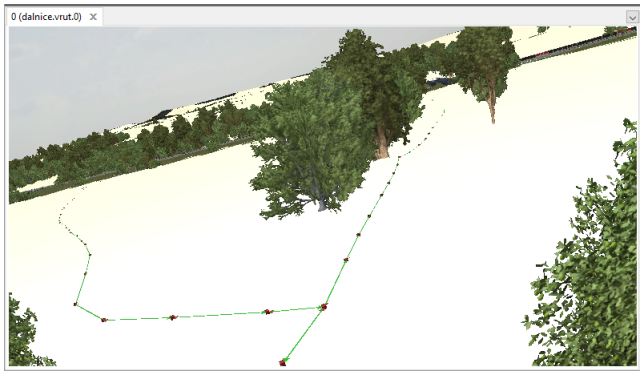

Figure 18: Souhrnný obrázek 2.

Další velikou výhodou editoru je možnost měnit snadno parametry jednotlivých uzlů (na rozdíl od hran, jejichž parametry jsou pouze pro čtení). Před tím bylo třeba daný parametr měnit v souboru XML, nyní postačí v uživatelském rozhraní daný parametr přepsat a zmáčknout Enter.

Figure 19: Změna parametrů se ukládá do struktury a po uložení grafu i do souboru.

Navíc změna některých konkrétních parametrů vyvolá okamžité vizuální změny. Tímto způsobem se například mění poloha uzlu ve scéně. Po uložení grafu se tato změna projeví i v XML souboru.

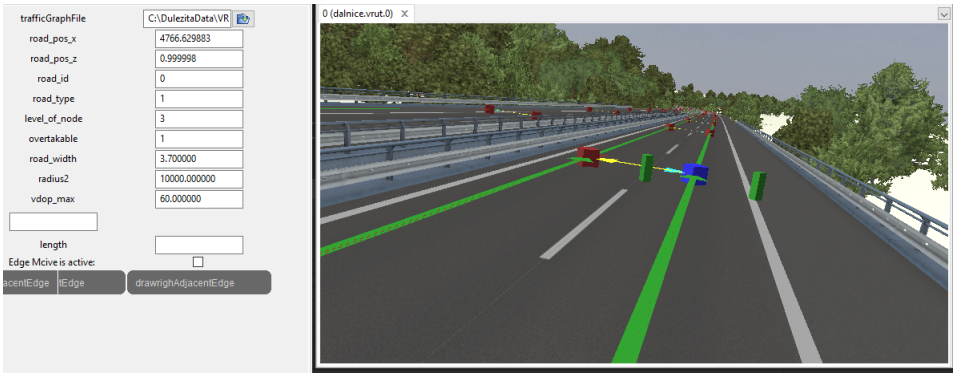

Figure 20: Počáteční poloha uzlu.

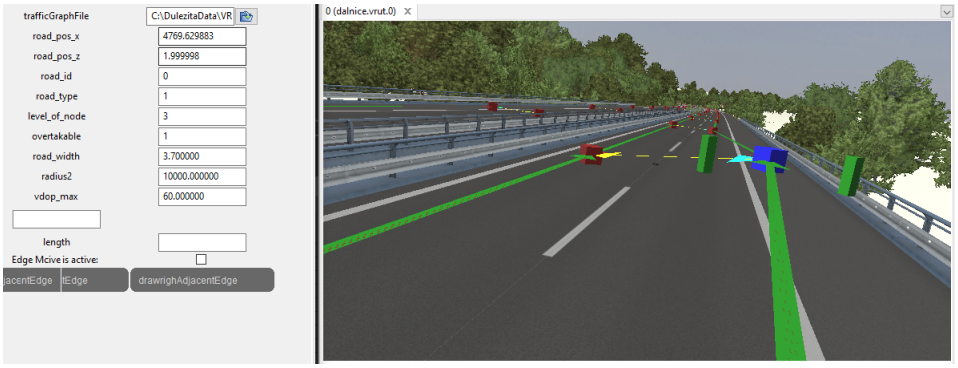

Figure 21: Poloha uzlu po přepsání parametrů pos.x a pos.z.

Tento editor také zvládá snadný návrat chybné editace. Na následujících obrázcích můžeme vidět výše editovaný uzel navrácený do původní polohy.

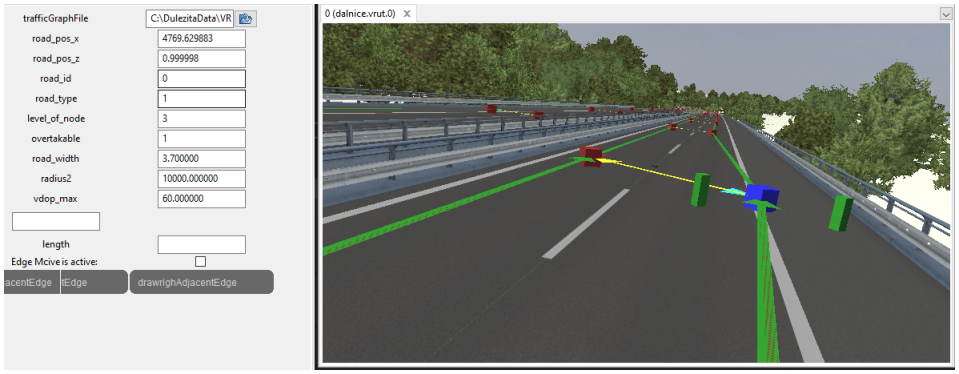

Figure 22: První krok Undo.

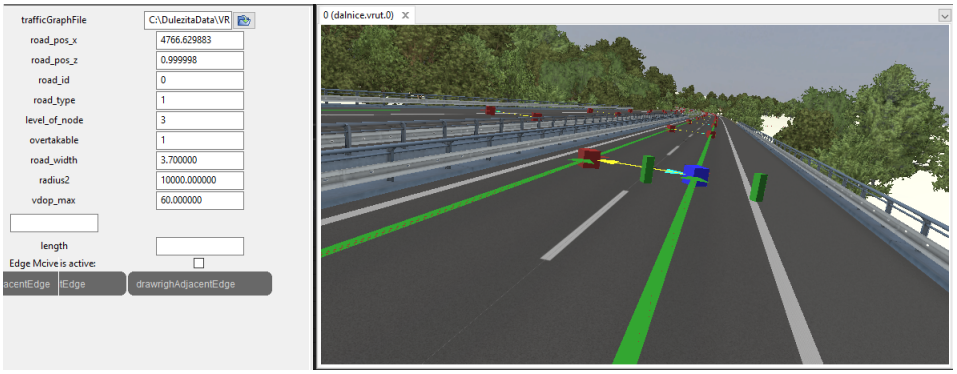

Figure 23: Druhý krok Undo.

Také změna šířky silnice způsobí překreslení geometrie, což je velmi praktické pro nastavení správné šířky.

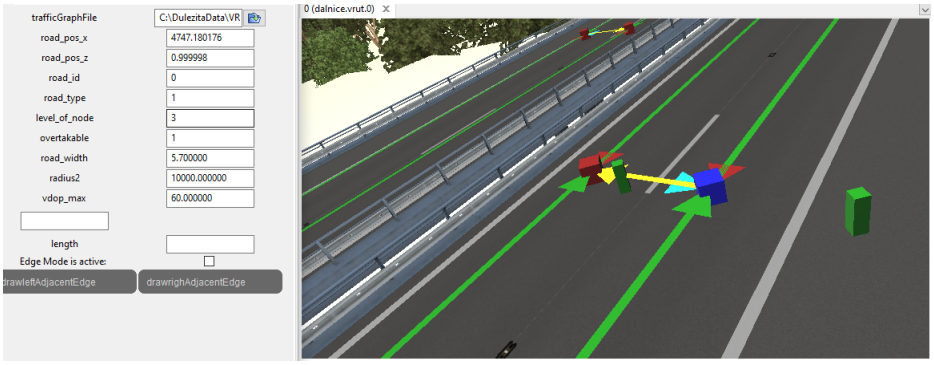

Figure 24: Změna šířky silnice.

Občas je dobré špatný uzel odstranit. To nám zajišťuje funkce DeleteNode.

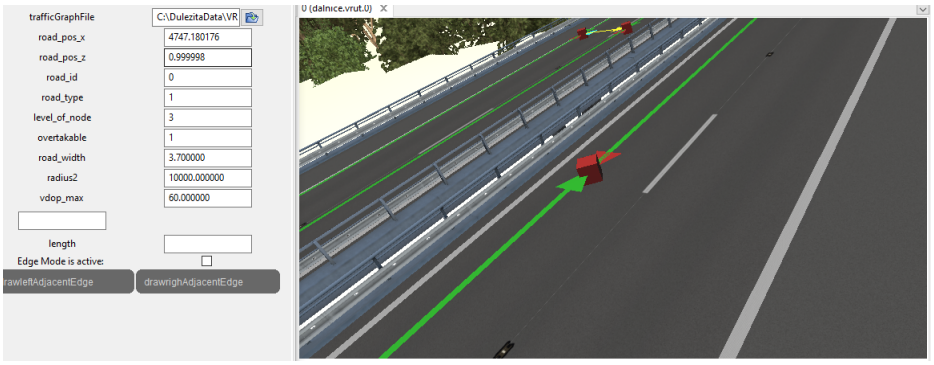

Figure 25: S uzlem se autoamticky odstraní i všechny hrany s ním spojené.

Avšak zde obzvlášť je nutné mít pro případ chyby možnost uzel vrátit zpět.

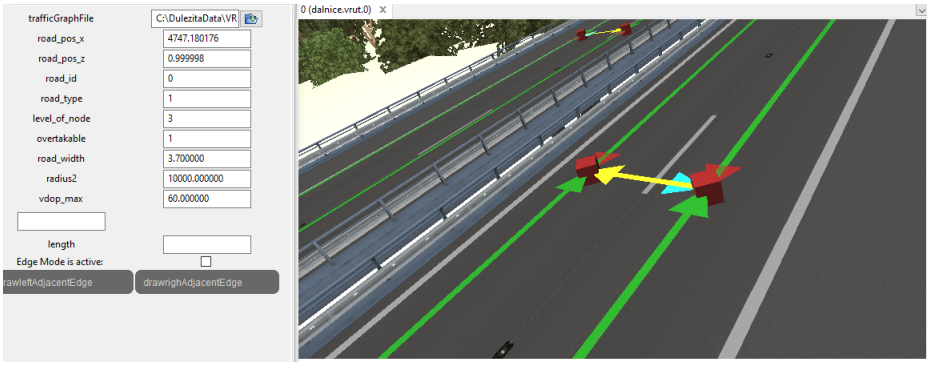

Figure 26: Po vrácení uzlu je struktura grafu zachována.

Pro kreslení hran mezi jednotlivými uzly slouží 4 tlačítka v dolní části uživatelského rozhraní.

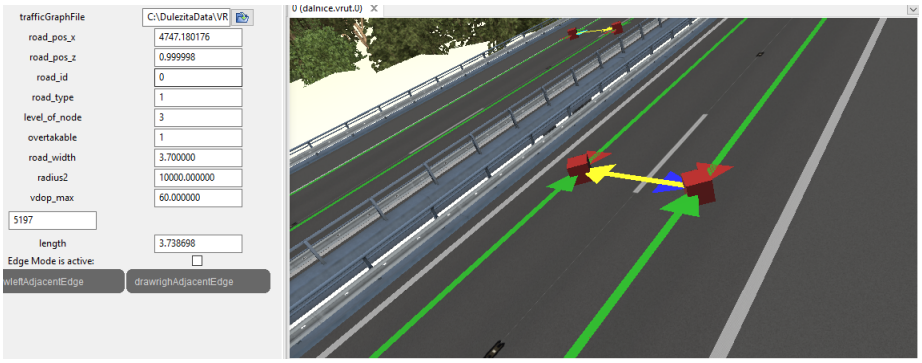

Figure 27: Hrana, kterou smažeme, je označena tmavě modře.

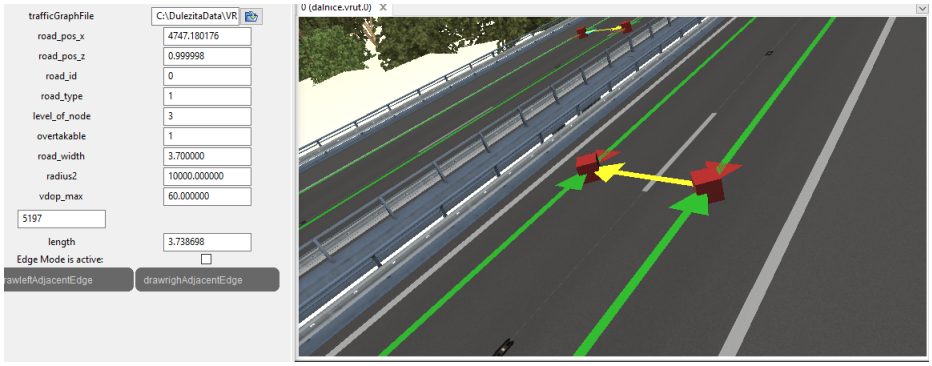

Figure 28: Hrana byla smazána- k jajímu navrácení lze použít vykreslení nebo funkce Undo na krok zpět.

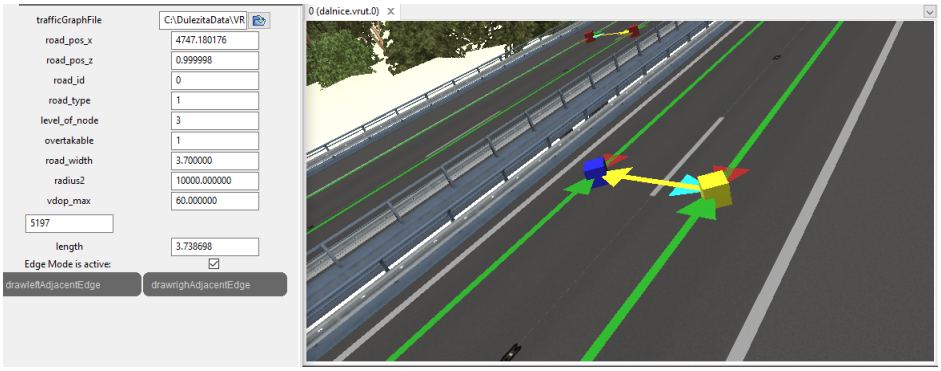

Figure 29: Hrana byla vykreslena zpět. Tmavě modře je označen uzel, ze kterého hrana vede, žlutě je označen uzel, do kterého hrana vede.

9.3 Test na vytvoření kompletně nového silničního grafu

Tento závěrečný test přinesl jisté velice užitečné poznatky. Odhalil ještě jednu poměrně zásadní chybu, která byla následně opravena. Jednalo se o počet povolených sousedů každého uzlu. Na snímcích níže můžete vidět scénu před editací a po editace (jedná se o celkem devět křižovatek).

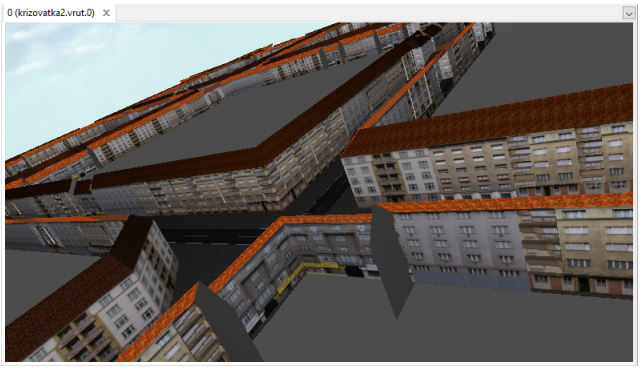

Figure 30: Needitovaná scéna.

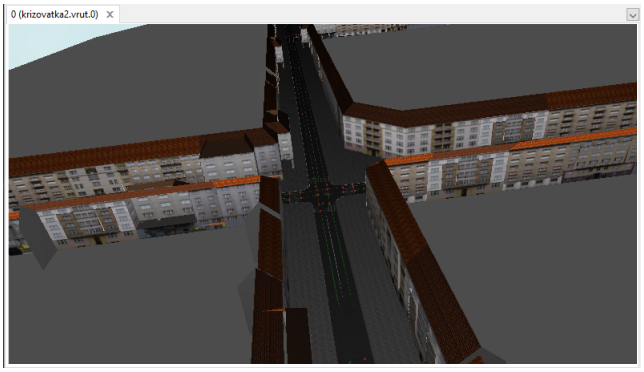

Figure 31: Editace dokončena. Můžete si povšimnout, že různé typy hran mají různou barvu.

Práce byla hotova přibližně za 1 hodinu a 30 minut. To znamená, že na jednu křižovatku bylo v průměru potřeba méně než deset minut (jednotlivé silnice mezi křižovatkami byly vykresleny do minuty). Nejdéle trvalo vykreslování jednotlivých hran, poněvadž pro vykreslení konkrétní hrany mezi konkrétní dva body je potřeba oba body ve správném pořadí označit a poté kliknout na příslušné tlačítko.

10 Závěr

Výsledkem této bakalářské práce měl být funkční intuitivní editor silničního grafu. To se nakonec povedlo a z výsledků testů vychází, že samotná editace se pro budoucí vývojáře mnohonásobně zrychlila. Nyní již pro úspěšnou úpravu silničního grafu není vůbec třeba fyzicky manipulovat s XML souborem a vše lze měnit přímo v prostředí systému VRUT. K zajištění této funkčnosti bylo třeba zasáhnout do stávající struktury silničního grafu. Jediné, co editoru ještě schází, je (jak již bylo zmíněno) funkce Drag and Drop pro editaci pozice daného uzlu a možnost spustit modul během probíhající simulace. Díky tomu je tu příležitost pro jiného studenta se na tomto projektu rovněž podílet. Doufám, že se takový student najde, protože systém VRUT si rozhodně zaslouží další vylepšení.

Literatura

[1] Bc. Jaroslav Minařík. Simulace okolních dopravních dějů, 2014.

[2] Pascal Neis. The openstreetmap contributors map aka who's around me?, 2013.

[3] Bc. Chrudoš Vorlíček. Prototyp turistického systému založeného na datech openstreetmap, 2013.

[4] OpenStreetMap Wiki. File:josm-2013.png, 2016.

[5] taj ČTK. Autonomní auta budeme v ulicích potkávat do deseti let, tvrdí expert, 2018.