

Západočeská univerzita v Plzni  
Fakulta aplikovaných věd  
Katedra informatiky a výpočetní techniky

## **Diplomová práce**

# **Grafický editor schémat dopravních systémů**

---

## **Poděkování**

Ráda bych poděkovala vedoucímu diplomové práce panu Doc. Ing. Pavlu Heroutovi, PhD. za odbornou pomoc a užitečné rady při zpracování mé diplomové práce. Nemenší dík patří také všem těm, kteří se mnou během mnoha měsíců, ve kterých jsem diplomovou práci vytvářela, měli téměř bezmeznou trpělivost, mé rodině i přátelům za jejich podporu a pochopení v průběhu celého mého studia.

---

## **Graphical Editor of the Traffic System Schemes**

This work describes the editor of street graphs, which is a part of system for a traffic simulation – Java Urban Traffic Simulator (JUTS). It enables the users to create a street graph from the given geographical data. The street graph is a detailed road map of a town district, it includes roads, crossroads, vehicle or parking lanes. Street graphs are intended to be used as an environment for a traffic simulation in Pilsen (Czech Republic), or in any other town which can provide the whole simulation system with all input data in the correct format.

All actions of the editor can be done in an elegant way, users do not need to be specialists in geographical systems. The users can create a new map in XML format.

---

---

## Obsah

1	Úvod.....	1
2	Systém JUTS.....	2
2.1	Editor.....	3
2.2	Formáty souborů.....	4
2.2.1	Souřadnice uzlových bodů.....	4
2.2.2	Souřadnice segmentů silnic.....	5
2.2.3	Popis křižovatek.....	6
2.2.4	Grafická část simulační mapy.....	9
2.2.5	Simulační část simulační mapy.....	12
3	GIS systémy.....	19
3.1	Existující prostředky.....	19
3.2	Funkčnost, výběr vhodných funkcí.....	20
4	Návrh editoru.....	21
4.1	Návrh grafického prostředí.....	21
4.2	Algoritmy využívané editorem.....	22
4.2.1	Vytvoření silniční sítě.....	23
4.2.2	Výběr uzlového bodu.....	23
4.2.3	Výběr segmentu silnice.....	24
4.2.4	Oprava chybného segmentu.....	25
4.2.5	Vytvoření sítě křižovatek.....	25
4.2.6	Vygenerování hranice křižovatky.....	28
4.2.7	Výpočet hranic jízdnic pruhů, jejich korekce.....	31
4.2.8	Generování odbočovacích jízdnic pruhů.....	33
4.2.9	Automatické generování buněk v křižovatce.....	34
4.2.10	Nalezení křižovatky.....	37
4.3	Funkce editoru.....	38
4.3.1	Funkce pro práci se souborem.....	38
4.3.2	Pohledové funkce.....	38
4.3.3	Funkce pro práci s elementy mapy.....	39
4.3.4	Nástroje.....	42
4.3.5	Nápověda.....	42
5	Implementace editoru.....	43
5.1	Grafické třídy.....	43
5.1.1	GAboutDlg.....	43
5.1.2	GColorDlg.....	44
5.1.3	GChooseCrossroadDlg.....	44
5.1.4	GNewMapDlg.....	44
5.1.5	GPointMoveDlg.....	44
5.1.6	GSaveMapDlg.....	45
5.1.7	GScaleDlg.....	45
5.1.8	GUserInterfaceDlg.....	45
5.1.9	GWindow.....	46
5.1.10	MyPanel.....	47

---

---

5.2	Podstatné negrafické třídy.....	48
5.2.1	CrossroadsDOM.....	48
5.2.2	GraphicXML.....	48
5.2.3	GDescription.....	48
5.2.4	JUTS_constants.....	49
5.2.5	Main.....	51
5.2.6	MapXML.....	51
5.3	Třídy pro elementy mapy.....	52
5.3.1	AccessPlace.....	52
5.3.2	CrossPoint.....	52
5.3.3	GenTerm.....	53
5.3.4	Krizovatka.....	53
5.3.5	NextPlace.....	53
5.3.6	Place.....	54
5.3.7	PruhDovnitř.....	54
5.3.8	PruhVen.....	54
5.3.9	RamenoKrizovatky.....	54
5.3.10	Road.....	54
5.3.11	RoadSegment.....	55
5.3.12	SuccPred.....	56
5.3.13	VehicleLane.....	56
5.3.14	Way.....	56
5.4	Pomocné třídy.....	57
5.4.1	Curve.....	57
5.4.2	DoublePoint.....	57
5.4.3	Line.....	57
5.4.4	ObsluhaChyb.....	57
6	Potíže při řešení, možná zlepšení.....	58
7	Závěr.....	60
	Literatura.....	61
	Elektronické zdroje.....	61

## **Přílohy:**

A – Uživatelská dokumentace

---

---

Prohlašuji, že jsem diplomovou práci vypracovala samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 15. 5. 2005

---

Jana Hájková

---

# 1 Úvod

Simulace dopravních systémů jsou oblastí, která se prudce rozvíjí také díky zvyšující se výkonnosti výpočetní techniky. Nezastupitelné místo v přípravě podkladů a ve vizualizaci výsledků či přípravě vstupních dat má i počítačová grafika.

Simulačních systémů existuje větší množství na různých úrovních detailnosti. V současné době město Plzeň využívá pro simulaci dopravy v Plzni systém založený na makroskopické simulaci, který pracuje pouze s toky vozidel po jednotlivých komunikacích a vychází z koncentrace obyvatel v jednotlivých částech města. Systém umožňující simulaci dopravy na mikroskopické úrovni (tzn. na úrovni jednotlivých vozidel) dosud nemá město k dispozici. Z tohoto důvodu se rozvinula spolupráce ZČU a města Plzně na projektu JUTS (Java Urban Traffic Simulator), umožňujícím zmíněnou mikroskopickou simulaci.

Projekt JUTS je komplexním projektem, na kterém v současné době pracuje tým lidí. Hlavní část projektu je simulační, ale kromě ní je potřeba několik dalších podpůrných aktivit, které se na vlastní simulaci přímo nepodílejí. Jednou z nich je příprava simulačních map, tj. vstupních dat pro simulaci, podle kterých simulace na určitém konkrétním úseku probíhá. Mapy musejí být připravovány na základě reálných geografických dat a dalších podkladů, popisujících např. dopravní značení apod. Těchto dat existuje relativně značné množství, jak co do objemu, tak i do různorodosti.

V první verzi projektu JUTS byly tyto mapy připraveny „ručně“. Tento způsob má víceméně pouze nevýhody, mezi hlavní z nich patří:

- přípravu může provádět pouze odborník detailně rozumějící způsobu simulace v systému JUTS,
- jedná se o časově velmi náročnou činnost,
- výsledek je značně náchylný k chybám, které se špatně detekují a odstraňují,
- jakékoliv změny mapy jsou velmi obtížné.

Z těchto důvodů bylo rozhodnuto o vývoji grafického editoru, který by přípravu simulačních map maximálně ulehčoval a odstranil všechny výše popsané problémy „ruční“ přípravy. Jeho ovládání musí být jednoduché a intuitivní, aby jej mohl používat i nezaškolený pracovník. Dále by editor měl vykazovat jistý stupeň „intelligence“, tj. provést automatizovaně všechny činnosti, které provést lze. Úkolem grafického editoru naopak není jakkoliv vizualizovat výsledky simulace.

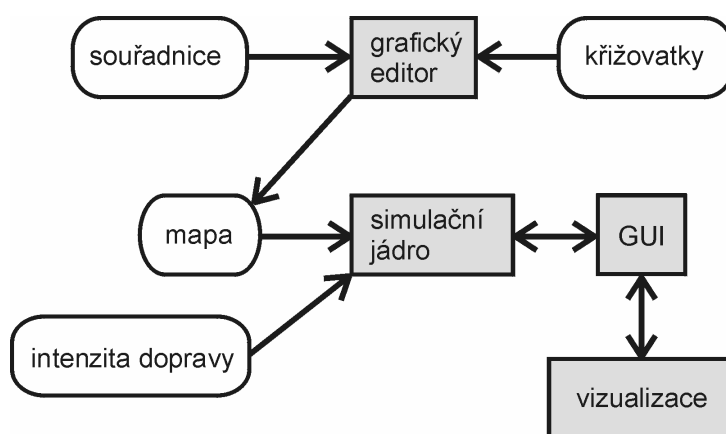
Pomocí grafického editoru by mělo být možné připravit simulační mapu za dobu v řádu desítek minut, zatímco „ruční“ příprava je otázkou řádově dnů.

## 2 Systém JUTS

Stejně jako v jiných větších městech, není ani v Plzni ideální dopravní situace, nejen v úzkých uličkách historického centra, ale také na několikaproudových magistrálách i na běžných komunikacích. Systém pro dopravní simulaci může být velmi užitečným nástrojem při předcházení dopravním zácpám, při zjišťování možných problémů při rekonstrukcích nebo při zjišťování, jak by se v dané oblasti změnila dopravní situace, kdyby byla postavena, nebo naopak zrušena paralelní komunikace. Pouhou změnou nastavených hodnot lze měnit parametry dopravní situace a silničního provozu. Uživatel může snadno získat pouze statistiky ze simulované oblasti (tzv. offline simulace) nebo celou simulaci detailně sledovat (online simulace).

Úsek koncepce a dopravního inženýrství Správy veřejného statku města Plzně [URL2] využívá pro simulaci dopravy v Plzni systém založený na makroskopické simulaci. Pro získání hodnot hustoty dopravy vychází ze zalidnění jednotlivých částí města a z umístění obchodních center a jiných oblastí se zvýšeným pohybem obyvatel. Protože dosud neexistuje žádný systém simulace dopravy na mikroskopické úrovni (tzn. na úrovni simulace jednotlivých vozidel), který by mělo město k dispozici, rozvinula se spolupráce na projektu JUTS (Java Urban Traffic Simulator). Díky této spolupráci jsou k dispozici data potřebná pro kompletní simulaci malé části města Plzně, jedná se o silniční okruh v plzeňské městské části Lochotín, zahrnující pět světelných křižovatek.

Na začátku vytváření JUTS projektu byl vytvořen návrh celého simulátoru a jeho mikroskopického modelu. Celý návrh je podrobně popsán v [Hart1]. V průběhu postupného vytváření simulátoru a následně rozšiřující se spolupráce s městem se celý projekt začal rozrůstat, vznikla řada subproblémů, systém bylo nutné rozdělit na několik částí a na celém projektu se začal podílet několikačlenný tým. V současné době se tedy systém skládá z několika základních částí zobrazených na obrázku 2.1.



Obr. 2.1: Struktura systému JUTS.

Hlavní část celého simulátoru tvoří simulační jádro [Hart2], které načte mapy spolu se všemi potřebnými daty udávajícími intenzitu provozu a zajišťuje samotnou simulaci dopravy. Na simulační jádro je navázáno grafické uživatelské rozhraní umožňující ovládání simulace a její vizualizaci. Jak již bylo zmíněno, do simulátoru je nutné dodat



potřebná vstupní data. Jedná se především o mapy, které vytvářejí prostředí pro simulovanou oblast, a o data udávající intenzitu provozu. Vstupní data je nutné simulátoru dodat v odpovídající podobě (XML formát), je tedy nutné jejich předzpracování. To je prováděno v částech systému oddělených od samotného simulátoru.

Hodnoty intenzity silničního provozu jsou měřeny pomocí detektorů (měřicích smyček), které jsou umístěny v jednotlivých jízdních pružích každé světelné křižovatky. Detektory zaznamenávají počet vozidel, která jízdním pruhem projedou, a tyto počty jsou mimo jiné agregovány do patnáctiminutových intervalů a ukládány do centrálního počítače dopravní ústředny. Data tedy obsahují hodnoty naměřené jednotlivými detektory v konkrétních časových úsecích a poskytují tak systému poměrně přesnou informaci o počtech vozidel projíždějících v danou dobu určitým územím. V současné době má systém k dispozici údaje za celý kalendářní rok 2004. Zpracováním těchto údajů se zabývá jiná práce.

Stejně jako data s hodnotami udávajícími intenzitu provozu je nutné předzpracovat i poskytnutá geografická data a vytvořit z nich mapy oblastí, ve které má simulace probíhat. Tyto mapy je možné vytvořit ručně – přímou editací XML souborů. Protože struktura mapy je ale poměrně složitá, vyžaduje její neautomatizované vytváření velké množství práce a času. Proto je v systému, jako další z jeho částí, vytvářen grafický editor umožňující pohodlnou a poměrně časově nenáročnou poloautomatickou přípravu mapy pro simulační systém.

Celý proces simulace může být prováděn online, tzn. simulace je vizualizována během celého svého průběhu, nebo offline, kdy je simulace spuštěna bez vizualizace a poté lze pracovat se získanými výslednými statistikami a údaji. Podrobnější popis celého systému JUTS je možné najít také v [Hart3].

## 2.1 Editor

Úkolem editoru je převádět načtené geografické souřadnice bodů udávajících možné pozice křižovatek (tzv. uzlových bodů) a souřadnice oblastí, ve kterých jsou umístěny silnice, popř. jejich části (tzv. segmenty silnic), do mapy reprezentované v podobě XML souboru. Celý převod je umožněn jednoduchým klikacím způsobem, editor poloautomaticky generuje výslednou mapu a umožňuje uživateli její základní úpravy.

Přestože existence editoru k samotné simulaci není nezbytná, je tento způsob grafické podpory simulace velmi důležitý. Přípravu mapy pro simulátor je možné realizovat ručně, ale její vytváření i pro nepřilíš složité silniční sítě je velmi náročné. Nejedná se pouze o přepočítání souřadnic jízdních pruhů nebo hranic křižovatky, ale také o zajištění vzájemné návaznosti a logického propojení všech elementů mapy, vytvoření generátorů a terminátorů a dalších prvků pro zajištění generování vozidel do sítě a jejich odebírání z ní a plynulé napojení všech částí mapy. Vše musí být provedeno tak, aby byla přesně dodržena požadovaná struktura XML souborů – grafické i simulační části mapy.

Editor pracuje s několika základními elementy mapy jako jsou silnice, jízdní pruh, křižovatka, buňka v křižovatce, propojovací místo, generátor, terminátor. Významy jednotlivých elementů mapy budou vysvětlovány průběžně při popisu návrhu a funkčnosti editoru.

## 2.2 Formáty souborů

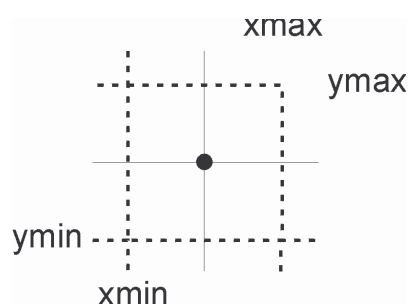
V celém projektu hrají velmi důležitou roli formáty vstupních a výstupních souborů. Vzhledem k tomu, že všechny součásti systému JUTS mezi sebou navzájem spolupracují, je výměna dat ve formě souborů nezbytná. Jako základní formát dat pro výměnu byl zvolen v současné době poměrně rozšířený formát XML. Výhodnost jeho použití je i v kvalitní podpoře zpracování XML dokumentů programovými prostředky jazyka Java, ve kterém jsou jednotlivé části systému implementovány.

Protože soubory slouží pro výměnu dat mezi navzájem oddělenými částmi systému, bylo nutné určit jejich strukturu hned na začátku příprav a návrhů jednotlivých částí. Zvolená struktura pak již zůstala, kromě několika drobných úprav, víceméně nezměněna.

Většina souborů používaných v celém systému JUTS tedy využívá XML formát. Jinak je tomu však u souborů poskytovaných městem. V tomto případě se jedná o běžné textové soubory obsahující data získaná buď z databáze GIS nebo z měřicí ústředny. Grafický editor pracuje se dvěma takovými soubory – souborem se souřadnicemi uzlových bodů (2.2.1) a souborem se souřadnicemi udávajícími ohraničující obdélníky pro jednotlivé segmenty silnic (2.2.2). Ostatní soubory, které editor načítá nebo generuje, jsou již v XML formátu. Jedná se o vstupní soubor s popisem křižovatek (2.2.3) a o dva výstupní soubory, které jsou editorem generovány – grafickou část mapy (2.2.4) a simulační část mapy (2.2.5).

### 2.2.1 Souřadnice uzlových bodů

Jedná se o vstupní soubor, který obsahuje souřadnice jednotlivých uzlových bodů. Uzlový bod je místo tvořící křižovatku, spojení dvou sousedních segmentů, nebo počáteční, popř. koncový bod segmentu. Každý řádek souboru obsahuje data pro jeden uzlový bod mapy. Pro jednotlivé body jsou zadávány souřadnice udávající značku pro hranici, kterou jsou uzlové body v databázi GIS města Plzně zobrazovány. Konečná pozice uzlového bodu je tedy dána jako střed čtverce zadaného souřadnicemi v souboru (viz obr. 2.2) – bod uprostřed naznačuje skutečnou pozici uzlového bodu. Další hodnotou patřící k jednotlivým uzlovým bodům je identifikátor. Jedná se o identifikátor používaný v rámci celé databáze města. Grafický editor ani jiná součást systému JUTS s těmito hodnotami nepracuje.



Obr. 2.2: Graficky znázorněný význam jednotlivých souřadnic pro jeden uzlový bod.

Souřadnice bodů jsou zadávány v tzv. Křovákově souřadném systému, neboli S-JTSK (Systém jednotné trigonometrické sítě katastrální) [Vever], [URL1]. Jedná se o kartografické zobrazení používané na území České republiky. V praxi to znamená, že

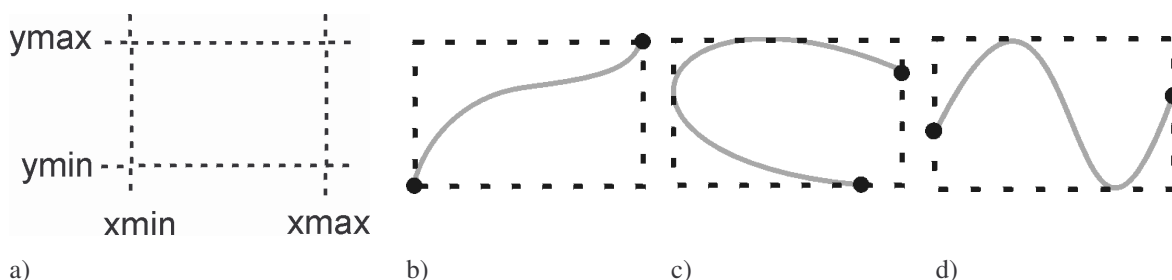
čísla udávající jednotlivé souřadnice jsou vztaženy k počátku této souřadné soustavy, kterou je místo poblíž města Talin. Příklad části souboru s několika definovanými uzlovými body je uveden na obrázku 2.3. Údaje jsou uvedeny v milimetrové přesnosti.

xmin	xmax	ymin	ymax	id_kriz
-822264282	-822260273	-1066020104	-1066016095	1839
-822319686	-822315677	-1065982446	-1065978437	519
-822143380	-822139371	-1066058842	-1066054833	1827

Obr. 2.3: Příklad části souboru s několika definovanými uzlovými body.

## 2.2.2 Souřadnice segmentů silnic

Jedná se o druhý textový soubor, který je do grafického editoru načítán. Soubor se souřadnicemi segmentů silnic má podobnou strukturu jako předchozí soubor, obsahující souřadnice uzlových bodů. Každá řádka obsahuje souřadnice minimálního a maximálního bodu. Tyto body udávají ohraničující obdélník jednoho segmentu silnice. Konkrétní tvar segmentu uvnitř ohraničujícího obdélníka není nijak definovaný, koncové body segmentu leží na hranici obdélníka. Na obrázku 2.4 a) je naznačen význam souřadnic vzhledem k ohraničujícímu obdélníku, na obrázcích b) až d) je naznačeno, jak mohou vypadat segmenty popisované v jednotlivých řádkách. Přesný tvar segmentu není v žádné formě obsažen ve vstupních datech.



Obr. 2.4: a) Graficky znázorněný význam jednotlivých souřadnic pro jeden segment silnice; b) až d) možné tvary segmentů silnic.

Jak je vidět na obrázku 2.5, každý řádek obsahuje řadu dalších informací o jednotlivých úsecích. Jedná se především o identifikátory sloužící k propojení segmentů s ostatními elementy v databázi města. Tyto identifikátory jsou však pro účely systému JUTS nepoužitelné, protože nesplňují potřebná kritéria – konkrétní popis viz kapitola 2.2.5, stejně jako ostatní položky popisující segmenty. Tyto hodnoty grafický editor ukládá (stejně jako identifikátory uzlových bodů z předchozího souboru), ale dále s nimi nepracuje, ani je nezahrnuje do výstupních popisů mapy.

XMIN	XMAX	YMIN	YMAX	L_PARAM	RC	ID_USEK	CC_ULICE
-822661442	-822495817	-1066512390	-1066255790	305410	S III. tř.	894	5921
-822325240	-822296720	-1066459950	-1066415304	52978	MK III. tř.	100	9899
-822557189	-822449246	-1066450958	-1066283694	199070	MK III. tř.	61	5921
-822196359	-822183002	-1066456996	-1066371313	87399	MK III. tř.	96	9899

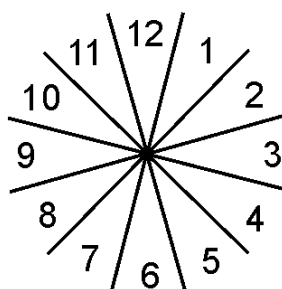
Obr. 2.5: Příklad části souboru s několika definovanými segmenty silnic. Pro nedostatek místa byly oproti skutečnému souboru použity zkratky S = silnice, MK = místní komunikace.

### 2.2.3 Popis křižovatek

Poslední vstupní soubor již narozdíl od předchozích dvou využívá formát XML. Tento soubor obsahuje popis reálných plzeňských křižovatek a je využíván v okamžiku mapování křižovatek na uzlové body silniční sítě, kdy jsou uzlovým bodům představujícím křižovatky přiřazovány konkrétní existující křižovatky. Aby bylo možné danou křižovatku jednoznačně určit, musí mít každá svůj jedinečný identifikátor (jeho konkrétní umístění v popisu je uvedeno níže).

Kořenovým elementem uložených křižovatek je element `<krizovatky>` (viz obr. 2.9). Ten v sobě obsahuje popisy jednotlivých křižovatek. Každá křižovatka `<krizovatka>` má dva atributy – číslo `<cislo>` odpovídající číslování křižovatek v centrále dopravního úseku města Plzně a popis `<popisKrizovatky>` udávající jména křížících se ulic, popř. ještě zažité jméno křižovatky. Uvnitř každé křižovatky jsou popisovány její další vlastnosti a také její grafická struktura. Je zde umístěn i identifikátor `<idJUTSKrizovatky>`, který křižovatku jednoznačně určuje. Připravená značka `<soucastOblasti>` pro číslo `<cislo>` a popis oblasti `<popisOblasti>`, ve které je křižovatka umístěna, zatím není využita, hodnoty těchto atributů budou do popisu křižovatek doplněny v budoucnu.

Každá křižovatka je nadále určena svými rameny. Pro každé z nich je v popisu určený samostatný tag `<ramenoKrizovatky>` s atributem `<jmenoUlice>` obsahujícím jméno ulice, kterou popisované rameno křižovatky prochází. Na začátku popisu každého ramena je zadána jeho orientace `<orientace>`. Ta může obsahovat hodnotu od 1 do 12 v závislosti na natočení ramena křižovatky. Tato hodnota odpovídá umístění těchto číslic na hodinách (viz obr. 2.6). Samozřejmě, že takové zadávání orientace křižovatky je poměrně nepřesné a pro uchování v XML souboru by bylo možné ukládat relativně přesnou hodnotu natočení segmentu. Zjišťování přesného úhlu by ale, vzhledem k tomu, že jsou popisy křižovatek vytvářeny ručně, přípravu souborů značně zpomalovalo a daná hodnota by i tak nemusela být zcela přesná. Hrubé (ale rychlé) zadání orientace pro tento účel plně postačuje, přesné zjištění úhlu natočení ramena je určeno později podle úhlu natočení segmentu silnice, ke kterému je rameno křižovatky přiřazeno. Následujícím tagem hned za orientací ramena křižovatky je umístěn odkaz na nejbližší křižovatku `<navazujiciKrizovatka>`, která je také popsána.



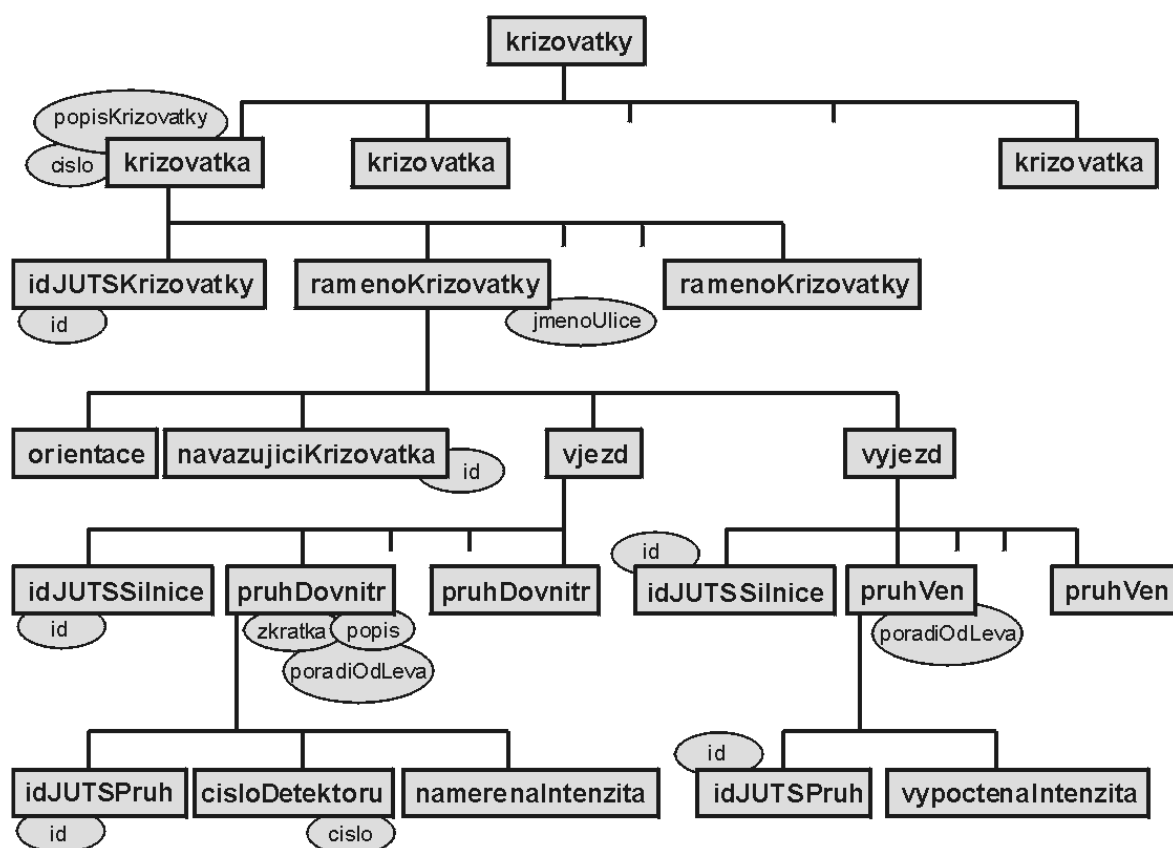
Obr. 2.6: Hodnoty pro hrubou orientaci popisu ramen křižovatky.

Každé rameno obsahuje dvě části – pro každý směr jízdy jednu. Ty kromě identifikátoru `<idJUTSSilnice>` obsahují popisy pro jednotlivé jízdní pruhy, které v silnici leží. Silnice ve směru vjezdu vozidel do křižovatky `<vjezd>` v sobě zahrnuje

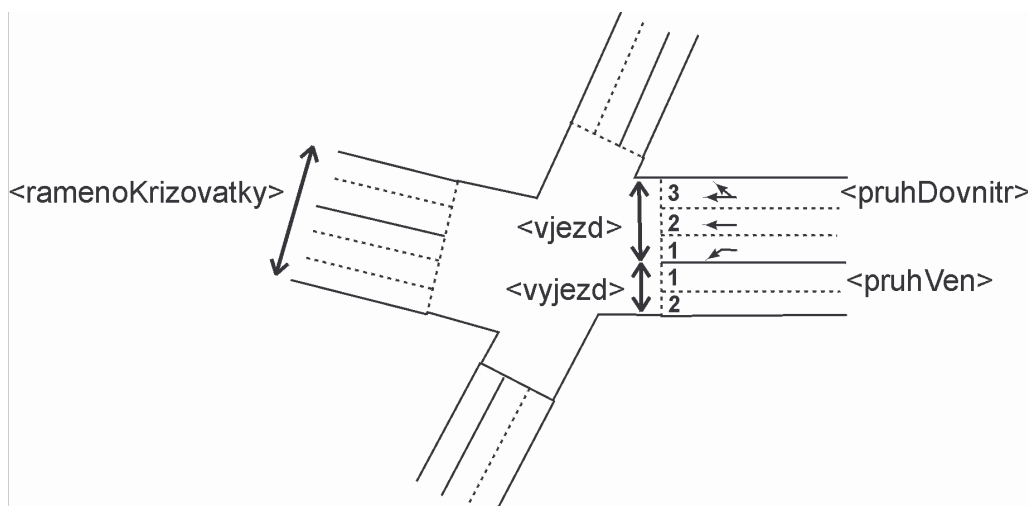
popisy všech jízdnic pruhů `<pruhDovnitř>`, kterými vozidla vjíždí do křižovatky. Každý jízdnic pruh má jako své atributy určené možné směry jízdy a to zkratkou `<zkratka>` (L = vlevo, R = rovně, P = vpravo, včetně všech libovolných kombinací), nebo plným popisem směru `<popis>`. Poslední atribut určuje pořadí jízdnic pruhu zleva ve směru jízdy `<poradiOdLeva>`. Pruh je pak popsán identifikátorem `<idJUTSPruh>`, číslem detektoru `<cisloDetektoru>`, který je v daném pruhu umístěn pod povrchem vozovky, a vzorcem pro výpočet intenzity průjezdu vozidel z detektorů v křižovatce `<namerenaIntenzita>`.

Popis jízdnic pruhů v silnici ve směru z křižovatky `<vyjezd>` je oproti vjezdu výrazně jednodušší. Každý jízdnic pruh `<pruhVen>` obsahuje pouze atribut pro určení pořadí jízdnic pruhu zleva ve směru jízdy `<poradiOdLeva>`, tag pro identifikátor jízdnic pruhu `<idJUTSPruh>` a vzorec pro výpočet intenzity průjezdu vozidel jízdnic pruhem, uložený ve značce `<vypoctenaIntenzita>`.

Příklad základní struktury XML dokumentu je ve formě stromu naznačený na obrázku 2.7. Jednotlivé značky jsou zobrazeny v obdélníčkách, jejich atributy v přilehlých elipsách. Význam několika základních značek a atributů vzhledem k reálné křižovatce je ukázán na obrázku 2.8. Část XML souboru pro zobrazenou vzorovou křižovatku je k dispozici na obrázku 2.9. Podrobně je zde popsáno to rameno křižovatky, které je v obrázku označeno do největších podrobností.



Obr. 2.7: Základní struktura XML, popsána formou stromu.



Obr. 2.8: Vztah jednotlivých značek k reálné křižovatce.

```

<krizovatky>
  <krizovatka cislo="k333" popisKrizovatky="Studentská - Komenského">
    <idJUTSKrizovatky id="120000001" />
    <soucastOblasti cislo="" popisOblasti="" />
    <ramenoKrizovatky jmenoUlice="Studentská">
      <orientace>3</orientace>
      <navazujiciKrizovatka id="120000013" />
      <vyjezd>
        <idJUTSSilnice id="110000001" />
        <pruhDovnitř zkratka="L" popis="vlevo" poradíOdLeva="1">
          <idJUTSPruh id="112000001" />
          <cisloDetektoru cislo="7" />
          <namerenaIntenzita>91</namerenaIntenzita>
        </pruhDovnitř>
        <pruhDovnitř zkratka="R" popis="rovně" poradíOdLeva="2">
          ...
        </pruhDovnitř>
        <pruhDovnitř zkratka="RP" popis="rovně a vpravo" poradíOdLeva="3">
          ...
        </pruhDovnitř>
      </vyjezd>
      <vyjezd>
        <idJUTSSilnice id="110000002" />
        <pruhVen poradíOdLeva="1">
          <idJUTSPruh id="112000005" />
          <vypoctenaIntenzita />
        </pruhVen>
        <pruhVen poradíOdLeva="2">
          ...
        </pruhVen>
      </vyjezd>
    </ramenoKrizovatky>

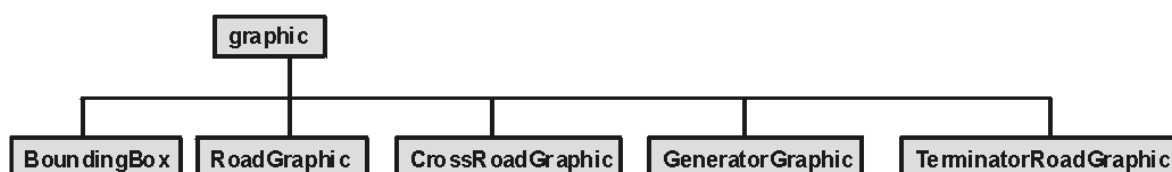
    <ramenoKrizovatky jmenoUlice="Kaznějovská"> ... </ramenoKrizovatky>
    <ramenoKrizovatky jmenoUlice="Studentská"> ... </ramenoKrizovatky>
    <ramenoKrizovatky jmenoUlice="Komenského"> ... </ramenoKrizovatky>
  </krizovatka>
</krizovatky>

```

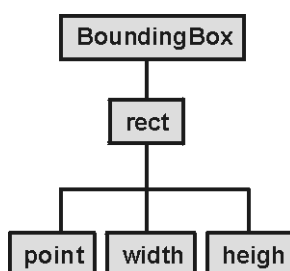
Obr. 2.9: Konkrétní příklad části XML souboru vztahující se ke křižovatce na obr. 2.8.

## 2.2.4 Grafická část simulační mapy

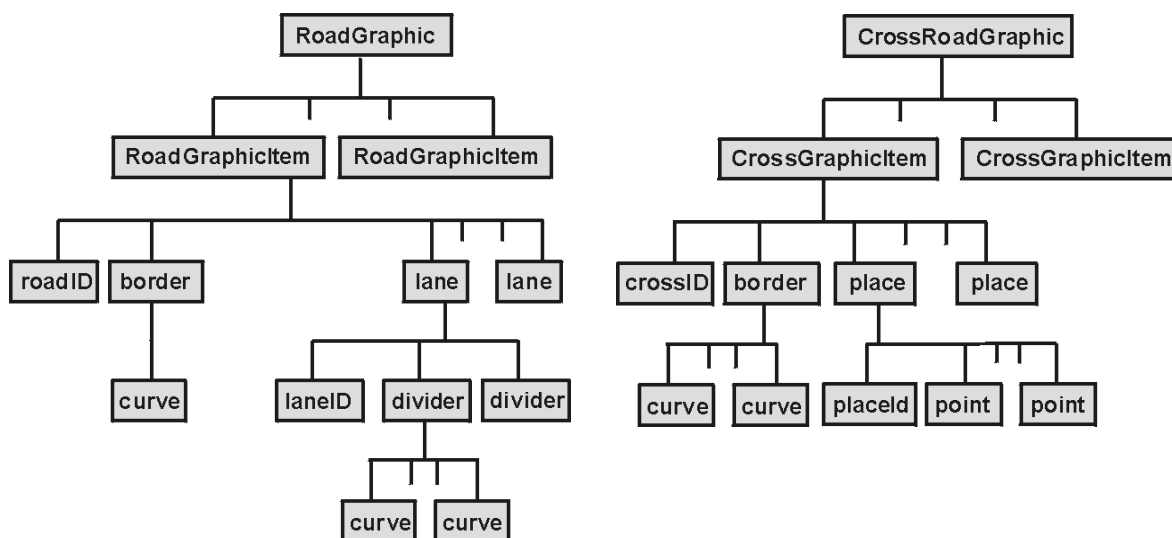
Tento soubor je prvním výstupním souborem editoru. Jedná se o grafickou část vytvořené mapy. Soubor v sobě ukládá souřadnice, umístění a rozměry ohraničujících oblastí apod., data slouží pro vykreslení mapy v simulátoru. Je zde popsán každý element, který se vykresluje, všechny popisované elementy mapy mají svůj jednoznačný identifikátor, pomocí kterého jsou napojené na druhou simulační část (popsaná v kapitole 2.2.5). Struktura popisovaného XML souboru je znázorněna na obrázcích 2.10 a) až f) ve formě stromu, rozděleného na několik částí. Pro rozdělení stromu jsem se rozhodla vzhledem k jeho velikosti a v závislosti na možnostech zobrazení.



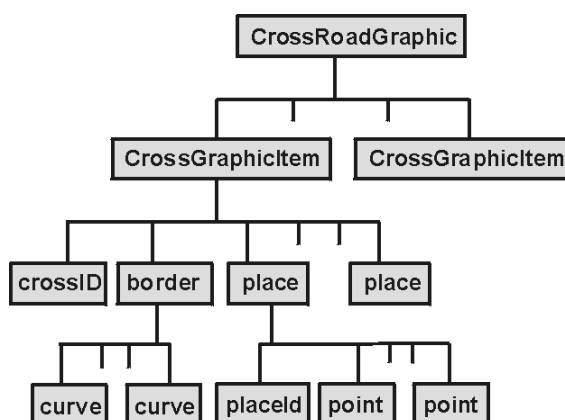
a)



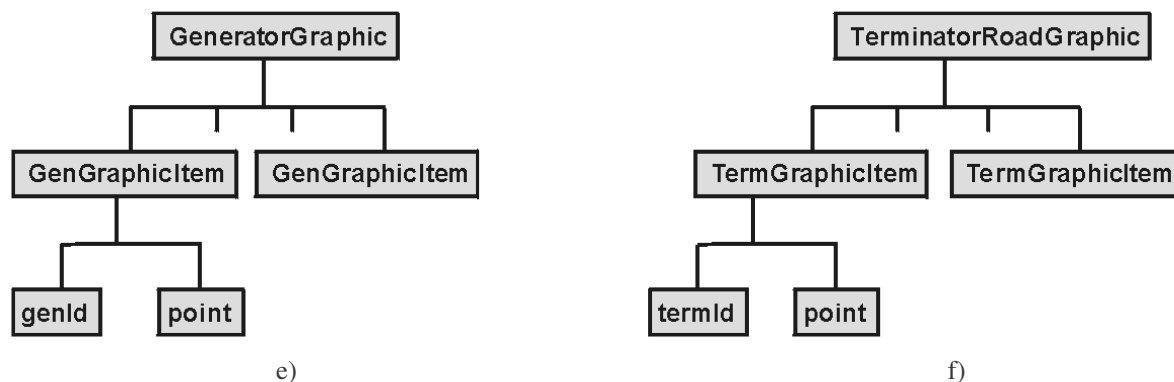
b)



c)



d)



Obr. 2.10: Struktura XML ve formě stromu: a) základní struktura; b) až f) podrobnější struktura jednotlivých částí.

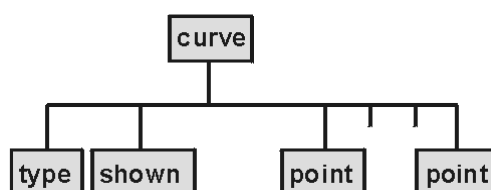
Kořenovým elementem je element `<graphics>`. Ten naznačuje, že se skutečně jedná o grafickou část mapy. Hlavní element v sobě zahrnuje určení ohraničující oblasti celé mapy `<BoundingBox>` a části pro popis jednotlivých skupin elementů mapy jako jsou: silnice `<RoadGraphic>`, křižovatky `<CrossRoadsGraphic>`, generátory `<GeneratorGraphic>`, terminátory `<TerminatorGraphic>`.

Ohraničující oblast celé mapy je určena obdélníkem `<rect>`. Ten je zadán bodem udávajícím levý horní roh `<point>`, šířkou `<width>` a výškou `<height>`, které udávají rozměr obdélníka. Značka `<point>` se v celém souboru používá velmi často. Jedná se o zadání souřadnic bodu. Její struktura je velmi jednoduchá, bod se skládá ze souřadnice x `<p_x>`, souřadnice y `<p_y>` a hodnoty udávající úroveň ve vertikálním směru `<p_h>`. Tato hodnota je určena pro řešení problému mimoúrovňového křížení apod. Nejčastější používanou hodnotou je 1, tzn. element se nachází na základní úrovni.

Další často používanou značkou je značka popisující křivku `<curve>`. Strom znázorňující její strukturu je vidět na obrázku 2.11. Křivka je určena typem `<type>` udávajícím její vlastnosti. Možné hodnoty a jejich význam jsou:

plná	0
přerušovaná	1
dvojitá	2
dvojitá, zleva přerušovaná	3
dvojitá, zprava přerušovaná	4

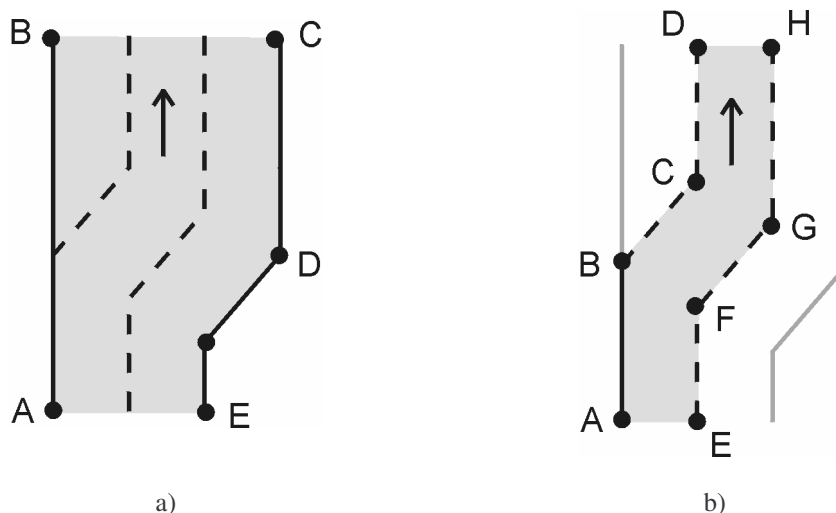
Vlastnosti čáry (zejména to, zda je možné ji přejíždět nebo ne) vyplývají z Pravidel silničního provozu. U každé křivky je značkou `<shown>` určeno, jestli má být při vykreslování zobrazována, nebo ne (0 = ne, 1 = ano). Po definici typu a zobrazování jsou v definici křivky uloženy body `<point>`, které ji tvoří. Element křivky je využíván pro popis hranice křižovatek, silnic, jízdních pruhů nebo pro určení oddělovačů jednotlivých jízdních pruhů.



Obr. 2.11: Struktura elementu popisujícího křivku.



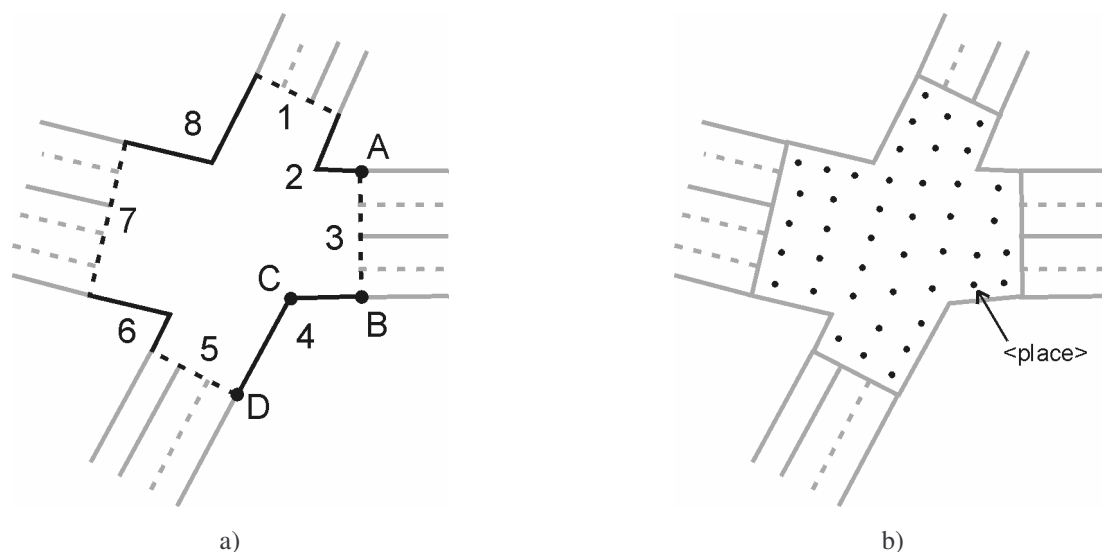
Jak již bylo zmíněno, grafická část mapy obsahuje popisy jednotlivých skupin elementů mapy. Jako první jsou v souboru uloženy popisy všech silnic. Každá silnice `<RoadGraphicItem>` je určena svým identifikátorem `<roadId>` a křivkou udávající hranici křižovatky `<border>`. Body jsou v křivce `<curve>` zadávány tak, jak je naznačeno na obrázku 2.12 a) v pořadí A, B, C, D a E, parametr pro zobrazování je u hraniční křivky nastaven na 0 (nezobrazovat). Každá silnice se skládá z několika jízdních pruhů `<lane>`. Jízdní pruh má svůj identifikátor `<laneId>` a levý a pravý oddělovač `<divider>`. Levý oddělovač je vždy uveden jako první. Každý oddělovač je tvořen jednou nebo více křivkami. Jestliže je celý oddělovač tvořen křivkou jednoho typu, stačí pro popis jediná křivka, pokud je nutné v průběhu oddělovače typ křivky měnit, je nutné k jeho určení několik křivek. Konkrétní příklad křivek v oddělovačích je uveden na obrázku 2.12 b). Levý oddělovač se skládá ze dvou křivek, první je plná a obsahuje body A, B, druhá je přerušovaná a tvoří ji body C a D. Pravý oddělovač je tvořen jednou křivkou určenou body E, F, G a H. Pro urychlení vykreslování v simulátoru je již přímo v mapě určeno, které oddělovače je nutné vykreslovat a které ne. Pokud to tedy okolnosti nevyžadují, je parametr vykreslování křivek levého oddělovače `<shown>` nastaven vždy na 0 (tzn. nevykreslovat). Výjimka platí vždy pouze pro levý oddělovač toho jízdního pruhu, který je nejvíce vlevo u jedné ze silnic ležících na stejném rameni křižovatky (to zajistí vykreslení dělicí čáry mezi silnicemi).



Obr. 2.12: Ukázka pořadí vkládání bodů do a) hranice silnice; b) křivek tvořících levý a pravý oddělovač jízdního pruhu.

V další části XML jsou popisovány křižovatky, které jsou v mapě obsažené. Každá křižovatka `<CrossGraphicItem>` je dána svým identifikátorem `<crossId>` a hranicí `<border>`. Hranice se skládá z několika křivek, ve kterých jsou body určující hranici ukládány tak, jak je naznačeno na obrázku 2.13 a). Čísla označují pořadí samotných křivek, křivka 3 obsahuje body v pořadí A, B a je typu 1 (přerušovaná), křivka 4 pak se skládá z bodů B, C, D a je typu 0 (plná). Parametr `<shown>` je pro všechny hraniční křivky křižovatky nastaven na 0, typ `<type>` jednotlivých křivek závisí na tom, zda se jedná o křivku hraničící s ramenem křižovatky (taková křivka je nastavená jako přerušovaná) nebo o část hranice, kterou není možné přejíždět (plná). Popis křižovatky obsahuje také popis všech buněk pro průjezd vozidel, které jsou v křižovatce nadefinovány. Buňka

`<place>` je určena svým identifikátorem `<placeId>` a bodem označujícím pozici, na které je umístěná `<point>`. Na obrázku 2.13 b) jsou naznačené možné pozice buněk definovaných v křižovatce.



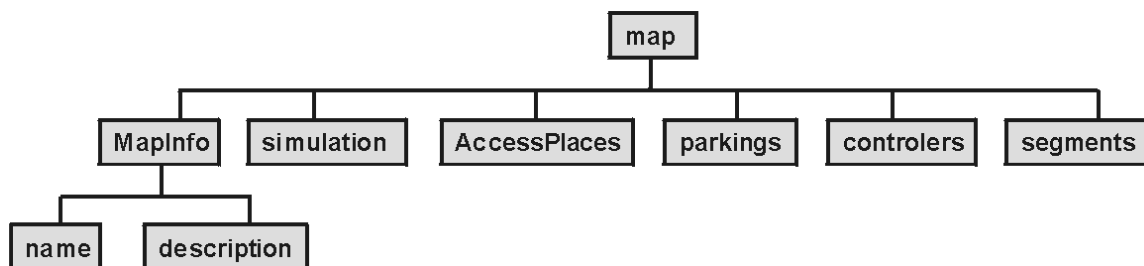
Obr. 2.13: Popis křižovatky: a) pořadí bodů vkládaných do hraničních křivek; b) možné pozice buněk v křižovatce definovaných pro průjezd vozidel křižovatkou.

Oblast pro popis generátorů obsahuje jednotlivé generátory obsažené v mapě `<GenGraphicItem>`. Každý generátor je určený identifikátorem `<genId>` a bodem, na kterém je umístěný `<point>`. Generátory mají za úkol emitovat vozidla do simulačního systému.

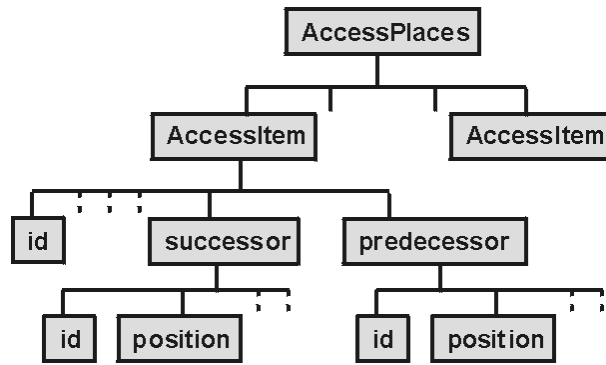
Stejnou strukturu jako generátory mají v popisu i všechny terminátory, v souboru se mírně liší jen používané značky XML. Každý terminátor `<TermGraphicItem>` má tedy svůj identifikátor `<termId>` a bod `<point>`. Přes terminátory vozidla opouštějí simulační systém, jsou v něm evidována a jejich objekty rušeny. Funkčnost generátorů a terminátorů zajišťuje simulační jádro, editor pouze popisuje jejich umístění.

## 2.2.5 Simulační část simulační mapy

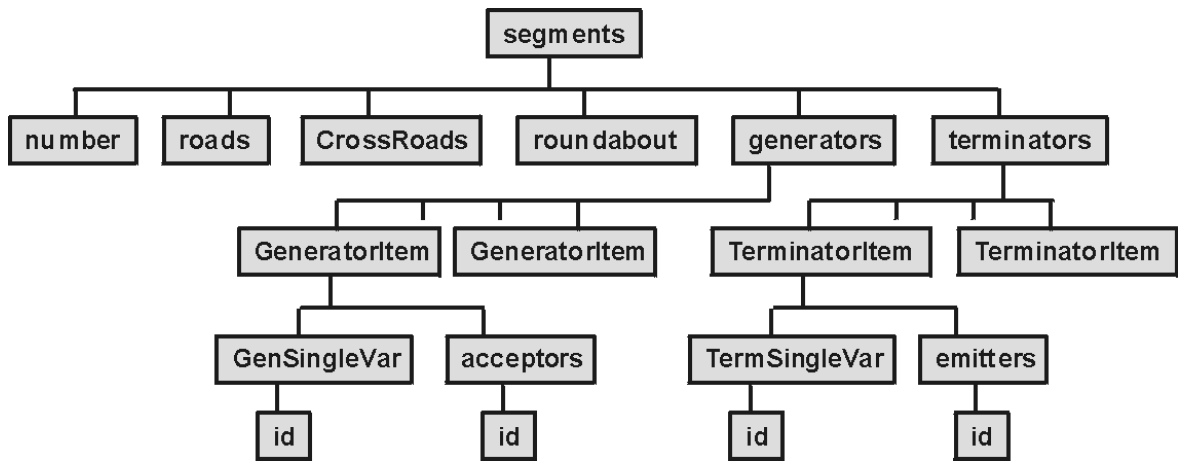
Posledním souborem, se kterým grafický editor pracuje, je druhá, simulační, část mapy, kterou editor vytváří v XML formátu. Tato struktura je opět znázorněna ve formě stromu, rozděleného na několik částí, na obrázcích 2.14. Jedná se o logickou strukturu uchovávající vzájemné návaznosti mezi jednotlivými částmi mapy. Tato část mapy neobsahuje žádné grafické informace, žádné souřadnice. Ty jsou uloženy v grafické části a jsou simulátoru dostupné přes identifikátory, které jsou editorem přiřazené jednotlivým elementům mapy.



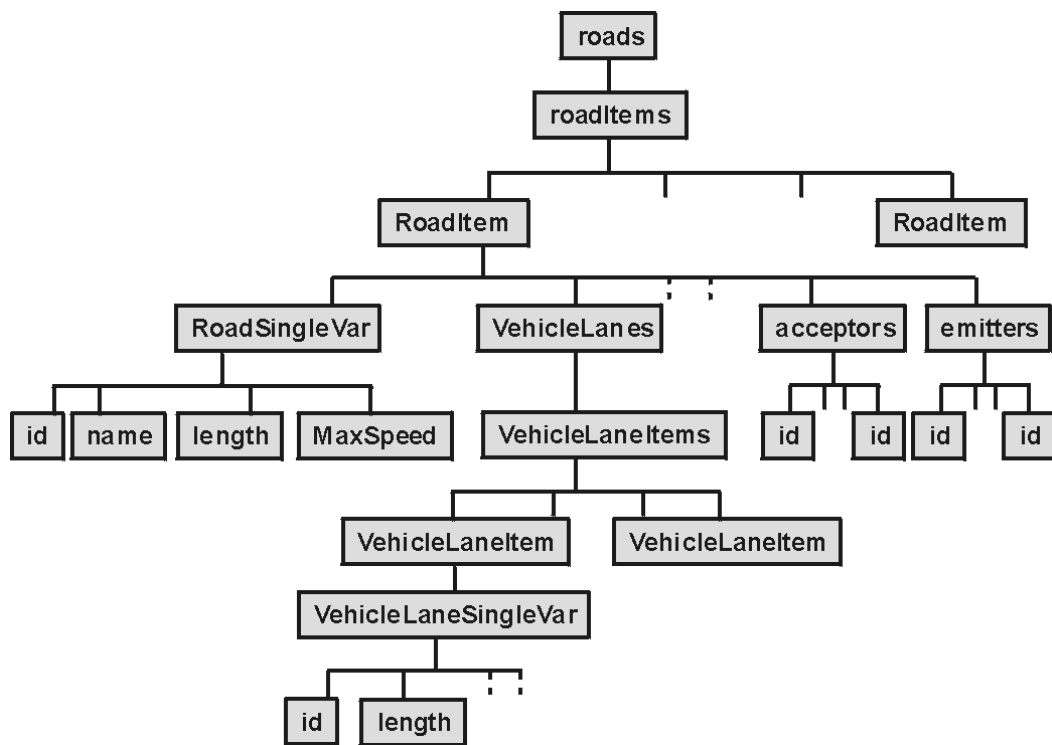
a)



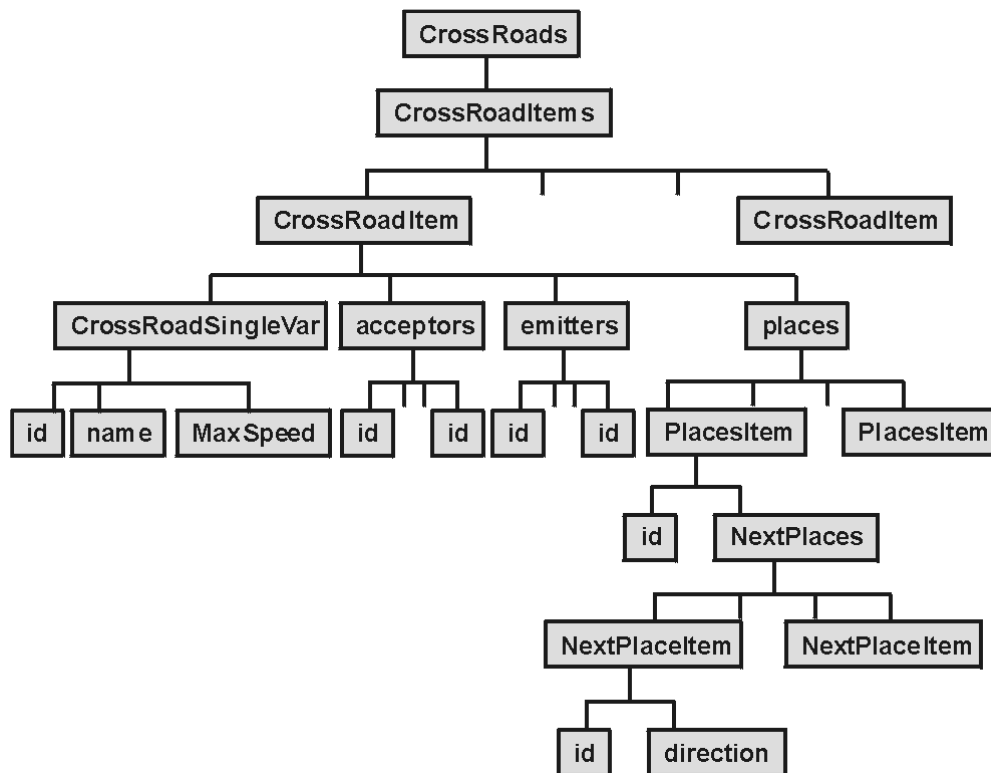
b)



c)



d)



e)

Obr. 2.14: Struktura XML ve formě stromu: a) základní struktura; b) až e) podrobnější struktura jednotlivých částí.

Celý systém přidělování identifikátorů má několik základních pravidel, která je při vytváření a přidělování identifikátorů třeba dodržovat:

- identifikátor elementu je v celé mapě jedinečný, tzn. není možné, aby se v jedné mapě vyskytovalo více elementů se stejným ID,
- identifikátor je vždy pouze číselný a skládá se vždy z devíti číslic,
- první tři číslice určují typ elementu, kterému je identifikátor přidělen, a to takto:

110	RoadItem	silnice
111	ParkLane	parkovací pruh
112	VehicleLaneItem	jízdní pruh
113	RailLaneItem	tramvajový pruh
120	CrossRoadItem	křižovatka
121	PlacesItem	buňka v křižovatce
130	RoundAboutItem	kruhový objezd
140	GeneratorItem	generátor
150	TerminatorItem	terminátor
160	ParkingItem	parkoviště
210	AccessItem	prvek pro propojení jízdního pruhu a křižovatky, popř. jízdního pruhu a generátoru nebo terminátoru.

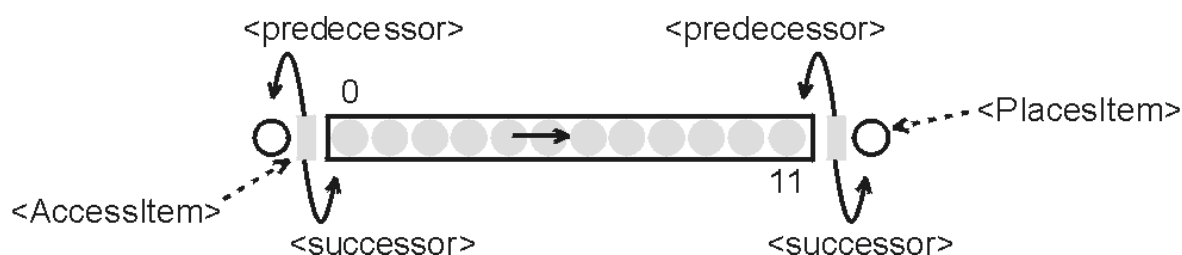
Systém může obsahovat i jiné identifikátory pro servery a řídicí elementy. Ty však nejsou vytvářeny grafickým editorem, ale jsou do mapy dodávány v jiných částech

systému JUTS, proto zde nebudou popisovány, stejně jako ty části souboru, které grafický editor sám nevytváří (např. `<simulation>`).

Kořenový element `<map>` v sobě obsahuje část pro uvedení informací o mapě `<MapInfo>`, která obsahuje jméno `<name>` a popis `<description>` mapy. Další částí XML souboru jsou informace pro simulaci `<simulation>`, dále pak popis propojovacích míst `<AccessPlaces>`, část pro parkoviště `<parkings>` a řídicí objekty `<controlers>`. S parkovišti ani řídicími objekty editor ani simulátor v současné době nepracují, budou zapracovány až v některé z následujících verzí. V poslední části souboru je vložena logická struktura všech segmentů mapy `<segments>`. Podrobněji je tedy zapotřebí popsat propojovací místa a strukturu tvořenou segmenty mapy.

Popis propojovacích míst `<AccessPlaces>` se skládá z jednotlivých položek. Každé propojovací místo `<AccessItem>` má svůj identifikátor `<id>` a kromě řady většinou prázdným tagů připravených pro rozšíření funkčnosti, jsou pro popis propojení důležité ještě elementy: `<successor>` – předchůdce a `<predecessor>` - následník. Jak následník, tak předchůdce jsou určeny identifikátorem `<id>` a pozicí `<position>`. Následníkem nebo předchůdcem propojovacího místa může být jízdní pruh nebo místo v křižovatce. Jako identifikátor předchůdce a následníka je tedy vždy uložen identifikátor příslušného elementu (jízdního pruhu nebo buňky v křižovatce). Pozice buňky v křižovatce je vždy rovna nule, jinak je tomu však v případě jízdního pruhu (jak je naznačeno na obrázku 2.15).

V závislosti na délce jízdního pruhu je možné ho rozdělit na určitý počet úseků po 2,5 m. Délka 2,5 m je základní velikost buňky celulárního automatu a je odvozena z platných norem ministerstva dopravy [Hart1]. Pokud číslo při dělení délky pruhu velikostí úseku nevyjde jako celá hodnota, je použito nejbližší nižší přirozené číslo. Získané úseky lze považovat za pozice v jízdním pruhu. Jestliže budeme tyto pozice číslovat od nuly ve směru jízdy, získáme tak na začátku jízdního pruhu pozici 0 a na jeho konci pozici  $n-1$ , kde  $n$  je právě zmiňovaný počet úseků. Takto získaná hodnota pozice je použita pro element `<position>` v předchůdci, popř. v následníkovi propojovacího místa. Záleží pouze na tom, kde se konkrétní popisované propojovací místo nachází. Ve struktuře XML je opět uvedeno několik dalších vlastností, ty však nijak nezávisí na vlastním grafickém editoru. Při generování jsou buď zadány jako konstantní hodnota, nebo jsou ponechány prázdné.

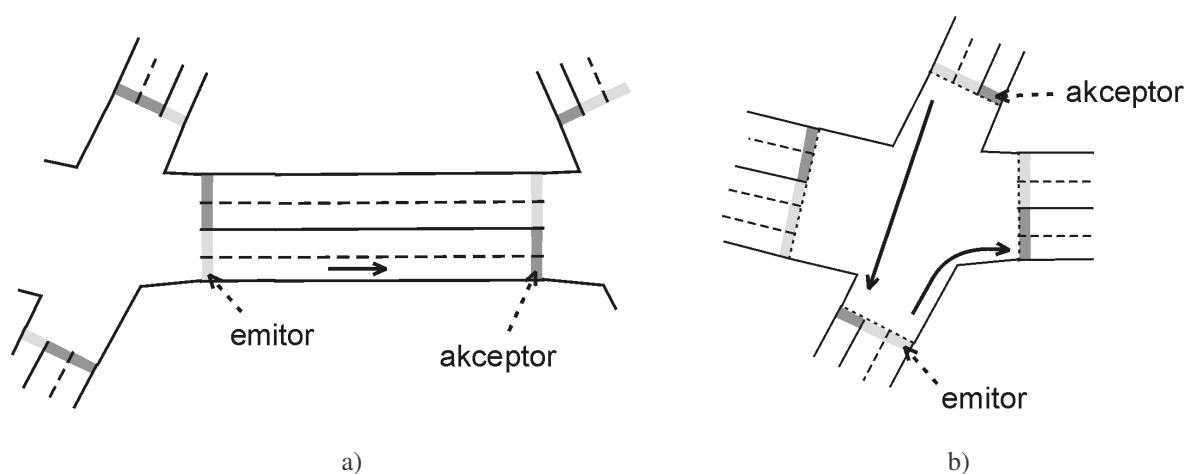


Obr. 2.15: Způsob získání hodnoty pro element `<position>` v popisu propojovacího místa.

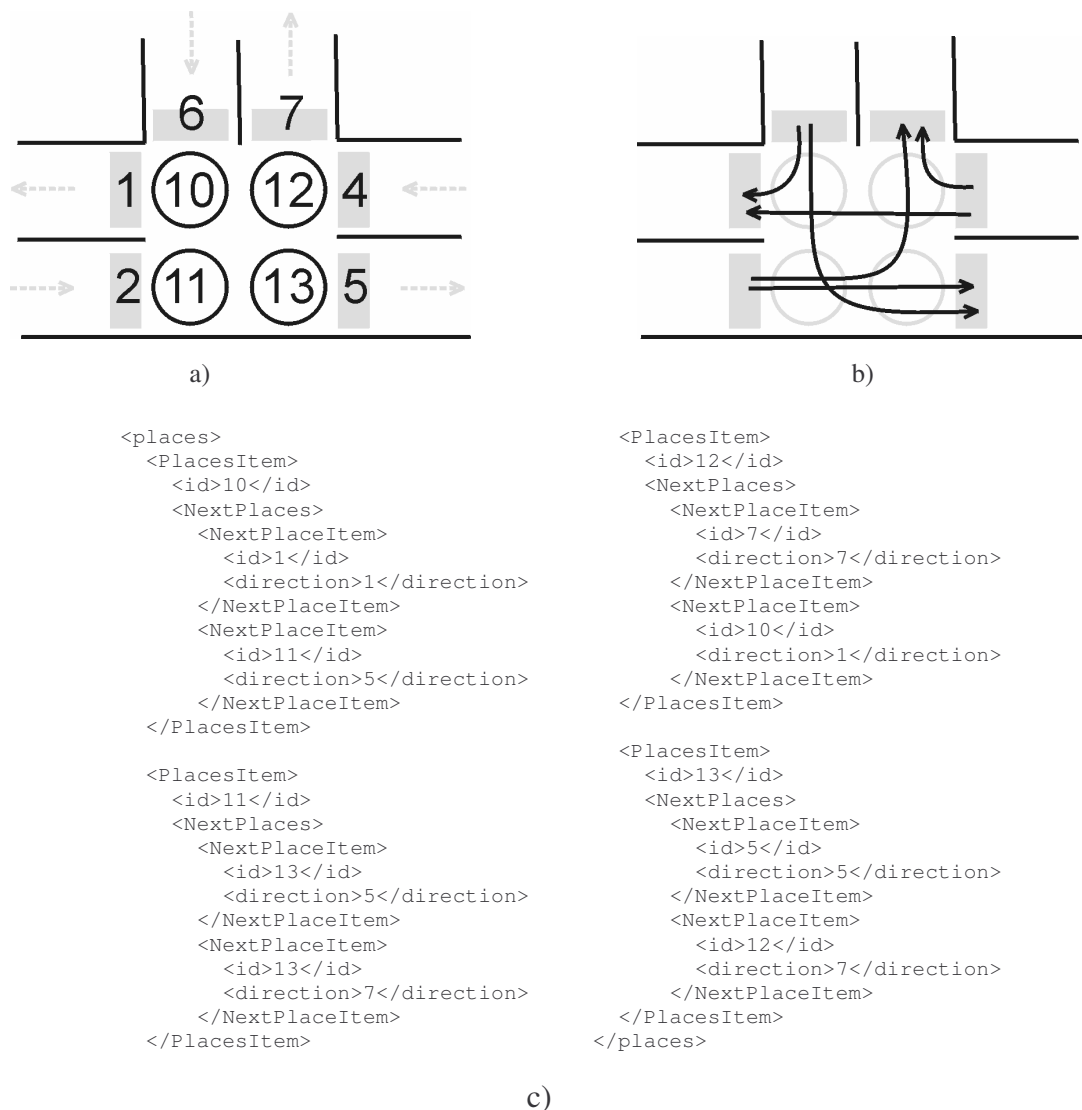
Část pro popis segmentů v mapě má oproti předchozí části výrazně složitější strukturu. Ještě před samotným popisem segmentů je uveden jejich celkový počet `<number>`. Poté následuje definice silnic `<roads>`, křižovatek `<CrossRoads>`, kruhových objezdů `<roundabout>`, generátorů `<generators>` a terminátorů `<terminators>`.

Nejdříve jsou v souboru popsány jednotlivé silnice `<RoadItem>`. Na začátku je silnici zadáno několik obecných informací jako jsou identifikátor `<id>`, jméno `<name>`, délka `<length>`, udávající počet 2,5 metrových úseků, které by se do silnice vešly, a maximální možná rychlost `<MaxSpeed>`. Po obecných informacích následují definice jízdních pruhů `<VehicleLanes>`. Za položky, které stojí za zmínku u jednotlivých jízdních pruhů `<VehicleLaneItem>`, považují identifikátor jízdního pruhu `<id>` a jeho délku `<length>` (počet 2,5 metrových úseků v jízdním pruhu). S dalšími položkami pro popis jízdních pruhů grafický editor nepracuje. Pro každou silnici jsou v souboru dále připravené elementy pro popis řady dalších vlastností, např. tramvajové nebo parkovací pruhy apod. Podstatnými v popisu každé silnice jsou ještě seznamy všech akceptorů `<acceptors>` a emitorů `<emitters>`, přiléhající k jednotlivým jízdním pruhům silnice. Za akceptory jsou považována taková propojovací místa, která akceptují vozidla ze silnice, tzn. vozidla ze silnice do nich vjíždí. Emitory jsou naopak ta propojovací místa, která vozidla do silnice emitují. Emitory i akceptory jsou určeny svými identifikátory `<id>`. Umístění akceptorů a emitorů vzhledem k jízdnímu pruhu je názorně ukázáno na obrázku 2.16 a).

Popis každé křižovatky `<CrossRoadItem>` začíná, stejně jako popis jednotlivých silnic, obecnými informacemi `<CrossRoadSingleVar>` jako je identifikátor `<id>`, jméno `<name>` a maximální rychlost `<MaxSpeed>`. Stejně jako silnice obsahuje také seznam akceptorů `<acceptors>` a emitorů `<emitters>`. V tomto případě je ale význam propojovacích míst brán z pohledu křižovatky a nikoli jízdního pruhu (viz obrázek 2.16 b). Z toho logicky vyplývá, že pokud bylo propojovací místo z pohledu silnice akceptorem (vozidla z jízdního pruhu do něho vjížděla), z pohledu křižovatky je totéž místo bráno jako emitor (vozidla z něho do křižovatky vyjíždějí). V každé křižovatce je v mapě definována řada směrů pro průjezd danou křižovatkou. Tyto směry jsou určeny pomocí tzv. buněk v křižovatce `<places>`. Každá buňka `<PlaceItem>` má stejně jako ostatní elementy mapy svůj jedinečný identifikátor `<id>`. Zároveň je pro buňku uložen seznam všech možných směrů `<NextPlaces>`, kterými lze z buňky pokračovat v jízdě křižovatkou. Každá položka seznamu `<NextPlaceItem>` se skládá z identifikátoru následujícího prvku mapy `<id>` a směru (přesněji řečeno akceptoru křižovatky v daném směru) `<direction>`. Příklad definice a následný popis situace v XML souboru je znázorněn na obrázcích 2.17. Jako následující prvek pro buňku v křižovatce může být dáno buď jiná buňka, popř. některý z akceptorů. V takovém případě se hodnoty zadané pro `<id>` a `<direction>` shodují.



Obr. 2.16: Rozdělení propojovacích míst na emitory a akceptory z pohledu a) silnice; b) křižovatky.



Obr. 2.17: Ukázka popisu navazujících buněk v křižovatce: a) číslování elementů; b) definice směru jízdy; c) příklad XML.

Část popisu kruhových objezdů je v současné době prázdná, doplnění systému pro kruhové objezdy je určeno pro další rozšíření.

Popis generátorů `<GeneratorItem>` a terminátorů `<TerminatorItem>` má podobnou strukturu. Každý je určen identifikátorem `<id>`. Ten je v případě generátorů zanořen do elementu `<GenSingleVar>`, u terminátorů pak `<TermSingleVar>`. Pro generátory je dále značkou `<acceptors>` určen identifikátor `<id>` propojovacího místa `<acceptors>`, přes který je generátor připojen k jízdnímu pruhu. Totéž propojení existuje u terminátorů, liší se pouze použitá značka, místo akceptoru je propojovací místo označeno jako emitor `<emitters>`.

V popisu struktury souboru nejsou uvedeny všechny značky, které se mohou v XML objevit. Většinou se jedná o prázdné elementy nebo elementy s konstantně zadanou hodnotou, na kterou nemá grafický editor žádný vliv. Popis všech těchto značek by nijak neprospěl zpřehlednění a zlepšení pochopení struktury souboru z pohledu grafického

editoru. I když byl formát souboru dohodnutý před vlastním zahájením tvorby jednotlivých částí systému, stále v něm dochází k drobným změnám a úpravám. Struktura souboru je tedy ještě stále v pracovní verzi. Před úplným dokončením souboru bude ještě zapotřebí odstranit nadbytečné elementy, které byly na začátku navrženy a v průběhu vývoje se přestaly používat. Bylo by také vhodné ujednotit názvy jednotlivých značek, zejména používání malých a velkých písmen. Všechny tyto úkony upravující strukturu souboru však vzhledem k tomu, že je soubor načítán simulačním jádrem, závisí na implementaci simulátoru. Grafický editor nemůže strukturu XML souboru nijak ovlivnit, podléhá požadavkům simulátoru.



## 3 GIS systémy

### 3.1 Existující prostředky

Pro práci s geografickými daty je určeno velké množství produktů. Podle [Cajt] se jen v ČR tvorbou GIS (geografických informačních systémů) pro veřejnou správu a samosprávu zabývá přes 20 firem (např. Bentley, Intergraph, ESRI, Foresta, Gepro, ...). Základní funkce GIS systémů jsou ve většině případů velmi podobné. Systémy se liší ve velikosti území, se kterým je možné pracovat, některé jsou použitelné v obecnějším smyslu, jiné jsou poměrně úzce specializované na konkrétní oblast lidské činnosti – např. lesní hospodářství nebo zpracování konkrétních adresních míst.

V závislosti na obecnosti a rozsahu systému uvádí [Cajt] možnost dělení GIS do dvou základních skupin. Do první skupiny patří velké obecné GIS, které uživatelům nabízejí značné množství funkcí a ke kterým se přidáváním dalších aplikací doplňuje další funkčnost. Tyto systémy neomezují uživatele svou úzkou specializací a nižší funkčností, jsou však poměrně finančně náročné. Patří mezi ně například produkty firem Intergraph (WebCity), Bentley (Microstation), apod. Do druhé skupiny se pak řadí systémy s nižší funkčností, ale zároveň s nižší pořizovací cenou. Jejich funkčnost většinou postačuje konkrétním potřebám uživatele a zároveň se jedná o finančně dostupnější systém. Nevýhodou těchto systémů může být však špatná kompatibilita formátů datových souborů mezi různými systémy. Důvodem je používání vlastních formátů pro uložení grafických dat. Mezi tyto samostatné produkty patří například systémy firem Gepro, s.r.o. (MISYS, Kokeš) nebo Foresta SG, a.s. (PUKNI2) a další.

Editor schémat uličních grafů, který vznikl ve spolupráci s Úsekem koncepce a dopravního inženýrství Správy veřejného statku města Plzně [URL2] a Správou informačních technologií města Plzně [URL3], by bylo možné zařadit právě do výše uvedené druhé skupiny produktů. Jedná se o silně specializovaný systém pro vytváření mapy, který v sobě musí spojovat některé ovládací prvky používané v běžných grafických editorech a speciální funkce sloužící pro práci s mapou a jejími částmi – křižovatkami, silnicemi, apod.

V průběhu vytváření grafického editoru se objevily informace o existenci systému pro simulaci dopravy, jehož součástí je i grafický editor určený pro přípravu křižovatek a silniční sítě. Jedná se o program pro mikroskopickou simulaci dopravy AIMSUN a editor komunikační sítě TEDI [URL4]. TEDI je rozsáhlý produkt umožňující přípravu křižovatek i silnic pro mikroskopickou simulaci. Stejně jako grafický editor JUTS pracuje s elementy na úrovni jízdních pruhů a definice směru jízdy v křižovatkách. Narozdíl od grafického editoru JUTS umožňuje dále také editaci složitějších dopravních uskupení jako jsou například kruhové objezdy nebo dálnice, jízdní pruhy pro MHD, včetně možnosti připojení aktuálních jízdních řádů, umožňuje zařazení semaforů, definici rychlosti jízdy v konkrétních úsecích, rychlosti jízdy pro odbočování, sklonu vozovky, reakční doby řidičů, ... Dále nabízí uživateli export dat do formátu standardního pro import do GIS.

Celý tento simulační projekt pracuje na částečně odlišné filosofii v porovnání se systémem JUTS. Zatímco v systému JUTS slouží grafický editor pouze k přípravě mapy, její grafické úpravě a vzájemnému logickému propojení jejích jednotlivých elementů, a samotná data pro simulaci jsou pak společně s mapou vkládána v zadaném formátu přímo do simulační části systému, editor TEDI vkládá do mapy přímo také informace o intenzitě

dopravy v jednotlivých úsecích. Přítomnost informace o hustotě provozu by mohla podle mého názoru omezit možnosti testování různých dopravních situací v daném místě (např. změnu provozu v důsledku uzavření paralelní komunikace, popřípadě situaci, kdy by v daném místě došlo k dopravní nehodě apod.). Vzhledem k tomu, že jsem však neměla možnost vyzkoušet si činnost simulátoru ani grafického editoru osobně, nemohu tuto domněnku podložit praktickými zkušenostmi.

### 3.2 Funkčnost, výběr vhodných funkcí

Jak jsem již zmínila v kapitole 3.1, velká část GIS systémů se shoduje zejména v základních funkcích. Řada těchto funkcí bývá zároveň i běžnou součástí nesespecializovaných grafických editorů. Jedná se především o funkce umožňující práci s pohledem (výběr oblasti, zoom, skrývání jednotlivých částí mapy), možnost výběru jednotlivých zobrazovaných elementů, práce se soubory apod.

GIS kladou velký význam na přesnost souřadných systémů, často také nabízejí možnost pro volbu konkrétního souřadného systému. Odlišnosti jednotlivých GIS je možné hledat také v závislosti na velikosti zpracovávaného území a účelu zpracování (tvorba mapy, zpracování statistik obyvatelstva, ...). Systémy poskytují uživateli další funkce jako výpočty výměry oblastí a vzdáleností, vzájemných vztahů a pozic elementů a jiné geodetické a konstrukční funkce, umožňují přesné nastavení měřítka, nabízí práci s černobílými i barevnými rastry, umožňují vektorizaci objektů, tvorbu 3D modelů apod. Neméně důležitým hlediskem při kategorizaci GIS je také velikost a detailnost zpracování území. Některé systémy jsou vhodné pro práci s daty pro území jednoho nebo několika států, jiné pracují se souřadnicemi jednotlivých parcel, silnic, plochami lesních porostů a jinými geografickými prvky. Některé systémy jsou přímo poměrně úzce specializované na konkrétní oblast lidské činnosti – např. lesní hospodářství nebo zpracování konkrétních adresních míst, jiné pracují na obecnější bázi.

V závislosti na všech těchto faktorech a po konzultaci se studenty a pracovníky katedry matematiky ZČU, odborníky v oblasti GIS, jsem se rozhodla jako referenční software pro svoji diplomovou práci zvolit systém Kokeš for Microsoft Windows, produkovaný firmou GEPRO s.r.o. Tento software detailně pracuje s nejrůznějšími geografickými elementy (zástavba, parcely, ...). Velikostí zpracovávané oblasti a detailností úprav jednotlivých prvků tedy odpovídá parametrům uličního grafu. Dalším důvodem pro volbu systému Kokeš byla spolupráce odborníků FAV ZČU na jeho vytváření. Abych mohla tento systém použít jako referenční, bylo nutné se seznámit s jeho funkcemi a možnostmi.

Mojí základní snahou bylo získat maximum informací týkajících se ovládní a stylu práce. Snažila jsem se najít takové funkce, které uživateli práci v editoru zjednodušují a urychlují. Tyto informace jsem získávala nejen vlastním zkoumáním programu a čtením dokumentace k systému Kokeš [GepDoc], ale také spoluprací s těmi, kteří tento systém běžně používají. Během mnoha konzultací jsem tak získala řadu námětů a doporučení, kterých jsem se během tvorby grafického editoru snažila držet.

## 4 Návrh editoru

Grafický editor schémat uličních grafů slouží k vytváření map pro systém JUTS. Systém umožňuje načtení všech potřebných vstupních dat, jejich úpravu a doplnění o další údaje. Jako vstupní data jsou používána skutečná geografická a dopravní data města Plzně, která byla poskytnuta Správou informačních technologií města Plzně [URL3]. Po dokončení editace uličního grafu lze výslednou mapu exportovat do souboru přesně popsaného XML formátu, který je možné použít jako vstupní soubor pro JUTS. Formáty všech vstupních i výstupních souborů jsou podrobně popsány v kapitole 2.2 – Formáty souborů.

Při návrhu samotného grafického editoru jsem vycházela z poznatků získaných důkladným prostudováním referenčního GIS – Kokeš for Microsoft Windows i v závislosti na zkušenostech s prací v obecných grafických editorech. Některé funkce bylo třeba zahrnout do systému nezávisle na jakýchkoliv již existujících systémech. Jedná se o funkce související s úzkou specializací programu, například úprava silniční sítě nebo definice směrů pro jízdu křižovatkou.

Po zvážení všech potřebných a vhodných funkcí bylo nutné z důvodu přehlednosti jednotlivé funkce logicky rozčlenit do kategorií a navrhnout samotné grafické rozhraní editoru (podrobnější popis návrhu grafického prostředí popisuje kapitola 4.1). Poté následovala implementace jednotlivých algoritmů (viz kapitola 4.2) a funkcí (popsány v kapitole 4.3). Původně navržená funkčnost i vzhled procházely během implementace upřesňujícími změnami a úpravami.

V dále uváděných vzorcích většinou nebudou detailně vysvětleny jednotlivé proměnné, vztahy nebudou odvozovány příliš podrobně. Je to z důvodu jednoduchosti vztahů a jejich všeobecné známosti.

### 4.1 Návrh grafického prostředí

Při prostudování referenčního systému i procházení běžných grafických softwarů, během vytváření návrhu i následné implementace jsem se snažila, aby samotné používání jednotlivých funkcí a celková příprava mapy byly maximálně intuitivní. Cílem bylo vytvořit snadno ovladatelný a přehledný editor pro zadaný účel – vytvoření silničního grafu ze zadaných vstupních dat a jeho export do výstupních souborů s odpovídajícím XML formátem.

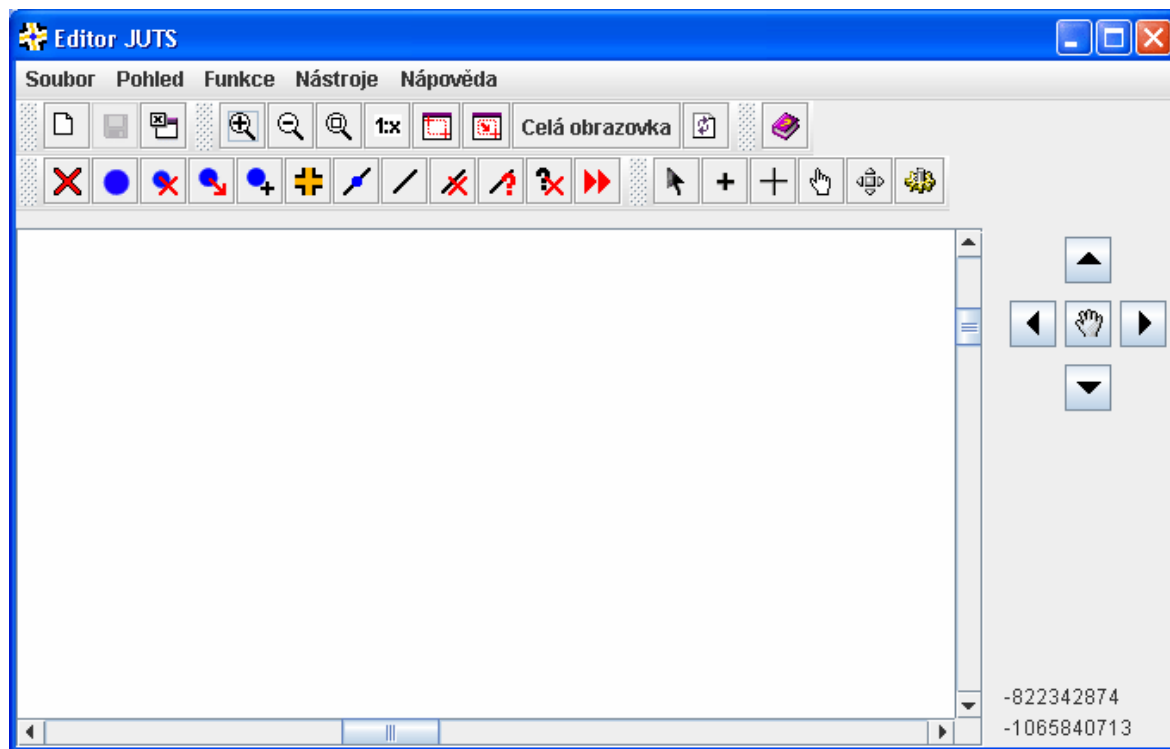
Před samotným návrhem grafického prostředí bylo nutné rozhodnout, jakým způsobem bude vlastně celý proces vytváření mapy probíhat, jaká činnost bude od uživatele vyžadována a v jakém rozsahu, zda bude vytváření mapy probíhat klikacím nebo dávkovým způsobem a jestli bude závislé jen na uživateli, nebo bude částečně automatizováno.

Při konečném rozhodování jsem se přiklonila k jedinému hlavnímu oknu, ve kterém jsou umístěny všechny grafické ovládací prvky – menu, nástrojové lišty, šipková růžice, i samotná kreslicí plocha. Aby byla uživateli umožněna co nejpohodlnější práce, umožňuje editor nejrůznější nastavení vykreslování (barva, styl, velikost). Zároveň lze skrývat a opětovně zobrazovat jednotlivé ovládací prvky. Tím je umožněna alespoň částečně variabilní velikost pracovní plochy. Rozmístění ovládacích prvků v okně za předpokladu, že jsou všechny zobrazené, je popsáno na obr. 4.1.

Grafický editor vytváří mapu poloautomaticky, s nutnými zásahy uživatele v okamžicích vyžadujících kontrolu stavu, popř. doplnění informací, které nejsou systému

dobu ve vstupních datech, jako např. přiřazení reálných křižovatek ke křižovatkám silniční sítě nebo definice směrů pro průjezd křižovatkou. Editor sám provádí určité kontroly správnosti stavu mapy, a pokud je to nutné, na případné chyby uživatele upozorní. Během přechodů je automaticky generován vzhled mapy a následně je uživateli umožněna dodatečná korekce automaticky vygenerovaného stavu a doplnění potřebných informací.

Styl práce je závislý zejména na používání myši, systém obsahuje velké množství klávesových zkratk. Celou mapu je ale možné vytvořit také pouze pomocí pohybu a klikání myši.



Obr. 4.1: Rozmístění všech grafických ovládacích prvků v hlavním okně editoru.

## 4.2 Algoritmy využívané editorem

Během vytváření mapy využívá editor několik algoritmů pro vyhledávání, určení pozice nebo automatické generování. Následující podkapitoly určují přehled jednotlivých algoritmů, jejich popis a vysvětlení s názornými ukázkami na obrázcích.

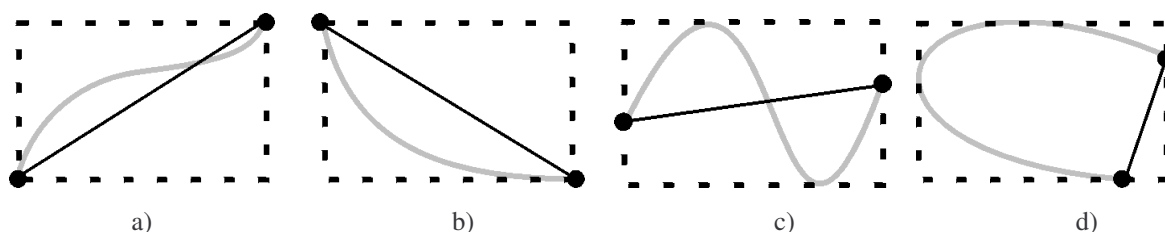
Konkrétní použití algoritmů ve funkcích editoru je popsáno společně s funkcemi editoru v kapitole 4.3 a dále pak v příloze A – Uživatelská dokumentace, v kapitole o typickém postupu přípravy mapy „od A do Z“, která se zabývá postupem vytváření mapy v grafickém editoru.

Grafický editor pracuje zároveň se dvěma souřadnými systémy – souřadným systémem kreslicí plochy (tento systém se používá při vykreslování, zjišťování pozice kurzoru v kreslicí ploše apod.) a souřadným systémem uličního grafu (pozice uzlových bodů, celkové rozměry mapy, ...) Proto je nutné často používat přepočty mezi oběma systémy. Způsob přepočtu je součástí popisu pro výběr uzlového bodu v podkapitole 4.2.2.

### 4.2.1 Vytvoření silniční sítě

Algoritmus slouží k automatickému vytvoření sítě silnic ze vstupních dat načtených na začátku vytváření nového uličního grafu. Podstatné jsou oba soubory v textovém formátu (první udávající ohraničující obdélníky segmentů silnic a druhý obsahující souřadnice uzlových bodů; přesný popis jejich formátu je uveden v kapitole 2.2). Pro vytvoření silniční sítě je potřeba vyhledat souvislosti mezi jednotlivými uzlovými body a hranicemi segmentů a zároveň zajistit správné natočení segmentů silnic (ohraničující obdélník segmentu silnice udává oblast, ve které se může daný segment vyskytovat).

V rámci ohraničujícího obdélníku může být segment libovolně zakřivený, jeho koncové body jsou však vždy umístěny na hranici dané obdélníkem (příklady viz obr. 4.2). Koncové body segmentu silnice jsou zároveň totožné se dvěma uzlovými body zadanými v druhém datovém souboru. Je tedy nutné pro každý načtený segment vyhledat právě dva odpovídající uzlové body, mezi kterými je segment proložen ve tvaru úsečky. Použitím aproximace dochází k částečné ztrátě přesnosti oproti reálnému tvaru segmentu, informace o přesném tvaru však není ve vstupních datech. Vzhledem k tomu, že však naprostá většina reálných úseků odpovídá situaci na obrázku 4.2 a), b), poskytuje tato aproximace pro simulaci dostatečnou přesnost.



Obr. 4.2: Aproximace segmentu silnice úsečkou – původní tvar je zobrazen šedou křivkou, aproximace pak černou úsečkou.

Pro každý segment jsou tedy vyhledány dva uzlové body a zároveň jsou navzájem propojeny odkazy. Jakmile jsou prohledány všechny segmenty obsažené v datovém souboru, vznikne silniční síť se vzájemnými vazbami mezi segmenty a uzlovými body. Vzhledem k malému počtu segmentů obsažených v datových souborech by předzpracování dat a následné použití rychlejšího algoritmu pro vyhledávání odpovídajících uzlových bodů pravděpodobně nemělo zásadní vliv na rychlost zpracování. Urychlení by bylo patrné při výrazném nárůstu počtu segmentů a uzlových bodů (řádově alespoň pro stovky objektů). Tyto počty jsou v závislosti na zadání v současné době málo pravděpodobné.

### 4.2.2 Výběr uzlového bodu

V průběhu úpravy mapy v prvním editačním kroku musí mít uživatel možnost upravovat pozice uzlových bodů, popřípadě jednotlivé body mazat. Je tedy zapotřebí algoritmus, který by v závislosti na zadání souřadnic kurzoru zjistil, zda je zadaná souřadnice v oblasti uzlového bodu, popř. který uzlový bod má k pozici kurzoru nejmenší vzdálenost.

Výběr zvoleného uzlového bodu z mapy je prováděn přepočtem souřadnic a následným vyhledáním takového existujícího uzlového bodu, který se na dané souřadnici nachází. Po kliknutí myši do oblasti mapy se v závislosti na měřítku zobrazení a posunu výřezu mapy přepočítají souřadnice bodu do geografického souřadného systému podle vztahů uvedených níže (1). Získané geografické souřadnice jsou s tolerancí odpovídající

průměru kruhové značky, která zobrazuje uzlový bod, použity pro vyhledávání existujícího uzlového bodu. Vzhledem k nepříliš vysokému celkovému počtu uzlových bodů je vyhledávání prováděno sekvenčně.

$$\begin{aligned} geoX &= mapLeft + (x + viewX) * scale \\ geoY &= mapTop - (y + viewY) * scale \end{aligned} \quad (1)$$

kde:  $[geoX, geoY]$  ... skutečné geografické souřadnice  
 $[x, y]$  ... souřadnice kliknutí v mapě  
 $[viewX, viewY]$  ... souřadnice pozice levého horního vrcholu výřezu mapy  
 $scale$  ... měřítko zobrazení mapy  
 $mapLeft, mapTop$  ... levý, resp. horní okraj mapy

Uzlový bod je vybrán, jestliže je vzdálenost pozice kurzoru a uzlového bodu menší než poloměr kruhové značky zobrazující daný uzlový bod. Jestliže je vyhledáván nejbližší uzlový bod k pozici kurzoru, je jako vybraný označený ten uzlový bod, jehož vzdálenost od pozice kurzoru je ze všech uzlových bodů v uličním grafu nejmenší.

### 4.2.3 Výběr segmentu silnice

Vzhledem k tomu, že editor umožňuje uživateli také práci s jednotlivými segmenty silnice (mazání, oprava), obsahuje editor také algoritmus pro výběr segmentu v mapě. Cílem algoritmu je tedy určit, zda kurzor, jehož souřadnice byly algoritmu předány, neleží blízko některého z existujících segmentů silnic. Velikost tolerance je určena poloměrem kružnice umístěné středem na pozici kurzoru (popis viz níže). Z důvodu obsažení celé mapy probíhají všechny výpočty v reálných geodetických souřadnicích.

Pro vyhledání označeného segmentu jsem využila zjištění vztahu mezi kružnicí vyjádřenou vztahem (2) a přímkou/úsečkou parametricky vyjádřenou vztahem (3). Souřadnice kliknutí jsou uvažovány jako střed kružnice s určitým tolerančním poloměrem. Tento poloměr byl zvolen v závislosti na provedených testech. Všechny existující segmenty jsou sekvenčně procházeny a testovány, zda může existovat průsečík segmentu a kružnice (tzn. že diskriminant rovnice (4) je kladný). Z důvodu urychlení se nejprve testuje možnost průsečíku s přímkou procházející segmentem, v kladném případě se pak zjišťuje konkrétní hodnota parametru v bodě průsečíku (5). Jestliže odpovídá rovnici úsečky, úsek je určen jako zvolený. Pro zjednodušení výpočtu je souřadný systém přepočítán tak, aby byl jeho počátek ve středu kružnice. Celý výpočet je založený na následujícím matematickém postupu:

$$\mathbf{x}^T \mathbf{x} - R^2 = 0 \quad (2)$$

$$\mathbf{x}(t) = \mathbf{x}_A + s\mathbf{t} \quad (3)$$

$$(\mathbf{x}_A + s\mathbf{t})^T (\mathbf{x}_A + s\mathbf{t}) - R^2 = 0 \quad (4)$$

$$D = b^2 - 4ac; \quad \begin{aligned} a &= \mathbf{s}^T \mathbf{s} \\ b &= 2 \cdot \mathbf{x}_A^T \mathbf{s} \\ c &= \mathbf{x}_A^T \mathbf{x}_A - R^2 \end{aligned}$$

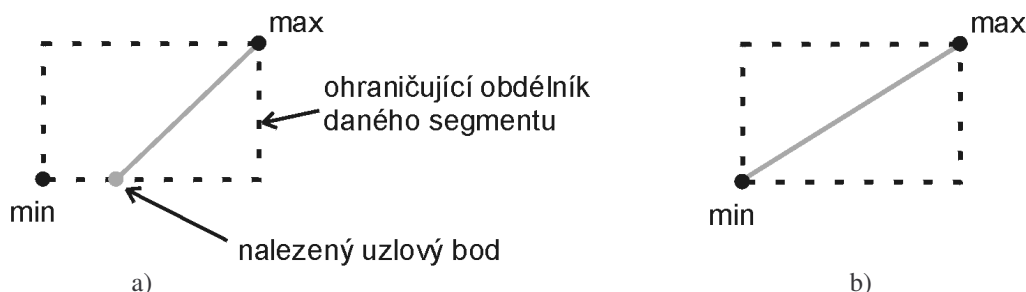
$$t_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \quad (5)$$

Jestliže je pro vypočtený parametr  $t$  splněna podmínka, že  $t \in \langle 0,1 \rangle$ , leží bod daný souřadnicemi kurzoru skutečně na úsečce udávající segment silnice nebo v její těsné blízkosti. Segment je v tomto případě určen jako vybraný. Při vyhledávání vybraného segmentu jsou sekvenčně procházeny všechny segmenty v mapě. Je totiž možné, že pro jednu pozici kurzoru bude nalezeno více odpovídajících segmentů a to v případě, že pozice kurzoru odpovídá pozici některého z uzlových bodů, který spojuje větší počet segmentů. V takovém případě jsou označeny všechny segmenty.

#### 4.2.4 Oprava chybného segmentu

Při ukládání vstupních dat jsou uzlové body a segmenty silnic načítány ze dvou různých souborů. Z tohoto důvodu může dojít k určité nekonzistenci dat, tzn. některým segmentům chybí po načtení jeden nebo oba uzlové body. Takový segment je editorem pokládán za chybný a je u něj potřeba chybějící bod, popř. oba body, doplnit. Výběr segmentu určeného pro opravu je prováděn algoritmem popsáním v kapitole 4.2.3. Při opravě segmentu mohou nastat dvě situace (oba případy jsou znázorněny na obrázku 4.3):

- Při načítání byl k segmentu přiřazen jen jeden uzlový bod – při opravě je jako jeden koncový bod segmentu zadán nalezený uzlový bod, jako druhý koncový bod je zvolen vzdálenější bod z dvojice minimum – maximum, udávající souřadnice ohraničujícího obdélníku – viz obrázek a).
- K segmentu nebyly ve vstupních datech nalezeny žádné uzlové body – jako koncové body segmentu jsou zadány minimální bod a maximální bod ohraničujícího obdélníka segmentu – viz obrázek b).



Obr. 4.3: Možné způsoby nalezení chybějících bodů u segmentů načtených jako chybné: a) jeden uzlový bod je nalezen, jako druhý je určen vzdálenější bod z dvojice minimum-maximum; b) oba uzlové body chybí, jsou jim přiřazeny pozice minima a maxima ohraničujícího obdélníka.

V obou případech dochází ke zkreslení. V prvním případě je zkreslení pouze částečné, jeden bod segmentu je zadán přesně. Ve druhé situaci může dojít k úplnému zkreslení pozice segmentu (koncové body segmentu mohou ležet kdekoli na hranici ohraničujícího obdélníka). V tomto případě nezbyvá než pozici koncových bodů upravit ručně.

#### 4.2.5 Vytvoření sítě křižovatek

Algoritmus pro vytvoření sítě křižovatek je volán v okamžiku převodu mapy z první editační fáze (úprava sítě uzlových bodů a segmentů silnic) do druhé editační fáze (práce

s křižovatkami, jízdními pruhy apod.). K jednotlivým uzlovým bodům je přiřazena informace o funkci daného bodu. Mapa tedy nyní obsahuje uzlové body, kterým byla definována jedna ze tří možných funkcí:

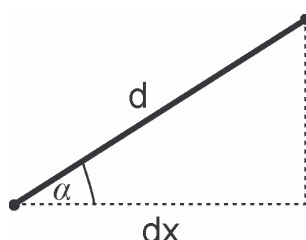
- koncový bod (přísluší jedinému segmentu; v předchozím kroku ho nebylo nutné ani možné explicitně definovat),
- spojovací bod (spojuje dva sousední segmenty silnic; uživatel ho mohl explicitně označit jako spojnicí),
- křižovatka (uzlový bod, na který byla namapována některá z křižovatek načtených ze souboru XML; při přechodu do této editační fáze byla automaticky provedena kontrola, zda počet segmentů vycházejících z uzlového bodu odpovídá počtu ramen namapované křižovatky).

Jestliže má bod funkci křižovatky, musí k němu být přiřazena reálná křižovatka načtená ze vstupního souboru. Při vytváření křižovatky je tedy nejdříve nutné přiřadit ke každému segmentu, který vychází z uzlového bodu, odpovídající rameno křižovatky.

Jestliže kdekoliv v mapě nastane situace, kdy jsou dvě ramena křižovatky spojena pouze jediným segmentem, je nutné tento segment uměle rozdělit na dva a umožnit tak vznikající silnici přiřazení ramen obou křižovatek (z každé strany jedno). Toto rozdělení je provedeno jednoduchým matematickým výpočtem s jediným omezením – segment nelze rozdělit přesně uprostřed, protože pak v následujících výpočtech dochází během vytváření jízdnic k problémům (dělení nulou apod.). Proto jsou tyto segmenty děleny v poměru 0,49:0,51 původní délky.

Ramena křižovatky jsou již při načítání XML souboru řazena za sebou podle orientace tak, aby jako první bylo v seznamu uloženo rameno s nejnižší hodnotou orientace. Toho lze s výhodou využít pro přiřazování skutečných ramen křižovatky ke konkrétním segmentům. Nejdříve je však nutné všechny segmenty seřadit stejným způsobem, jakým jsou seřazena ramena.

Pro zjištění natočení jednotlivých segmentů byl použitý následující jednoduchý výpočet založený na vyjádření funkce kosinus v pravouhlém trojúhelníku:  $\cos \alpha = dx / d$ , významy jednotlivých vzdáleností odpovídají obrázku 4.4.



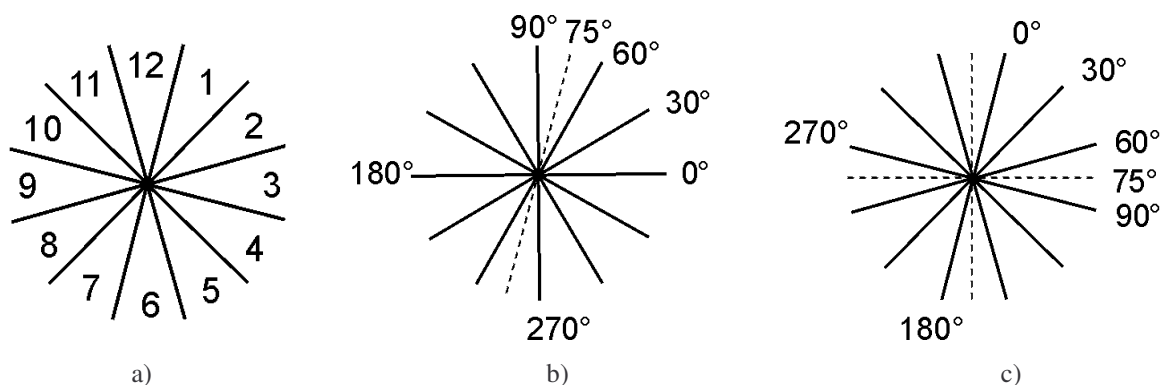
Obr. 4.4: Skutečný význam jednotlivých částí výrazu pro výpočet natočení segmentu silnice.

Kvůli jednoduššímu přiřazení segmentů a ramen křižovatky je dále prováděn následující přepočítání úhlu (6) do orientace odpovídající orientaci uváděné u ramen křižovatky:

$$\begin{aligned} \text{orientace} &= 75^\circ - \alpha && \text{pro } \forall \alpha \leq 75^\circ \\ \text{orientace} &= 75^\circ - \alpha + 360^\circ && \text{pro } \forall \alpha > 75^\circ \end{aligned} \quad (6)$$



Na obrázcích 4.5 je možné vidět způsob udávání orientace u ramen křižovatek (a), číselný význam úhlu  $\alpha$  (b) a orientaci segmentů silnic po přepočtu, tzn. číselný význam hodnoty *orientace* (c).



Obr. 4.5: Způsoby zadávání úhlu natožení: a) u křižovatek; b) u segmentů před přepočtem; c) u segmentů po přepočtu.

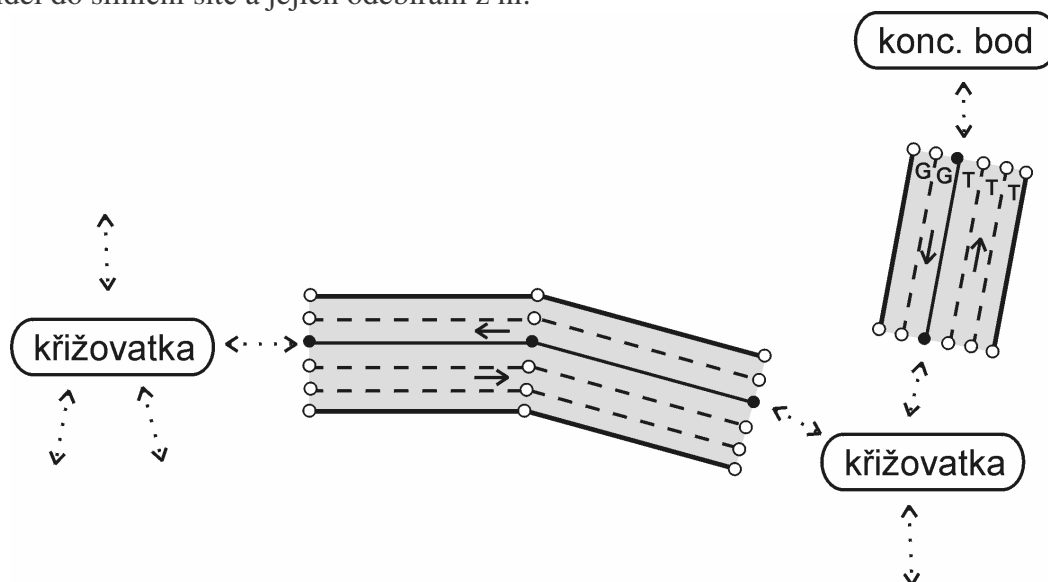
Jednotlivá ramena křižovatky jsou pak segmentům přiřazována postupně od nejnižší hodnoty orientace k nejvyšší. Pokud by tedy uživatel v předchozím kroku na uzlový bod namapoval nevhodnou křižovatku, u které by se orientace jejích ramen zásadně lišila od směřování segmentů, může dojít k chybnému přiřazení. K takové situaci by však mohlo dojít pouze v případě chybných nebo nereálných vstupních dat nebo při snaze uživatele přiřadit uzlovému bodu křižovatku, která ve skutečnosti odpovídá zcela odlišnému umístění. Protože se předpokládá, že editor bude používat zaškolený pracovník, nebyly tyto případy ošetřovány.

Ke každému segmentu uzlového bodu je tedy přiřazeno rameno reálné křižovatky. Jednotlivé křižovatky je nyní nutné spojit a vytvořit mezi nimi strukturu silnice, zahrnující jízdní pruhy a další potřebné informace. Systém proto pro všechna ramena křižovatek postupným procházením navazujících segmentů vytváří silnici, do které ukládá počáteční a koncový uzlový bod, počty jízdních pruhů na začátku a na konci obou směrů silnice, seznam všech segmentů, ze kterých silnice vznikla, a seznam bodů udávajících středovou linii vozovky (tzn. dělicí čáru mezi oběma směry v silnici). Uzlové body jsou postupně odmazávány tak, jak jsou přiřazeny k nově vytvářeným silnicím, takže na konci této operace zůstávají uloženy jen uzlové body s koncovou funkcí nebo body odpovídající křižovatce. Silnice vzniká vždy od křižovatky ke křižovatce nebo k uzlovému bodu, body udávající linii středu vozovky jsou udávány ve směru od počátečního ke koncovému bodu křižovatky.

Struktura silnice je zobrazena na obrázku 4.6. Každá silnice se skládá ze dvou směrů. Pro každý směr je definováno několik jízdních pruhů. Každý směr je zároveň považován za samostatnou část, body levého a pravého oddělovače jízdních pruhů jsou zadávány vždy ve směru jízdy. Jízdní pruh je určen levým a pravým oddělovačem, které tvoří hranici jízdního pruhu po obou stranách. V každém oddělovači je uloženo několik křivek, daných jednotlivými body a jejich souřadnicemi. Křivky mají zároveň definovaný i styl, kterým mají být vykreslovány (plná čára, přerušovaná čára apod.). V této chvíli je již nepřehlédnutelná podoba struktury pro uložení dat s formátem výstupních souborů

(grafické i simulační části). Tato podoba je v systému zavedena úmyslně z důvodu snazšího vytváření XML souborů.

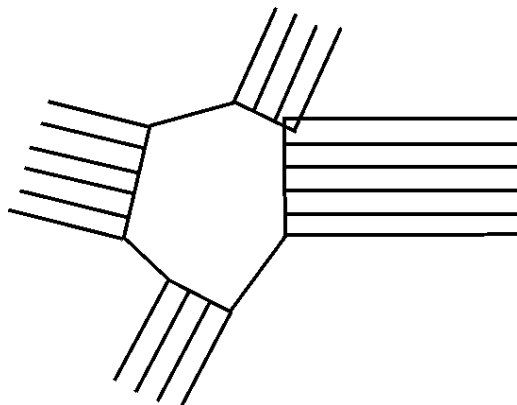
Jestliže vede silnice mezi křižovatkou a koncovým bodem, jsou na jejím konci vytvořeny generátory a terminátory (viz písmena G, T v obrázku 4.6) sloužící pro dodávání vozidel do silniční sítě a jejich odebrání z ní.



Obr. 4.6: Struktura popisující síť křižovatek a silnic mezi nimi.

#### 4.2.6 Vygenerování hranice křižovatky

Pro každou křižovatku, která v mapě vznikla, je nutné vygenerovat její hranici. Hranice křižovatky je určena počtem jízdních pruhů v obou směrech všech ramen křižovatky. Křižovatku je nutné vytvářet tak, aby se žádný jízdní pruh nepřekrýval se svým sousedem, viz obr. 4.7. Proto jsem se rozhodla založit generování křižovatky na základě tzv. kružnice křižovatce opsané. Střed této kružnice je totožný se středem křižovatky (uzlovým bodem, jehož souřadnice udávají přesnou pozici křižovatky). Poloměr je určen jako maximální vzdálenost hranice krajního jízdního pruhu od středu křižovatky, která je nutná pro to, aby se sousední jízdní pruhy nepřekrývaly. Jestliže je tato maximální vzdálenost dodržena pro nejproblématictější místo (nejmenší úhel mezi rameny křižovatky, do kterého je zároveň nutné umístit největší počet jízdních pruhů), pak v ostatních případech nemůže dojít k problémům.

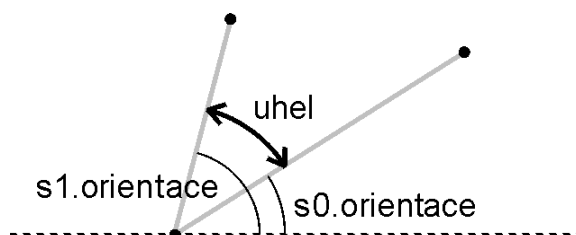


Obr. 4.7: Chybné určení hranice křižovatky, jízdní pruhy sousedních ramen křižovatky se překrývají.

Nejdříve se musí určit úhly mezi sousedními segmenty. Tuto hodnotu je možné zjistit jako rozdíl úhlů určujících přesnou orientaci segmentu (viz obr. 4.8)

$$uhel = s1.orientace - s0.orientace$$

Velikost tohoto úhlu by bylo možné spočítat i jinými metodami (např. pomocí skalárního součinu), ale tento způsob je nejjednodušší a nejrychlejší vzhledem k tomu, že pro každé rameno křižovatky je již z předchozího výpočtu známa jeho orientace.

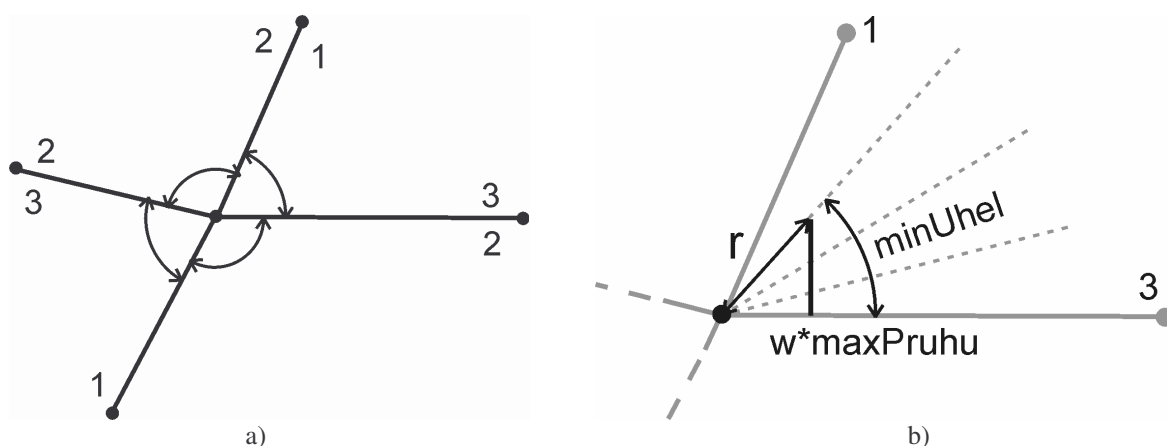


Obr. 4.8: Výpočet úhlů mezi jednotlivými rameny křižovatky.

Zároveň je nutné zjistit, který úhel v křižovatce je kritický. Jedná se o ten úhel, ve kterém je minimální hodnota  $minUhel$ . Úhly a počty pruhů v křižovatce jsou uvedeny na obr. 4.9 a). Hodnota minimálního úhlu udává dvě ramena křižovatky, mezi kterými musí být hranice umístěna v největší vzdálenosti od středu křižovatky. V tomto místě je tedy nutné spočítat poloměr kružnice křižovatce opsané –  $r$  (viz obr. 4.9 b). Vztahy pro daný výpočet jsou uvedeny v (7).

$$minUhel = \frac{uhel}{celkPruhu} * maxPruhu \quad (7)$$

$$r = (maxPruhu * w) / \sin(minUhel)$$



Obr. 4.9: Postup při určení kritického úhlu v křižovatce: a) zjištění úhlů mezi rameny křižovatky; b) výpočet poloměru kružnice křižovatce opsané, jakmile byl určen kritický úhel.

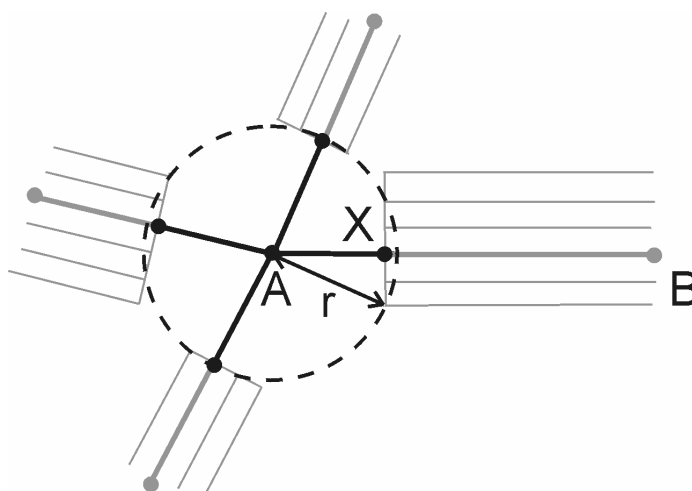
Jakmile je pro křižovatku určený poloměr kružnice, je možné pro všechna ramena křižovatky dopočítat počáteční souřadnice středu vozovky (místo, kde začíná středová čára vozovky). Výpočet probíhá podle následující sekvence rovnic. Minimální počet pruhů je volen z důvodu zamezení problémů u ramen s rozdílným počtem pruhů na vjezdu a výjezdu. Obrázek 4.10 znázorňuje situaci v okamžiku výpočtu podle vzorců (8).

$$sirka = minPocetPruhu * w \quad (8)$$

$$t = \frac{\sqrt{r^2 - sirka^2}}{\sqrt{(ax - bx)^2 + (ay - by)^2}}$$

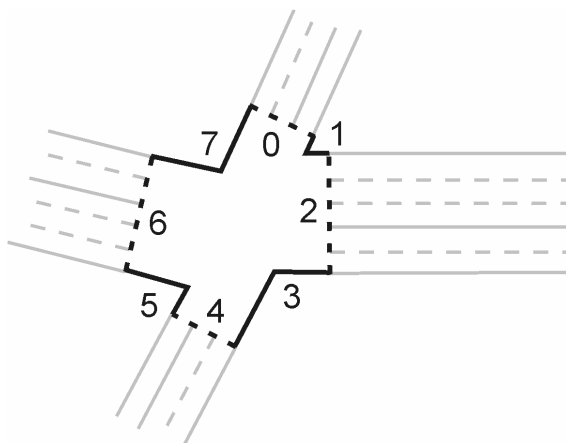
$$x = ax + (bx - ax) * t$$

$$y = ay + (by - ay) * t$$



Obr. 4.10: Výpočet hranice křižovatky, jestliže byl určen poloměr kružnice křižovatce opsané.

Po dopočítání hranice křižovatky lze vygenerovat i křivky, které hranici jednotlivých křižovatek udávají. Ty jsou zadávány ve směru orientace ramen křižovatky postupně od první až po poslední, jak je naznačeno na obrázku 4.11. Hranice křižovatky je později ještě zpřesněna a křivky doplněny o průsečíky pravých křivek nejkrajnějších jízdních pruhů sousedních ramen křižovatky. Po dopočítání křižovatek je ještě nutné dodatečně upravit hodnoty počátečních a koncových bodů, udávajících středovou linii vozovky.



Obr. 4.11: Konkrétní zadání křivek určujících hranici křižovatky. Křivky nesou zároveň také informaci o způsobu vykreslování jednotlivých částí hranice křižovatky.

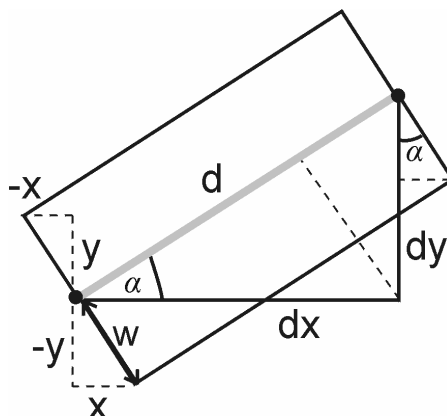
Tento postup sice nezaručuje zcela optimální výsledný tvar křižovatky (ve skutečnosti by bylo možné posunout středy některých ramen blíže ke středu). Pro automatické generování je však tento způsob poměrně jednoduchý a v poměru s výsledky dostatečně robustní.

#### 4.2.7 Výpočet hranic jízdních pruhů, jejich korekce

S využitím znalosti souřadnic bodů udávající lomenou čáru, která tvoří středovou čáru vozovky, je možné s využitím Pythagorovy věty dopočítat souřadnice pro potřebný počet jízdních pruhů. Výpočet je naznačen na obrázku 4.12. Křivky jsou ukládány odděleně pro každý směr jízdy vždy od středu vozovky – viz 4.13 a). Na obrázku 4.13 b) je vykreslená křižovatka po vygenerování jízdních pruhů a jejich oddělení křivkami odpovídajícího stylu. Výpočet je prováděn podle následujících vzorců (9):

$$y = w \cdot \cos \alpha = \frac{w \cdot dx}{d} \quad (9)$$

$$x = w \cdot \sin \alpha = \frac{w \cdot dy}{d}$$



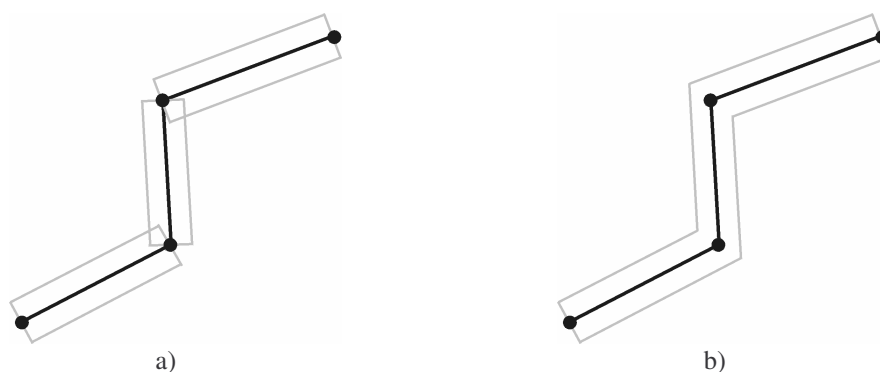
Obr. 4.12: Výpočet jednotlivých jízdních pruhů.



Obr. 4.13: Styly vykreslení oddělovačů jízdních pruhů pro a) jednu silnici; b) celou křižovatku.

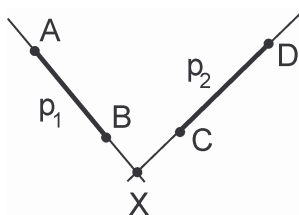
Jestliže se jízdní pruh skládá z několika úseků, tvoří v podstatě lomenou čáru. Jestliže algoritmem popsaným v kapitole 4.2.6 spočítáme pravou křivku jízdního pruhu, dojdeme k situaci znázorněné na obrázku 4.14 a); jako levý oddělovač je možné použít pravý oddělovač sousedícího pruhu. Proto je po výpočtu hrubé hranice pravé křivky jízdního pruhu nutné provést korekční přepočít souřadnic pro zlomy křivek (tzn. zatáčky jízdního pruhu).

Pro tuto korekci jsou využita vztahy pro výpočet průsečíku dvou přímek a následný přepočít souřadnic bodu udávající hraniční křivku jízdního pruhu. Vzhled jízdních pruhů po korekci je vidět na obr. 4.14 b).



Obr. 4.14: Jízdní pruh složený z jednotlivých úseků: a) bez korekce; b) po korekčním výpočtu.

Průběh výpočtu (jednotlivé proměnné odpovídají značení na obr. 4.15) podle vztahů (10), výpočet výsledných souřadnic je prováděn podle vztahu (11):



Obr. 4.15: Označení používané ve výpočtu pro zpřesnění jízdních pruhů.

$$\begin{aligned} p_1: \mathbf{x} &= \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A) \cdot t \\ p_2: ax + by + c &= 0 \end{aligned} \quad (10)$$

$$\begin{aligned} \begin{vmatrix} a & b & c \\ x_C & y_C & 1 \\ x_D & y_D & 1 \end{vmatrix} = 0 &\Rightarrow a = \begin{vmatrix} y_C & 1 \\ y_D & 1 \end{vmatrix} = y_C - y_D \\ b &= -\begin{vmatrix} x_C & 1 \\ x_D & 1 \end{vmatrix} = x_D - x_C \\ c &= \begin{vmatrix} x_C & y_C \\ x_D & y_D \end{vmatrix} = x_C \cdot y_D - x_D \cdot y_C \end{aligned}$$

Po dosazení do  $ax + by + c = 0$ :

$$a \cdot (x_A + (x_B - x_A) \cdot t) + b \cdot (y_A + (y_B - y_A) \cdot t) + c = 0$$

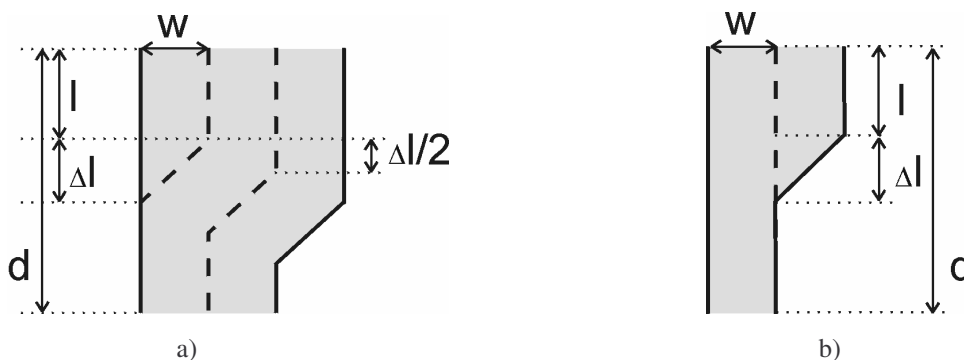
$$t = -\frac{c + a \cdot x_A + b \cdot y_A}{a \cdot (x_B - x_A) + b \cdot (y_B - y_A)}$$

$$\mathbf{x} = \mathbf{x}_A + (\mathbf{x}_B - \mathbf{x}_A) \cdot t \quad (11)$$

#### 4.2.8 Generování odbočovacích jízdních pruhů

V silničním provozu je běžná existence odbočovacích pruhů. Proto i editor musí umožňovat generování pruhů pro odbočení vlevo a vpravo. K této úpravě dochází automaticky při vytváření silnic a křižovatek ze silničního grafu. O směru pro odbočení rozhodne systém v závislosti na skutečných směrech jízdních pruhů odpovídajícího ramene křižovatky podle následujícího pravidla: Jestliže je v silnici jízdní pruh, který odbočuje pouze vlevo, generuje editor odbočovací jízdní pruh doleva, jinak vytvoří pruh pro odbočení vpravo. V závislosti na pozorování reálného stavu považují tuto heuristiku za dostačující.

Během úpravy jízdních pruhů na odbočovací dochází v podstatě pouze k přepočtu souřadnic již předem vytvořených jízdních pruhů. Generování odbočovacích pruhů je znázorněno na obrázku 4.16. Nové souřadnice bodů jsou přepočítávány z parametrického vyjádření úsečky z délky  $d$  na novou délku, hodnoty konstant  $l$  a  $\Delta l$  jsou nastaveny v závislosti na pozorování reálné situace.



Obr. 4.16: Generování jízdních pruhů pro odbočování: a) vlevo; b) vpravo.

### 4.2.9 Automatické generování buněk v křižovatce

Pro definování směru jízdy křižovatkou je nutné umístit do křižovatky tzv. buňky. Každá buňka je definována souřadnicemi, na kterých je umístěna. Buňky jsou v křižovatce rozmístěny tak, jako by každé odpovídala kružnice o průměru 2,5 m. Jako kružnice o průměru 2,5 m jsou i jednotlivé buňky v křižovatce vykreslovány v mapě. Vzhledem k tomu, že v průběhu simulace slouží buňky jako místa výskytu vozidel (v každé buňce se může v jednom okamžiku nacházet pouze jediné vozidlo), nesmí se kružnice, která buňky reprezentují, překrývat. Je tak zamezeno vzniku nereálných situací, jako např. křížení dvou vozidel apod. Stejně tak není vhodné, aby buňky přesahovaly oblast křižovatky, tzn. aby střed buňky ležel v menší vzdálenosti než 1,25 metru od hranice křižovatky.

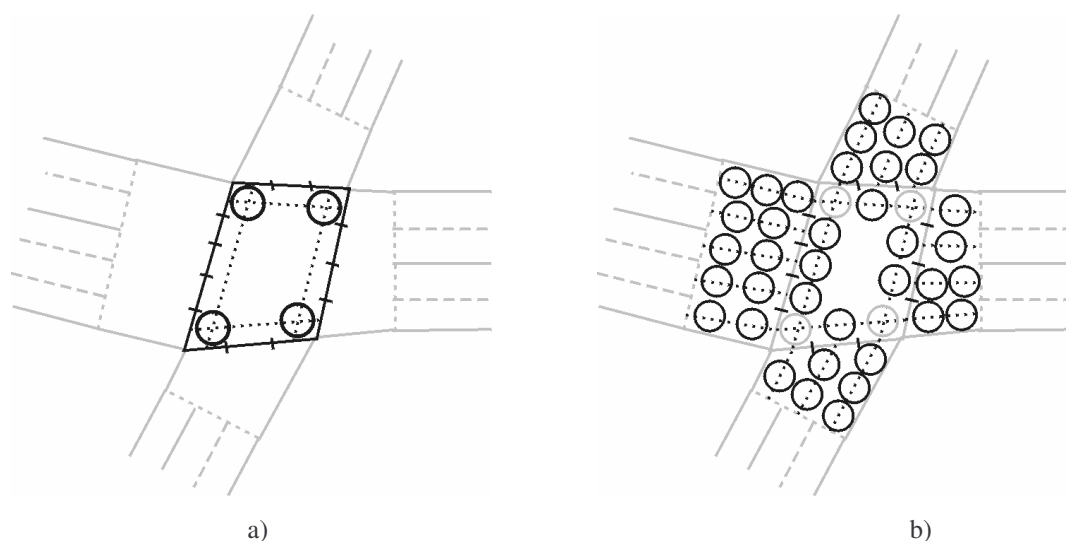
Aby uživatel nemusel vkládat do křižovatky jednotlivé buňky, rozhodla jsem se vytvořit algoritmus pro dávkové vygenerování buněk ve všech křižovatkách. Jak vyplývá z předchozího popisu, pro generování buněk je dáno několik základních omezení. Cílem algoritmu je všechna tato omezení dodržet (buňky se nesmí navzájem překrývat, musí být celé umístěny uvnitř oblasti křižovatky) a zároveň jich musí být do křižovatky umístěn maximální možný počet tak, aby bylo možné bez větších problémů definovat směry jízdy v křižovatce. Omezení jsou při automatickém generování dodržována striktně, ostatní podmínky pak maximálně, jak je to možné.

Uvnitř každé křižovatky se nachází polygon, jehož vrcholy tvoří průsečíky křivek pravých oddělovačů nejkrajnějších jízdních pruhů sousedních ramen křižovatky. Způsob vyplnění tohoto polygonu jednotlivými buňkami ovlivňuje následnou možnost definování směru jízdy. Od polygonu jsou pak buňky generovány přímo k jednotlivým jízdním pruhům tak, jak dovoluje zbývající volný prostor.

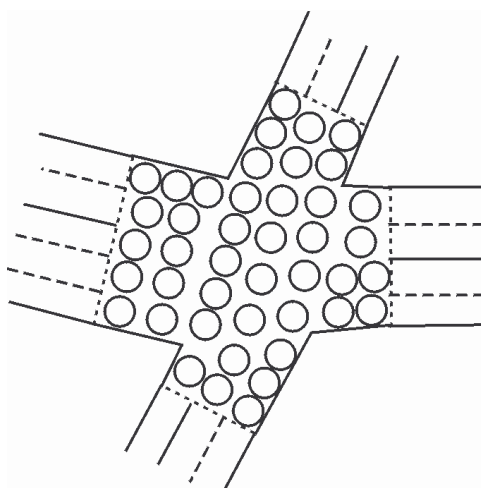
Aby bylo možné zajistit výjezdy ze všech jízdních pruhů vstupujících do křižovatky a vjezdy do všech, které z křižovatky vycházejí, je nutné pro každý jízdní pruh vygenerovat na okraji alespoň jednu buňku. Hrany polygonu uvnitř křižovatky je tedy nutné rozdělit podle počtu jízdních pruhů ramena křižovatky, které k dané hraně přiléhá, na příslušný počet dílků. Poté lze zjistit souřadnice rohových buněk ležících na průsečících přímk, které spojují středy krajních dílků sousedních hran polygonů (viz obr. 4.17 a). Mezi rohovými buňkami lze pak z parametrického vyjádření úsečky jednoduchou změnou parametru dopočítat ostatní buňky ležící na hranách polygonu. Stejným postupem lze vygenerovat i buňky mezi středy dílků hran polygonu a středem příslušného jízdního pruhu (viz obr. 4.17 b). Vzorová křižovatka s kompletně vygenerovanými buňkami je zobrazena na obr. 4.18.

Přestože je na obrázku 4.18 ukázána již celá křižovatka zaplněná automaticky vygenerovanými buňkami, zbývá ještě popsat druhou část algoritmu. Do pokrytí celé křižovatky zbývá doplnit už jen vnitřní oblast polygonu. Zde již nelze využít žádná pravidla spojená s křižovatkou, popř. jízdními pruhy. Úkolem je pouze vyplnění konvexní oblasti maximálním možným počtem buněk (tzn. nepřekrývajících se kružnic). Pro výpočet jsem zvolila rekurentní postup založený na vnitřním polygonu křižovatky. Do výpočtu již nevstupuje původní polygon, ale polygon zmenšený o první úroveň buněk vygenerovaných pro každou hranu na začátku celého postupu.





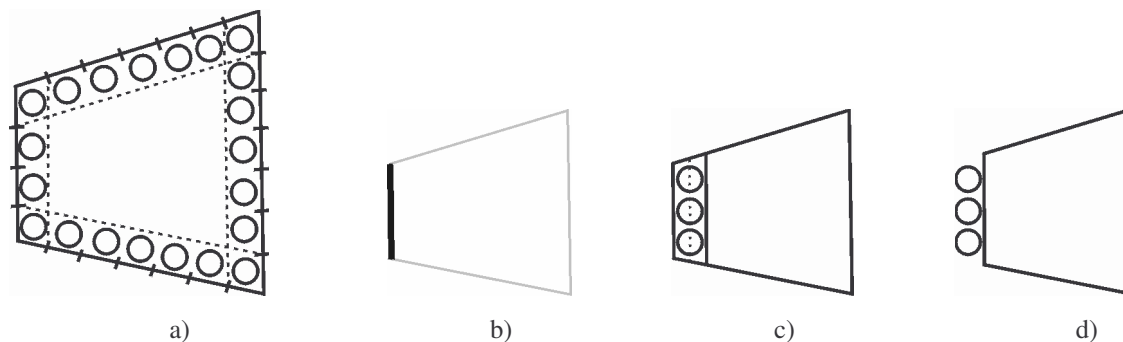
Obr. 4.17: Postupné generování buněk pro definování směru jízdy v křižovatce: a) vygenerované rohové buňky; b) buňky v křižovatce vygenerované po obvodu polygonu uvnitř křižovatky a buňky vyplňující prostor mezi polygonem a jednotlivými jízdními pruhy.



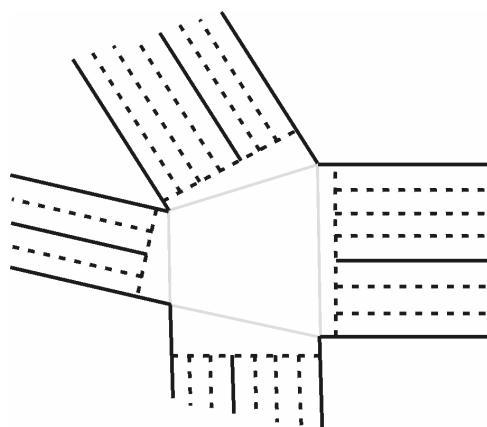
Obr. 4.18: Křižovatka kompletně zaplněná automaticky vygenerovanými buňkami v křižovatce pro definici směru jízdy vozidel.

Celý algoritmus se skládá ze čtyř základních kroků. Výchozí stav je zobrazený na obrázku 4.20 a), jednotlivé kroky algoritmu jsou pak názorně předvedeny na obrázcích 4.20 b) až d):

- ze všech hran polygonu je vybrána hrana s minimální délkou (obr. b),
- u minimální hrany je vygenerovaný maximální možný počet buněk (obr. c),
- polygon je zmenšen o velikost buňky (o ubranou část) (obr. d),
- pokud je nejkratší hrana polygonu větší než průměr kružnice reprezentující buňku v křižovatce, celý výpočet se opakuje.

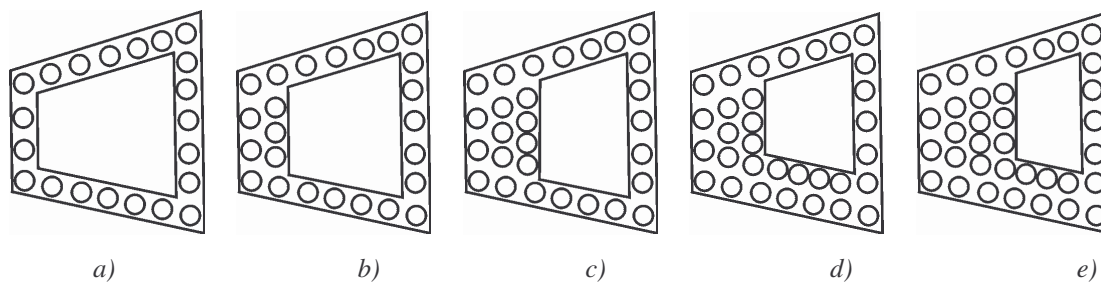


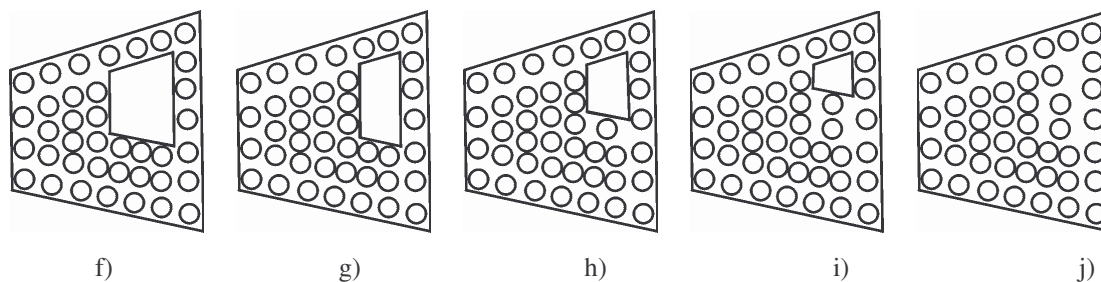
Obr. 4.20: Čtyři základní kroky algoritmu při zaplňování vnitřního polygonu křižovatky buňkami.



Obr. 4.21: Křižovatka zvolená pro popis algoritmu pro generování buněk ve vnitřním polygonu křižovatky.

Jednotlivé kroky zaplnění celého polygonu znázorňují obrázky 4.22 a) až j). Pro názornou ukázkou algoritmu jsem úmyslně zvolila odlišný případ než vzorovou křižovatku (celá nově zvolená křižovatka by mohla vypadat tak, jak je ukázáno na obrázku 4.21). Tento případ je obtížnější vzhledem k nesterannému počtu dílků na protilehlých hranách polygonu. Na obrázcích znázorňujících postupné zaplňování polygonu je patrné, že v první úrovni jsou buňky umístěné dále od sebe. Tato situace je způsobena rozdílnou šířkou jízdního pruhu (3 m) a průměrem kružnice reprezentující buňku (2.5 m). Také je možné postřehnout, že v průběhu generování jsou jednotlivé buňky v křižovatce rozmístěny s nepravidelnými rozestupy. Nepravidelné rozmístění je způsobeno postupným zmenšováním oblasti polygonu. Uživatel má možnost, pokud to uzná za vhodné, upravit rozmístění buněk v křižovatce ručně.





Obr. 4.22: Postupné zaplňování vnitřního polygonu křížovky buňkami.

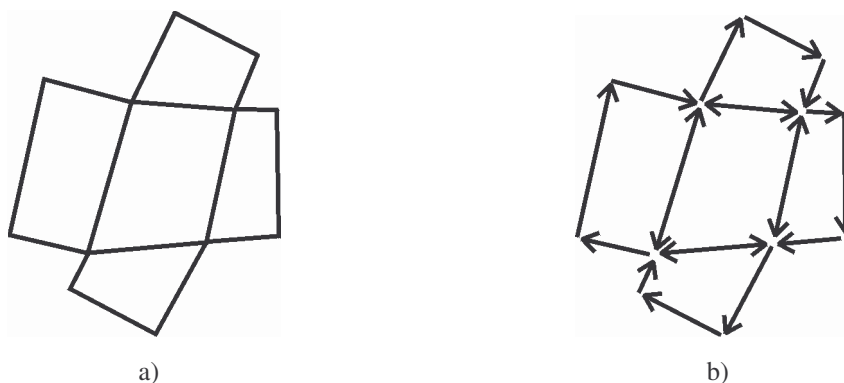
#### 4.2.10 Nalezení křížovky

Pro přidání buňky do mapy je nutné určit, ke které křížovatce bude nově vytvořená buňka patřit. Proto jsem musela do editoru zakomponovat také algoritmus pro nalezení křížovky na základě zadaných souřadnic (v tomto případě souřadnic, na kterých se nachází kurzor myši). Jedná se v podstatě o problematiku určení, zda bod leží uvnitř nebo vně oblasti křížovky. Křížovatka je v obecném případě nekonvexní polygon. Problém nalezení polohy bodu vůči nekonvexnímu polygonu je poměrně složitý. Proto jsem se rozhodla převést ho na problém nalezení pozice bodu vůči několika konvexním polygonům.

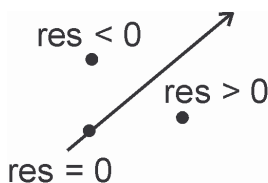
Každou křížovatku je možné převést na několik konvexních oblastí (jak je ukázáno na obrázku 4.23 a) – vnitřní polygon křížovky a oblasti mezi tímto polygonem a jednotlivými jízdnicími pruhy. Jak je naznačeno v obrázku 4.23 b), tyto oblasti jsou ukládány ve formě směrových úseček procházejících hranami oblastí. Výpočet koeficientů je totožný s výpočtem koeficientů přímky v případě korekce jízdnicích pruhů (viz kapitola 4.2.7). Pro každou hranu jsou tak při vytváření reprezentace oblasti vypočítány koeficienty  $a$ ,  $b$ ,  $c$  pro analytické vyjádření přímky. Pro samotné určení, zda bod leží uvnitř oblasti je použitý jednoduchý znaménkový test využívající vzorce (12), kde  $[x_0, y_0]$  jsou souřadnice hledaného bodu:

$$\text{res} = a \cdot x_0 + b \cdot y_0 + c \quad (12)$$

Znaménko výsledku ( $\text{res}$ ) určuje, na které straně od přímky se bod nachází nebo zda leží přímo na ní. Jednotlivé možnosti jsou znázorněny na obrázku 4.24. Bod leží uvnitř celé křížovky, pokud leží uvnitř, nebo na hraně některé z jejích konvexních podoblastí.



Obr. 4.23: Popis křížovky: a) rozdělení křížovky na konvexní oblasti; b) reprezentace konvexních oblastí hraničními přímkami.



Obr. 4.24: Možné výsledky znaménkového testu.

## 4.3 Funkce editoru

Funkce lze rozdělit do několika skupin: funkce pro práci se souborem, pohledové funkce, funkce určené pro práci s jednotlivými elementy mapy a funkce pro nastavení pracovního prostředí (tzv. nástroje). Použití jednotlivých funkcí je podrobně popsáno v příloze A – Uživatelská dokumentace. V následujícím popisu je uveden pouze přehled jednotlivých funkcí spolu s popisy, jakými byly funkce navrženy. Součástí popisu jsou také odkazy na uživatelskou dokumentaci (obsah posledního sloupce tabulky).

### 4.3.1 Funkce pro práci se souborem

Těchto několik funkcí je určeno pro práci se soubory a samotným programem. Při otvírání nové mapy jsou najednou načteny všechny tři vstupní soubory a pomocí algoritmu 4.2.1 je ze vstupních dat – souřadnic uzlových bodů a segmentů silnic – vygenerována silniční síť. Pro export mapy do XML souboru není zapotřebí žádný složitý algoritmus, pro export jsou použity standardní funkce DOM (Document Object Model), které poskytuje JavaCoreAPI. Funkce pro ukončení programu a zavření mapy pokládám za intuitivní.

Název funkce	Stručný popis	Str.
Nová mapa	Načte všechny potřebné vstupní soubory a otevře novou mapu pro editaci.	3
Uložit	Umožní export mapy do výstupních XML souborů (uživatel může volit, zda vytvoří grafickou, nebo simulační část nebo jestli provede export mapy do obou výstupních souborů zároveň).	4
Zavřít mapu	Zavře mapu a vrátí se do stavu před jejím otevřením.	4
Konec	Ukončí běh programu.	4

Tab. 4.1: Seznam funkcí pro práci se souborem.

### 4.3.2 Pohledové funkce

Funkce pro práci s pohledem jsou určeny k manipulaci s mapou během jejího vytváření. Umožňují posun mapy, přiblížení a oddálení, výběr oblasti pro zobrazení apod. Některé funkce pro posun mapy (funkce scrollbarů, použití kolečka myši) jsou automaticky obsluhovány metodami použitých komponent. Šipková růžice a pohyb pomocí šipkových kláves využívají posun viewportu komponenty JScrollPane, na které je umístěna kreslicí plocha. Posun viewportu je v implementaci funkcí často kombinován se změnou měřítka.

Název funkce	Stručný popis	Str.
Zvětšit	Lupa pro přiblížení mapy.	5
Zmenšit	Lupa pro oddálení mapy.	5
Best fit	Nastavení měřítka na takovou hodnotu, aby byla celá mapa zobrazena v kreslicí oblasti.	5
Nastavit měřítko	Přesné nastavení měřítka (1:x).	5
Výběr oblasti dvěma body	Umožňuje výběr obdélníkové oblasti mapy pomocí dvou kliknutí myši. Vybraná oblast je určena obdélníkem, jehož dva protilehlé body jsou určeny kliknutím myši.	5
Výběr oblasti tažením	Umožňuje výběr obdélníkové oblasti mapy stiskem tlačítka myši, tažením a následným uvolněním tlačítka. Souřadnice vrcholů obdélníka jsou určeny v místě stisku a uvolnění tlačítka.	5
Celá obrazovka (Fullscreen)	Slouží pro přechod do fullscreen módu, ve kterém jsou všechny ovládací prvky skryty.	6
Refresh pohledu	Vyvolá překreslení kreslicí plochy v případě chybného vykreslení mapy.	6

Tab. 4.2: Seznam pohledových funkcí.

### 4.3.3 Funkce pro práci s elementy mapy

Tyto funkce je možné rozdělit do dvou základních skupin podle toho, v jaké fázi editace mapy se používají. Jediná funkce, která je použita pro obě fáze editace, je funkce zrušení operace. Ta slouží pro zrušení operace, která probíhá ve více krocích (např. posun uzlového bodu). Zrušení operace slouží i pro zrušení výběru.

Do první skupiny funkcí (pro první editační krok) patří ty, které slouží pro úpravu silniční sítě vytvořené po načtení vstupních dat nově vytvářené mapy. Ty umožňují uživateli převést silniční graf do takové podoby, aby bylo možné převést mapu do druhého editačního kroku. Jedná se o funkce uvedené v tabulce 4.3.

Funkce pracující s uzlovými body využívají pro výběr konkrétního uzlového bodu algoritmus 4.2.2. pro výběr uzlového bodu. Funkce pro operace se segmenty pak pro vyhledání odpovídajícího segmentu silnice používají algoritmus popsany v kapitole 4.2.3, v případě oprav chybných segmentů pak také algoritmus 4.2.4. Při přechodu mapy do následujícího kroku je ze silniční sítě vytvářena struktura křižovatek a silnic. Zde jsou využívány hned čtyři postupy popsany v předchozích kapitolách. Nejdříve je z existující silniční sítě vytvořena síť křižovatek (4.2.5). Poté, co jsou na jednotlivé uzlové body označující křižovatky namapovány reálné křižovatky a jednotlivé křižovatky jsou spojeny silnicemi, lze pro všechny nově vznikající křižovatky určit jejich hranici (4.2.6) a následně pro všechny silnice a jejich oba směry vygenerovat potřebný počet jízdnic pruhů včetně korekcí (4.2.7) a odpovídající počet odbočovacích pruhů (4.2.8) v závislosti na reálných datech křižovatek.

Při přechodu do druhé editační fáze dostává uživatel k dispozici novou sadu funkcí, kterými může automaticky vygenerovanou síť křižovatek a silnic dále upravovat až do stavu, kdy je uliční graf připraven k exportu do výstupních souborů. Jednotlivé funkce jsou uvedeny v tabulce 4.4.

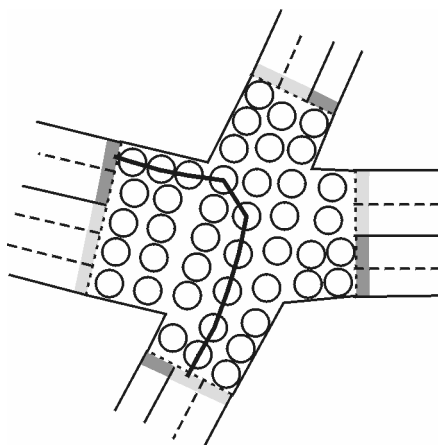
I ve funkcích druhého editačního kroku jsou používány některé popsany algoritmy. Jedná se o algoritmus pro automatické generování buněk v křižovatce (4.2.9), který je používán ve stejnojmenné funkci. Poslední popsany algoritmus 4.2.10 pro nalezení vztahu křižovatky a bodu daného souřadnicemi kurzoru je zapotřebí hned v několika funkcích.

Protože buňky v křižovatce patří vždy některé z existujících křižovatek, nelze vkládat nové buňky mimo oblast některé z křižovatek nebo při posunu umístit buňku jinam než do téže křižovatky. V těchto případech je vhodné tento algoritmus využívat.

Název funkce	Stručný popis	Str.
Přidat nový bod	Vloží do mapy nový uzlový bod a umístí ho na pozici kliknutí.	7
Smazat uzlový bod	Smaže zvolený uzlový bod z mapy. Jestliže byl daný uzlový bod spojený s jedním nebo více segmenty silnic, jsou odstraněny i ty.	7
Posun uzlového bodu	Umožňuje posun uzlového bodu. Stiskem levého tlačítka je daný bod uchopen, poté se spojitě s pohybem myši mění jeho umístění a po uvolnění tlačítka je umístěn na novou pozici v uličním grafu.	7
Přesné zadání souřadnic uzl. bodu	Nastavuje souřadnice zvoleného uzlového bodu na přesnou pozici danou hodnotami zadanými ve vyvolaném dialogovém okně.	8
Namapovat existující křižovatku	Slouží pro namapování reálné křižovatky, načtené ze vstupních dat, na uzlový bod silniční sítě. Pro výběr křižovatek je použitý speciální dialog.	8
Označit bod jako spojovací	Vloží na uzlový bod, který spojuje dva segmenty silnic, grafickou značku a označí ho tak za spojnicu těchto dvou segmentů. Během přechodu do druhého kroku editace mapy jsou takto spojené segmenty spojené do jedné silnice.	10
Smazat segment silnice	Smaže segment silnice z mapy.	10
Přidat segment silnice	Vloží do mapy segment silnice. Segmenty je možné vkládat pouze mezi existující uzlové body.	10
Opravit chybný segment	Opraví segment, který byl při načítání vstupních dat (z důvodu neúplnosti dat) načtený jako chybný.	11
Smazat chybné segmenty	Umožňuje vymazání všech chybných segmentů z mapy, jestliže je v mapě obsažený alespoň jeden chybný segment.	11
Přejít do následujícího kroku	Jestliže je uliční graf v pořádku a neobsahuje žádné závažné chyby, slouží tato funkce pro přechod do další editační fáze. Před přechodem provede editor několik kontrol. Jestliže jsou chyby závažné, musí je uživatel před přechodem upravit, u méně závažných chyb nabízí editor automatickou korekci.	11

Tab. 4.3: Seznam funkcí pro práci s elementy mapy pro první editační krok.

Podstatná je také funkce pro přidání cesty do křižovatky. Cesty je nutné do křižovatek přidávat jako definice směrů jízdy vozidel křižovatkou v průběhu simulace. K definování cest v křižovatkách slouží buňky rozmístěné v křižovatce, emitory a akceptory. Definici cesty je nutné zahájit v některém z emitorů. V okamžiku zahájení vytváření cesty je do emitoru nová cesta uložena a nadále upravována. Po volbě začátku cesty (emitoru) je nutné pokračovat v určení buněk, které budou do nově vznikající cesty patřit. Buněk může být v cestě zadáno libovolné množství, vždy však musí být alespoň jedna. Pro uzavření cesty musí uživatel zadat některý z akceptorů. Jeho volbou je cesta ukončena. Příklad jedné cesty definované v křižovatce je uveden na obr. 4.25.



Obr. 4.25: Příklad definice jedné cesty pro určení směru jízdy vozidel křižovatkou.

Občas je nutné vyhledávat v mapě i jiné objekty než křižovatku nebo buňku v křižovatce (jedná se zejména o emitory a akceptory). V těchto případech je použitý stejný algoritmus jako při vyhledávání křižovatek, pouze s tím rozdílem, že kontrola neprobíhá pro několik konvexních částí nekonvexního celku, ale pozice je porovnávána jen vzhledem k jediné konvexní oblasti. Z tohoto důvodu jsou všechna propojovací místa (emitory, akceptory) ukládána s parametry křivek procházejících hranami zobrazovaných čtyřúhelníků.

Název funkce	Stručný popis	Str.
Generovat buňky v křižovatce	Automaticky vygeneruje buňky pro definování směru jízdy křižovatkou ve všech křižovatkách obsažených v mapě.	11
Přidat buňku do křižovatky	Vloží novou buňku do křižovatky na pozici určenou kliknutím myši.	12
Smazat buňku z křižovatky	Smaže označenou buňku z křižovatky. Mazat lze pouze takové buňky, přes které není definována žádná cesta.	12
Posunout buňku v křižovatce	Slouží pro upravení pozice buňky v křižovatce. Stiskem levého tlačítka myši je buňka uchopena, poté je její pozice plynule měněna podle pozice myši a uvolněním tlačítka je určena nová pozice. Uživateli není umožněno posunout buňku mimo oblast křižovatky.	12
Vytvořit cestu pro průjezd křižovatkou	Umožňuje zadání cesty pro definování směru průjezdu křižovatkou. Zadávání probíhá v pořadí: emitor některého z jízdních pruhů (ze kterého vozidla do křižovatky vjíždí), jedna nebo více buněk definovaných v křižovatce, akceptor jízdního pruhu křižovatky (do kterého vozidla z křižovatky vjíždí).	12
Smazat cesty z emitru	Smaže všechny definované cesty vedoucí z vybraného emitru.	13
Zobrazit/skrýt cesty z emitru	Slouží pro skrytí a opětovné zobrazení cest definovaných v křižovatce pro zvolený emitor. Použití této funkce zvyšuje přehlednost křižovatky při definování směru pro průjezd vozidel.	13

Tab. 4.4: Seznam funkcí pro práci s elementy mapy pro druhý editační krok.

### 4.3.4 Nástroje

Tyto funkce slouží pro nejrůznější uživatelská nastavení. Jejich průběh tak spočívá z větší části v nastavování nejrůznějších parametrů a vlastností jednotlivých vykreslovacích nastavení, změně kurzorů, skrývání a zobrazování ovládacích prvků a komponent grafického okna.

Název funkce	Stručný popis	Str.
Zobrazit šipky	Určuje, zda má být šipková růžice zobrazena nebo ne.	13
Toolbary	Nabízí uživateli skrytí/zobrazení jednotlivých nástrojových lišt, popřípadě všech toolbarů najednou.	13
Kurzor	Umožňuje volbu kurzoru nezávisle na automatickém nastavení kurzoru.	13
Souřadnice	Slouží pro nastavení zobrazování souřadnic při pohybu kurzoru po kreslicí ploše. Lze volit mezi reálnými geodetickými souřadnicemi, souřadnicemi kreslicí plochy a úplným vypnutím zobrazování souřadnic.	14
Nastavení uživatelského rozhraní	Slouží pro vyvolání dialogu umožňujícího nejrůznější nastavení uživatelského prostředí (barva, velikost, styl, ...). Pro obě editační fáze se dialogové okno liší.	14

Tab. 4.5: Seznam funkcí pro uživatelská nastavení (nástroje).

### 4.3.5 Náповěda

V této skupině funkcí je k dispozici pouze jediná – funkce pro otevření dialogového okna s informacemi o programu, jeho autorovi a verzi.



## 5 Implementace editoru

Grafický editor je implementován v programovacím jazyce Java 1.5.0. Všechny vytvořené třídy jsou zanořeny do balíku `jut.s`. Vytvořené třídy lze rozdělit do čtyř základních skupin:

- grafické třídy, pro každé dialogové okno je vytvořena samostatná třída,
- důležité negrafické třídy, které nepopisují konkrétní element mapy, ale jsou zásadní pro základní funkčnost editoru,
- třídy pro jednotlivé elementy mapy, jako jsou silnice, jízdní pruh, křižovatka atd.,
- pomocné třídy, například pro bod obsahující reálné souřadnice apod.

Pro každou třídu obsahuje následující popis vysvětlení jejího základního významu a popis důležitých funkcí. Přehled všech funkcí implementovaných v jednotlivých třídách je k dispozici v podobě automaticky generované dokumentace na přiloženém CD.

Pro vytvoření datové struktury editované mapy je mezi instancemi tříd reprezentujících jednotlivé elementy vytvořen systém odkazů a propojení. Základem celé sítě jsou uzlové body a segmenty silnic v prvním editačním kroku a křižovatky a silnice v druhé etapě. Mezi nimi jsou vždy vytvořena spojení odpovídající skutečným návaznostem v silniční síti. Ostatní komponenty jsou pak postupně navazovány na silnice nebo křižovatky a celá struktura se tak logicky rozrůstá. Ke křižovatkám jsou přiřazeny seznamy buněk definovaných v křižovatce, emitory, akceptory, struktura popisující reálnou křižovatku namapovanou na křižovatku silniční sítě, tvořenou uzlovým bodem. K jednotlivým silnicím pak náleží příslušná ramena křižovatek, struktury jízdních pruhů, generátory, terminátory a další.

Jednotlivé třídy využívají standardní třídy pro zpracování XML, práci s grafickými komponentami a obsluhou událostí, jejich členské metody, proměnné a konstanty. Pro vytvoření grafického uživatelského rozhraní byly použity grafické komponenty balíku Swing.

### 5.1 Grafické třídy

V programovacím prostředí Java je pro každý používaný dialog nutné vytvořit samostatnou třídu oddělenou od některé z existujících kontejnerových komponent.

Pro grafický editor jsem se rozhodla vytvořit hlavní okno programu zděděním od třídy `JFrame`, která pro okno vytvoří v pravém horním rohu tři standardní funkční tlačítka pro změnu velikosti a zavření okna. Ostatní okna v editoru, sloužící pro různá nastavení apod., jsou odvozena z třídy `JDialog`, mají tedy ve svém pravém horním rohu pouze tlačítko pro zavření. Všechna dialogová okna jsou nastavena jako modální, tzn. že po otevření dialogu se hlavní okno stává neaktivním až do opětovného uzavření dialogového okna.

#### 5.1.1 GAboutDlg

Třída `GAboutDlg` slouží pro vytvoření dialogového okna s informacemi o verzi programu a autorovi. Text v popisu informací (`textLB`) je popsán ve formátu HTML. Tlačítko (`okB`) slouží pro uzavření okna.

### 5.1.2 GColorDlg

Popisuje dialogové okno pro nastavení barvy tlačítka, předaného v parametru konstruktoru. Tato třída je volána při nastavování uživatelského rozhraní.

Pomocí tří existujících tlačítek umožňuje: zavřít dialog, aniž je provedena jakákoliv změna (`zavritB`), použít vybranou barvu a nechat okno otevřené (`pouzitB`), nebo nastavit barvu pozadí tlačítka na barvu zvolenou v dialogu barev (`colorChooser`) a dialogové okno zavřít (`okB`).

### 5.1.3 GChooseCrossroadDlg

Třída obsahuje dialogové okno pro výběr reálné křižovatky ze seznamu načteného ze vstupního souboru při mapování křižovatky na uzlový bod. Kromě tlačítek pro zrušení namapování (`noB`), zavření okna (`zavriB`), nebo potvrzení výběru (`okB`) obsahuje také tlačítko pro znovunačtení XML souboru s popisem křižovatek (`reloadB`).

Obsluha znovunačítání XML souboru využívá přetíženou variantu konstrukturu třídy `CrossroadsDOM` (viz kapitola 5.2.1). Po opětovném načtení, stejně jako při samotném vytváření dialogového okna, je do komponenty `JList` (`prvkyL`) načten seznam všech křižovatek obsažených ve vstupním souboru. Z těch je možné při mapování na uzlový bod vybírat.

Proměnná `state` určuje, zda byla pro namapování na uzlový bod zvolena křižovatka (`state = true`), nebo bylo použito tlačítko pro zrušení namapování křižovatky (`noB`). V takovém případě `state = false`.

### 5.1.4 GNewMapDlg

Dialogové okno tvořené třídou `GNewMapDlg` slouží pro otevírání souborů při vytváření nové mapy.

Pro každý ze tří načítaných souborů je určena jedna řádka v dialogovém okně. Každá řádka obsahuje kromě popisku, vyjadřujícího význam načítaného souboru, také textové pole pro přímé zadávání jména souboru a tlačítko pro vyvolání dialogu pro výběr souboru. Pro zadávání souborů jsou použité standardní dialogy systému – `JFileDialog`. Jestliže je soubor zadáván přes dialog, po ukončení výběru se jméno objeví v textovém poli. Jestliže je jméno souboru zadáváno přímo do textového pole, dialog zaregistruje tuto změnu po stisku tlačítka `Enter`, nebo přesunu kurzoru na jiný ovládací prvek. Dialog musí zaregistrovat změnu proto, aby v případě, že již byla vyplněna všechna textová pole, umožnil použití tlačítka `OK`. Pokud je již tlačítko funkční a byla provedena pouze změna v názvu souboru, není `Enter` ani přesun kurzoru nutný.

### 5.1.5 GPointMoveDlg

Dialogové okno vytvářené třídou `GPointDlg` slouží pro přesné zadávání souřadnic uzlovým bodům.

S využitím komponent `JSpin` umožňuje přesné nastavení souřadnice `x` (`spinX`) i souřadnice `y` (`spinY`). Tlačítka slouží pro zavření okna bez jakékoli změny (`zavriB`) a použití nastavených hodnot spolu se zavřením dialogu (`okB`).

### 5.1.6 GSaveMapDlg

Dialogové okno tvořené touto třídou je vzhledově velmi podobné dialogu pro otevírání souborů při vytváření nové mapy. Jeho úkolem je získat od uživatele jména souborů, do kterých má být proveden export vytvořené mapy. Uživatel může zvolit také pouze jedno jméno souboru, v tom případě je pro mapu vytvořen pouze tento jediný soubor.

Tlačítko pro zahájení exportu jednoho nebo obou souborů (`okB`) je funkční pouze v případě, že je zadáno alespoň jedno jméno souboru. To lze zadat buď použitím standardního systémového dialogu pro ukládání souborů, nebo zapsáním jména do připraveného textového pole. Ve druhém případě je změna zaregistrována až v okamžiku, kdy je stisknutý `Enter` nebo je kurzor přesunutý na jiný ovládací prvek, např. do druhého textového pole. Stejně jako v případě dialogu pro načítání souborů slouží zaregistrování změny dialogem k eventuálnímu zpřístupnění tlačítka `OK`. Pro zavření okna slouží tlačítko `zavriB`.

### 5.1.7 GScaleDlg

Toto dialogové okno je určeno pro přesnou změnu měřítka zobrazované mapy.

`JComboBox` (obsažený v okně) pro nastavení nové hodnoty měřítka (`scaleCB`) nabízí uživateli několik možných hodnot pro nastavení měřítka, uživatel však může zadat i libovolnou hodnotu. Předdefinované i uživatelem zadávané hodnoty jsou vždy celočíselné. Je-li zadaná hodnota záporná nebo nulová, je měřítko mapy automaticky nastaveno na 1:1. Okno samozřejmě opět obsahuje tlačítka pro potvrzení nastavení (`okB`) a zavření dialogu bez použití nastavených hodnot (`zavriB`).

### 5.1.8 GUserInterfaceDlg

Třída `GUserInterfaceDlg` reprezentuje dialog pro nejrůznější nastavení uživatelského prostředí. Podoba okna se liší v závislosti na aktuální fázi editace mapy, dialog obsahuje vždy takové prvky, které s danou fází souvisí.

Okno obsahuje velké množství ovládacích prvků, které pracují víceméně na podobném principu. Jejich význam je patrný z pojmenování příslušných proměnných. Nemá proto příliš velký význam popisovat jednotlivě všechna tlačítka a jiné komponenty, za vhodnější považují vysvětlení základních principů.

Protože dialog slouží pro nastavení parametrů pro vykreslování, často se pracuje s nastavením barvy. To je realizováno pomocí malého čtvercového tlačítka vykresleného barvou, odpovídající aktuálně nastavené barvě. Stisknutím tohoto tlačítka je vyvolán dialog pro nastavení barvy (viz 5.1.2), pomocí něhož lze barvu změnit.

Častým prvkem objevujícím se v okně je také zaškrtačací políčko komponenty `JCheckBox`. Každý element mapy, u kterého je možné nastavovat a měnit vlastnosti pro zobrazení, je zároveň možné zcela skrýt nebo ho opětovně nastavit jako zobrazovaný. Nastavení se provádí změnou `boolean` konstant umístěných ve třídě `JUTS_constants` (5.2.4).

Je-li kdekoli zapotřebí nastavit velikost nebo jiný číselný rozměr, je pro to využita komponenta `JSpinner`. V závislosti na nastaveném kroku a minimální a maximální hodnotě může uživatel nastavovat např. velikost značky pro zobrazení uzlových bodů (`sizeCrossPoints`), tloušťku čáry zobrazující segmenty (`widthRoadSegmentS`), popř.

tloušťku hraniční čáry pro vykreslení silnic a křižovatek (`lines`). Číselně nastavovanými parametry jsou také kroky pro zoomování (`zoomStepS`) a posun mapy (`moveStepS`).

Samostatnou nastavitelnou vlastností uživatelského prostředí je nastavení cesty k pracovnímu adresáři. Jako implicitní hodnota na začátku běhu programu je tato proměnná nastavena na vrchol adresářové struktury, ze které byl program spuštěn. Po změně cesty, volbou adresáře ve standardním dialogu nebo přímým zadáním cesty do textového pole (`pathTF`), je hodnota nastavena jako systémová vlastnost:

```
System.setProperty("user.dir", pathTF.getText());
```

Pokud není uživatel se změnou nastavení spokojený, může se použitím tlačítka pro nastavení standardních hodnot (`standardB`) vrátit k defaultním hodnotám. Toto je zajištěno obslužnou třídou k již zmiňovanému tlačítku. Všechny parametry vykreslování jsou nastaveny na hodnoty dané konstantami třídy `JUTS_constants`.

Samotné použití nastavených hodnot je zajištěno metodou `useSetting()`. Ta upravuje všechny parametry v závislosti na nově zvolených hodnotách. Metoda je volána ve dvou případech – při stisku tlačítka `useB`, kdy zůstává dialogové okno otevřené, a při potvrzení nastavených hodnot tlačítkem `okB`.

### 5.1.9 GWindow

Třída `GWindow` je hlavní třídou celého programu, vytváří hlavní okno grafického editoru a plně zajišťuje jeho funkčnost obsluhou všech grafických ovládacích prvků. V této třídě je také umístěna metoda `main()`, která slouží ke spuštění celého programu. Ta spustí program vytvořením hlavního okna s pracovní plochou a potřebnými ovládacími prvky.

Převážnou většinu členských proměnných tvoří grafické komponenty – menu, jeho položky, toolbary se všemi potřebnými tlačítky, obě popup menu a jejich položky, šipková růžice a samozřejmě také samotná kreslicí plocha, která je umístěna do komponenty `JScrollBar`. Ta umožňuje posun mapy v případě, že je mapa zobrazovaná na kreslicí ploše větší než samotná kreslicí plocha. Kromě proměnných pro jednotlivé ovládací prvky však třída obsahuje i řadu dalších vlastností a metod:

- měřítko zobrazení celé mapy – `scale`,
- proměnnou uchovávající všechna data – `data`,
- parametry a příznaky pro vykreslování, určující, zda mají být vykreslovány některé z pomocných prvků; tyto parametry také určují přesnou činnost funkce obsluhující pohyb myši apod.  
(`movingCrossPoint`, `drawCrossPoint`, `drawCursorLines`, `movingPlace`, `drawPlace`, `creatingWay`, `arrowHandActive`, `chooseRegionPoint`, `chooseRegion`),
- pomocné proměnné pro dočasné uchování souřadnic pohybu kurzoru, pro uchování původních hodnot pro případ zrušení prováděné operace apod.  
(`x1`, `y1`, `x2`, `y2`, `clickedX`, `clickedY`, `oldMouseX`, `oldMouseY`, `oldPlace`),
- jména souborů (`fileNameSegments`, `fileNamePoints`, `fileNameXML`),
- proměnné pro uložení aktuálně vybraných elementů mapy (např. `crossroadActual`, `pActual`, `emitterActual`, `acceptorActual`, `wayActual`, ...),
- proměnné určující, jaká operace je aktuálně zvolena: pro první krok – `operation`, pro druhý krok editace mapy – `operation2`,
- parametr udávající, ve kterém editačním kroku se mapa právě nachází – `step`,

- pole pro uchování identifikátorů křižovatek, které už byly do systému načteny z XML souboru/souborů – `indexes`,
- seznam křižovatek načtených z XML souboru/souborů – `crossroadsDOM` a další.

Pokud je zapotřebí, aby byla proměnná dostupná i vně třídy, popř. aby bylo možné ji mimo třídu měnit, je pro ni vytvořena metoda `get`, popř. `set`.

Třída obsahuje řadu metod, z nichž některé slouží pro samotné vytváření grafického rozhraní. Protože je vytváření celého okna a všech ovládacích prvků v něm poměrně rozsáhlé, slouží tyto metody k celkovému přehlednění.

Protože editor pracuje ve dvou souřadných systémech, jsou velmi časně přepočty mezi nimi, především přepočty ze souřadné soustavy kreslicí plochy do systému reálných souřadnic. K tomu jsou určeny metody `map2realX()` pro x-ovou složku a `map2realY()` pro y-ovou část.

Editor umožňuje různá barevná nastavení, proto je občas zapotřebí změnit barvu všech elementů jednoho typu najednou. K tomu jsou pro jednotlivé elementy určeny příslušné funkce (např. `allCrossPoints()`, `allRoadSegments()`, `allPlaces()`, apod.). K dispozici je i podobná funkce pro tlačítka funkcí v nástrojových lištách editoru (`allButtons()`).

Do další skupiny patří metody pro zajištění funkčnosti jednotlivých tlačítek (např. `closeOperation()`, `deleteCrossPoint()`, `moveCrossPoint()`, `repareRoadSegment()`, `zoomIn()`, ...). Ty jsou pak volány při obsluze konkrétních ovládacích prvků. Funkce `init()` slouží pro hromadné nastavení možnosti použití konkrétních ovládacích prvků.

Pro zajištění funkčnosti jednotlivých ovládacích prvků slouží řada obslužných tříd. Většina z nich volá ve svém průběhu některou z členských metod třídy `GWindow`, popřípadě nastavuje hodnoty příznaků přesně určující funkci myši pro aktuálně nastavenou operaci. Pro obsluhu funkce myši pro komponentu `JScrollPane`, ve které je umístěna kreslicí plocha, jsou vytvořené potřebné třídy pro obsluhu pohybu myši a stisku jejích tlačítek.

Pro obsluhu pohybu myši je určena třída `MMScrollPane`. Překrytím její metody `mouseMoved()` je zajištěna veškerá funkčnost editoru týkající se pohybu myši po kreslicí ploše. V závislosti na nastavených parametrech, udávajících, zda je právě zapnuta operace přidávání nového bodu nebo buňky v křižovatce, zda se přidává cesta do křižovatky nebo jestli je posouváno s některým z objektů apod., určuje nastavení k tomu určených členských proměnných třídy `GWindow`.

Na podobném principu jako obsluha pohybu myši (tzn. funkčnost závislá na nastavení parametrů) jsou zajištěny i obsluhy ostatních operací prováděných při používání myši. K tomuto účelu slouží třída `MLScrollPane`, jejímž metodám `mouseExited()`, `mouseEntered()`, `mousePressed()`, `mouseReleased()` a `mouseClicked()` je přetížením přidělena konkrétní potřebná funkčnost.

### 5.1.10 MyPanel

Třída `MyPanel` slouží pro vykreslování mapy do oblasti kreslicí plochy. Třída je zděděná od komponenty `JPanel`, pro ovládání zobrazování bylo nutné překrýt metodu `paintComponent()`.

V překryté metodě jsou pak s využitím metod 2D grafiky a v závislosti na nastavení jednotlivých konstant určujících, které komponenty mají být zobrazeny a které nezobrazovat, vykreslovány všechny požadované elementy mapy. Tyto konstanty a jejich významy jsou podrobněji popsány v kapitole 5.2.4 v popisu třídy `JUTS_constants`.

V závislosti na nastavení některých proměnných hlavního okna mohou být vykreslovány také pomocné prvky, jako např. obdélník pro výběr regionu, svislá a vodorovná linka při nastavení souřadnicového kříže, pomocné značky při vytváření nebo přesouvání elementů mapy apod.

## 5.2 Podstatné negrafické třídy

Do zvláštní skupiny tříd jsem se rozhodla vyčlenit třídy pro manipulaci s XML soubory a jiné negrafické třídy, které jsou pro implementaci grafického editoru zásadní. Jedná se o třídu zajišťující popis komponent ze souboru, obalující třídu pro všechny konstanty, které jsou v editoru používány, a nakonec třídu uchovávající všechna potřebná data a struktury.

Aby zůstal zachovaný základní princip objektově orientovaného programování, jsou všechny členské proměnné nastaveny jako `private`, každá třída pak obsahuje řadu `get` a `set` funkcí pro manipulaci s členskými proměnnými. V implementaci se často používají seznamy pro uložení elementů stejného typu. Ve všech případech jsou použity kolekce, konkrétně instance třídy `ArrayList`.

### 5.2.1 CrossroadsDOM

Třída `CrossroadsDOM` slouží pro parserování vstupního souboru s popisy reálných křižovatek. Během načítání je soubor zpracováván a jednotlivé elementy XML struktury jsou ukládány do datových struktur křižovatek. Přesná struktura XML souboru je popsána v kapitole 2.2.3 – Popis křižovatek.

Třída má k dispozici dva přetížené konstruktory. První obsahuje jediný parametr a slouží pro načítání prvního XML souboru (tzn. načítání souboru v okamžiku, kdy ještě nebyl načten žádný jiný soubor s popisem křižovatek). Druhý konstruktor při načítání křižovatek kontroluje, zda již některá z aktuálně načítaných křižovatek nebyla obsažena v předchozím souboru. Křižovatka je načtena pouze v případě, že se v seznamu všech reálných křižovatek vytvořených ze vstupního souboru nevyskytuje křižovatka se stejným identifikátorem. Přidávané křižovatky jsou umístovány vždy na konec seznamu.

Ostatní privátní metody slouží k samotnému parserování souboru, k jeho zpřehlednění a zvýšení modularity.

### 5.2.2 GraphicXML

Třída `GraphicXML` slouží pro vytvoření prvního ze dvou výstupních souborů – grafické části vytvořené mapy ve formátu XML, splňujícím požadovanou strukturu.

Po vytvoření potřebných instancí tříd pro zapisování XML do souboru zadaného jménem v parametru konstruktoru vypisuje třída s využitím několika privátních metod data do souboru XML. Data jsou třídě dodána druhým parametrem konstruktoru. Přesný popis výstupního souboru je uveden v kapitole 2.2.4 – Grafická část výstupního XML popisu.

### 5.2.3 GDescription

Jméno třídy `GDescription` začíná písmenem G, což by mělo značit, že se jedná o třídu reprezentující grafické okno programu. Není tomu tak, přesto má jméno třídy svůj význam. Jedná se o třídu zajišťující poskytování popisků pro všechna grafická okna programu.

Ve svém konstruktoru získává jméno souboru s jednotlivými popisky a inicializuje hashmapu, ve které budou popisky uloženy. Třída dále nabízí dvě metody pro tvorbu a používání popisků:

- `readFile()` – metoda pro samotné načtení souboru a uložení jednotlivých popisků do hashmapy ve formě dvojic klíč:hodnota
- `get(String key)` – metoda pro získávání popisků z hashmapy podle klíče zadaného v parametru.

Pro názornou představu použití je možné uvést příklad. Je-li popisek uložen v souboru jako `titleGWindow>>Editor JUTS`, je možné ho použít `desc.get("titleGWindow")`, kde `desc` je instance třídy `GDescription`. Každý popisek je v souboru umístěn na samostatné řádce ve formátu `klíč>>hodnota`.

Tímto způsobem jsou popisovány všechny ovládací prvky všech oken editoru, tzn. že pokud by při spouštění editoru nebyl soubor s popisky na svém místě v adresářové struktuře, budou ovládací prvky vytvořeny bez popisku (prázdné).

### 5.2.4 JUTS\_constants

Třída `JUTS_constants` v sobě zapouzdřuje všechny konstanty a nastavení používané v rámci celého editoru. Jedná se především o nastavení parametrů pro vykreslování jako jsou barvy, velikosti objektů, tloušťky čar apod. `JUTS_constants` obsahuje také konstanty pro určení typu křivky nebo rozlišení propojovacích míst na emitery a akceptory, konstanty označující generátor, terminátor a další. Podrobnější přehled konstant s vysvětlením jejich významu je uveden v tabulce 5.1.

Název konstanty	Stručný popis významu	Hodnota
<code>sttBackgroundColor</code>	Barva pozadí.	<code>Color.white</code>
Nastavení vlastností uzlového bodu:		
<code>sttColorCrossPointA</code>	Barva aktivního uzlového bodu.	<code>Color.red</code>
<code>sttColorCrossPointIA</code>	Barva neaktivního uzlového bodu.	<code>Color.blue</code>
<code>sttColorCrossPointMaped</code>	Barva uzlového bodu s nastavenou funkčností.	<code>Color.green</code>
<code>sttPointSize</code>	Velikost značky pro zobrazení uzlového bodu.	10
<code>sttShownCrossPoint</code>	Zobrazení uzlových bodů.	<code>true</code>
Nastavení vlastností segmentu silnice:		
<code>sttColorRoadSegmentA</code>	Barva aktivního segmentu.	<code>Color.pink</code>
<code>sttColorRoadSegmentIA</code>	Barva neaktivního segmentu.	<code>Color.black</code>
<code>sttColorRoadSegmentFail</code>	Barva chybného segmentu.	<code>Color.red</code>
<code>sttWidthRoadSegment</code>	Tloušťka čáry pro zobrazení segmentu.	3.0f
<code>sttStyleRoadSegment</code>	Styl čáry pro vykreslení segmentu (význam hodnoty viz dále – typy čar).	1
<code>sttShownRoadSegment</code>	Zobrazení segmentů silnic.	<code>true</code>
Nastavení parametrů pro vykreslování a výpočty křižovatek a jízdnic pruhů:		
<code>sttLineColor</code>	Barva čáry pro vykreslení obrysů silnic a křižovatek.	<code>Color.black</code>
<code>sttWidthLine</code>	Tloušťka čáry pro vykreslení obrysů silnic a křižovatek.	3
<code>sttColorCrossroadA</code>	Barva aktivní křižovatky.	<code>Color.blue</code>
<code>sttColorCrossroadIA</code>	Barva neaktivní křižovatky.	<code>Color.lightGray</code>
<code>sttShownCrossroad</code>	Zobrazení křižovatek.	<code>true</code>
<code>crossroadSizeLimit</code>	Minimální poloměr kružnice křižovatce opsané. Hodnota musí být z důvodu prováděných výpočtů větší než šířka jízdnic pruhu.	4000f

sttColorRoadA	Barva pro vykreslení aktivní silnice.	Color.blue
sttColorRoadIA	Barva pro vykreslení neaktivní silnice.	Color.lightGray
sttShownRoad	Zobrazení silnic.	true
turnLaneLength	Délka jízdního pruhu pro odbočení.	30000f
turnLaneLengthDiff	Posunutí sousedních jízdních pruhů pro výpočet odbočovacích pruhů (viz obr. 4.16 – $\Delta l$ ).	3000f
vehicleLaneWidth	Šířka jízdního pruhu.	3000f
<b>Nastavení generátorů a terminátorů:</b>		
generator	Instance třídy GenTerm je generátor.	0
terminator	Instance třídy GenTerm je terminátor.	1
generatorID	Prefix pro identifikátor generátoru.	"140"
terminatorID	Prefix pro identifikátor terminátoru.	"150"
sttColorGenerators	Barva generátorů.	Color.green
sttShownGenerators	Zobrazení generátorů.	true
sttColorTerminators	Barva terminátorů.	Color.magenta
sttShownTerminators	Zobrazení terminátorů.	true
<b>Nastavení vlastností pro buňky v křižovatce:</b>		
placeID	Prefix pro identifikátor buňky v křižovatce.	"121"
sizeP	Průměr kružnice znázorňující buňku v křižovatce. Hodnota se používá pro vykreslení i pro výpočty.	2500
sttColorPlaceA	Barva pro zobrazení aktivní buňky.	Color.yellow
sttColorPlaceIA	Barva pro zobrazení neaktivní buňky.	Color.red
sttShownPlaces	Zobrazení buněk v křižovatce.	true
<b>Nastavení emitörů a akceptorů:</b>		
acceptor	Instance třídy AccessPlace je akceptorem vzhledem ke křižovatce.	0
emitter	Instance třídy AccessPlace je emitorem vzhledem ke křižovatce.	1
accessPlaceID	Prefix pro identifikátor akceptoru a emitöru.	"210"
sttColorAcceptorA	Barva pro zobrazení aktivního akceptoru.	Color.red
sttColorAcceptorIA	Barva pro zobrazení neaktivního akceptoru.	Color.orange
sttShownAcceptors	Zobrazení akceptorů.	true
sttColorEmitterA	Barva pro zobrazení aktivního emitöru.	Color.red
sttColorEmitterIA	Barva pro zobrazení neaktivního emitöru.	Color.cyan
sttShownEmitters	Zobrazení emitörů.	true
<b>Typy čar:</b>		
solidLine	Plná.	0
dashedLine	Přerušovaná.	1
doubleLine	Dvojitá.	2
doubleLeftLine	Dvojitá, přerušovaná zleva.	3
doubleRightLine	Dvojitá, přerušovaná zprava.	4
<b>Určení parametru pro vykreslování ve výstupních souborech:</b>		
show	Zobrazovat.	1
unshow	Nezobrazovat.	0
<b>Barvy tlačítek:</b>		
sttButtonColor	Standardní barva neaktivního a nestisknutého tlačítka (její hodnota je nastavena při spuštění programu na defaultní barvu komponenty JButton).	-
sttActiveButtonColor	Barva stisknutého tlačítka.	Color.lightGray
<b>Nastavení kroků:</b>		
sttMoveStep	Krok pro posun mapy šipkovou rúžicí.	20
sttZoomStep	Krok pro zoomování mapy.	1,5f

Tab. 5.1: Seznam konstant obsažených ve třídě JUTS\_constants.



Třída obsahuje i řadu konstant, které zatím nejsou grafickým editorem využívány (např. barva aktivní silnice, typy čar apod.). Tyto proměnné a konstanty byly vytvořeny z důvodu celistvosti programu a jsou připraveny pro další rozšiřování editoru.

### 5.2.5 Main

Třída `Main` slouží pro uchování všech potřebných dat a struktur týkajících se mapy. Jsou zde uloženy souřadnice pro určení hranice mapy: `mapRight`, `mapTop`, `mapLeft` a `mapBottom`. Aby nebyly elementy umístěné na kraji mapy zobrazované na úplném okraji, je konstantou `mapMargin` určen nevyužitý okraj, o který je mapa zvětšena.

Ostatní členské proměnné třídy `Main` slouží k uchování seznamů prvků mapy. Uloženy jsou zde seznamy všech segmentů silnic (`roadSegments`) a všech uzlových bodů (`points`) načtených ze vstupního souboru, seznam silnic (`roads`), akceptorů (`acceptors`), emitörů (`emitters`), generátorů (`generators`), terminátorů (`terminators`) a také seznam všech cest definujících směr jízdy křižovatkami (`ways`). Tyto seznamy samozřejmě nemohou zachycovat žádnou logickou strukturu mapy, slouží však zejména pro vykreslování a vyhledávání, které je zapotřebí provádět se všemi elementy jednoho typu najednou.

Zbývá popsat poslední dvě členské proměnné. První z nich, `segmentNumber`, slouží pro uchování celkového počtu všech elementů mapy, ve kterých se může vyskytovat vozidlo (jedná se o silnice, křižovatky, generátory a terminátory). Druhá proměnná, `roadsOK`, určuje, zda mapa (pokud se nachází v prvním editačním kroku) obsahuje nějaké chybné segmenty silnic.

Při vytváření instance třídy `Main` jsou v jejím konstruktoru načteny ze vstupních souborů všechny uzlové body (metoda `savePoint()`) a segmenty silnic (metoda `saveRoad()`). Třída dále poskytuje několik metod pro práci s mapou.

Jako první je využívána metoda pro vytvoření silniční sítě z uzlových bodů a segmentů silnic - `createNet()`. Zde je implementován algoritmus popsaný v kapitole 4.2.1. Dále je zde několik metod pro vyhledávání elementů v mapě v závislosti na souřadnicích zadaných jako parametry metod:

- `nearestCrossPoint()` pro vyhledání nejbližšího uzlového bodu (algoritmus 4.2.2),
- `getNearestP()` pro nalezení nejbližší buňky v křižovatce k zadaným souřadnicím,
- `getCrossRoad()` sloužící k nalezení křižovatky, ve které se nachází bod daný souřadnicemi (algoritmus 4.2.10),
- `getAccessPlace()` určí, zda se bod zadaný souřadnicemi nachází v oblasti některého z akceptorů nebo emitörů.

### 5.2.6 MapXML

Jedná se o druhou třídu pro vytváření výstupního XML souboru. Tentokrát jde o simulační část mapy.

Funkčnost celé třídy je velmi podobná funkčnosti třídy `GraphicXML` (5.2.2). Po všech potřebných nastaveních a vytvoření potřebných instancí tříd DOMu je zahájen výpis dat zadaných parametrem do souboru, jehož jméno je také zadáno parametrem konstruktoru. Přesný formát struktury XML ve vypisovaném souboru je uveden v kapitole 2.2.5 – Simulační část výstupního XML popisu.

## 5.3 Třídy pro elementy mapy

Pro každý element vyskytující se v editované mapě je vytvořena příslušná třída, která jej reprezentuje. Jednotlivé členské proměnné pak odpovídají vlastnostem konkrétního elementu. Členské proměnné jsou používány jako privátní. Pro umožnění přístupu z okolí přísluší ke každé proměnné funkce `set`, `get`, v případě seznamů i funkce pro manipulaci s jednotlivými prvky, je-li to zapotřebí.

### 5.3.1 AccessPlace

Třída `AccessPlace` je určena pro reprezentaci propojovacího místa mezi křižovatkou a jízdním pruhem. Členské proměnné třídy tedy odpovídají všem parametrům, které je nutné pro jednotlivá místa ukládat.

Několik proměnných slouží pro uchování grafických vlastností propojovacího místa: souřadnice, na kterých je místo v mapě (`x`, `y`), body polygonu pro vykreslení místa do mapy (`points`) a seznam parametru analytického vyjádření jednotlivých hran polygonu (`lines`) pro rychlejší vyhledávání propojovacího místa v křižovatce. Pro každé propojovací místo je také uložena barva (`color`) pro jeho vykreslení a parametr (`shown`) určující, zda mají být cesty vedené z propojovacího místa (pokud se jedná o emitore) zobrazeny (`true`) nebo nemají (`false`). V případě, že nemají být zobrazeny, je přes značku emitore vykreslené tenké proškrtnutí.

Každé propojovací místo má v systému svůj identifikátor (`id`) a je určen jeho typ (`type`). To, zda se jedná o akceptor nebo emitore, je určováno vždy vzhledem ke křižovatce. Pro jízdni pruh je nutné k typu přistupovat opačně (viz obrázek 2.16). Je-li propojovací místo emitorem vzhledem ke křižovatce, uchovává v sobě také obrázek pro popis přiléhajícího jízdniho pruhu (`arrow`), jeho natočení (`rotate`) a souřadnice, na které má být vykreslován (`imageX`, `imageY`).

Některé členské proměnné slouží pro uchování logické struktury. Jedná se o seznam cest (`ways`), které jsou pro propojovací místo definovány, jízdni pruh (`laneID`), ke kterému místo připojuje křižovátku (`crossroad`), poslední pozici v přiléhajícím jízdni pruhu (`lastPosition`) a přímého předchůdce (`predecessor`) a následníka (`successor`) propojovacího místa.

### 5.3.2 CrossPoint

Struktura elementu uzlového bodu a následně křižovaty je reprezentována třídou `CrossPoint`. Jedná se asi o nejsložitější třídu pro element mapy. Jsou zde uloženy informace pro uzlový bod od jeho načtení ze vstupního souboru až po křižovátku před exportem mapy do výstupních XML souborů.

Po načtení hodnot ze vstupního souboru je do uzlového bodu uložen identifikátor načtený z dat – `idKriz` (tato hodnota je uložena ze vstupního souboru, ale systém s ní prozatím dále nijak nepracuje) a souřadnice, na kterých se uzlový bod nachází (`x`, `y`). Bodu je také přiřazena barva pro jeho vykreslení (`color`). Po vytvoření silniční sítě jsou k bodu uloženy odkazy na segmenty silnic, které s bodem sousedí (`segments`).

V dalších krocích práce s mapou je bodu přiřazen samotný identifikátor pro systém JUTS (`id`), získaný z křižovaty namapované na bod (`crossroad`), ukládán je styl bodu (`style`), který udává, zda se jedná o křižovátku, spojovací nebo koncový bod. Některé proměnné slouží při generování hranic křižovatek. Jedná se zejména o poloměr kružnice

křižovatce opsané ( $r$ ), hodnoty pro velikosti úhlů ( $\text{minUhel}$ ,  $\text{MaxUhel}$ ) a maximální počet jízdnic pruhů v ramenech křižovatky ( $\text{maxPruhu}$ ).

Jakmile je z uzlového bodu vygenerována křižovatka, je nutné pro ni uchovávat řadu parametrů, např. hraniční křivky ( $\text{curves}$ ), body tvořící vnitřní polygon křižovatky ( $\text{boundaryPoints}$ ), seznam konvexních částí, na které je křižovatka rozdělena ( $\text{parts}$ ), a jejich přímkovou reprezentaci ( $\text{lines}$ ), počty jízdnic pruhů v jednotlivých ramenech ( $\text{lanesNumber}$ ), seznam emitörů ( $\text{emitters}$ ) a akceptorů ( $\text{acceptors}$ ) náležících ke křižovatce, všechny buňky definované v křižovatce ( $\text{places}$ ) a několik dalších.

Pro podrobný popis jednotlivých metod doporučuji nahlédnout do dokumentace k programu na přiloženém CD. Zde zmíním jen nejdůležitější z nich. Metoda `createCrossroads()` slouží pro vytvoření křižovatky z uzlového bodu, vlastnosti uzlového bodu upravují v závislosti na jeho typu také metody pro úpravu bodů určených jako křižovatky (`finishCrossroad()`), koncové body (`finishEndPoint()`) nebo body určené jako spojovací (`finishContactPoints()`). K dispozici je pak také metoda `countCrossroadPolygonPoints()` určená pro výpočet bodů vnitřního polygonu křižovatky, `createCrossRoadParts()` slouží k rozdělení křižovatky na několik konvexních oblastí. K automatickému vygenerování buněk v křižovatce je určena metoda `generatePlaces()`. Ta ke svému běhu využívá řadu privátních metod třídy.

### 5.3.3 GenTerm

Tato třída je určena k reprezentaci generátorů a terminátorů. Informace, zda se jedná o terminátor, nebo generátor, je uložena v proměnné `type`. Dalšími hodnotami, které je nutné u generátorů a terminátorů ukládat, jsou identifikátor (`id`) a souřadnice, na kterých je objekt v mapě umístěn (`x`, `y`).

### 5.3.4 Krizovatka

Struktura definovaná třídou `Krizovatka` v sobě uchovává informace o křižovatce načtené z datového souboru. Takto uložená křižovatka je pak při namapování křižovatky na uzlový bod k danému uzlovému bodu přiřazena.

Jsou zde uchovány informace zahrnující číslo (`cislo`) a popis křižovatky (`popisKrizovatky`), její identifikátor v systému JUTS (`idJUTSKrizovatky_id`), číslo (`soucastOblasti_cislo`) a popis oblasti (`soucastOblasti_popisOblasti`), ve které se skutečná křižovatka rozkládá, a seznam všech ramen (`ramenaKrizovatky`) k ní přiléhajících.

### 5.3.5 NextPlace

Aby bylo možné po definování cest v křižovatkách určit pro každou buňku v křižovatce všechny možné následující směry průjezdu, je pro každou buňku uložen seznam následujících míst (`NextPlaces`). Každé z těchto následujících míst v sobě obsahuje identifikátor následující buňky (`id`) a směr (tzn. propojovací místo), do kterého je možné touto cestou dojet (`direction`).

### 5.3.6 Place

Třída `Place` slouží jako struktura pro popis buňky v křižovatce. Ta je v mapě určena identifikátorem (`id`), a souřadnicemi `x` a `y`. Aby bylo možné barevně odlišovat aktuálně zvolenou buňku, je pro každou buňku v proměnné `color` uložena i barva, kterou je konkrétní buňka vykreslována. Pro každou buňku je zároveň uložen seznam všech míst, do kterých je možné z dané buňky pokračovat v průjezdu křižovatkou (`NextPlaces`). Parametr `free` určuje, zda je buňkou nadefinovaná nějaká cesta, pokud ano, buňku nelze smazat (`free = false`).

### 5.3.7 PruhDovnitř

Tato třída reprezentuje jízdní pruh vedoucí do křižovatky. Jízdní pruh má několik základních vlastností, ty jsou ve třídě uloženy ve formě členských proměnných.

Jedná se o jedinečný identifikátor jízdního pruhu (`id`), zkratku (`zkratka`) popisující směr, kterým mohou vozidla z daného jízdního pruhu pokračovat v jízdě křižovatkou, a její slovní vysvětlení (`popis`) a pořadí jízdního pruhu v silnici ve směru od středu vozovky (`poradiOdLeva`). Pro jízdní pruh je zároveň uchováváno číslo detektoru pro snímání četnosti průjezdu vozidel (`cisloDetektoru`) a vzorec pro výpočet intenzity průjezdu vozidel jízdním pruhem (`namerenaIntenzita`).

### 5.3.8 PruhVen

Struktura jízdního pruhu vycházejícího z křižovatky je mnohem jednodušší než popis pruhu v opačném směru. Třída proto obsahuje pouze dvě členské proměnné – identifikátor jízdního pruhu (`id`) a pořadí pruhu udávané ve směru od středu vozovky (`poradiOdLeva`).

### 5.3.9 RamenoKrizovatky

Tato třída reprezentuje strukturu jednotlivých ramen křižovatek načtených ze vstupního souboru.

Každé rameno je určeno několika parametry: jménem (`jmenoUlice`) a svou orientací (`orientace`) určenou v rozmezí 1 až 12 podle umístění těchto číselic na hodinách. Z datového souboru je načten také identifikátor nejbližší v souboru popsání křižovatky v daném směru (`navazujiciKrizovatka_id`). V každém rameni křižovatky leží dvě silnice – vjezd do křižovatky a výjezd z ní. Pro každou je uložen identifikátor (`vjezd_id`, `vyjezd_id`) a seznam jízdních pruhů, které jsou v ní umístěné (`vjezdy`, `vyjezdy`). Pro připojení ramena křižovatky na datovou strukturu silnice (`Road`) slouží proměnná `road`.

### 5.3.10 Road

Pro uložení silnice ve druhém editačním kroku mapy je určena třída `Road`. Ta má své jméno (`name`) a zahrnuje v sobě dva směry jízdy, které jsou popisovány ve většině případů jako `from` a `to`. Každá silnice má dva uzlové body – dvě křižovatky, popř. křižovatku a koncový bod, které spojuje. Tyto body jsou uloženy jako `crossroadsFrom` a `crossroadsTo`. Protože vytváření sítě křižovatek a silnic probíhá vždy ve směru od některé z křižovatek, je v `crossroadsFrom` vždy uložený uzlový bod určený jako křižovatka. V `crossroadsTo` je pak uložena buď křižovatka, nebo uzlový bod, který

přiléhá pouze k jedinému segmentu, a byl proto označen jako koncový. V každé instanci třídy `Road` jsou tedy uloženy dvě silnice z pohledu výstupního XML: `from` která vede ve směru `crossroadsFrom` → `crossroadsTo`, a `to` v opačném směru. Samozřejmě by bylo možné (a v některých ohledech i jednodušší pro zpracování) vytvořit pro každý směr samostatnou instanci třídy reprezentující silnici ve smyslu elementu výstupních XML souborů. Při návrhu jsem však zvolila silnici ve smyslu jednoho ramene křižovatky, které z křižovatky vychází, a dále jsem se tohoto přístupu držela. Obě varianty přináší své výhody i nevýhody.

Pro každou silnici je tedy uložena řada hodnot pro jednotlivé směry: identifikátory pro oba směry (`idFrom`, `idTo`), seznamy jízdnic pruhů (`lanesFrom`, `lanesTo`), počty směrů na začátku a na konci směru jízdy (`lanesFromStart`, `lanesFromEnd`, `lanesToStart`, `lanesToEnd`) a pořadí koncového segmentu silnice mezi seřazenými segmenty, uloženými v křižovatkách, které daná silnice spojuje (`indexFrom`, `indexTo`).

Pro silnici jsou pak dále ukládány seznamy všech segmentů, ze kterých je silnice vytvořena (`segments`), seznam bodů, které tvoří střed vozovky (`points`), a seznam odchylek ve směru os  $x$  a  $y$  pro urychlení výpočtu a korekci jednotlivých jízdnic pruhů ( $xy$ ). Zároveň je pro silnici uložena její délka, označující, kolik 2,5 metrových úseků by bylo možné do celé délky silnice naskládat (`length`).

Celá silnice je ve směru od jedné křižovatky k druhé vytvářena postupně z na sebe navazujících segmentů silnic v konstruktoru třídy. Další členské metody slouží převážně pro práci s jízdnicími pruhy. Metoda `precountXY()` slouží pro předpočítání hodnot do seznamu odchylek. Ty jsou pak využívány při výpočtu hrubých hranic jednotlivých jízdnic pruhů (`countRoughLanes()`). Hrubě spočítané hranice jsou dále zpřesněny (viz algoritmus 4.2.7) – `finishDirection()`, `finishLanes()` a v případě potřeby vytvoření jízdnic pruhů jsou k dispozici příslušné metody pro úpravu jízdnic pruhů k odbočení doprava `changeLaneTurnRight()` a doleva `changeLaneTurnLeft()`. Všechny jízdnicí pruhy jsou pak s využitím výše popsanych metod vytvořeny metodou `createLanes()`.

### 5.3.11 RoadSegment

Třída `RoadSegment` popisuje segment silnice tvořící základní strukturu silniční sítě v prvním editačním kroku. Během převodu mapy do druhého editačního kroku jsou z jednotlivých segmentů vytvořeny silnice, na segmenty jsou nabaleny další informace. Aby bylo možné při tomto převodu mapy provádět řazení segmentů, implementuje třída `RoadSegment` rozhraní `Comparable` a překrývá třídu `compareTo()` na řazení instancí třídy podle úhlu natočení segmentu. Proměnná `color` slouží pro určení barvy, kterou má být konkrétní segment silnice vykreslován.

Segment je z datového hlediska určen svým identifikátorem (`id`), hodnotami udávajícími ohraničující obdélník segmentu při jeho načítání (`xMin`, `xMax`, `yMin`, `yMax`) a jménem (`name`). K vytvoření logické struktury silniční sítě v prvním editačním kroku mapy slouží uchování uzlových bodů přiléhajících k segmentu (`points`) a proměnná `ok`, udávající, zda byl segment při načítání vstupního souboru načten jako chybný (`false`) nebo bezchybný (`true`).

Při přechodu mapy do druhé editační fáze je ke každému segmentu přiřazeno rameno křižovatky (`ramenoKrizovatky`). Ještě předtím je však nutné spočítat úhel natočení segmentu (`uhel`) a po seřazení segmentů podle tohoto úhlu lze určit úhel, který segment svírá se svým předcházejícím sousedem (`uhelPredchozi`). Ostatní proměnné jsou

určené pro uložení informací o jízdnicích pruzích popsaných v přiřazeném rameni křižovatky. Tyto proměnné slouží k výpočtu hranice křižovatky z uzlového bodu, ke kterému segment přiléhá.

Za zmínku stojí i dvě metody náležící do třídy `RoadSegment`. Jedná se o metodu `split()`, která je určená pro rozdělení segmentu silnice na dvě části, jestliže segment přímo spojuje dva uzlové body označené jako křižovatky (podrobnosti viz kapitola 4.2.5 – Vytvoření sítě křižovatek). Druhá metoda slouží pro vytvoření jízdnic pruhů začínajících nebo končících v daném segmentu – `createLanes()`. Při samotném vytváření silnic a křižovatek na začátku druhého editačního kroku je tato metoda volána pro každý segment pouze jednou, tzn. silnice jsou vytvářeny vždy od jedné křižovatky ke druhé.

### 5.3.12 SuccPred

Tato třída je svojí jednoduchou strukturou velmi podobná třídě `NextPlace` (5.3.5). Jednotlivé instance slouží pro uložení předchůdce nebo následníka propojovacího místa. Jako členské proměnné třídy jsou použity identifikátor (`id`) následníka/předchůdce a hodnota pozice (`position`), na kterou je propojovací místo připojeno.

### 5.3.13 VehicleLane

Jedná se o třídu pro uložení jízdnicího pruhu v silniční síti. Jízdní pruh je určen svým identifikátorem (`id`).

Pro jízdní pruh je uchováváno několik seznamů křivek a bodů. Některé z nich slouží pouze k dočasnému uchování mezivýsledků výpočtu (jako např. `right` – seznam bodů pro hrubě předpočítaný pravý oddělovač jízdnicího pruhu). Další dva seznamy jsou určeny k uložení křivek náležících do levého (`leftDivider`) a pravého (`rightDivider`) oddělovače pruhu. Poslední seznam bodů (`points`) slouží k uložení bodů tvořících polygon daného pruhu. Ten je využíván při vykreslování barevné plochy, kterou je jízdní pruh zobrazen v mapě. Seznam `points` je vytvářen metodou `createLanes()`.

Ke každému pruhu patří proměnné pro uložení emitů a akceptorů. Jestliže jízdní pruh nepatří do silnice spojující dvě křižovatky, je na jeho volném konci umístěn generátor nebo terminátor, podle toho, zda se z volného konce do pruhu vjíždí, nebo se z něho vyjíždí. Odkaz na generátor/terminátor je uložen v proměnné `genTerm`. Pro budoucí umožnění výběru jednotlivých jízdnicíh pruhů a jejich barevné odlišení je pro každý pruh uložena barva, kterou má být jízdní pruh vykreslován (`color`). Parametr `full` určuje, zda se jedná o odbočovací pruh, který je kratší, tzn. nevede od křižovatky ke křižovatce (`false`), nebo nejedná (`true`).

### 5.3.14 Way

Jestliže je v křižovatce definována cesta pro průjezd vozidla křižovatkou, je vytvořena jako instance třídy `Way`. Ta v sobě obsahuje potřebnou strukturu: `emitter` (propojovací místo jízdnicího pruhu, ze kterého vozidla do křižovatky vjíždějí), `places` (seznam buněk, které tvoří samotnou cestu křižovatkou) a `acceptor` (propojovací místo pruhu, do kterého vozidla z křižovatky vjíždějí).

Parametr `ready` určuje, zda je již cesta zadána kompletně, tzn. včetně akceptoru na jejím konci. Tato informace je důležitá pro vykreslování (hotová cesta je vykreslována jinou barvou než cesta nedokončená). Poslední proměnnou je parametr `shown`, který

určuje, zda má být konkrétní cesta vykreslována v mapě nebo ne. To umožňuje skrývání cest vycházejících z jednotlivých emitů v křižovatce.

Jakmile je zakládána nová cesta (je vytvářena nová instance třídy `Way`), je už v jejím konstruktoru zadáván emit, ze kterého daná cesta vychází. Třída dále obsahuje metody pro přidání buňky a akceptoru do cesty. Přidáním akceptoru je cesta ukončena a prohlášena za hotovou. Poslední dvě metody, které považuji za podstatné, jsou metody `addWayToPlaces()` a `deleteWayFromPlaces()`. Ty slouží k uložení/smazání propojení obsažených v cestě do/ze všech buněk, které jsou součástí cesty. V obou případech se pracuje se seznamy navazujících míst (`NextPlace`) v jednotlivých buňkách (`Place`).

## 5.4 Pomocné třídy

Jedná se o třídy používané jako pomocné struktury, uchovávající data v potřebné formě. Protože jsou tyto třídy používány ve více třídách celého `juts` balíku, není možné je implementovat jako privátní. Stejně jako všechny třídy popisované v předchozích kapitolách jsou všechny členské proměnné nastavené jako `private`. Pro každou z nich tedy existují metody `get` a `set` pro možnost nastavení a získání hodnoty proměnné i mimo vlastní třídu.

### 5.4.1 Curve

Třída `Curve` reprezentuje křivku udávající například hranici křižovatky, nebo část oddělovače jízdního pruhu. Křivka je dána seznamem bodů (`points`), které udávají její průběh, a typem, který popisuje její vzhled – `type` (plná, přerušovaná, ...). Hodnoty pro proměnnou `typ` jsou určeny konstantami ve třídě `JUTS_constants` (5.2.4).

### 5.4.2 DoublePoint

Třída `DoublePoint` slouží jako pomocná struktura pro uložení bodu daného souřadnicemi `x` a `y`.

### 5.4.3 Line

Třída ukládá přímku danou dvěma body třídy `DoublePoint` a uchovává ji v podobě koeficientů pro analytické vyjádření přímky – `a`, `b`, `c`.

### 5.4.4 ObsluhaChyb

Tato třída zajišťuje obsluhu chybových hlášení při parserování vstupních a výstupních XML souborů. V případě, že je v souboru nalezena chyba, vypíše tuto informaci. Chyba je zároveň označena jako varování, chyba nebo fatální chyba, v závislosti na jejím rozsahu. Výpis chyby obsahuje řádek, sloupec a druh chyby, která se v parserovaném souboru vyskytla.

## 6 Potíže při řešení, možná zlepšení

Během vytváření grafického editoru jsem se postupně setkávala s potížemi a problémy, které bylo třeba vyřešit. Problémy se začaly objevovat již při seznamování se se systémem a zejména se vstupními daty, která nám byla poskytnuta.

Získání konečné podoby dat, která jsou v editoru používána a která jsou také popsána mezi formáty souborů (kapitola 2.2), předcházelo několik pracovních variant. Jako vstupní geografická data nám byl na úplném počátku spolupráce s úřady města poskytnutý vstupní soubor s ohraničujícími úseky silnic. Spolu se souborem jsme měli k dispozici informaci, že zadané body odpovídají počátečním a koncovým bodům úseků silnic. Po podrobnějším prozkoumání dat se ale ukázalo, že zadaná data odpovídají pouze ohraničujícím obdélníkům úseků. Bylo tedy nutné získat data, která by umožňovala určení koncových bodů segmentů, a tak i souřadnice případných křižovatek. Po další schůzce s odpovědnými pracovníky se nám podařilo získat soubor se souřadnicemi uzlových bodů. Následovaly krátké neshody ohledně vzájemné návaznosti obou souborů pomocí interně používaných identifikátorů. Ty se na poslední informativní schůzce vyřešily zjištěním, že žádná vzájemná souvislost mezi soubory neexistuje. V konečném řešení editoru se použitou aproximací vyřešil i problém nejednoznačnosti umístění a tvaru segmentu silnice v ohraničujícím obdélníku. Koncové body leží vždy na hranici dané právě ohraničujícím obdélníkem, celý segment je umístěn uvnitř obdélníka a může mít libovolný tvar.

V průběhu samotného návrhu a implementace se pak průběžně objevovaly drobné problémy a nepříjemnosti, které bylo nutné vyřešit. Za poměrně nepříjemnou považuji práci s reálnými souřadnicemi. Vzhledem k tomu, že se jedná o velká záporná čísla, ladění a hledání chyb vznikajících například během přepočtů, není příliš přehledné. Mezi ne zcela triviální záležitosti bych zařadila také problém generování křižovatek tak, aby nedocházelo k překrývání jízdních pruhů sousedních silnic a nekonvexitám v obrysu křižovatky, dále pak výpočet odbočovacích jízdních pruhů, popř. automatické generování buněk v křižovatkách. Jak je vidět v kapitole 4.2, všechny tyto problémy byly nakonec vyřešeny a následně naimplementovány. Přestože algoritmy nejsou nijak náročné, nebylo jejich vytvoření zcela přímočaré, vzhledem k tomu, že by výsledná mapa, vytvořená v grafickém editoru měla alespoň v základních rysech korespondovat s reálným vzhledem křižovatek a silnic a zároveň splňovat požadavky simulačního systému na parametry mapy.

Editor by bylo samozřejmě možné rozšířit a zdokonalit. V případě zájmu a potřeby by bylo možné editor uličních grafů zobecnit, a umožnit tak vytváření mapy zcela bez načtených vstupních dat – „od nuly“. Tímto způsobem by bylo možné vytvářet i mapy pro simulaci dosud neexistujících silnic. V diplomové práci nebyla tato funkce implementována vzhledem k tomu, že nebyla zadavatelem projektu požadována.

Funkčnost editoru by bylo zároveň možno rozšířit a doplnit o funkce pro úpravu samotných jízdních pruhů (nastavení odbočovacích pruhů a jejich délky, změnu počtu jízdních pruhů apod.) a křižovatek (např. posun hranice křižovatky). Další možnosti rozšíření se nabízejí v oblasti pohledových funkcí, uživatel zvyklý na práci v GIS by jistě ocenil funkci pro volbu oblasti zadáním středu a měřítko. Za praktické funkce bych rovněž pokládala např. funkci „krok zpět“, tzn. zrušení poslední provedené akce, možnost zobrazení mřížky pro přichytávání objektů nebo možnost vložení obrázku na pozadí uličního grafu. To by umožňovalo uživateli „obkreslení“ silniční sítě například z vloženého leteckého snímku. Rozšíření funkčnosti by se mohlo týkat nejen pohledové a mapové oblasti, ale také souborových funkcí. Jako další vhodné funkce by určitě našly své



uplatnění možnost tisku uličního grafu, práce s projektovými soubory popisujícími všechny soubory potřebné na vstupu pro vytváření nové mapy. S postupným rozšiřováním editoru v budoucnu by bylo praktické, aby editor umožňoval kromě vytváření nové mapy také úpravu již existující mapy po načtení grafické i simulační části XML. Vhodná by byla také možnost uložení nastavení uživatelského prostředí.

Většina z těchto funkcí byla součástí původních návrhů grafického editoru. Celá práce tak dosáhla poměrně velkých rozměrů. Je zde tedy prostor pro další rozšiřování a zdokonalování systému a možnosti jeho přizpůsobení požadavkům zadavatele.

## 7 Závěr

Diplomová práce popisuje ucelenou aplikaci první verze jednoúčelového grafického editoru pro přípravu simulačních map. Editor je organicky začleněn do skupiny podpurných prostředků simulačního systému JUTS. Během práce bylo nutné akceptovat zásady práce na kolektivním rozsáhlém projektu.

V současné době je editor plně funkční a splňuje všechny požadavky, které byly stanoveny na počátku práce. S jeho pomocí lze uživatelsky příjemným způsobem připravit simulační mapu pro pilotní projekt systému JUTS (mapu oblasti Lochotína) za necelých 20 minut. Lze očekávat, že zaškolená obsluha bude dosahovat ještě lepších časů, využije-li nabídky funkčních kláves a dalších běžně známých věcí pro usnadnění činností, které editor obsahuje.

Práce nebyla pouze rutinní činností, ale bylo nutné navrhnout a implementovat různé, pro tuto aplikaci specifické postupy, jejichž popis lze nalézt v předchozích kapitolách.

Během programování byla navržena mnohá vylepšení, která však z časových důvodů nebyla všechna realizována. Tato vylepšení plus další možnosti rozvoje editoru byly podrobně popsány v předchozí kapitole.

## Literatura

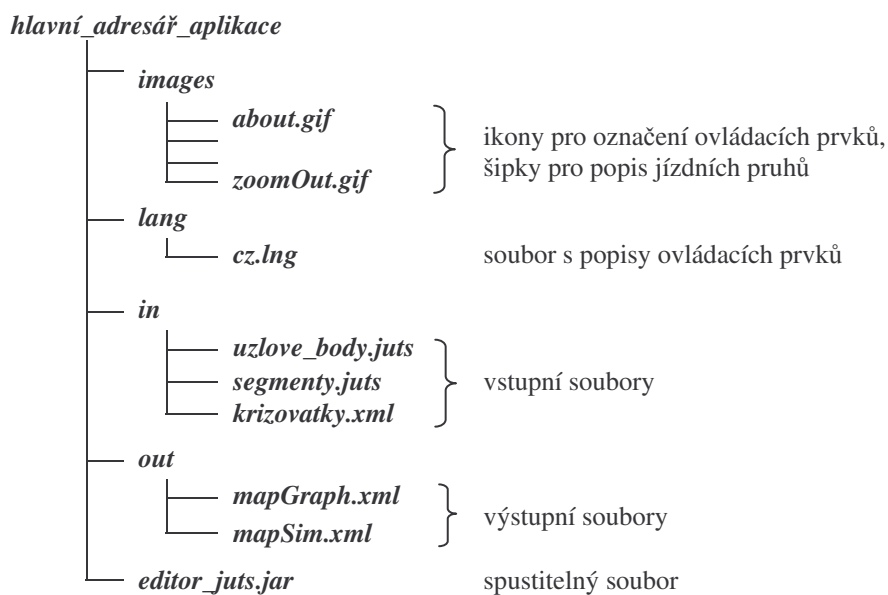
- [Cajt] Cajthaml, J.: *The present state of geographic information systems for towns and cities in Czech Republic*. Sborník GIS Ostrava 2005.
- [GepDoc] GEPRO s.r.o., *Dokumentace pro interakční grafický systém*. 2002
- [Hajk] Hájková, J.: *Graphical Support of the Traffic Simulation System*, Proceedings of the 9<sup>th</sup> Central European Seminar on Computer Graphics, Budmerice, pp. 51-57, 2005
- [Hart1] Hartman, D.: *Modelování dopravní situace pomocí systémů hromadné obsluhy*, Diplomová práce KIV ZČU, 2003
- [Hart2] Hartman, D.: *Head Leading Algorithm for Urban Traffic Model*. Proceeding of European Simulation Symposium. pp. 297-302, 2004
- [Hart3] Hartman, D.: *Implementation of Head Leading Algorithm in Simulation of Traffic in Pilsen*. Eighth United Kingdom Simulation Society Conference, Oxford, UK, 2005
- [McLau] McLaughlin, B.: *Java & XML*, O'Really, 2001
- [Rourke] O'Rourke, J.: *Computational Geometry in C*. Cambridge University Press, 2000
- [Zara] Žára, J., Beneš, B., Felkel, P.: *Moderní počítačová grafika*. Computer Press, 1998.

## Elektronické zdroje

- [Vever] Veverka, B.: *Krovak Projection*,  
[http://krovak.webpark.cz/e\\_version/krovak.pdf](http://krovak.webpark.cz/e_version/krovak.pdf)
- [URL1] Křovákovo zobrazení <http://krovak.webpark.cz/>
- [URL2] Úsek koncepce a dopravního inženýrství Správy veřejného statku města Plzně (SVSMP) <http://info.plzen-city.cz/svs/>
- [URL3] Správa informačních technologií města Plzně <http://info.plzen-city.cz/gis/>
- [URL4] Systém pro simulaci AIMSUN, grafický editor TEDI <http://www.aimsun.com/>

# 1 Obecné informace pro spuštění a používání Editoru map JUTS

Pro spuštění systému není nutná žádná instalace, jedinou podmínkou spustitelnosti SW je instalace JDK 1.5 a vyšší. Editor je možné spustit otevřením souboru `editor_juts.jar`, umístěném v základním adresáři. Zároveň je nutné, aby byla zachována potřebná adresářová struktura zahrnující interpretované třídy, soubor s popisy ovládacích prvků, adresář se všemi potřebnými ikonami apod. Adresářová struktura je zobrazena na obr. 1.1.



Obr. 1.1: Adresářová struktura potřebná pro správnou funkčnost editoru.

Pro práci v editoru je uživateli k dispozici několik ovládacích prvků, užitečných funkcí a možností, které je vhodné blíže popsat hned na začátku:

## Menu

Funkce jsou v menu rozděleny do pěti základních skupin – Soubor, Pohled, Funkce, Nástroje a Nápověda. Podrobný popis jednotlivých funkcí je uveden dále v kapitole 2. Menu je v hlavním okně programu zobrazeno vždy, kromě režimu Fullscreen (= celá obrazovka), během kterého jsou všechny ovládací prvky skryty.

## Nástrojové lišty (toolbar)

Slouží ke zrychlení ovládání editace mapy, protože umožňují uživateli rychlou a přehlednou volbu často používaných funkcí. Funkce jsou stejně jako v menu rozdělené do několika základních skupin. Pro každou skupinu existuje v editoru samostatná nástrojová lišta (toolbar). Popis jednotlivých funkcí je uveden v kapitole 2.

Jednotlivé nástrojové lišty je možné skrýt nebo opětovně zobrazit volbou v menu (*Nástroje* → *Toolbary* → ...). K dispozici je i volba pro skrytí/zobrazení všech toolbarů zároveň. Toolbary je zároveň možné kliknutím levého tlačítka myši uchopit a vytáhnout mimo prostor určený pro nástrojové lišty. Pro toolbar se pak vytvoří samostatný dialog,

kterým je možné libovolně pohybovat po pracovní ploše. Po zavření dialogu se toolbar vrátí zpět do oblasti pro nástrojové lišty.

### Popup menu

Jakmile je v editoru otevřená mapa určená k editaci, má uživatel k dispozici kontextové (popup) menu. To je možné otevřít kliknutím pravým tlačítkem do kreslicí plochy editoru (tzn. do oblasti mapy). Popup menu obsahuje základní funkce pro práci s jednotlivými prvky mapy. V závislosti na pozici, ze které bylo menu vyvoláno, jsou příslušné funkce aktivní nebo neaktivní. Aktivní jsou vždy takové funkce, které je možné použít k práci s objektem objekt, na který uživatel klikl. Podrobnější popis jednotlivých funkcí je uveden v kapitole 2.

Aktivací popup menu je zrušena funkce aktuálně zvolená z menu nebo nástrojové lišty, stejně tak jako je zrušený výběr regionu obdélníkem nebo posunutí mapy pomocí ručičky. Funkce obsažené v popup menu se liší v závislosti na editační fázi mapy.

### Spouštění funkcí

K pohodlné editaci mapy disponuje uživatel odpovídajícím množstvím potřebných funkcí. Většinu z nich je možné spouštět více způsoby, editor nabízí řadu klávesových zkratk. S výjimkou funkcí pro přímou manipulaci s jednotlivými prvky mapy platí pro ostatní funkce, že při jakémkoli spuštění má funkce vždy stejné vlastnosti a shodné použití. Jinak je tomu však u funkcí pro editaci mapy (přidávání prvků, posun, mazání apod.).

Jestliže je některá z těchto funkcí spuštěna z popup menu, lze ji (narozdíl od situace, kdy byla funkce spuštěna z nástrojové lišty nebo menu) použít pouze jedenkrát, a to na objekt, na kterém byla vyvolána. To neplatí pouze pro výjimky jako jsou například přidávání uzlového bodu nebo pozice v křížovatce a některé další (viz popis jednotlivých funkcí). V těchto dvou případech jsou funkce aktivní vždy, nezávisle na objektu, na kterém bylo menu vyvoláno, a vkládaný objekt je možné umístit na libovolnou pozici. I v tomto případě se však jedná o „funkci na jedno použití“. Pro vložení dalšího prvku je nutné volbu funkce opakovat.

Jestliže byl kliknutím označen některý z objektů, je barevným odlišením zvýrazněn. Kliknutím do oblasti křížovatky je označen příslušný uzlový bod, i všechny segmenty s ním sousedící.

Pokud je funkce aktivována v menu nebo nástrojové liště a pokud to základní princip funkce umožňuje, lze ji používat opakovaně, ale např. po smazání všech chybných segmentů nemá význam tuto akci provádět znovu. Po ukončení požadované činnosti je nutné funkci vypnout opětovným kliknutím nebo přepnout na jinou funkci. Funkce je zrušena také kliknutím pravým tlačítkem myši v kreslicí oblasti a vyvoláním popup menu.

### Souřadnice

Souřadný systém editoru odpovídá souřadnému systému vstupních dat – tzn. souřadnému systému Křovákova zobrazení. Osa  $x$  je orientována vodorovně s kladnými hodnotami rostoucími směrem doprava, osa  $y$  je orientována svisle s rostoucími hodnotami směrem nahoru. Všechny souřadnice ve vstupních datech jsou zadávány v záporných hodnotách s přesností na milimetry.

Při posunu uzlových bodů umožňuje editor kromě přibližného nastavení pozice pomocí myši také použít dialogové okno pro přesné nastavení souřadnic. Zároveň je možné nastavit zobrazování souřadnic kreslicí plochy nebo reálných geografických

souřadnic, na kterých se právě nachází kurzor myši, v pravém spodním rohu hlavního pracovního okna editoru. Zobrazování souřadnic je také možné zcela vypnout. Příslušnou volbu lze provést v popup menu, které lze zobrazit kliknutím pravým tlačítkem myši na pravý spodní roh okna, popř. volbou v menu (*Nástroje* → *Souřadnice* → ...).

### Posun mapy

Mapu je možné posouvat ve vertikálním i horizontálním směru hned několika způsoby. První možností je použití scrollbarů na pravé a na spodní straně mapy. K dispozici je též šipková růžice pro posun mapy ve vertikálním a horizontálním směru šipkami a pro posun mapy v libovolném směru ručičkou. Ve vertikálním směru je navíc možné použít kolečko myši ke scrollování mapy.

Při posouvání mapy pomocí šipkové růžice je nastavena kontrola hranic posuvu, tzn. není možný posuv mimo vlastní oblast uličního grafu (mapy). V případě, že je velikost mapy totožná s velikostí pracovní plochy, nemá použití posunovacích prvků žádný význam.

## 2 Uživatelské funkce

Funkce, které editor nabízí, jsou logicky rozděleny do několika skupin:

- Funkce pro práci se souborem
- Pohledové funkce
- Funkce pro práci s elementy mapy
- Nástroje
- Nápověda

Pokud je možné spuštění funkce použitím klávesové zkratky, je daná zkratka u funkce uvedena. Rozdělení funkcí odpovídá jejich logickému rozčlenění v menu a nástrojových lištách. Některé funkce při svém spuštění otevírají funkční dialogová okna, popsaná u jednotlivých funkcí.

Všechna okna mají však několik shodných prvků. Jedná se především o tlačítka *OK* a *Zavřít* – tlačítko *OK* potvrdí všechna nastavení a dialogové okno zavře, tlačítko *Zavřít* zavře dialog, aniž by byla provedena jakákoliv změna. Některá dialogová okna obsahují ještě tlačítko *Použít*. To provede v programu změny odpovídající přednastaveným parametrům, ale ponechá dialogové okno otevřené.

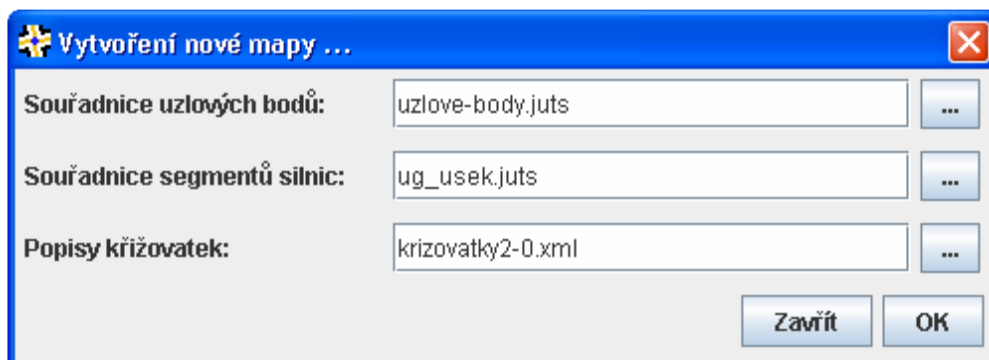
### 2.1 Funkce pro práci se souborem

Tyto funkce slouží pro souborovou práci s mapou – vytvoření nové mapy, uložení do výstupních XML souborů, uzavření celé mapy, popř. ukončení běhu celého programu. Menu *Soubor* nabízí jako jednu ze svých položek také seznam všech otevřených vstupních souborů. Uživatel tak může kdykoli zkontrolovat, jaké soubory pro vytváření mapy použil.

#### Nová mapa

**CTRL + N**

Slouží pro vytvoření mapy na základě vstupních souborů. Protože je nutné, aby všechny vstupní soubory byly načítány na začátku editace mapy, je pro vytvoření nové mapy použitý dialog, do kterého je nutné názvy všech tří vstupních souborů zadat. Dialogové okno pro vytvoření nové mapy je zobrazeno na obr. 2.1.



Obr. 2.1: Dialogové okno pro vytvoření nové mapy ze souborů.

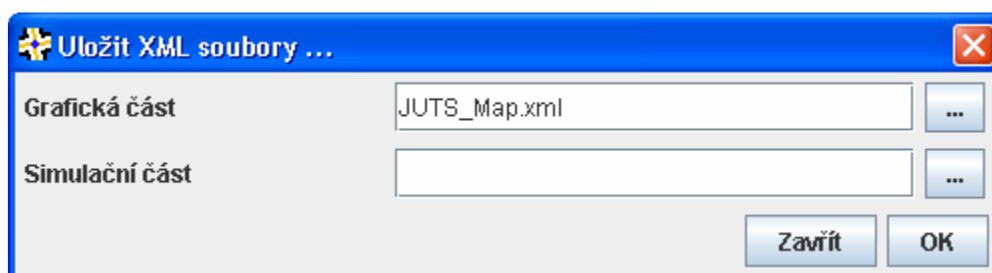
Jméno souboru je možné zapsat buď ručně do připravených textových polí, nebo soubor zvolit ve standardním dialogu pro otevření souborů. Souborový dialog je vyvolán stiskem tlačítka [...] u příslušné řádky. Dokud nejsou zadány názvy všech vstupních souborů, tlačítko *OK* je neaktivní.

### Uložit

**CTRL + S**

Funkce je určena k exportu vytvořené mapy do příslušných XML výstupních souborů. Funkce není aktivní, dokud není editovaná mapa v takové fázi, aby pro ni mohly být výstupní XML vůbec vytvořeny.

Pro ukládání mapy je použitý podobný dialog jako pro vytváření nové mapy, podobně fungují i jednotlivé ovládací prvky. V tomto případě však není nutné vždy zadávat oba výstupní soubory. Uživatel si může libovolně zvolit, zda mají být pro mapu generovány obě, nebo zda se má vytvořit pouze jedna část výstupního XML (v tomto případě jméno souboru pro druhou část mapy zůstává nevyplněno). Dialogové okno je k dispozici na obr. 2.2.



Obr. 2.2: Dialogové okno pro export mapy do XML.

### Zavřít mapu

**CTRL + W**

Tato funkce slouží pro ukončení editace právě otevřené mapy. Po zavření mapy se editor nachází ve stejném stavu jako před jejím otevřením.

### Konec

**ALT + F4**

Po volbě této funkce je ukončena činnost celého grafického editoru. Program se však pro jistotu ještě před samotným ukončením zeptá uživatele, zda si přeje činnost programu skutečně ukončit. V závislosti na volbě uživatele program je ukončen nebo není.

Jestliže uživatel ukončí činnost souboru použitím tlačítka křížku v pravém horním rohu hlavního okna programu, dotaz na potvrzení se neobjeví a program je ukončen okamžitě.

Toolbar *Soubor* obsahuje výběr souborových funkcí – vytvoření nové mapy, export mapy do XML, uzavření mapy.

## 2.2 Pohledové funkce

Funkce pro práci s pohledem jsou určeny k manipulaci s mapou během jejího vytváření. Umožňují posun mapy, přiblížení a oddálení, výběr oblasti pro zobrazení apod.

**Zvětšit**  +

Lupa slouží pro zvětšení mapy a přiblížení detailu. Volbou funkce se měřítko mapy upraví v závislosti na nastaveném kroku. (Implicitní hodnota je nastavena tak, že je měřítko mapy zmenšeno 1,5krát, tuto hodnotu lze změnit v nastavení uživatelského prostředí.) Při přibližování mapy zůstává zachován střed zobrazované oblasti.

**Zmenšit**  -

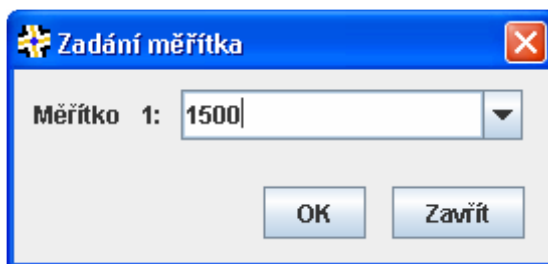
Tato funkce je určena k oddálení zobrazované oblasti. Stejně jako v případě zvětšení, dochází při této funkci ke změně měřítka mapy v závislosti na nastaveném kroku pro zoom. Při zmenšování mapy zůstává zachován střed zobrazované oblasti.

**Best fit**  \*

Best fit zajišťuje takové nastavení měřítka mapy, při kterém jsou všechny její části zobrazené na pracovní ploše.

**Nastavit měřítko** **1:x**

Slouží pro přesné nastavení konkrétní hodnoty měřítka zobrazení mapy. (Čím je nastavené číslo vyšší, tím je mapa zobrazena menší.) Dialogové okno je zobrazeno na obr. 2.3.



Obr. 2.3: Dialogové okno pro přesné nastavení měřítka mapy.

**Výběr oblasti** 

**Mezerník**

Výběr detailu obdélníkem umožňuje přesnější určení vybrané podoblasti mapy. Funkce umožňuje zadáním dvou bodů zvolit obdélníkovou oblast, která bude roztažena přes celou kreslicí plochu. Po zadání prvního bodu obdélníka stiskem levého tlačítka myši a jejím následným tažením je vykreslován obdélník zobrazující vybranou oblast. Po uvolnění



tlačítka je zadán druhý bod a současně vykreslen detail. Pro opakované použití výběru je nutná opakovaná volba funkce.

### **Celá obrazovka (Fullscreen)**

**CTRL + F**

Tato volba slouží pro přepnutí editoru do režimu Fullscreen. Režim slouží ke zvětšení pracovní plochy na maximum. Všechny ovládací prvky jsou skryty, aktivní zůstává pouze popup menu s funkcemi pro jednotlivé části mapy. Ukončení režimu FullScreen je možné použitím klávesy Esc.

### **Refresh pohledu**

**CTRL + R**

Funkce refresh slouží k překreslení kreslicí plochy v případě, že by došlo k chybnému vykreslení mapy na obrazovku.

V toolbaru *Pohled* jsou uživatelé k dispozici tytéž funkce jako v menu hlavního okna. V tomto případě jsou zde umístěny všechny, ne pouze výběr, jako tomu bylo u menu *Soubor*.

## **2.3 Funkce pro práci s elementy mapy**

Samotná tvorba mapy probíhá v několika krocích. Pro každý krok je typická určitá skupina funkcí. V následujícím popisu jsou funkce pro práci s mapou členěny podle fází při tvorbě mapy. Pro první editační fázi (úprava sítě segmentů silnic a uzlových bodů) jsou určeny funkce popsané v kapitole 2.4.1, pro druhou editační fázi (editace křižovatek) pak slouží funkce v kapitole 2.4.2.

### **2.4.1 Funkce pro úpravu silniční sítě uličního grafu**

Po načtení všech potřebných vstupních datových souborů je mapa zobrazena jako síť silnic v uličním grafu skládající se ze segmentů silnic a z uzlových bodů. Uzlové body jsou zobrazeny jako kruhové značky a označují místa, ve kterých se nachází buď křižovatka, nebo změna v průběhu silnice (např. zatáčka, výjezd apod.). Segmenty silnic (dále jen segmenty) jsou zobrazeny úsečkou a určují jednotlivé části silnic spojené uzlovými body.

V tomto okamžiku je možné provádět v zobrazené síti úpravy týkající se přidávání nebo ubírání uzlových bodů a segmentů, případně jejich posun, opravu nebo úplné odstranění chybných segmentů (tzn. segmentů, ke kterým nebyly ve vstupních souborech načtené potřebné uzlové body). Některé operace jsou prováděny jednokrokově, jiné probíhají ve více krocích. Vícekrokové operace je možné v jakémkoli kroku zrušit.

Je také nutné definovat, zda je uzlový bod křižovatkou nebo pouhým spojením dvou segmentů. Definice se provede přiřazením některé z křižovatek z načteného vstupního souboru s popisem křižovatek. Jestliže je uzlový bod označen jako spojení dvou segmentů, vytvoří segmenty, které na tento uzlový bod navazují, během přechodu do dalšího kroku tvorby mapy jednu silnici.

Všechny funkce je možné spouštět z menu, z nástrojové lišty funkcí nebo z popup menu, otevíraného kliknutím pravým tlačítkem kamkoli do oblasti mapy. Podrobnější popis používání funkcí je uveden v kapitole 1 v odstavci o použití funkcí. V případě, že jsou k úpravě uličního grafu používány funkce z nástrojové lišty, jejich použití se od funkcí spouštěných z popup menu nepatrně liší. Konkrétní odlišnosti jsou popsány u jednotlivých funkcí.

**Zrušit operaci** 

Ecs

Tato funkce slouží pro zrušení vícekrokové operace s některým z prvků mapy. Aktuálně zvolená funkce je nejenom vypnuta, ale zároveň je zrušena i právě probíhající operace a všechny objekty mapy jsou vráceny do původního stavu.

**Přidat uzlový bod** 

F1

Touto funkcí umožníme následným kliknutím levým tlačítkem myši přidání nového uzlového bodu kamkoli do mapy. Pro vložení bodu je nastaven kurzor nitkového kříže a současně s pohybem kurzoru je vykreslována prázdná kruhová značka odpovídající velikosti vytvářeného uzlového bodu. Po skončení vkládání je kurzor opět změněn na původní šipku. S nově vytvořeným bodem je možné okamžitě a bez dalších úprav provádět stejné operace jako s body načtenými z datového souboru (např. mazat ho, přidávat k němu segmenty nebo na něj mapovat křížovatku).

**Smazat uzlový bod** 

F2

Kliknutím pravým tlačítkem nad objektem uzlového bodu a volbou jeho smazání je daný uzlový bod vymazán z mapy i ze všech datových struktur. Při volbě funkce z menu nebo nástrojové lišty lze uzlové body mazat opakovaně.

Po stisku tlačítka pro mazání uzlových bodů v nástrojové liště nebo v menu je kurzor změněn na výběrovou ručičku. Poté je možné kliknutím myši mazat jednotlivé uzlové body. Dokud je funkce mazání zapnuta, je při pohybu myši po ploše uličního grafu označován vždy nejbližší uzlový bod, který bude v případě stisku levého tlačítka myši smazán. Funkci lze zrušit volbou jiné funkce nebo opětovným stiskem tlačítka. Poté se kurzor opět změní buď na standardní šipku, nebo na kurzor příslušný jiné zvolené funkci.

Jestliže je uzlový bod spojen s některým ze segmentů silnic, jsou během mazání bodu smazány i všechny s ním propojené segmenty.

**Posun uzlového bodu** 

F3

Po volbě funkce pro posun uzlového bodu z popup menu je možné pohybem myši plynule měnit pozici bodu, jehož volba byla provedena. Po stisku levého tlačítka myši je bod umístěn na nové souřadnice. Spolu s uzlovým bodem se překreslují i všechny segmenty, které s bodem sousedí. Po dobu posunu uzlového bodu je jako kurzor použita ručička. Po dokončení operace se kurzor opět automaticky přepne na standardní šipku.

Pokud chce uživatel posouvat více uzlových bodů, je vhodné použít tlačítko z nástrojové lišty nebo položku v menu. Po dané volbě lze barevně zvýrazněné uzlové body stiskem levého tlačítka myši uchopit, posunout a uvolněním tlačítka umístit na novou pozici. Funkci posouvání uzlových bodů lze vypnout opakovaným stiskem tlačítka nebo zrušit přepnutím na jinou funkci.

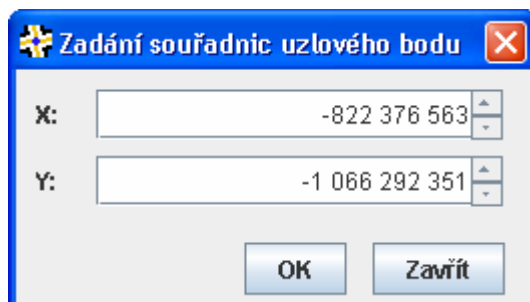
V průběhu operace je možné celou akci zrušit. Uzlový bod i všechny segmenty se pak automaticky vrátí na původní pozici. Není možné posouvat uzlové body mimo oblast mapy. Pokud si to situace žádá a je zapotřebí posunout bod za hranici mapy, je nutné použít funkci pro přesné nastavení souřadnic uzlového bodu.

**Přesné zadání souřadnic uzlového bodu** 

F4

Editor umožňuje přesné nastavení souřadnic jednotlivých uzlových bodů mapy. Po kliknutí pravým tlačítkem myši na objekt uzlového bodu a volbou položky *Upravit uzlový bod*

v popup menu je možné zadat uzlovému bodu přesné geodetické (reálné) souřadnice. Během úpravy souřadnic je zvolený bod zvýrazněn červenou barvou. Pro změnu souřadnic je použitý dialog zobrazený na obr. 2.7. Pokud je jako nová souřadnice zadána hodnota, která nespadá do rozsahu současné mapy, je hranice mapy posunuta.



Obr. 2.7: Dialogové okno pro přesné nastavení souřadnic uzlového bodu.

Pro změnu více bodů slouží tlačítko ve funkční nástrojové liště nebo v menu. Po jeho stisku lze kliknutím do mapy upravovat souřadnice nejbližšího uzlového bodu. Ten je barevně zvýrazněn při pohybu myši nad oblastí mapy. Pro výběr bodu je použitý kurzor ručičky.

Vzhledem k vzájemnému propojení celé silniční sítě a závislosti segmentů silnic na uzlových bodech dochází při posunu uzlového bodu také automaticky k úpravě s ním sousedících segmentů silnic.

### Namapovat existující křižovatku

**F5**

Po spuštění funkce pro zadaný bod je zobrazeno dialogové okno se seznamem všech křižovatek načtených z datového souboru (dialog viz obr. 2.8). Uživatel má možnost na daný bod namapovat příslušnou křižovatku uvedenou v seznamu. Jestliže je na uzlový bod namapována křižovatka, dojde ke grafickému zobrazení volby v mapě – na pozici uzlového bodu je zvolená křižovatka zhruba načrtnuta v závislosti na načtených datech.


V tuto chvíli je ještě možné upravit pozice jednotlivých uzlových bodů, zejména těch, které musely být po načtení vstupních dat opravovány a u kterých tak mohlo dojít k chybnému umístění. Dále lze přidat nové uzlové body a segmenty silnic (například v okrajových oblastech mapy, kde některé z existujících segmentů ve skutečnosti nebyly uvedeny ve vstupních datech). Není však nutné (a ani vhodné) přesně nastavovat směry všech segmentů v závislosti na zobrazeném náčrtku. Vzhledem k tomu, že orientace jednotlivých ramen křižovatek jsou ve vstupním souboru uvedeny s přesností 15°, nemusí se směr segmentu silnice a hrubého nákresu křižovatky zcela shodovat. K přesnému určení směru ramene křižovatky dojde automaticky v průběhu přechodu do následujícího kroku tvorby mapy. Pro systém jsou primární přesné souřadnice segmentu. Směr ramena křižovatky je jim proto během přechodu přizpůsoben.

Jakmile je křižovatka přiřazena k některému z uzlových bodů, je automaticky odstraněna ze seznamu. Jestliže se uživatel v průběhu přiřazování křižovatek k uzlovým bodům rozhodne již namapovaný bod smazat, zařadí se na něj původně namapovaná křižovatka na konec seznamu křižovatek načtených ze vstupního souboru. V případě, že se uživatel rozhodne k uzlovému bodu, na který již byla namapována křižovatka, přiřadit novou křižovatku načtenou z datového souboru, je původní křižovatka zobrazena

v seznamu jako první a je označena jako vybraná. Namapování je možné zcela odstranit použitím tlačítka *Žádná* v dialogu pro výběr křižovatky. Pokud uživatel zvolí jako novou volbu jinou křižovatku ze seznamu, původní křižovatka již zůstane na začátku seznamu.



Obr. 2.8: Dialog pro namapování křižovatky na uzlový bod.

Na uzlový bod lze mapovat pouze takové křižovatky, které jsou uvedeny ve vstupních datech. Pokud některá křižovatka chybí, je nutné ji do souboru s využitím jiného programu doplnit a do editoru opět načíst. K tomu slouží tlačítko Reload XML , umístěné v dialogu pro výběr křižovatek. Při novém načítání souboru jsou do seznamu přidány pouze křižovatky, které dosud nebyly do editoru načteny. Pro načtení chybějících křižovatek může být použit i úplně jiný XML soubor s odpovídající strukturou (tzn. data křižovatek mohou být rozdělena do více souborů). Křižovatka s daným identifikátorem může být do systému načtena pouze jedenkrát.

### Označit bod jako spojovací

F6

Pokud uzlový bod neoznačuje křižovatku a není ani jedním z okrajových bodů, slouží ke spojení dvou sousedních segmentů silnic. Pokud uživatel neoznačí takový bod jako křižovatku, je automaticky považován za spojnicu dvou segmentů. Pro zvýšení přehlednosti lze takový bod explicitně označit a umístit na něj zřetelnou značku. Tuto značku je možné umístit pouze na ty body, které sousedí se dvěma segmenty.

Jestliže uživatel v průběhu editace označí uzlový bod jako spojnicu segmentů a poté k uzlovému bodu přidá nebo od něj odebere segment, může dojít v závislosti na změně počtu navazujících segmentů ke zrušení původního nastavení. Např. uzlový bod je nastaven jako spojení dvou segmentů a uživatel se rozhodne jeden z těchto segmentů smazat; od této chvíle už bod nemůže sloužit jako spoj, protože nemá co spojovat.

**Smazat segment silnice ****F7**

Vymazat segment silnice z mapy je možné kliknutím pravým tlačítkem nad některým ze segmentů. Jestliže je volba segmentu dostatečně přesná, segment se z mapy vymaže. Pokud je kliknutím označeno více segmentů (bylo kliknuto přímo do oblasti uzlového bodu), jsou smazány všechny označené segmenty.

Při použití tlačítka z menu nebo funkční nástrojové lišty je opět zapnuta funkce pro opakované mazání segmentů silnic. Při pohybu kurzoru (výběrové ručičky) po mapě jsou zvýrazňovány segmenty, nad kterými je kurzor umístěn. Po stisku tlačítka je aktivní segment smazán z mapy i všech datových struktur. Po ukončení mazání je nutné opět funkci vypnout nebo přepnout.

**Přidat segment silnice ****F8**

Kliknutím pravým tlačítkem na některý z uzlových bodů je možné ke zvolenému uzlovému bodu přímo přidat segment silnice. Z tohoto bodu je veden nový segment silnice a kliknutím levým tlačítkem myši je nutné určit jeho druhý uzlový bod. Při pohybu kurzoru v mapě je automaticky zvýrazňován nejbližší uzlový bod, který by byl v případě stisku levého tlačítka myši zvolen jako koncový bod nově vkládaného segmentu.

Při stisku tlačítka v nástrojové liště funkcí je zapnuto přidávání segmentů. Přidávání je možné libovolně opakovat. Nový segment lze v tomto případě vložit dvojím kliknutím myši do oblasti mapy. Prvním kliknutím je zvolen počáteční bod, druhým koncový bod segmentu. Nejbližší uzlový bod je při pohybu myši nad mapou barevně zvýrazňován. Po určení počátečního bodu je současně s pohybem kurzoru vykreslována navazující čára určující vytvářený segment.

Segment silnice je možné vytvořit jen mezi dvěma již existujícími uzlovými body. V průběhu přidávání nového segmentu je možné operaci vytváření segmentu kdykoliv zrušit. Nový segment je do mapy vložen po zadání obou koncových bodů. Pro segment silnice není k dispozici funkce přesného zadání souřadnic. Vzhledem k tomu, že každý segment je závislý na dvou uzlových bodech, jehož souřadnice určují souřadnice koncových bodů segmentu, lze souřadnice segmentu měnit pouze posunem příslušného uzlového bodu.

**Opravit chybný segment ****F9**

Při načítání vstupních dat jsou uzlové body a segmenty silnic načítány ze dvou různých souborů. Z tohoto důvodu může dojít k určité nekonzistenci dat, tzn. některým segmentům chybí po načtení jeden nebo oba uzlové body a s některými nesousedí žádný segment. Takový segment je pak označený jako chybný, a pokud není uživatelem opraven, je při přechodu do další editační fáze z mapy odstraněn.

Opravu segmentu je možné provést kliknutím na segment, který si uživatel přeje opravit. Jako kurzor pro opakované opravy je použita výběrová ručička. Jestliže je zapnuta funkce pro opakovanou opravu, je nutné ji po skončení všech potřebných oprav opět vypnout opětovným stiskem tlačítka nebo přepnutím na jinou funkci.

Funkci je možné spouštět pouze v případě, že uliční graf obsahuje alespoň jeden chybný segment.

**Smazat chybné segmenty  F10**

Jestliže po načtení vstupních dat obsahuje uliční graf chybné segmenty silnic, je možné je výše popsanými funkcemi jednotlivě mazat nebo opravovat. Pokud ale uživatel nechce tyto segmenty do mapy vůbec zahrnovat, lze je všechny najednou použitím funkce pro smazání všech chybných segmentů z mapy odstranit. Jestliže mapa neobsahuje chybné segmenty, funkce je neaktivní.

**Přejít do následujícího editačního kroku  Enter**

Před samotným přechodem do následujícího editačního kroku je prováděno několik kontrol, zda je mapa v pořádku. Jedná se o kontrolu existence volných uzlových bodů, chybných sektorů, nedefinovaných spojení dvou sousedních úseků silnic. Pokud editor v těchto případech odhalí nějaké nedostatky, oznámí to uživateli a nabídne mu automatickou korekci. Pokud uživatel s opravou souhlasí, editor volné uzlové body a chybné segmenty smaže a dodefinuje spoje segmentů.

V mapě se však mohou objevit i závažnější nedostatky, jako například chybějící definice křižovatky, popřípadě neshodující se počet ramen namapované křižovatky a počet segmentů vycházejících z daného uzlového bodu. V takovémto případě je uživateli chyba oznámena a ten ji musí v mapě opravit.

Dokud nejsou z mapy odstraněny všechny nedostatky, není možné přechod do následujícího editačního kroku provést.

## 2.4.2 Funkce pro editaci křižovatek

**Zrušit operaci  Esc**

Tato funkce pracuje stejně jako v prvním editačním kroku.

**Generovat buňky v křižovatce  F1**

Tato funkce jednorázově vygeneruje maximální počet nepřekrývajících se buněk v křižovatce. V případě, že je funkce použita opakovaně nebo ve chvíli, kdy už jsou v křižovatce nějaké buňky vloženy, ponechá všechny již existující buňky a vloží další.

**Přidat buňku do křižovatky  F2**

Tato funkce slouží pro přidání nové buňky do oblasti křižovatky. Novou buňku lze přidat pouze do oblasti některé z křižovatek. Rozhodující je střed buňky, tzn. je možné, aby část kružnice, která buňku zobrazuje, přesahovala přes okraj křižovatky. Takové umístění však není vhodné (vozidla, která by projížděla cestou přes ni nadefinovanou, by v podstatě při průběhu simulace dopravy jako by vyjížděla z křižovatky). Stejně tak není vhodné, aby se buňky v křižovatce navzájem překrývaly (vozidla by se při průjezdu křižovatkou srážela).

Funkce přidání buňky do křižovatky funguje na stejném principu jako přidávání uzlových bodů v prvním editačním kroku.

**Smazat buňku z křižovatky  F3**

Jestliže uživatel potřebuje automaticky vygenerovanou nebo ručně přidanou buňku z křižovatky odstranit, má k dispozici funkci pro mazání. Smazána je vždy aktuálně vybraná zvýrazněná buňka (tzn. barevně odlišená), popř. ta buňka, na které bylo vyvoláno

popup menu. Buňku z křižovatky nelze smazat, pokud je přes ni definována cesta. Před smazáním buňky je nutné smazat i cestu.

#### **Posunout buňku v křižovatce**

F4

Buňku je možné posouvat v rámci dané křižovatky. Pokud se uživatel pokusí přesunout buňku mimo oblast křižovatky, zůstane buňka na okraji oblasti a dál ji posunout nelze.

Není vhodné, aby se jednotlivé buňky překrývaly nebo byly umístěny tak, že částečně přesahují okraj křižovatky. Zodpovědnost za správné umístění buněk v křižovatce je na uživateli, který buňky do křižovatky vkládá nebo upravuje jejich pozici. Buňky, která byly vygenerovány automaticky, se nepřekrývají a jsou celou svou plochou umístěny v křižovatce.

Posun buňky lze provádět stisknutím levého tlačítka myši (kružnice zobrazující dané buňku je uchopena), následným tažením myši (plynulý posun buňky v křižovatce) a konečným uvolněním tlačítka myši (vlození buňky na nové místo). Celou akci je možné zrušit volbou pro zavření operace. Při použití funkce z popup menu je vybraná buňka plynule posouvána v závislosti na pohybu kurzoru, kliknutím je určena její nová pozice.

#### **Vytvořit cestu pro průjezd křižovatkou**

F5

Aby bylo možné simulovat dopravu, je nutné nadefinovat směry, kterými se mohou vozidla pohybovat křižovatkami. K tomu slouží buňky rozmístěné v křižovatkách, emitory a akceptory.

Definici cesty je nutné zahájit v některém z emitorů. Po volbě začátku cesty (emitoru) je nutné pokračovat v určení buněk v křižovatce, které budou do nově vznikající cesty patřit. Buněk může být v cestě zadáno libovolné množství, vždy však musí být alespoň jedna. Jednotlivé buňky jsou zadávány kliknutím na některou z buněk nadefinovaných v křižovatce. Pro uzavření cesty musí uživatel zadat některý z akceptorů. Jeho volbou je cesta ukončena.

Při vytváření cesty jsou v jednotlivých fázích při pohybu myši zvýrazňovány ty elementy, které je možné do cesty přidat. Nejdříve je možné volit pouze z emitorů, jakmile je vytvořený začátek cesty, jsou zvýrazňovány buňky v křižovatce, a po zadání jednoho (povinného) místa může uživatel volit mezi buňkami a akceptory. Volbou akceptoru pak cestu ukončí.

#### **Smazat cesty z emitoru**

F6

Jestliže je v některém z emitorů definovaná cesta nebo několik cest, je možné všechny cesty vedoucí z daného emitoru jednorázově smazat. Mazat cesty lze i v případě, že jsou právě skryté. Značka skrytí je pak z emitoru odstraněna.

#### **Zobrazit / skrýt cesty z emitoru**

F7

Jestliže je v jedné křižovatce definováno větší množství různých cest, stává se celá křižovatka poměrně nepřehlednou. Z tohoto důvodu je možné některé z cest skrýt a celou křižovátku tak pro následnou editaci nebo kontrolu zpřehlednit. Skrytí cest je možné provést kliknutím na některý z emitorů křižovatky. Funkce je aktivní pouze pro ty emitory, které již nějakou cestu obsahují.

Po skrytí cest je příslušný emitor označen vykreslením tenkého kříže. Cesty je možné znovu zobrazit opětovným použitím stejné funkce.

## 2.4 Nástroje

Funkce ve skupině *Nástroje* jsou určeny k nejrůznějším uživatelským nastavením. Jejich úkolem je zpříjemnit a zjednodušit uživateli práci v průběhu editace uličního grafu.

### Zobrazit šipky

**CTRL + A**

Tento přepínač určuje zobrazení, popř. skrytí šipkové růžice v hlavním okně editoru. Je-li šipková růžice zobrazena, je umístěna v pravé části okna a skládá se z pěti tlačítek – čtyři pro pohyb ve svislém a vodorovném směru a prostředního tlačítka s funkcí ručičky.

Všechna tlačítka slouží pro posun mapy v rámci zobrazovací plochy. Krok šipkových tlačítek je možné určit v rámci nastavení uživatelského prostředí. Ručička umožňuje uchopení mapy (stisknutím levého tlačítka myši) a následný posun mapy po pracovní ploše. Ukončení posunu se provede opětovným uvolněním tlačítka myši. Stav, ve kterém se ručička aktuálně nachází, je indikován změnou kurzoru myši.

### Toolbary

Jedná se o podmenu nabízející skrytí nebo zobrazení jednotlivých nástrojových lišt. Jednou z možností je také zobrazení nebo skrytí všech nástrojových lišt najednou. Pokud je příslušná položka odpovídající určité nástrojové liště odškrtnuta, je daný toolbar viditelný, v opačném případě zůstává nástrojová lišta skrytá.

### Kurzor

Při aktivaci některé z funkcí je automaticky upraven vzhled kurzoru tak, aby umožňoval jednoduchou a názornou činnost. Pokud však automaticky nastavený kurzor uživateli nevyhovuje, může si jej libovolně nastavit. K dispozici jsou čtyři typy kurzorů:

	šipka	<b>1</b>
	nitkový kříž	<b>2</b>
	souřadnicový kříž	<b>3</b>
	ručička	<b>4</b>

### Souřadnice

Editor nabízí uživateli možnost zobrazovat v pravém dolním rohu hlavního okna souřadnice. Je možné volit dva způsoby výpisu – reálné geografické souřadnice, nebo souřadnice kreslicí plochy. Zobrazování souřadnic v hlavním okně může být také zcela vypnuté.

### Nastavení uživatelského rozhraní

**CTRL + U**

Pro uživatelské nastavení pracovního prostředí slouží dialogové okno. Konkrétní vzhled dialogu závisí na editační fázi mapy. Lze zde nastavit barvy pro zobrazení jednotlivých zobrazovaných objektů, jejich velikost, popřípadě další vlastnosti upravující vykreslování. Volitelná je také barva pozadí pracovního prostředí (pozadí mapy), volba kroků pro posun mapy v kreslicí oblasti a zoom. Samotné nastavení se provede použitím odpovídajícího

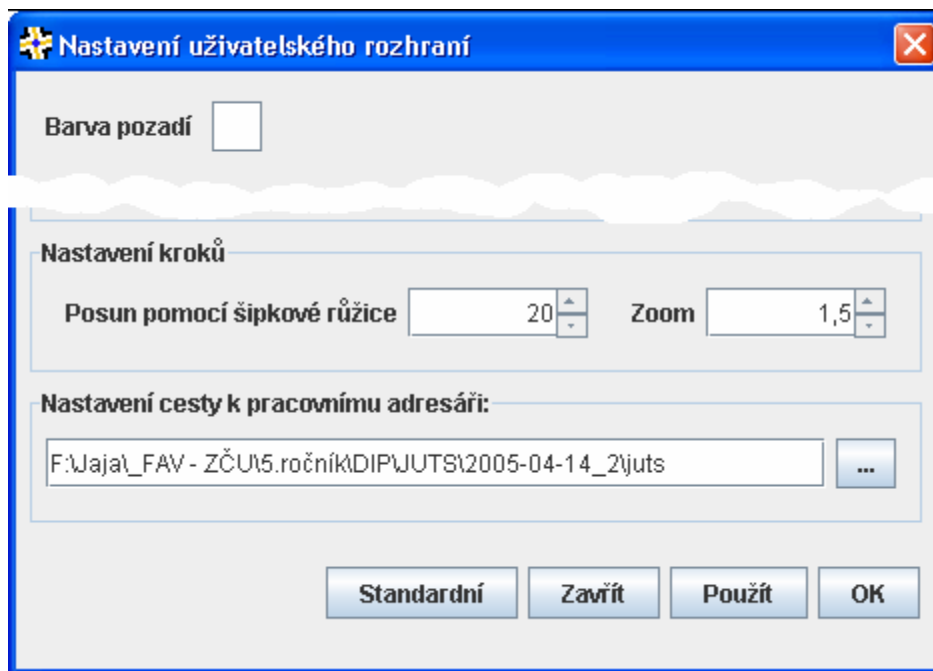


ovládacího prvku, nastavení barvy lze provést kliknutím na barevné políčko a následnou volbou v paletě barev.

Nastavení parametrů pro vykreslování mapy nemá žádný vliv na její výsledné uložení. Možnosti nastavení slouží pouze pro zvýšení komfortu při editaci mapy.

Lze také zpětně nastavit standardní hodnoty pro všechny nastavitelné položky uživatelského prostředí. Dané hodnoty (barva, krok apod.) jsou dále uváděny u jednotlivých konkrétních nastavení:

Nastavitelné položky shodné pro oba editační kroky (část dialogu viz obr. 2.4):



Obr. 2.4: Část dialogu pro nastavení uživatelského prostředí – položky shodné pro oba editační kroky.

Nastavovaná veličina	Popis	Standardní hodnota
Barva pozadí	Nastavení barvy pozadí editovaného uličního grafu	Bílá
Posun pomocí šipkové růžice	Krok pro posun mapy s využitím šipkové růžice. Hodnota udává počet pixelů při posunu mapy ve vertikálním nebo horizontálním směru.	20 bodů min = 1, max = 100
Zoom	Určuje rychlost přibližování a oddalování mapy při použití ±lupy. Čím je zadaná hodnota větší, tím větší je přiblížení nebo oddálení mapy při zoomování.	1,5 min = 0,1, max = 100
Cesta k pracovnímu adresáři	Určení adresáře, který je používán jako pracovní.	Adresář, ze kterého byl program spuštěn

Možnosti nastavení v prvním editačním kroku (úprava sítě segmentů silnic a uzlových bodů), obr. 2.5:

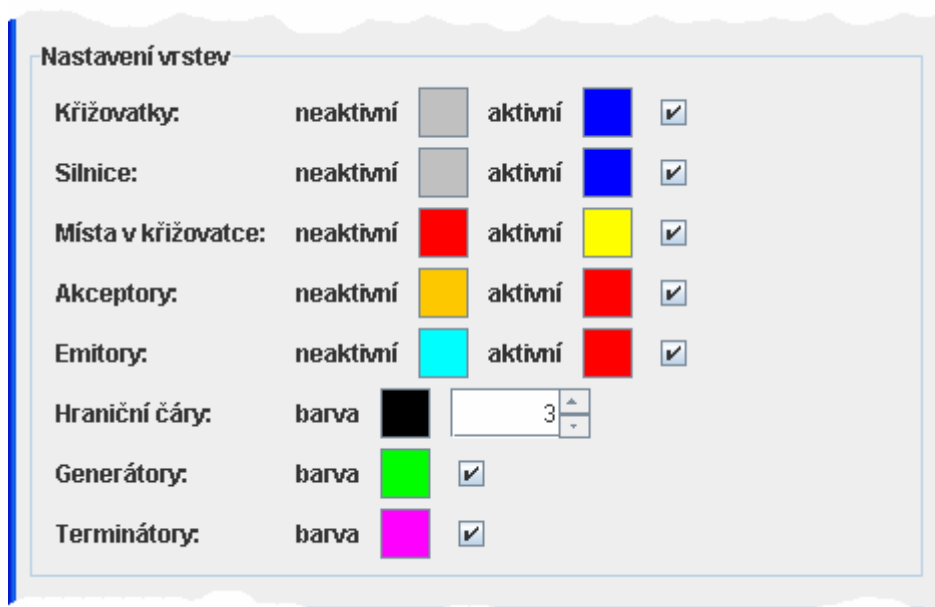


Obr. 2.5: Část dialogu pro nastavení uživatelského prostředí v prvním kroku editace mapy.

Nastavovaná veličina	Popis	Standardní hodnota
<b>Uzlové body</b>	Nastavení parametrů pro vykreslování uzlových bodů. Je možné nastavit několik vlastností:	
Barva aktivního bodu	Barva bodu, se kterým lze provádět danou operaci.	Červená
Barva neaktivního bodu	Barva bodu, se kterým není aktuálně prováděna žádná akce.	Modrá
Barva bodu s definovanou funkcí	Barva bodu, kterému je již definována funkce (spojnice dvou sousedních úseků, popř. je na něj namapována křižovatka).	Světle zelená
Velikost značky	Udává velikost kruhové značky pro zobrazení uzlového bodu. Hodnota je zadávána v pixelech. Velikost značky není závislá na měřítku zobrazení, tzn. že při přiblížení nebo oddálení mapy zůstává velikost vykreslovaného uzlového bodu stejná.	10 min = 1, max = 100
Zobrazení	Zaškrtnutím (eventuelně odškrtnutím) okénka lze zapnout nebo vypnout zobrazování všech uzlových bodů.	Zobrazit
<b>Nastavovaná veličina</b>	<b>Popis</b>	<b>Standardní hodnota</b>
<b>Segmenty silnic</b>	Umožňuje několik nastavení pro zobrazování segmentů silnic:	
Barva aktivního segmentu	Barva segmentu, se kterým je možné provádět zvolenou operaci.	Růžová
Barva neaktivního segmentu	Barva segmentu, se kterým není aktuálně prováděna žádná operace.	Černá
Barva chybného segmentu	Nastavení barvy segmentu, který byl načten jako chybný.	Červená
Tloušťka čáry	Určuje tloušťku čáry, kterou jsou segmenty silnic vykreslovány. Tato hodnota je udávána v pixelech a není závislá na velikosti měřítko.	3 min = 0, max = 100
Styl vykreslování čáry	Pro zobrazení stylu jsou k dispozici dvě varianty – plná a přerušovaná čára.	Plná
Zobrazení	Zaškrtnutím (eventuelně odškrtnutím) čtvercového okénka lze zapnout nebo vypnout zobrazování všech segmentů silnic.	Zobrazit

Možnosti nastavení v druhém editačním kroku (editace křižovatek), obr. 2.6:

Nastavovaná veličina	Popis																								
<b>Barva pro aktivní a neaktivní část mapy</b>	Nastavení barev je totožné pro několik prvků mapy. Vždy je možné nastavit barvu pro aktivní i neaktivní variantu. U některých prvků lze nastavit pouze barvu jako takovou. Jedná se o ty části mapy, které není možné uživatelsky upravovat. Pro jednotlivé elementy lze zároveň určit, zda mají nebo nemají být zobrazovány.																								
	<table border="1"> <thead> <tr> <th>Prvek mapy</th> <th>Aktivní barva</th> <th>Neaktivní barva</th> </tr> </thead> <tbody> <tr> <td>Křižovatka</td> <td>Modrá</td> <td>Světle šedá</td> </tr> <tr> <td>Silnice</td> <td>Modrá</td> <td>Světle šedá</td> </tr> <tr> <td>Buňka v křižovatce</td> <td>Žlutá</td> <td>Červená</td> </tr> <tr> <td>Akceptor</td> <td>Červená</td> <td>Oranžová</td> </tr> <tr> <td>Emitor</td> <td>Červená</td> <td>Tyrkysová</td> </tr> <tr> <td>Generátor</td> <td>-</td> <td>Světle zelená</td> </tr> <tr> <td>Terminátor</td> <td>-</td> <td>Purpurová</td> </tr> </tbody> </table>	Prvek mapy	Aktivní barva	Neaktivní barva	Křižovatka	Modrá	Světle šedá	Silnice	Modrá	Světle šedá	Buňka v křižovatce	Žlutá	Červená	Akceptor	Červená	Oranžová	Emitor	Červená	Tyrkysová	Generátor	-	Světle zelená	Terminátor	-	Purpurová
Prvek mapy	Aktivní barva	Neaktivní barva																							
Křižovatka	Modrá	Světle šedá																							
Silnice	Modrá	Světle šedá																							
Buňka v křižovatce	Žlutá	Červená																							
Akceptor	Červená	Oranžová																							
Emitor	Červená	Tyrkysová																							
Generátor	-	Světle zelená																							
Terminátor	-	Purpurová																							
Nastavovaná veličina	Popis																								
<b>Čára pro vykreslení hranic</b>	Hranice křižovatek a jízdních pruhů jsou vykreslovány v závislosti na jejich vlastnostech plnou nebo přerušovanou čarou. Význam vykreslení odpovídá pravidlům silničního provozu – plnou čáru nelze přejíždět, přerušovanou ano. Pro všechny tyto hraniční linie je možné nastavit barvu vykreslení a tloušťku vykreslované čáry.																								
Nastavovaná veličina	Popis	Standardní hodnota																							
Tloušťka čáry	Nastavení tloušťky čáry pro hraniční linie. (Jestliže se měřítko mapy dostane pod 1:500, jsou všechny čáry vykreslovány tloušťkou 1, při hodnotě 0 se čára nezobrazuje.)	3 min = 0, max = 100																							
Barva čáry	Barva vykreslovaných hraničních čar.	Černá																							



Obr. 2.6: Dialog pro nastavení uživatelského prostředí ve druhém kroku editace mapy.

V toolbaru *Nástroje* může uživatel využívat tlačítka pro změnu jednotlivých kurzorů, zobrazení/skrytí šipkové růžice a otevření dialogu pro nastavení uživatelského prostředí.

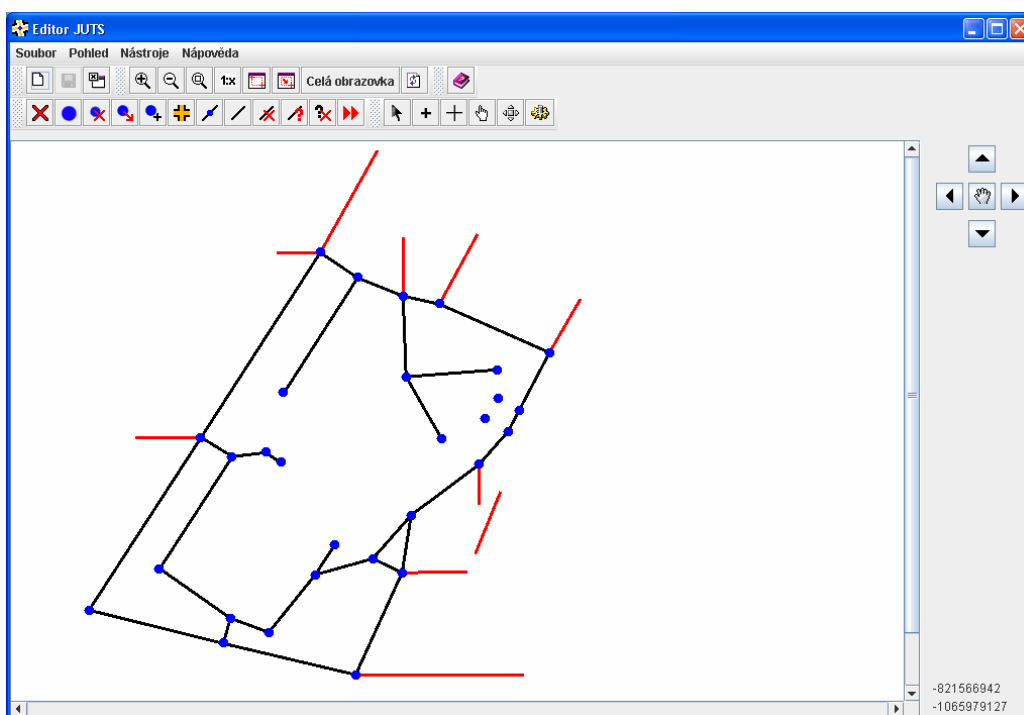
## **2.5 Nápověda**

Menu pro nápovědu obsahuje pouze položku pro zobrazení informací o programu (autora, verzi, apod.). Toolbar *Nápověda* není k dispozici.

### 3 Příklad typického postupu při editaci mapy (vytvoření mapy „od A do Z“)

Celý proces vytváření a editace mapy probíhá v několika krocích. V této kapitole bych proto ráda uvedla možný způsob využití editoru pro vytvoření mapy ve formátu vhodném pro načtení do simulačního systému JUTS. Zároveň s popisem základních kroků zde budou uvedeny obrázky pro názornější ukázky typických úkonů při práci v editoru.

Uživatel po spuštění systému načte všechny tři potřebné vstupní soubory – textové soubory s uzlovými body a segmenty silnic i XML soubor, který obsahuje popis křižovatek v dané oblasti. Editor automaticky vytvoří síť segmentů a uzlových bodů a vykreslí ji (viz obr 3.1). Uživatel může začít se sítí pracovat a upravovat ji, jak potřebuje.



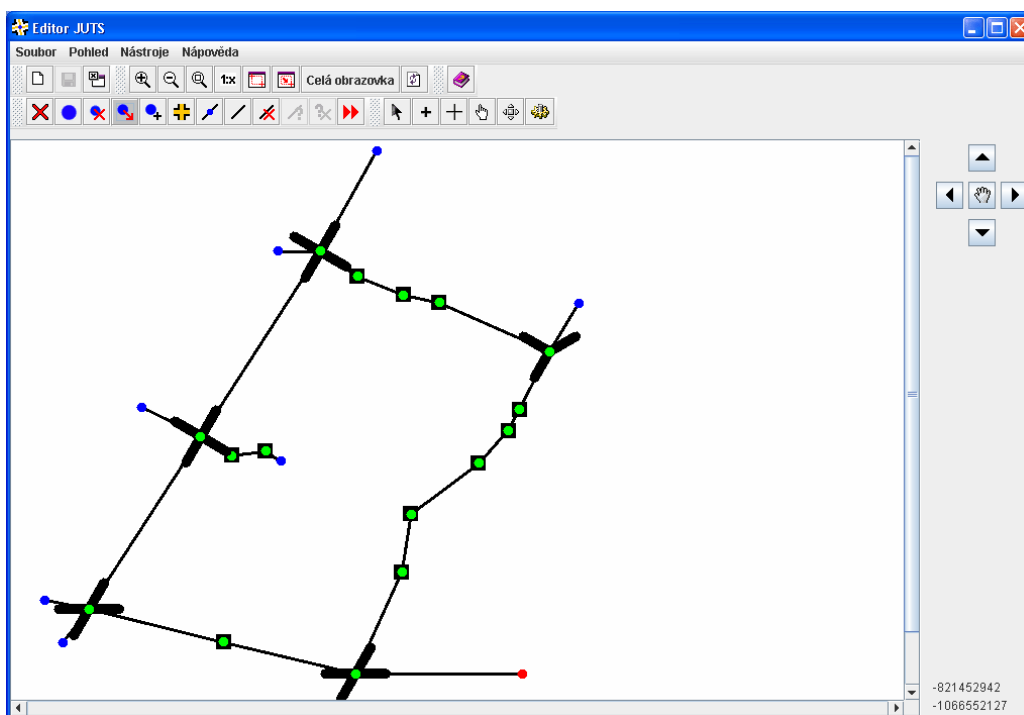
Obr. 3.1: Stav editoru po načtení vstupních dat.

Protože soubory se segmenty a s uzlovými body nebyly pravděpodobně pořizovány současně, objevilo se ve vykreslené mapě několik chybných segmentů. Jestliže chce uživatel některé z nich vložit do mapy, rozhodne se je upravit, poté ostatní smaže. V mapované oblasti několik segmentů a uzlových bodů chybí, proto je nutné je ručně doplnit. Řekněme, že uživatel má k dispozici mapu se všemi potřebnými souřadnicemi úseků, které v mapě chybí. Nic mu tedy nebrání vložit do mapy několik nových uzlových bodů a poté přesně nastavit jejich souřadnice. K novým uzlům je nutné pochopitelně vložit také příslušné segmenty. Některé segmenty a uzlové body jsou v mapě naopak navíc – např. uživatel nebude simulovat dopravu v ulicích procházejících sídlištích. Tyto segmenty a uzlové body tedy smaže. Protože pozice všech uzlových bodů již odpovídají skutečnosti, není zapotřebí používat funkci pro jejich posun.

Teď je tedy silniční síť připravena, a uživatel může začít na jednotlivé uzlové body mapovat křižovatky. Postupně mapuje odpovídající křižovatky ze seznamu na uzlové body až do chvíle, kdy potřebnou křižovatku nenajde. Tato křižovatka chyběla ve vstupních datech, je tedy nutné ji do vstupního XML přidat. Pro tento případ je k dispozici speciální program (úprava vstupních souborů není součástí editoru). Uživatel tedy soubor s popisem křižovatek upraví a stiskem tlačítka *reload* v dialogu pro mapování křižovatek se nově přidané křižovatky do seznamu doplní. Poté může uživatel pokračovat v jejich mapování na uzlové body. Pokud některé z opravovaných segmentů neodpovídají směru načrtnuté křižovatky (což by se vzhledem k tomu, že při přípravě samotné sítě pečlivě překontroloval souřadnice opravovaných segmentů, stát nemělo), uživatel poopraví jejich pozici. Náčrtky křižovatek jsou jen hrubé, v dalším kroku se orientace jednotlivých ramen křižovatky přizpůsobí přesným souřadnicím uzlových bodů, není tedy zapotřebí podle načrtnuté křižovatky přesně upravovat jednotlivé segmenty.

Všechny ostatní zbývající uzlové body, které spojují pouze dva segmenty, se teď označí jako jejich spojení. Okrajové body není nutné nijak označovat, systém je v následujícím kroku rozpozná automaticky.

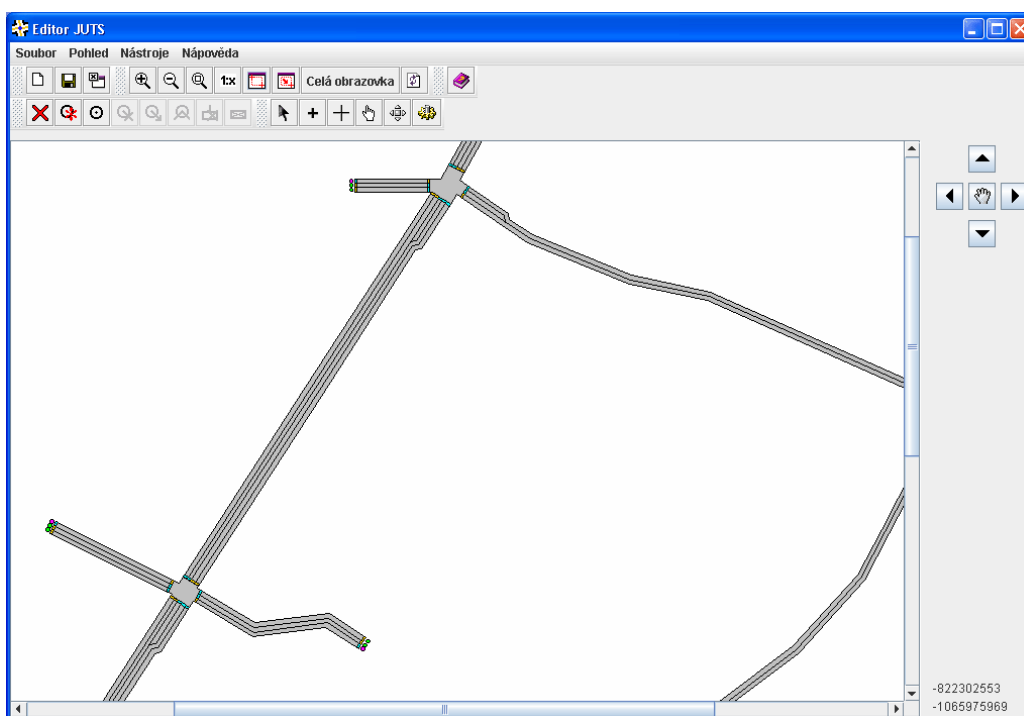
Před přechodem do další fáze úpravy mapy překontroluje uživatel zběžným pohledem celou mapu a zjistí, že oproti namapované křižovatce chybí v silničním grafu jeden segment, který buď nebyl uveden ve vstupních datech, nebo ho uživatel při úpravě sítě omylem smazal. Není však žádný problém jej do mapy doplnit vytvořením segmentu mezi daným uzlovým bodem a jeho o kousek vzdálenějším sousedem. Ten už byl však označený jako spojení dvou segmentů, které na něj navazují. Protože po přidání segmentu k tomuto bodu už na bod nenavazují jen dva, ale už tři segmenty, je jeho označení jako spojení zrušeno, a uživatel na něj musí namapovat některou z křižovatek. V této chvíli tedy mapa vypadá tak, jak je ukázáno na obrázku 3.2.



Obr. 3.2: Mapa připravená pro přechod do druhé editační fáze.

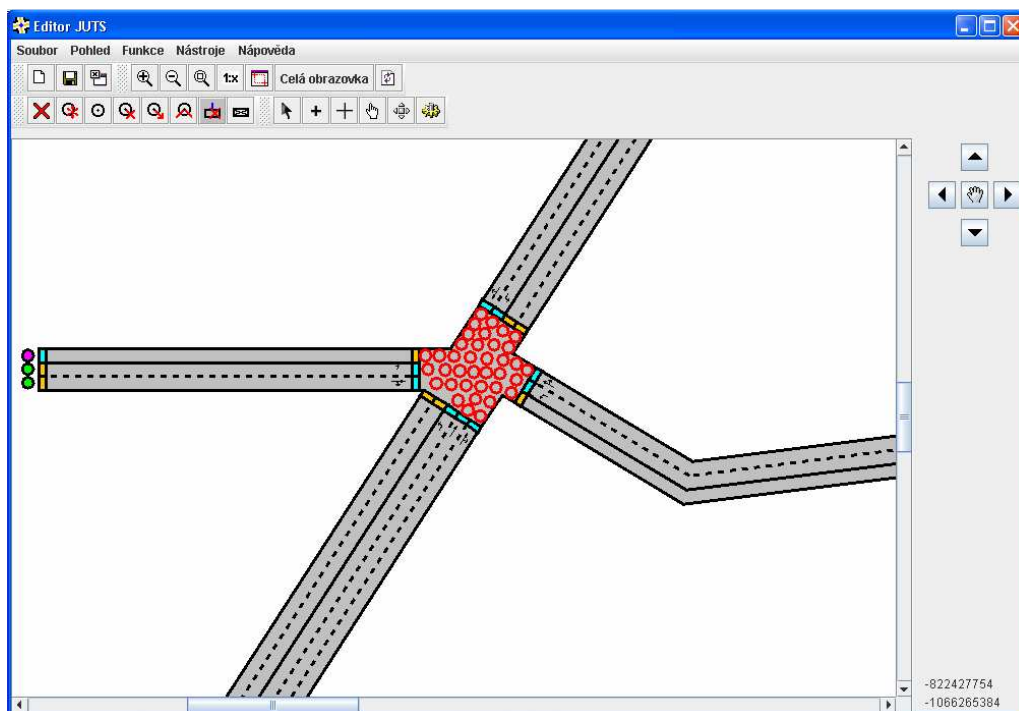
Mapa je nyní připravena, je tedy možné stisknout tlačítko pro přechod do dalšího editačního kroku. Pokud by se přece jen v této chvíli vyskytovaly v mapě nějaké chyby (volné uzlové body, chybné segmenty, některým uzlovým bodům s více segmenty by nebyla nadefinována funkčnost nebo počet segmentů navazujících na uzlový bod by nesouhlasil s počtem ramen na něj namapované křižovatky), uživatel by byl upozorněn. V případě volných bodů, chybných segmentů nebo nedefinovaných spojovacích bodů by byla korekce provedena automaticky, pokud by ji uživatel potvrdil v zobrazeném dialogovém okně. Pokud by se jednalo o závažnější chyby (nedefinovaná křižovatka, chybný počet ramen), musel by uživatel tyto nedostatky opravit. Pokud je však vše v pořádku, je možno pokročit do následujícího kroku editace mapy.

Systém nyní automaticky vykreslí jednotlivé křižovatky a všechny potřebné jízdní pruhy, vytvoří jízdní pruhy pro odbočování, vygeneruje generátory a terminátory, které v silniční síti zajišťují dodávání a odebírání vozidel. Zároveň je i pro každé spojení jízdního pruhu s křižovatkou vytvořený emitor, popř. akceptor. Mapa tedy momentálně vypadá tak, jak je ukázáno na obrázku 3.3.



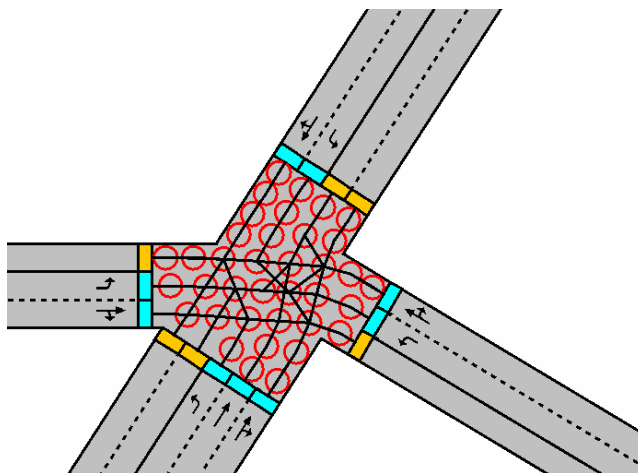
Obr. 3.3: Výřez z mapy po přechodu do druhého editačního kroku – křižovatky a jízdní pruhy jsou automaticky vygenerovány.

V tomto editačním kroku má uživatel k dispozici funkce odlišné od kroku předchozího. Aby mohla být celá mapa vyexportována do výstupních XML souborů, je ještě nutné definovat cesty pro průjezd křižovatkou. Pro běžné použití stačí zvolit automatické vygenerování buněk v křižovatce. Při vyšších nárocích na definování přesnosti průjezdu vozidel křižovatkou, lze jednotlivé buňky do křižovatky naklikat postupně, popř. automaticky vytvořené buňky upravit, posunout, smazat. Ukázka automaticky vygenerovaných buněk pro jednu zvolenou křižovatkou je k dispozici na obrázku 3.4.



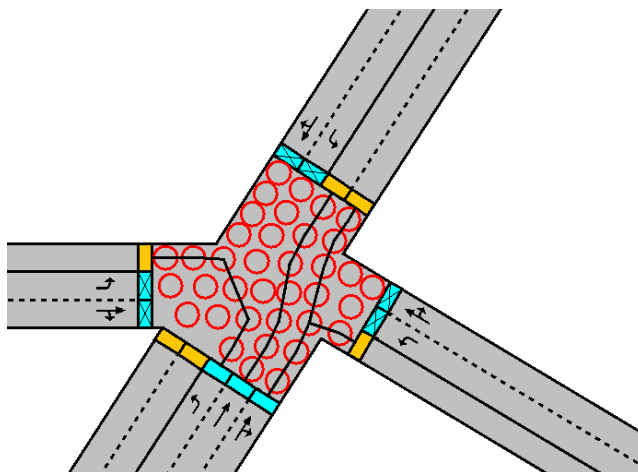
Obr. 3.4: Buňky vygenerované automaticky pro konkrétní křižovatku.

Definování cest v křižovatkách už nyní nic nebrání. Uživatel tedy v závislosti na směrech v dané křižovatce zadá do všech křižovatek možné cesty, kterými budou moci při provádění samotné simulace vozidla projíždět. Možné směry jízdy z jednotlivých jízdních pruhů jsou dány vykreslenými směrovými šipkami. K dispozici má zároveň funkce pro smazání již vytvořených cest z některého z emitorů nebo možnost existující cesty z konkrétního emitoru skrýt a později znovu zobrazit. Tato funkce by měla sloužit ke zvýšení přehlednosti při zadávání všech možných cest pro průjezd křižovatkou. Na obrázcích 3.5 a 3.6 je k dispozici ukázka všech zadaných cest v křižovatce s použitím a bez použití skrývání cest z jednotlivých emitorů.



Obr. 3.5: Všechny cesty definované v křižovatce bez skrývání.





Obr. 3.6: Všechny cesty definované v křižovatce s využitím skrývání cest z jednotlivých emitů.

Pokud je uživatel se svým dílem spokojený, tj. nadefinoval všechny potřebné cesty, eventuálně opravil pozice buněk v křižovatkách tak, aby měly cesty odpovídající tvar, může vytvořit výstupní XML soubory.