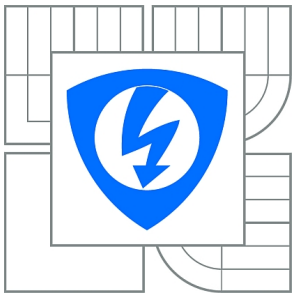
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

VÝVOJ DOPRAVNÍHO SIMULÁTORU - OBJEKT KŘIŽOVATKA

TRAFFIC SIMULATOR

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

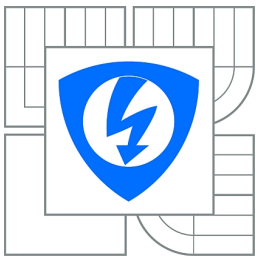
AUTOR PRÁCE — LUKÁŠ KUČERA

AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR HONZÍK, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Bakalářská práce

bakalářský studijní obor
Automatizační a měřicí technika

Student:	Lukáš Kučera	***ID:***	106581
Ročník:	3	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Vývoj dopravního simulátoru - objekt křižovatka

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je seznámit se s dopravním simulátorem TRASI vyvinutým na UAMT a seznámit se s algoritmem použitým pro logiku křižovatky. Dalším úkolem je najít chybu v tomto algoritmu a navrhnout rozšíření, které chybu odstraní. Posledním úkolem je implementace nové metody do TRASI a její ověření na několika jednoduchých dopravních uzlech.

DOPORUČENÁ LITERATURA:

Nagel Ch. - kol.: C# 2008. Praha, Computer Press, 2008.

Termín zadání:	8.2.2010	***Termín odevzdání:***	31.5.2010

Vedoucí práce: Ing. Petr Honzík, Ph.D.

prof. Ing. Pavel Jura, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem bakalářské práce bylo prostudovat kód dopravního simulátoru TRASI, zjistit příčiny nefunkčnosti udílení předností v křižovatce a tyto příčiny opravit. Bylo navrženo několik způsobů řešení, z nichž jedno bylo implementováno do simulátoru (systém s maskou předností). Funkčnost řešení byla ověřena na třech jednoduchých modelových situacích. Na křižovatce s hlavní silnicí, na křižovatce s předností zprava a na jednoduché světelné křižovatce. Jejich návrh je také součástí bakalářské práce. Dále je v práci uveden stručný návod na simulátor jak pro uživatele, tak pro tvůrce dalších programových rozšíření.

Klíčová slova

Dopravní simulátor, TRASI, křižovatka, maska předností.

Abstract

The object of the bachelor's thesis was to study a code of a traffic simulator TRASI, to find causes of a malfunction in giving way at the crossing and to fix these causes. Several methods of solution were suggested and one was selected and implemented into the simulator (the priority mask system). The solution's functionality has been tested on three simple model situations. On a crossing with a main road, on a crossing with a right road priority and on a crossing with traffic lights. The design of these situations is as well the part of the bachelor's thesis. Then there is a brief manual for the simulator, both for users and the following programmers, in the thesis.

Key words

Traffic simulator, TRASI, crossing, priority mask.

Bibliografická citace

KUČERA, L. *Vývoj dopravního simulátoru - objekt křižovatka*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 46 s. Vedoucí bakalářské práce Ing. Petr Honzík, Ph.D.

Prohlášení

„Prohlašuji, že svou bakalářskou práci na téma ‚Vývoj dopravního simulátoru - objekt křižovatka' jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb."

V Brně dne: **31. května 2010** ……
podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Honzíkovi, Ph.D. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé bakalářské práce.

V Brně dne: **31. května 2010** ……
podpis autora

OBSAH

1. ÚVOD

Cílem bakalářské práce je navázat na již rozpracovaný simulátor TRASI (TRAffic SImulator), který modeluje síť silnic a křižovatek, na nichž se pohybují automobily. Ty, každý sám za sebe, rozhodují své akce v závislosti na okolním dění. Chování automobilů v předchozí verzi simulátoru nebylo reálné a vznikaly patové (všechna auta stojí) nebo kolizní (auta se „projedou“) situace. Cílem práce je seznámit se s kódem simulátoru, pochopit ho a zjistit příčiny problémů. Následně je třeba navrhnout způsoby, jakými vyřešit reálné chování automobilů při příjezdu ke křižovatce a při jejím průjezdu. Dalším úkolem je vybrat vhodnou metodu a tuto implementovat do simulátoru. Pro ukázku funkčnosti je třeba zprovoznit několik jednoduchých modelových situací. Navržené řešení musí umožňovat použití v libovolné obecné situaci. Při vývoji nového systému pro řešení předností a hlídání přítomných automobilů je třeba přihlížet i k budoucím možnostem úprav, kupříkladu rozšíření na více pruhů.

2. REŠERŠE

V následující kapitole se nachází přehled dopravních pravidel, funkcí převzatého simulátoru a fyzikální rozbor pohybu vozidel.

2.1 STRUČNÝ PŘEHLED DOPRAVNÍCH PRAVIDEL

V dopravním simulátoru je simulován provoz dopravních prostředků tak, aby se co nejvíce podobal skutečnému provozu. Ten se řídí mimo jiné níže uvedenými dopravními pravidly.

2.1.1 Křižovatka a její varianty

Za křižovatku lze považovat jakékoliv křížení nebo spojení silnic. Pokud není jinak určeno, přednosti projíždějících automobilů se řídí předností zprava. To znamená, že každý automobil musí dát přednost těm, které se nacházejí napravo od jeho trasy a trasu mu kříží. Tento systém je však nevhodný pro velký provoz, protože by mohla nastat situace, kdy přijedou na křižovatku ze všech silnic auta a navzájem si budou dokola dávat přednost.

Proto je na většině křižovatek určena hlavní silnice. Potom pro vedlejší silnice navzájem platí předchozí pravidlo, ale s tím rozdílem, že je třeba dávat přednost všem automobilům, které jedou po silnici hlavní nebo z ní odbočují a je jim křížena cesta. Automobily na hlavní silnici jedoucí po ní nebo z ní odbočující doprava nikomu přednost dávat nemusí. Pokud odbočují doleva, musí dát přednost předchozím, pokud jim kříží cestu.

Křižovatka může mít proměnné udílení předností, a to ať už řízené světelně, či za pomoci policisty. V tomto případě je přímo určeno, kdo smí jet a kdo ne. Pokud není přímo určeno, že smějí jet automobily odbočující doleva (na světelné křižovatce naznačeno svítící šipkou), tyto musí dát přednost všem, kterým kříží cestu a smějí jet. U světelné křižovatky svítící zelené světlo určuje, že smí automobily z příslušné silnice jet, oranžové světlo svítí, když mají projet ty, které by již nestihly zastavit a červené světlo svítí, když auta nesmí do křižovatky vjet. Před rozsvícením zeleného světla svítí v České republice červená společně s oranžovou.

Kruhový objezd je speciální typ křižovatky, poslední dobou čím dál častěji používaný. Automobily v něm mají ze všech silnic stejnou přednost, proto je vhodný pro silnice se stejným provozem, ale na rozdíl od křižovatky bez určených předností zde nenastanou patové situace, kdy si auta navzájem dávají přednost. Jak je patrno z názvu, kruhový objezd je kus silnice ve tvaru kruhu, do nějž jsou zavedeny ostatní silnice. Při příjezdu automobil najede do kruhu v kladném směru otáčení (proti směru hodinových ručiček), pokud ovšem již v kruhu nejede automobil jiný. Ten by měl před nově přijetým přednost. Neplatí tu tedy přednost zprava, ale mají tu přednost již přítomné automobily [2].

2.1.2 Ostatní dopravní situace

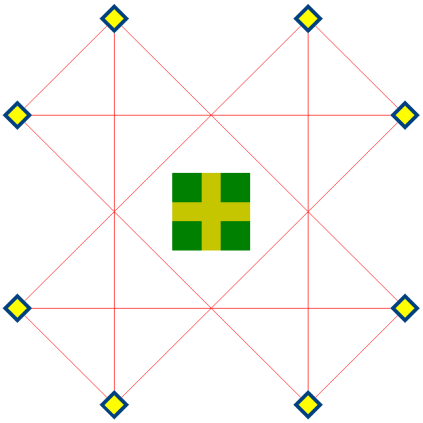
Křižovatky jsou propojeny silnicemi, ve kterých je povolena určitá maximální rychlost, kterou by automobily neměly překročit. U křižovatek i kdekoliv jinde se na silnicích může vyskytnout přechod pro chodce. V České republice mají chodci přednost před vozidly vyjma tramvají. Proto pokud se blíží řidič k přechodu, musí zpozornět a pokud je na přechodu chodec, nebo se někdo zjevně chystá přechod využít, musí mu přejití řidič umožnit zpomalením až zastavením.

Při nehodě nebo práci na silnici může nastat situace, kdy musí být jeden pruh silnice odstaven. Pokud je provoz sveden do protisměru, řídí se oblast buď signalizací pověřených osob, nebo signalizací světelnou. Jestliže není provoz řízen, mají přednost automobily jedoucí ve svém pruhu, není-li určeno jinak dopravní značkou [2].

2.2 DOPRAVNÍ SIMULÁTOR TRASI

V převzatém simulátoru bylo již naprogramováno grafické a uživatelské prostředí a provizorní umělá inteligence. Veškeré vyhodnocování probíhá ve funkcích třídy `BasicCarAI`, která má přiřazený jeden model auta (třída `BasicCarModel`), jehož problémy řeší. Model auta má uložené souřadnice své polohy a pomocí vyhodnocovací jednotky porovnává své vzdálenosti od ostatních automobilů na jejich společné komunikaci a buď brzdí, nebo zrychluje. Dále se umělá inteligence stará o maximální povolenou rychlost a křižovatky.

Silnice jsou řešeny pomocí třídy `Edge`. V ní jsou uloženy body, kde se mění směr silnice (zatáčky) a počáteční a koncový bod silnice. Tyto dva body jsou objekty třídy `Node`. Ta má uloženo, které silnice směřují do ní a které z ní. Třída `Crossing` realizující křižovatku obsahuje seznam uzlů (`Node`) a jejich spojnic (`Edge`). Viz obrázek 1. Žlutomodré čtverce jsou grafickým vyobrazením uzlů. Křižovatku tvaru kříže tvoří 4 vstupy a 4 výstupy. Z každého vstupu vedou červené čáry, reprezentující hrany (`Edge`), do výstupů všech zbývajících silnic. Čtvercový obrázek uprostřed křižovatky pouze symbolizuje, že se zde nachází objekt křižovatka (`Crossing`).



Obrázek 1 - Křižovatka (`Crossing`) tvořená hranami (`Edge`) a uzly (`Node`)

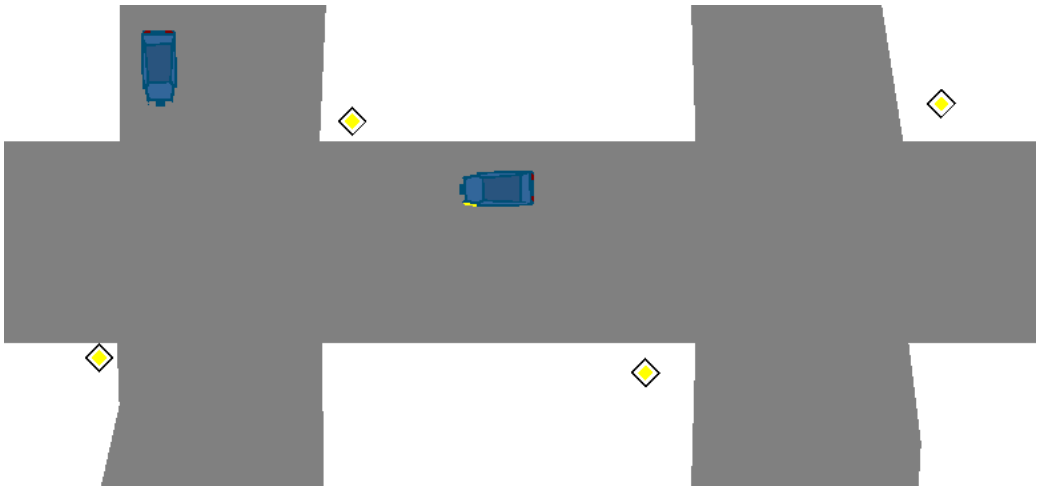
Když automobil přijede na křižovatku, směr, kterým se bude ubírat, se určí náhodně, s ohledem na nastavené přitažlivosti jednotlivých silnic (parametr `Attractivity` třídy `Edge`). Posunutí a natočení auta se určuje podle jeho rychlosti a natočení volantu. Natočení volantu se počítá podle toho, kde se nachází následující bod komunikace (`Node`, nebo zlomový bod v `Edge`). Auta potom i přes ostré hrany projíždějí s plynulým zatáčením.

Speciálním typem `Node` je třída typu `CarGen`, která generuje automobily. Činí tak v předem zadaném intervalu (parametr `Interval`) s náhodným rozptylem, jehož velikost lze také nastavit (parametr `Diffusion`).

Veškerá grafika programu je vektorová. Načítá se ze souborů vytvořených ve speciálním editoru PolyEdit, jenž je plně funkční. Každý objekt (obrázek) vytvořený

v editoru je složen z dílčích polygonů (n-úhelníků), kterým lze měnit barvu obrysu a barvu výplně.

V samotném simulátoru SimCity lze přidávat nové objekty typu `Crossing`, `Node`, `Edge` a `CarGen`. Těmto lze upravovat určité parametry dle požadavků. Vozidla i grafické znázornění celé silniční sítě je vytvořeno v PolyEditu. Po spuštění simulace se vozidla pohybují po vozovce podle trajektorií vytvořených v SimCity. Celkové propojení grafiky a trajektorií je uloženo jako workspace. V původním programu bylo pro načítání již uloženého workspace třeba listovat složkami, a tak byla do programu přidělána položka Recent, která načte poslední otevřený workspace. Grafické zobrazení hran a uzlů lze vypnout. Ukázku z původní simulace lze vidět na obrázku 2 [1].



Obrázek 2 - Grafické zobrazení silnic a aut v SimCity

2.3 MODELOVÁNÍ SILNIČNÍHO PROVOZU

K modelování silničního provozu lze volit dva způsoby. Automobily jako jednotlivce (mikroskopický model) nebo dopravní tok (makroskopický model). Dopravní tok si lze pro lepší pochopení představit jako tok krve tepnou [4]. Narozdíl od krve je ale každý z automobilů aktivní částicí, protože mají různé technické parametry a každý řidič se chová jinak. Může být zkušený i nezkušený, bojácný i agresivní. Při modelování toku lze tyto rozdílnosti simulovat pomocí náhodných proměnných [5]. Nejběžnějším je však využití průměrných hodnot. Jedná se

konkrétně o počet automobilů, které projedou určitým bodem za jednotku času (tzv. průtoková rychlost), o vzdálenost, kterou ujedou za jednotku času (tzv. rychlost dopravního toku) a počet automobilů, které může daný úsek silnice pojmout, vzhledem k jeho délce a počtu pruhů (tzv. hustota dopravy). Mezi těmito parametry jsou definovány matematické vztahy [4]. Jimi se však tato práce nezabývá.

V simulátoru byl již zaveden pouze primitivní fyzikální model pohybu automobilu jako jednotlivce. To znamená bez jakéhokoliv tření či smyků. Jedná se zde pouze o zrychlený pohyb s diskrétně proměnným zrychlením. Do jeho průběhu nebylo nijak zasahováno, ačkoliv zde neseděly jednotky, tudíž není simulace ve správném měřítku. To může být buď mikroskopické (je zde vidět každý model), nebo makroskopické (jednotlivé modely nelze vidět, jsou zobrazovány pouze masy modelů) [5]. Pro programové rozšíření určování předností je třeba vypočítat čas, kdy automobil přijede ke křižovatce. To už se řídí korektními fyzikálními pravidly.

Pro výpočet času (t), za který bude auto křižovatkou projíždět, je třeba znát současnou rychlost automobilu (v_0), maximální povolenou rychlost na silnici (v_{max}), zrychlení (a) a vzdálenost od uzlu křižovatky (s). Pro odvození výpočtu použijeme pomocnou proměnnou t_{max}, která určuje čas, za který dosáhne automobil při současném zrychlení maximální povolené rychlosti. Od tohoto okamžiku totiž pojede bez dalšího zrychlování.

Abychom získali dráhu a rychlost rovnoměrného zrychlení, vyjdeme z konstantního zrychlení a. Platí, že

$$v = \int a\,dt = at + v_0\,, \tag{1}$$

$$s = \int v\,dt = \int (at + v_0)\,dt = \frac{1}{2}at^2 + v_0 t + s_0\,. \tag{2}$$

Z rovnice 1 plyne, že pro čas dosažení rychlosti v_{max} platí:

$$t_{max} = \frac{v_{max} - v_0}{a}\,. \tag{3}$$

Uvažujeme, že $s_0 = 0$, čili že doposud nebyla ujeta žádná dráha. Z rovnice 2 lze dosazením $a = 0$ získat rovnici pro pohyb s konstantní rychlostí:

$$s = vt\,. \tag{4}$$

Před dosažením maximální rychlosti se pohyb řídí rovnicí 2 a po dosažení rovnicí 4, přičemž za v se dosadí v_{max}. Po sečtení dvou drah získáváme výslednou:

$$s = \frac{1}{2} a t_{max}^{\ 2} + v_0 t_{max} + v_{max}\left(t - t_{max}\right) + s_0 . \qquad (5)$$

Z rovnice 5 lze při uvažování $s_0 = 0$ vyjádřit požadovaný čas t:

$$t = \frac{s - \frac{1}{2} a t_{max}^{\ 2} - v_0 t_{max}}{v_{max}} + t_{max} . \qquad (6)$$

Pokud však auto přijede do cíle dřív, než zrychlí na maximum, vyjde celkový čas t menší, než t_{max}. Pro tento případ je třeba v programu ošetřit, aby se čas počítal dle vzorce odvozeného z rovnice 2 ($s_0 = 0$):

$$t = \frac{-v_0 + \sqrt{v_0^{\ 2} + 2as}}{a} . \qquad (7)$$

Pokud by měla vyjít část pod odmocninou záporná, znamená to, že zrychlení je záporné, tudíž auto brzdí natolik, že nedojede na konec dráhy, takže se čas nastaví na maximální hodnotu.

Při příjezdu automobilu ke křižovatce se počítá čas příjezdu na první uzel křižovatky a čas, za který ujede vzdálenost od vjezdu do křižovatky k výjezdu z ní. K tomuto času je dobré přičíst ještě jednu sekundu jako časovou rezervu pro skutečné vyjetí z křižovatky.

3. REALIZOVANÁ ROZŠÍŘENÍ PROGRAMU

Tato kapitola pojednává o způsobech, kterými lze řešit funkčnost křižovatky a které byly pro řešení vybrány. Dále jsou zde popsána alternativní využití křižovatky, princip generování masky, která je využívána pro udílení předností v křižovatce, vyřešení patové situace a návrh dalších vylepšení simulátoru.

3.1 ZPŮSOBY ŘEŠENÍ PŘEDNOSTÍ V KŘIŽOVATCE

Úkolem je zařídit, aby se dal v libovolné křižovatce určit automobil, který pojede dřív a který musí čekat. V nadcházejících podkapitolách je rozbor různých řešení tohoto problému.

3.1.1 Původní realizace udílení předností

V původním programu byla křižovatka realizována pouze na zkoušku a provizorně. Určování předností měla na starosti funkce `CanIPassTrough()`. V ní byl určen pouze jeden směr, ze kterého do křižovatky v konkrétní čas směla vjíždět auta. Směr se určoval podle předepsaných předností. Ty byly natvrdo implementovány v programu typem podmínky, po jejímž splnění se směr, odkud mají automobily přednost, uložil do proměnné určující, ze které silnice smí auta právě jezdit.

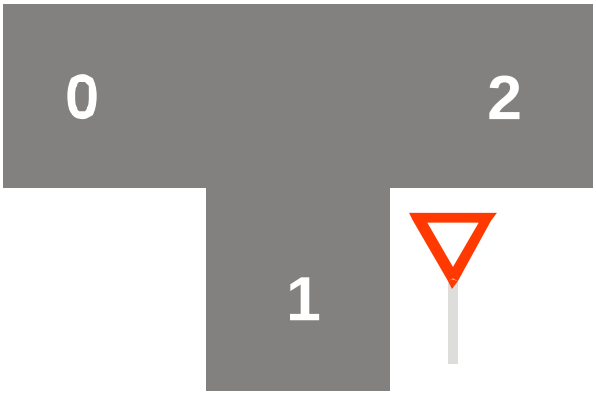
Auto, které přijelo na křižovatku, se zastavilo těsně na jejím vnitřku, a tím se dostalo na začátek spojnice dvou silnic. Podmínka určující přednost potom vyhodnocovala, zda smí jet dál, nebo musí zastavit a dát přednost autu jinému. Případ, kdy již auto projíždí křižovatkou, byl řešen pomocí časové prodlevy mezi změnou povoleného směru. Tím bylo dosaženo, že auta nejezdila přes sebe.

Tento systém se ale zdaleka nepodobá realitě. Původní verze programu navíc neměla podmínky pro přednost správně nastavené, a tak je bylo třeba přeprogramovat. Po úpravě již nedocházelo k patovým situacím, kdy si všechna auta dávala přednost. Zůstával však bohužel fakt, že se stávalo, že automobily z vedlejší silnice jely dřív než vozy z hlavní silnice, protože zrovna na vedlejší silnici ukazovala proměnná určující směr právě nastavený na projíždění.

3.1.2 Realizace pomocí tabulky priorit

Hlavní myšlenkou tabulky priorit je, že křižovatkou smí projet automobil, který má vyšší nebo stejnou prioritu v porovnání s ostatními automobily. Čili pokud má přednost před všemi vozidly, má prioritu nejvyšší. Pokud by naopak měl křižovatkou podle silničních předpisů projet jako poslední, má prioritu nejnižší.

Tento systém však má to úskalí, že jej nelze realizovat jednoduchou tabulkou, jejíž číslo řádku by bylo číslem příjezdu do křižovatky a číslo sloupce by bylo číslem označujícím příslušný výjezd z křižovatky. V této tabulce by potom byly uloženy priority z určitého směru do směru druhého. Jak je však ukázáno na obrázku 3, tak ačkoliv je to myšlenka hezká, nelze udělat tabulku priorit ani pro obyčejnou, nejjednodušší křižovatku tvaru T s jednou silnicí vedlejší.



	0	1	2
0	X	0	0
1		X	?
2	0	1	X

Obrázek 3 - Křižovatka tvaru T s příslušnou prioritní tabulkou

Když bychom se pokusili napsat tabulku priorit, narazíme na problém hned při řešení vedlejší silnice. Na hlavní diagonále není důležité, jaké číslo bude, protože automobily nemohou vyjet stejnou silnicí, kterou přijely. Pojedeme-li rovně po hlavní silnici, nemusíme dávat přednost nikomu, tudíž priority [0,2] a [2,0] nastavíme na nulu (nejvyšší). Zároveň pokud odbočujeme z hlavní silnice doprava, přednost nikomu nedáváme, proto je priorita [0,1] rovna nule. Pokud odbočujeme z hlavní silnice doleva, musíme dát přednost všem předchozím případům. Proto je priorita [2,1] nastavena na jedničku. Doposud se metoda zdá spolehlivá, zbývají již jenom dvě políčka v tabulce, ale právě u jednoho z nich narazíme na nepřekonatelný problém. Totiž když budeme řešit průjezd z 1 do 2. Zatáčíme doprava z vedlejší na hlavní silnici. Automobilům, které jedou z 2 do 0, nebo které jedou z 0 do 1, cestu

nekřižujeme, ale automobilům z 0 do 2 cestu křižujeme. Všechny tyto směry ale mají stejnou prioritu. Pokud by se nastavila hodnota [1,2] na jedničku, nesmyslně by byla dávána přednost automobilům, kterým by správně dávána být neměla. Změnou priority směru [0,2] bychom zase dosáhli toho, že při tomto přímém průjezdu křižovatkou bychom dávali přednost protijedoucím vozidlům, která nám v jízdě nijak nebrání.

Systém priorit tedy není možné realizovat jednoduchou tabulkou, a proto byl navržen jiný způsob určování, které auto projet smí a které má čekat.

3.1.3 Realizace pomocí masky předností

Maska předností je tabulka, která určuje, kterému autu má tázající se automobil dát přednost a pokud se v křižovatce tento automobil nenachází, může tázající se křižovatkou projet. Označí-li se počet vjezdů do křižovatky x a počet výjezdů z křižovatky y, bude počet prvků v prioritní tabulce $x \cdot y$. U masky předností však počet prvků vzroste na $x^2 \cdot y^2$. Na rozdíl od prioritní tabulky, kde je každým prvkem číslo, jsou v masce předností pouze prvky typu `bool`.

Způsob práce s maskou předností je opět ukázán na křižovatce tvaru T s jednou vedlejší silnicí, která je na obrázku 3. Jsou v ní tři vjezdy a tři výjezdy, čili velikost masky bude $3^2 \cdot 3^2 = 81$. Odečtou se prvky diagonál, které nejsou pro tento případ křižovatky podstatné ze stejného důvodu, jako tomu bylo u prioritní tabulky. Pro tento případ jich je 45. Zbylých 36 prvků je i tak oproti předešlým 6 prvkům prioritní tabulky hodně. Ošetření priorit s maskou je však univerzální a lze jeho návrh plně automatizovat.

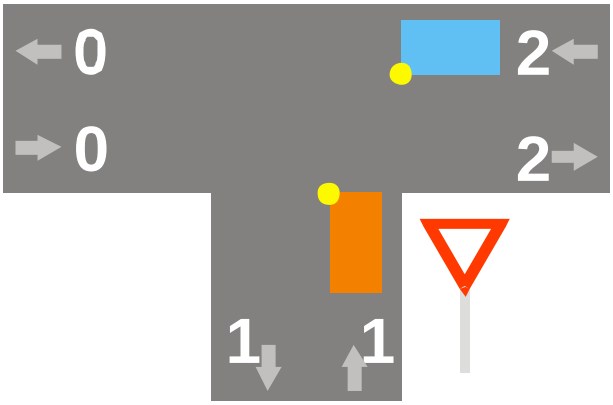
Na obrázku 4 je vidět, že maska předností je vlastně čtyřrozměrné pole. První dvě souřadnice určují trasu, kudy jede automobil, který si žádá informace, zda smí projet. Čili první souřadnice je, odkud přijíždí, a druhá, kam míří. Když je známo, který automobil si informaci žádá, přistoupí se k podtabulce, jež určuje, kterým automobilům musí dávat přednost. Její souřadnice opět určují, odkud (první) kam (druhá) jedou automobily, kterým je nutné dát přednost. Tabulka na obrázku 4 je navíc rozšířena o popisky vnitřních souřadnic z důvodu lepší orientace. Tato čísla ve výsledné masce nebudou.

Z \ DO	0				1				2			
0		**0**	**1**	**2**		**0**	**1**	**2**		**0**	**1**	**2**
	0	X	X	X	**0**	X	0	0	**0**	X	0	0
	1	X	X	X	**1**	0	X	0	**1**	0	X	0
	2	X	X	X	**2**	0	0	X	**2**	0	0	X
1		**0**	**1**	**2**		**0**	**1**	**2**		**0**	**1**	**2**
	0	X	0	1	**0**	X	X	X	**0**	X	0	1
	1	0	X	0	**1**	X	X	X	**1**	0	X	0
	2	1	1	X	**2**	X	X	X	**2**	0	0	X
2		**0**	**1**	**2**		**0**	**1**	**2**		**0**	**1**	**2**
	0	X	0	0	**0**	X	1	1	**0**	X	X	X
	1	0	X	0	**1**	0	X	0	**1**	X	X	X
	2	0	0	X	**2**	0	0	X	**2**	X	X	X

Obrázek 4 - Maska předností křižovatky tvaru T s vedlejší silnicí (číslo 1)

Při pohledu do hlavní tabulky na [0,0] je patrno, že se v ní na obrázku 4 nacházejí samé křížky. Tyto budou ve výsledném programu nahrazeny nulami. Pro lepší pochopení jsou zde ale křížky, symbolizující nepodstatnost políčka. Tato situace není v zadání uvažována, ale kdyby byl v budoucnu požadavek na otáčení automobilů v křižovatce, bylo by pole stejně podstatné jako ostatní. Při pohledu do políčka [0,1], reprezentujícího tázané auto jedoucí po hlavní silnici z nuly a odbočující na vedlejší silnici doprava, je patrné, že nemusí dávat přednost nikomu. Proto jsou všude mimo diagonálu v podtabulce samé nuly (hodnota `false`). Na ukázku situace, kdy automobil přednost dávat musí, je vhodné pole [1,0], které reprezentuje automobily jedoucí z vedlejší silnice na hlavní a odbočující při tom doleva. Jede-li auto ze silnice číslo jedna, nezajímají ho vozidla ze stejné silnice, proto jsou na řádku 1 podtabulky samé nuly. Automobilům jedoucím po hlavní silnici je třeba dát přednost vždy, kromě případu, kdy automobil jede z hlavní silnice

na vedlejší a odbočuje doprava, tudíž nekřižuje cestu autu jedoucímu z téže silnice. Proto jsou ve zbytku podtabulky jedničky (reprezentující `true`), kromě buňky [1,0][0,1], která je právě případ odbočení z hlavní silnice doprava.



TAB. KŘIŽOVATKY

	0	1	2
0	X	0	0
1	1	X	0
2	0	1	X

PODTABULKA [1,0] x TAB. KŘIŽOVATKY = VÝSLEDNÁ TABULKA 1

	0	1	2
0	X	0	1
1	0	X	0
2	1	1	X

X

	0	1	2
0	X	0	0
1	1	X	0
2	0	1	X

=

	0	1	2
0	X	0	0
1	0	X	0
2	0	1	X

X

PODTABULKA [2,1] x TAB. KŘIŽOVATKY = VÝSLEDNÁ TABULKA 2

	0	1	2
0	X	1	1
1	0	X	0
2	0	0	X

X

	0	1	2
0	X	0	0
1	1	X	0
2	0	1	X

=

	0	1	2
0	X	0	0
1	0	X	0
2	0	0	X

√

Obrázek 5 - Ukázka aplikace masky předností na tabulku křižovatky

Myšlenka masky spoléhá na to, že bude mít každá křižovatka vlastní tabulku, ve které budou v průběhu času evidovány automobily, které jedou v určitém směru. Tato tabulka bude mít rozměry $x \cdot y$ a bude se na ni aplikovat vždy ta podtabulka masky, ke které se vztahuje tázající se auto. Příklad je uveden na obrázku 5. Jsou použity křižovatky z předchozího příkladu z důvodu již vytvořené masky. Tabulku přítomných aut (v obrázku nazvanou tabulka křižovatky) lze jednoduše odvodit. Máme zde dvě auta, jedno jede doleva z vedlejší silnice (směr [1,0]) a druhé

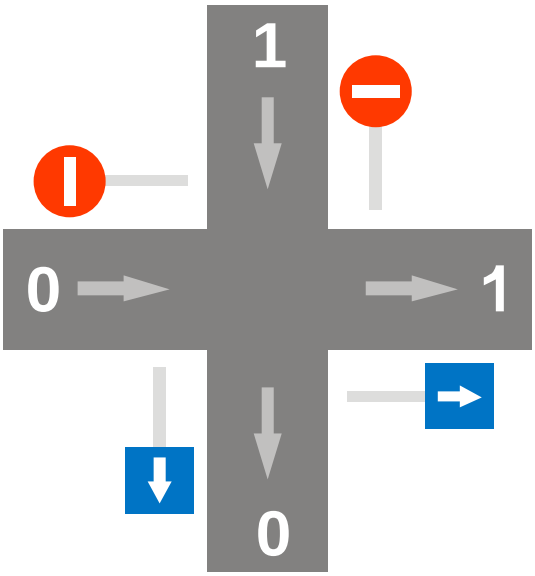
odbočuje doleva z hlavní silnice na vedlejší (ve směru [2,1]). Proto jsou v těchto prvcích jedničky a v ostatních nuly.

To, zda může auto projet, se vyhodnocuje tak, že se vynásobí každý prvek tabulky křižovatky s prvkem se stejnou souřadnicí příslušné podtabulky. Pokud se za celou dobu nenarazí na jiné číslo než na nulu, auto projet může. V opačném případě projet nesmí, protože musí dávat přednost.

Na obrázku 5 je násobena podtabulka [1,0] oranžového automobilu jedoucího z 1 do 0 s maskou křižovatky a narážíme zde na jedničku. Proto oranžový automobil musí dávat přednost. Podtabulka [2,1], náležící modrému automobilu, jedoucímu z 2 do 1, vynásobena tabulkou křižovatky, obsahuje samé nuly, proto modrý automobil v tomto okamžiku projet křižovatkou smí.

Mohlo by se zdát, že v masce jsou diagonální prvky naprosto zbytečné, když se s nimi nikde nepočítá a když na nich nezáleží. To však není pravda. Toto pravidlo platilo pouze na příkladu s křižovatkou tvaru T s obousměrným provozem silnic. Naprosté využití všech prvků masky lze vidět u následujícího příkladu.

Mějme křižovatku křížového tvaru, avšak všechny silnice jsou pouze jednosměrné. Dvě vedou do křižovatky a dvě z ní. Čili jsou zde vstupní silnice 0 a 1 a výstupní silnice 0 a 1. Na obrázku 6 je situace nakreslená pro lepší pochopení.

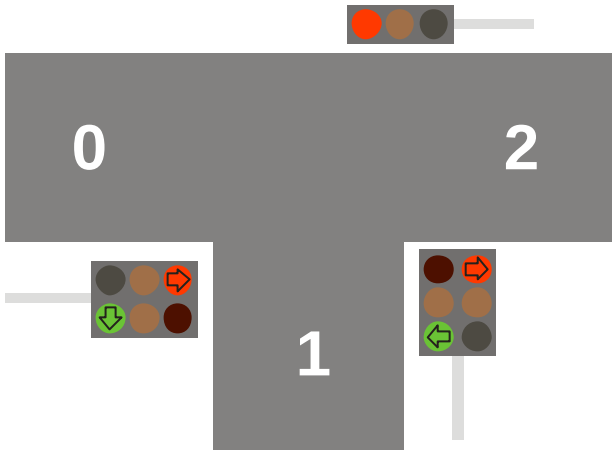


Z \ DO	0			1		
		0	**1**		**0**	**1**
0	**0**	0	0	**0**	0	0
	1	0	0	**1**	0	0
1		**0**	**1**		**0**	**1**
	0	1	1	**0**	0	1
	1	0	0	**1**	0	0

Obrázek 6 - Křižovatka s pouze jednosměrnými silnicemi s příslušnou maskou

Nyní tedy lze jet ze silnice 0 do silnice 0, protože každé číslo reprezentuje jinou komunikaci. Pro ukázkovou křižovatku platí přednost zprava, a proto ať jede auto ze silnice 0 kamkoliv, jsou v masce samé nuly. Pokud jede ze silnice 1, musí dávat přednost automobilům jedoucím ze silnice 0. Samo sobě přednost auto nedává, a proto je poslední řádek nulový. Ještě jedna nula se nachází v buňce [1,1][0,0] a ta vyjadřuje, že pokud jede auto ze silnice 1 doleva a ze silnice 0 jede jiné auto doprava, mohou projet nezávisle na sobě.

Z \ DO	0				1				2			
0		0	1	2		0	1	2		0	1	2
	0	X	X	X	0	X	0	0	0	X	+	1
	1	X	X	X	1	0	X	0	1	+	X	+
	2	X	X	X	2	0	0	X	2	+	+	X
1		0	1	2		0	1	2		0	1	2
	0	X	0	0	0	X	X	X	0	X	+	+
	1	0	X	0	1	X	X	X	1	+	X	1
	2	0	0	X	2	X	X	X	2	+	+	X
2		0	1	2		0	1	2		0	1	2
	0	X	+	+	0	X	+	+	0	X	X	X
	1	+	X	+	1	+	X	+	1	X	X	X
	2	1	+	X	2	+	1	X	2	X	X	X



Obrázek 7 - Jeden ze stavů světelné křižovatky s příslušnou maskou předností

3.1.4 Proměnné udílení předností maskou

Nemělo by cenu vyzdvihovat univerzálnost masky předností, kdyby selhala už jenom u obyčejné světelné křižovatky. Tu lze ale jednoduše vyřešit více typy masek pro každou variantu světel. Pro každou variantu svítících světel bude jedna maska a ty se budou postupně podle přednastavených intervalů měnit.

Na obrázku 7 lze vidět příklad světelné křižovatky v určitém stavu i s příslušející maskou. Tento stav sice neodpovídá skutečné křižovatce, ale zde je pouze pro ukázku. V tomto stavu smí jezdit jenom auta z 1 do 0 a zpět. Ostatní musí stát. Díky univerzálnosti masky stačí v jednom políčku změnit přednost a o zastavení celého pruhu je postaráno. Pokud se totiž například do [2,0][2,0] dá jednička, auto, které odtud přijede, nepojede dál a bude dávat přednost samo sobě. Ostatní auta nejsou podstatná, a proto jsou v tabulce na obrázku nakresleny plusy. Místo nich lze dát nuly i jedničky. Místo křížků patří nuly. Naopak směry, ze kterých smí auta jet, vždy mají ve všech prvcích samé nuly. Pro náš příklad se jedná o [1,0] a [0,1].

Proměnná přednost v křižovatce nenastává pouze při světelné křižovatce, ale třeba i při řízení policistou. Tu zde ale již není třeba rozebírat, protože se jedná o stejné řešení jako u předchozího případu. Prakticky se jedná o sérii masek, reprezentovanou jedním pětirozměrným polem, jehož první rozměr je číslo masky.

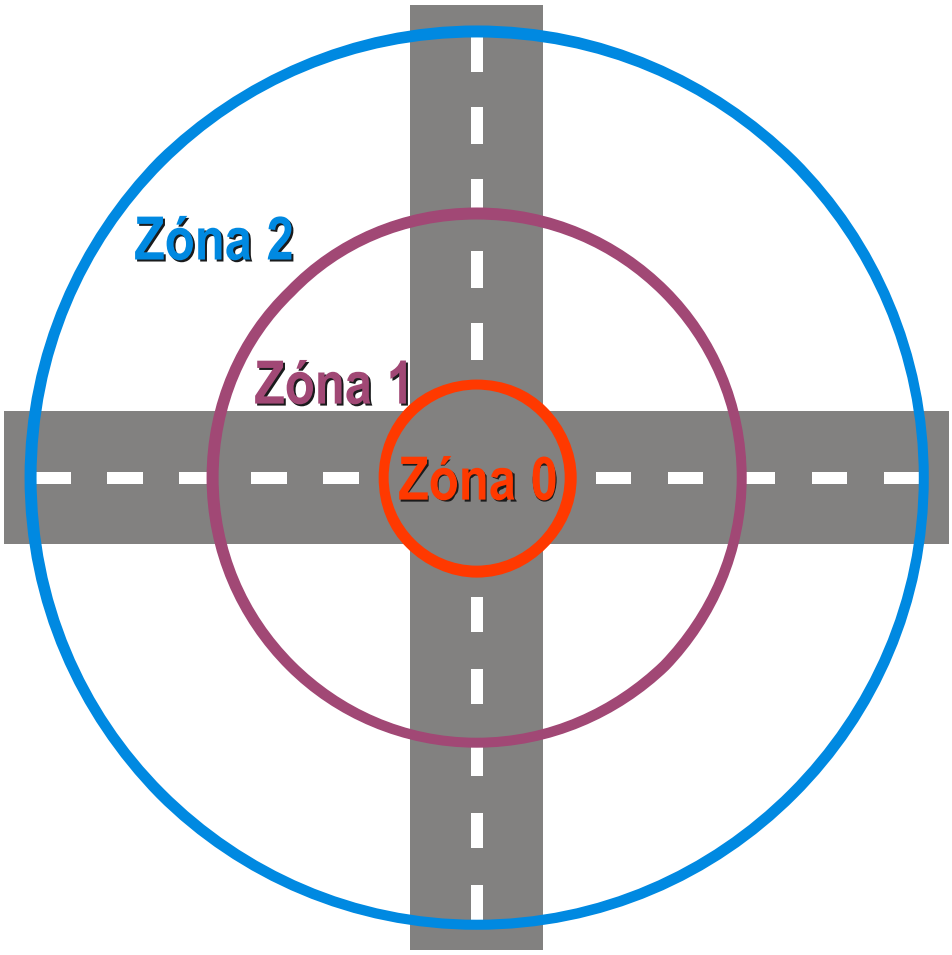
3.2 ZPŮSOBY VYHODNOCENÍ PŘÍTOMNÝCH AUT

Při určování přednosti je v každém již zmíněném případě potřeba znát, kde se v křižovatce a jejím blízkém okolí nacházejí automobily a kam směřují. Bez této informace by vlastně ani nemělo cenu přednost určovat. Proto je zpracováno i toto téma.

3.2.1 Původní vyhodnocování

V programu, na kterém bylo třeba pokračovat a vylepšit v něm křižovatku, se zjišťovaly pouze automobily, které se nacházejí již přímo v křižovatce. Čili nebylo vyhodnocováno okolí křižovatky. Pouze automobily, které ke křižovatce přijížděly, si hlídaly, zda křižovatkou smí projet. Pokud byl zrovna otevřen směr, ze kterého přijížděly, projet směly. Určování směru, odkud se smí jet, však na těchto přijíždějících automobilech nezáviselo. Rozhodovala pouze auta, která stála

v křižovatce, nebo jí projížděla. Jejich přítomnost byla vyhodnocována funkcí objektu `Crossing CarsOnEdgeFromTo()`, která měla jako parametry čísla vstupního a výstupního uzlu křižovatky, mezi nimiž se mohla nacházet auta. Tato funkce si podle uzlů zjistila, o který objekt typu `Edge` se jedná a pokud mu příslušela některá auta, program vyhodnotil `true`. Pomocí toho bylo možno určit, zda například jede vozidlo z komunikace 1 do komunikace 2. Jak již bylo zmíněno, tento způsob nezahrnoval automobily ke křižovatce přijíždějící.



Obrázek 8 - Ukázka zóny 0 až zóny 3 u křižovatky křížového tvaru

3.2.2 Rozdělení na zóny

Myšlenkou rozdělení na zóny je rozdělení okolí křižovatky na dílčí části, ve kterých má přítomnost automobilů jinou prioritu. Vhodným rozdělením se zdá rozdělení na tři zóny. Na obrázku 8 můžeme vidět zónu 0, která se nachází pouze uvnitř křižovatky. Tato zóna je totožná s kompletní oblastí sledovanou v původním

programu. Další zóny číslo 1 a 2 rozšiřují sledovanou oblast. Vzdálenosti, ze kterých by se vyhodnocovaly automobily, by se měnily podle rychlosti. Závisely by tedy na čase, potřebném k zastavení vozidla před křižovatkou.

Každý objekt typu `Crossing`, čili každá křižovatka, by měl v sobě uložena tři pole. V každém poli by se potom ukládal počet automobilů v příslušné zóně, které míří směrem podle souřadnic v poli, stejným způsobem, jako byla tabulka křižovatky v kapitole 3.1.3. Přijíždějící automobil by dle své rychlosti a vzdálenosti od křižovatky vypočítal, za jak dlouho se dostane ke křižovatce, a tím by se zařadil do některé ze tří zón do příslušného políčka.

V zóně 1 i 2 by se hlídala přítomnost aut z ostatních silnic v téže zóně a na tyto by byla aplikována například maska předností. V zóně 2 by se navíc hlídala i zóna 1. Než by se automobil ze zóny 2 dostal do zóny 0, automobily, které křižovatkou projížděly, když byl náš v zóně 2, by jí už dávno projely. Proto se ze zóny 2 zóna 0 nehlídá. Pro auta, která se nacházejí v zóně 1, naopak nejsou důležité automobily ze zóny 2, protože jsou od křižovatky příliš daleko, ale automobily ze zóny 0 důležité jsou, a proto je potřeba tuto zónu hlídat. Ze samotné zóny 0, čili během průjezdu křižovatkou, se již autům přednost neudílí, a tak se z této zóny maska neaplikuje na nic. Tato zóna je zde pouze pro informaci ostatním zónám.

Tento způsob udílení předností je již funkčnější než původní, ale při přejezdech mezi zónami dochází ke skokovým změnám způsobu řízení. Tím je myšleno, že při přejezdu ze zóny 2 do zóny 1 je najednou třeba hlídat i zónu 0, a auto, které si doposud myslelo, že jet smí, najednou zjistí, že nesmí, začne brzdit, když vtom se zóna 0 uvolní a ono může opět zrychlit.

3.2.3 Čas průjezdu křižovatkou

Poslední způsob vyhodnocování přítomných aut využívá přesný čas, kdy bude automobil křižovatkou projíždět. Pokud se v tomto čase již v křižovatce nachází auto, kterému je třeba dát přednost, auto začne zpomalovat, a tím se změní i čas, kdy se do křižovatky dostane.

Jednodušší, avšak méně funkční, je způsob, kdy z každé silnice se hlásí o přednost pouze jeden automobil, a ten nahlásí křižovatce, kdy a v jakém směru hodlá křižovatku projet. V poli bude tedy vždy pouze *n* položek určujících čas vjezdu do

křižovatky a výjezdu z ní, přičemž *n* je počet silnic vedoucích do křižovatky. Pokud mezi těmito časy bude v křižovatce auto, kterému je třeba dát přednost, bude tázající se automobil brzdit.

Tento způsob však neřeší automobily, které jedou v závěsu za prvním a mají jiný směr. Tyto by potom v některých případech musely prudce brzdit, jakmile by auto před nimi projelo křižovatkou a ony by konečně dostaly informaci, zda smí jet dál bez změny rychlosti. Je zjevné, že by to nemohly reálně ubrzdit, a proto by musela být aplikována nereálná brzdná dráha, jako tomu bylo i v původním programu.

Pokud chceme reálné řešení křižovatky, je třeba určit, kolik automobilů je třeba sledovat, abychom dosáhli plynulého provozu bez nereálných brzdných drah. Vzhledem k délce automobilů a bezpečných mezer by mělo stačit měřit 3 nejbližší auta, ale pro obecnější popis si tento počet označme *x*. Při počtu vstupních silnic *n* potom získáme $n \cdot x$ automobilů ovlivňujících dění v křižovatce.

Stejně jako u předchozího případu si auta rezervují čas, kdy budou v křižovatce, ale tentokrát má i druhé až *x*-té auto informaci o ostatních autech v křižovatce. Tím pádem předešlý popsaný problém mizí a tento systém řešení přítomných aut se nejvíce blíží realitě.

Každá křižovatka bude muset opět mít *x* polí, ve kterých budou uloženy časy vjezdů a odjezdů. Čili by se jednalo o pole typu double o rozměrech [*n, m,* 2], kde *n* je počet silnic vedoucích do křižovatky, *m* je počet silnic vedoucích z křižovatky a třetí parametr určuje, zda se jedná o čas vjezdu nebo o čas výjezdu. Pro případ křižovatky o třech vstupech i výstupech a třech hlídaných autech by tedy byla tři pole o rozměrech [3, 3, 2].

3.3 ALTERNATIVNÍ VYUŽITÍ KŘIŽOVATKY

Objekt typu křižovatka lze ve výsledném programu díky své univerzálnosti použít i na modelování dalších specifických situací. V této kapitole budou popsány tři vybrané typy využití. Kruhový objezd, uzavření jednoho pruhu a uzavření obou pruhů na určitý čas.

3.3.1 Kruhový objezd

Křižovatka sama o sobě neřeší speciální typ propojení silnic pomocí kruhového objezdu, ale když se na něj podíváme, skládá se vlastně z dílčích malých křižovatek o 2 vstupech a 2 výstupech. Každý vjezd do kruhového objezdu reprezentuje jeden vstup a jeden výstup a zbylý vstup a výstup je potom samotná část kruhu, která je jenom jednosměrná. Pokud by však byl kruhový objezd malý a mezi silnicemi by byl krátký prostor, je z důvodu vyhodnocování přítomných aut lepší vyřešit celý kruhový objezd jako jednu křižovatku.

3.3.2 Autonehoda a konstrukční práce

Pokud se stane na normální silnici nehoda, nejedná se vlastně o křižovatku, ale dá se pomocí ní simulovat. Jedná se o křižovatku se dvěma vstupy i výstupy, přičemž se dá jet jenom z jednoho vstupu do téhož výstupu a z jednoho směru je třeba dávat přednost tomu směru, kde není nehoda.

Případ konstrukcí na silnici je podobný autonehodě. Opět je zde uzavřen jeden pruh, nebo, což je tu nově, je doprava řízena semafory. Což je realizovatelné světelnou křižovatkou o dvou vstupech a dvou výstupech.

3.3.3 Železniční přejezd a přechod pro chodce

Poslední alternativní využití křižovatky je realizace železničního přejezdu. Je už jedno, zda je se závorami, nebo jenom světelný. Jedná se pokaždé o uzavření obou pruhů na určitý čas. I zde se dá tedy využít objekt `Crossing` se dvěma vstupy a dvěma výstupy. Vlaky jezdí převážně dle předepsaného jízdního řádu, ale i pro malý časový rozptyl (zpoždění vlaků) lze využít náhodné určování časů.

Přechod pro chodce je velmi podobný případu železničního přejezdu. Může být buď bez semaforu nebo se semaforem. Bez semaforu se jedná o kompletně náhodný systém přerušení jízdy. Se semaforem je potom nastavené nějaké minimum času, za který smějí teprve chodci opět silnici přecházet, a tím zastavit provoz. Zatímco u přechodu bez semaforu je náhodná i rychlost přejití silnice, u semaforu je tento čas stálý.

3.4 AUTOMATICKÉ GENEROVÁNÍ MASKY KŘIŽOVATKY

Algoritmus pro automatické generování masky funguje pro neomezený počet silnic a nepočítá s možností otáčení se v křižovatce (vyjetí stejnou silnicí, po které auto přijelo). Umožňuje zpracovat pouze základní typy křižovatky, které mají stejný počet vjezdů jako výjezdů. V algoritmu je třeba nastavit celkový počet silnic do parametru `Silnic` a které dvě silnice jsou hlavní do proměnných `HlavniA` a `HlavniB`. Očíslovány musí být tak, že `HlavniA` má menší hodnotu než `HlavniB`. Pokud bude `HlavniA` větší nebo rovna `HlavniB`, vyhodnotí se křižovatka jako křižovatka bez určených předností a bude se řídit pouze předností zprava.

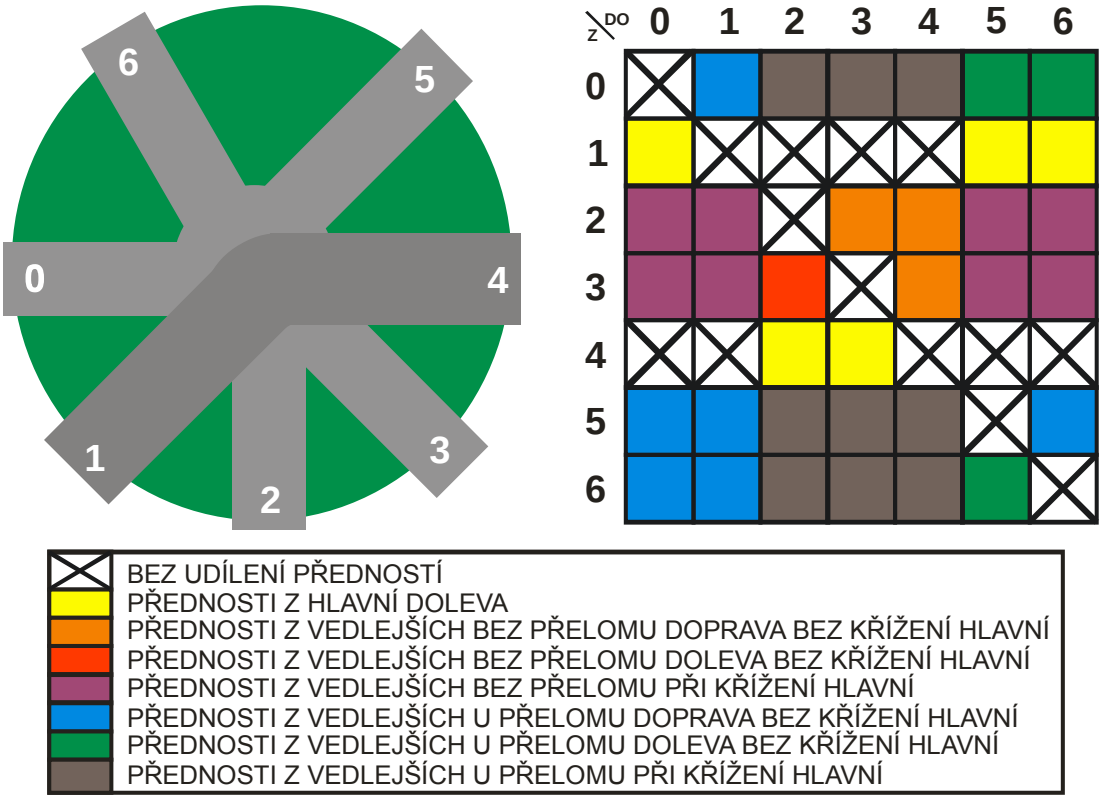
Před samotným generováním masky se celá projede a veškeré hodnoty se nastaví na `false`. Poté je možno masku vyplnit příslušným algoritmem závislým na nastavení hlavních silnic.

3.4.1 Generování pro křižovatku s hlavní silnicí

Jsou-li nastaveny hlavní silnice tak, že je `HlavniA` menší než `HlavniB`, projede algoritmus masku nejprve po vnějších souřadnicích (odkud a kam) a u každé se řídí následujícími pravidly. Pokud je souřadnice `odkud` rovna `HlavniA` nebo `HlavniB` a odbočuje se doprava od protější hlavní silnice, nebo na ni, přednost se neudílí nikomu. V tom případě se tato podtabulka celá nechá, jak je (čili samé hodnoty `false`) a přejde se na vyhodnocení další. Když se odbočuje z hlavní silnice doleva, nastaví se `true` do buněk podtabulky, které reprezentují automobily jedoucí z druhé části hlavní silnice a kterým je křížena cesta. Těmto musí být dána přednost.

Pokud není souřadnice `odkud` rovna žádné z hlavních silnic, jsou tři možnosti, kam se automobil ubírá. První možnost je, že zatáčí doprava a nekřižuje hlavní silnici. Maximálně na ni najíždí. V tomto případě se do podtabulky uloží `true` na souřadnice reprezentující automobily, které jedou z hlavních silnic a kterým je křížena cesta a automobily z vedlejších silnic napravo, jimž je křížena cesta (je jim udílena přednost zprava). Druhá možnost, kam se automobil ubírá, je ta, že odbočuje doleva a nekříží hlavní silnici. V tomto případě je třeba dát přednost všem, kteří se nachází napravo od silnice, odkud automobil přijíždí, až po silnici, kam automobil směřuje a kříží jim cestu. Pozor na situaci, kdy najíždí na hlavní silnici, protože

v tom případě ji svým způsobem kříží, a proto sem tato situace nepatří. Ta patří až do třetí možnosti, kdy automobil křižuje hlavní silnici. V tomto případě je třeba do podtabulky zaznamenat, že mají přednost automobily z obou hlavních silnic, pokud jim je křížena cesta a také všechny automobily napravo od trasy, kterým je cesta křížena. Ukázka všech možností (včetně problematiky přelomu – viz dále) algoritmů pro vyplňování podtabulek je naznačena na obrázku 9 pro křižovatku se 7 silnicemi, přičemž 1-4 je hlavní.



Obrázek 9 - Části algoritmu generování masky pro křižovatku s hlavní 1-4

3.4.2 Generování pro křižovatku bez určených předností a dodatky

Pokud se `HlavniA` rovná `HlavniB`, nebo pokud je `HlavniB` menší, nastala situace, na kterou není předchozí algoritmus připraven. To lze využít na předání informace, že se jedná o křižovatku bez nastavených předností, čili se zde řídí provoz předností zprava. Algoritmus na přednost zprava je značně jednodušší než pro přednost s hlavními silnicemi. Podobně jako v předešlém se projede celá maska po vnějších souřadnicích a pro každou buňku se vyhodnocuje, kterým automobilům

situovaným napravo od trasy se kříží cesta. Těmto je třeba udílet přednost, čili jsou u nich v podtabulce nastaveny hodnoty `true`.

V obou variantách (hlavní silnice a přednost zprava) se nastavují na hlavní diagonálu samé hodnoty `false`. Jelikož jsou silnice křižovatky indexovány v kladném směru otáčení (proti směru hodinových ručiček) od 0 do *n*, kde *n*+1 je počet silnic, je třeba ošetřit for cykly v programu, aby se nepřesáhl počet silnic. Čili že napravo od silnice *n* je silnice 0 a ne silnice *n*+1. Proto jsou ve výsledném algoritmu situace rozděleny ještě na části, kdy se jede přes přelom a kdy ne. Pokud se jede přes přelom, silnice se nejdříve procházejí do *n* a potom ještě dál od nuly.

Automatické generování masky počítá s tím, že automobil, který jede například z 0 do 1, nekřižuje trasu automobilu jedoucímu z 2 do 3 a podobně. Takže pokud je například hlavní silnice číslo 0 a 3, tak nesmí automobily jedoucí z 2 do 1 vůbec vjet na hlavní silnici. Pokud by však byl v budoucnu požadavek na křižovatku, kde by při tomto manévru automobily najížděly do hlavní silnice, lze to v algoritmu změnit.

3.5 VYŘEŠENÍ VZNIKLÉ PATOVÉ SITUACE

Stejně jako ve skutečném provozu, i v simulátoru nastává problém s nedokonalostí křižovatky s předností zprava. Když totiž přijede na křižovatku z každé silnice auto přibližně ve stejný čas, dává každé každému přednost. Všechny mají napravo od sebe automobil, kterému musí dát přednost. Takto nastává patová situace, kdy všechny automobily stojí.

V reálném světě je proto těchto křižovatek málo a když už se někde nacházejí, bývá to většinou v místech s malým provozem, kde je malá pravděpodobnost, že k situaci dojde. Když už však tato situace nastane, zpravidla se vyřeší tak, že nejrazantnější ze řidičů prostě vjede do křižovatky, a rozhodne tak o následujícím průběhu. Uvolní se tak totiž jednomu ze zbylých silnice napravo od něj a může jet on.

Toto řešení bylo zavedeno i do programu. Každá křižovatka má pro tento účel zavedeny parametry `MoznaNejasnost`, `SumaRychlosti`, `Dir` a `Odpocet`. Pokud je parametr `MoznaNejasnost` nastaven na `true`, znamená to, že v křižovatce může

nastat při neproměnném provozu patová situace. Ta se vyhodnotí podle součtu rychlostí (ukládaného do `SumaRychlosti`) nejbližších automobilů u křižovatky. Pokud je téměř nulový, náhodně se vybere jedna ze silnic, uloží se do proměnné `Dir` a začnou se z ní po maximálně 2 sekundy (k tomu slouží proměnná `Odpocet`) pouštět automobily. Pokud se takto situace vyřeší, řídí se křižovatka dál svou maskou. Pokud ne, změní se `Dir` a začne nový odpočet.

3.6 NÁVRHY NA DALŠÍ ROZŠÍŘENÍ SIMULÁTORU

Objekt křižovatky `Crossing` je koncepčně hotový. Není však v simulátoru uživatelsky pohodlně vkládatelný. Je zde z původní verze simulátoru ponechána možnost vkládat křižovatku jako objekt složený z 8 `Node` a 12 propojovacích `Edge`, čili křižovatku se 4 vstupy a 4 výstupy bez možnosti otáčení (viz obrázek 1). Mazáním jednotlivých uzlů by se sice dalo dokázat zmenšení počtu silnic v křižovatce, avšak maska zůstává stejná a mohou nastat problémy s ukazováním automobilů. Proto by bylo dobré se v některé z dalších verzí simulátoru zaměřit na uživatelské rozhraní umožňující vkládání křižovatky s libovolným počtem silnic a s možností určit, které jsou hlavní, nebo určit průběh semaforů. Také je třeba vyladit ukazování automobilů, které se chová správně pouze v křižovatce ve tvaru kříže.

Světelná křižovatka funguje zatím pouze jenom jedna na jednu mapu a pouze jako jeden typ (křížová, kdy se pouští buď jeden přímý směr, nebo druhý přímý směr). Její detailní nastavení by bylo třeba provádět při vkládání v uživatelském prostředí, což nebylo zadáním práce. Toto by bylo vhodné v dalších verzích také vylepšit.

Objekt křižovatka je obecně použitelný i pro více pruhů z jednoho směru. Pro tuto křižovatku by tedy bylo třeba přivést ke křižovatce dvě hrany `Edge`. To v současné verzi není problém, stačí přidat pouze uzel `Node` a z něj vyvést dvě silnice. Pokud by se ale měly silnice sbíhat (sjíždění pruhů), nastává problém s realizací. Pro přednost z jednoho pruhu lze použít objekt křižovatka, ale pokud by se vyžadovalo použití metody ZIP, jedná se o speciální situaci, kterou objekt `Crossing` za pomoci masky předností nezvládá. Pro tento speciální typ křižovatky je třeba naprogramovat speciální typ křižovatky nebo přímo nový objekt.

Jízdu ve více pruzích s vzájemným přejížděním z pruhu do pruhu bude asi těžké do programu implementovat, ale pro komplexní simulaci reálného provozu, jak má simulátor sloužit, je toto rozšíření téměř povinností.

Jak již bylo zmíněno, automobily se pohybují podle pohybových rovnic, kde nesedí jednotky, takže simulace neprobíhá v měřítku. Tento nedostatek byl již v převzaté verzi a nebylo v zadání ho měnit. Avšak v budoucích verzích by asi tento zásah bylo vhodné provést pro lepší využitelnost simulátoru.

Rozhodování automobilů, kam na křižovatce jet, probíhá pouze náhodně. V některé z dalších verzí by bylo dobré navrhnout systém, který naplánuje automobilům trasu, kterou se budou řídit. Jednalo by se o seznam uzlů, kterými musí projet.

Mapa (obrázek silnic a okolí), po které se automobily pohybují, je ve stávající verzi vektorová. Vzhledem k tomu, že se nijak neotáčí, aplikuje se na ni pouze zvětšování a zmenšování, šlo by použít přímo rastrovou reprezentaci grafiky. Při větším přiblížení se sice ztratí ostrost, ale celkově bude běh simulace vizuálně hezčí.

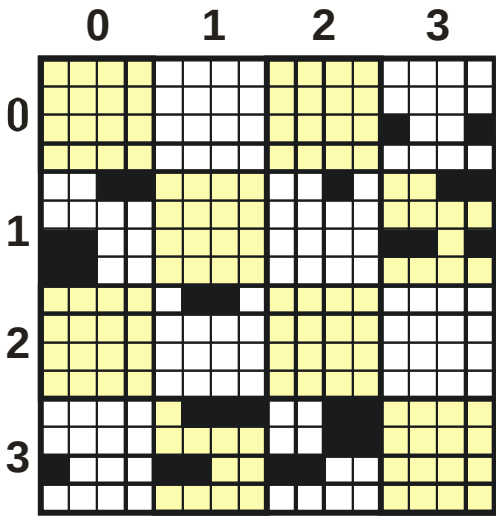
Vzhledem k cílům simulátoru (simulovat skutečný provoz) by bylo dobré implementovat nějaký algoritmus pro automatické vkládání objektů systému silnic (`Edge`, `CarGen`, `Node` a `Crossing`) například z mapy.

4. OVĚŘENÍ FUNKČNOSTI KŘIŽOVATKY

V následujících podkapitolách budou graficky vyobrazeny a okomentovány tři vybrané křižovatky, na nichž byla ověřena funkčnost udílení předností v modelu křižovatky (objekt `Crossing`) pomocí masky předností. Jedná se konkrétně o křižovatku s hlavní silnicí, křižovatku bez určených předností (s předností zprava) a o jednoduchou světelnou křižovatku. Všechny ukázkové křižovatky mají 4 vstupy a 4 výstupy, bez možnosti otáčet se v křižovatce, čili jet z *n*-tého vstupu do *n*-tého výstupu není možno.

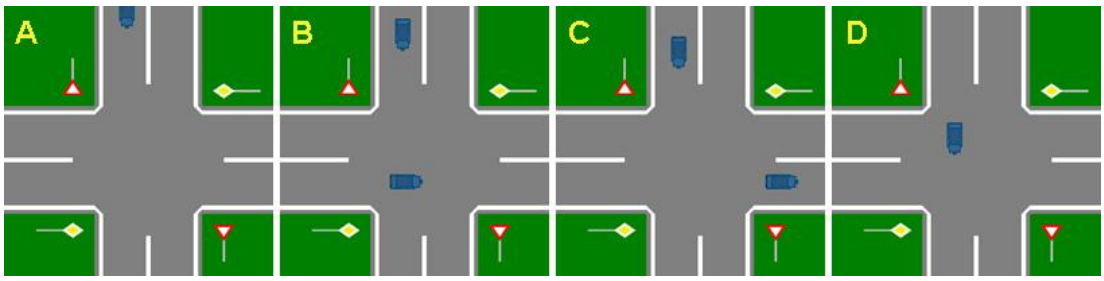
4.1 KŘIŽOVATKA S HLAVNÍ SILNICÍ

První ukázková křižovatka je křižovatka s hlavní silnicí. Její maska (na obrázku 10 – vyčerněná políčka jsou `true`, nevyčerněná `false`) se vyplňuje automatickým generátorem masky, jemuž se předají parametry udávající 4 silnice, přičemž ty s indexy 0 a 2 jsou hlavní.



Obrázek 10 - Automaticky vygenerovaná maska křižovatky s hlavní silnicí

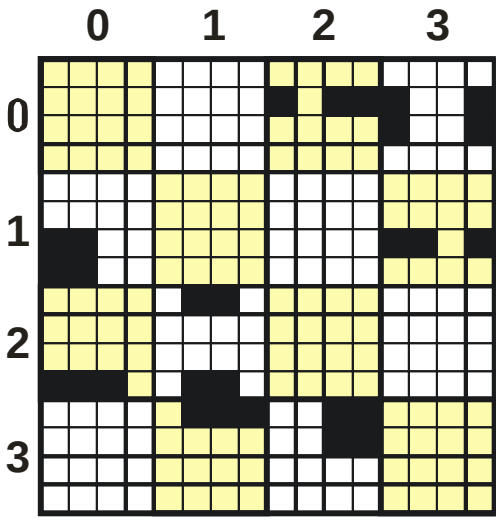
Simulace křižovatky s hlavní silnicí probíhá v grafickém prostředí TRASI s doplňujícími ilustracemi dopravních značek pro lepší přehlednost. Na obrázku 11 je ukázka udělení přednosti (zpomalením) automobilem přijíždějícím z horní vedlejší silnice automobilu projíždějícímu křižovatkou po hlavní silnici zleva. Obrázky ze simulace probíhají chronologicky od obrázku A až po obrázek D.



Obrázek 11 - Průběh simulace křižovatky s hlavní silnicí v TRASI

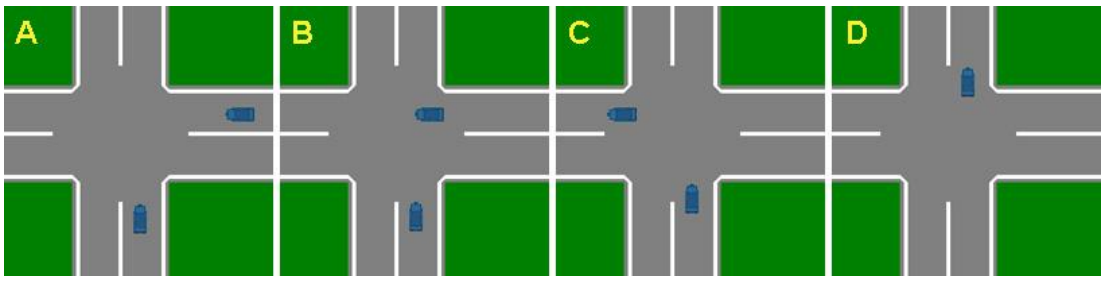
4.2 KŘIŽOVATKA S PŘEDNOSTÍ ZPRAVA

Druhá ukázková křižovatka je křižovatka bez určených předností, čili se řídí předností zprava. Její maska (na obrázku 12 – vyčerněná políčka jsou `true`, nevyčerněná `false`) se vyplňuje automatickým generátorem masky, jemuž se předají parametry udávající 4 silnice, přičemž indexy hlavních silnic se nastaví stejné, aby generátor rozpoznal, že se jedná o křižovatku s předností zprava.



Obrázek 12 - Automaticky vygenerovaná maska křižovatky s předností zprava

Ukázka simulace křižovatky s předností zprava je na obrázku 13. Dolní automobil musí dát přednost zprava automobilu jedoucímu z pravé silnice. Jakmile je již téměř pryč z křižovatky, smí se dolní automobil rozjet a projet křižovatkou. Obrázky ze simulace probíhají chronologicky od obrázku A až po obrázek D.



Obrázek 13 - Průběh simulace křižovatky s předností zprava v TRASI

4.3 JEDNODUCHÁ SVĚTELNÁ KŘIŽOVATKA

Ukázková světelná křižovatka je pouze jednoduchá křížová křižovatka, kde se v první fázi pouští jeden přímý směr a v druhé fázi směr kolmý na předešlý. V simulaci není možné rozhodovat, zda automobily ještě stihnout projet křižovatku na oranžovou, a proto jsou již při oranžovém světle pouštěny pouze automobily odbočující doleva. Ty by byly ve skutečném provozu již najeté v křižovatce a opustily by ji nejpozději v tento moment.

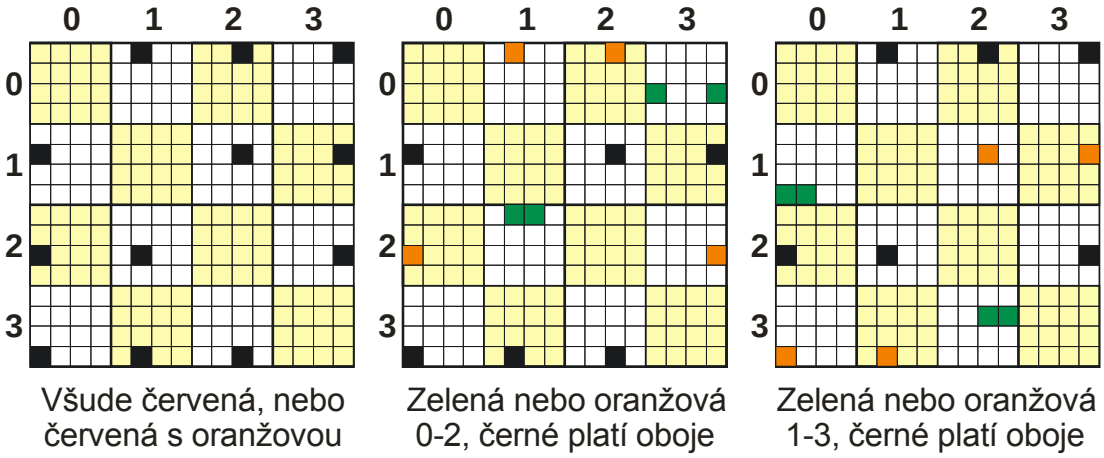
Při svícení červené a oranžové současně se ve skutečném provozu v ČR řidiči chystají na rozjezd. Jelikož tento úkon v simulátoru není třeba, je pro tuto variantu svícení semaforů zakázána jízda všem automobilům, stejně jako při svícení červené na všech silnicích. Tento stav semaforu například v Americe vůbec nemají. Po červené se rozsvítí rovnou zelená.

Chod semaforu probíhá cyklicky v těchto osmi krocích:

1. Všude svítí červená (1500 ms)
2. Na silnicích 0 a 2 svítí červená s oranžovou, na silnicích 1 a 3 červená (1000 ms)
3. Na silnicích 0 a 2 svítí zelená, na silnicích 1 a 3 červená (5000 ms)
4. Na silnicích 0 a 2 svítí oranžová, na silnicích 1 a 3 červená (1000 ms)
5. Všude svítí červená (1500 ms)
6. Na silnicích 0 a 2 svítí červená, na silnicích 1 a 3 červená s oranžovou (1000 ms)
7. Na silnicích 0 a 2 svítí červená, na silnicích 1 a 3 zelená (5000 ms)
8. Na silnicích 0 a 2 svítí červená, na silnicích 1 a 3 oranžová (1000 ms)

Těchto osm kroků využívá pět variant masky z obrázku 14. Vybarvená políčka podtabulek symbolizují hodnoty `true` a nevybarvená (bílá nebo světle žlutá)

`false`. Kroky 1, 2, 5 a 6 používají levou masku. Krok 3 využívá zelenou variantu prostřední masky a krok 4 oranžovou variantu prostřední masky. Krok 7 využívá zelenou variantu pravé masky a osmý krok používá oranžovou variantu pravé masky. Černá políčka v prostřední a pravé masce jsou společná vždy pro obě varianty (zelenou i oranžovou). Narozdíl od předchozích dvou křižovatek se zde používá pole masek `SemaforMask`, které je napevno zadáno (nevyplňuje se automatickým generátorem maky).

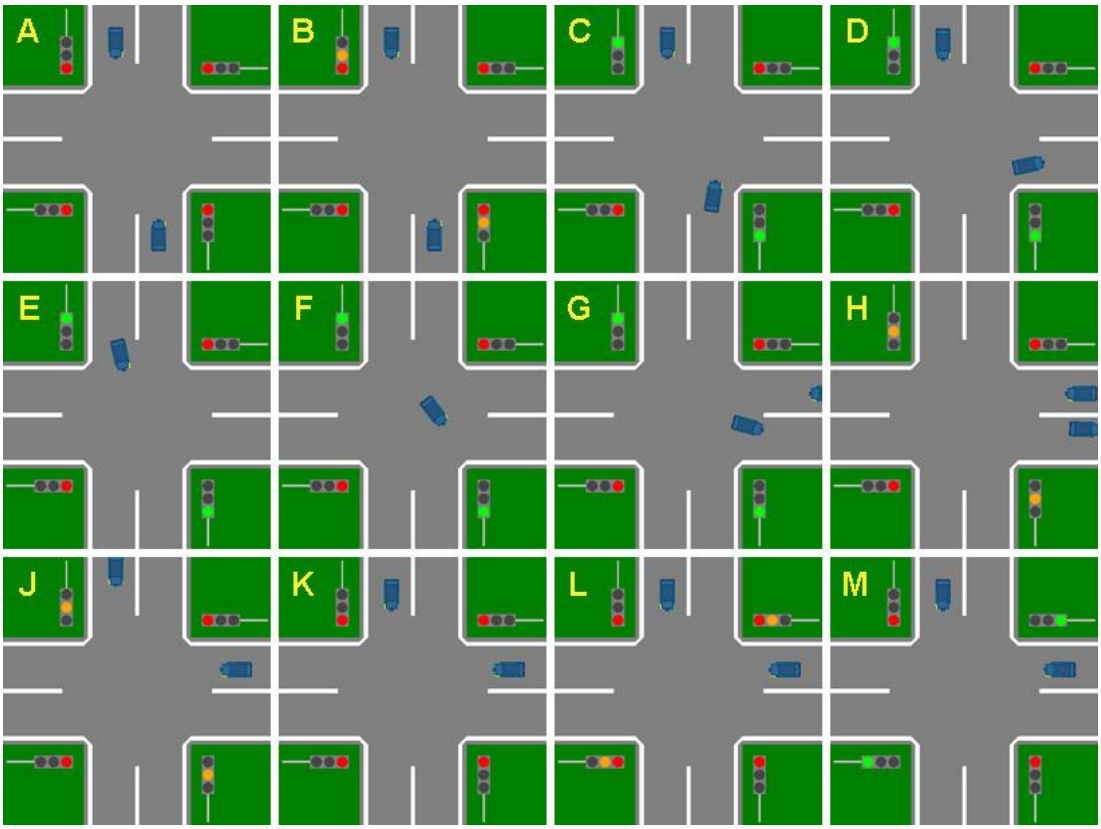


Obrázek 14 - Varianty masky použité pro ukázkovou světelnou křižovatku

Ukázka funkčnosti semaforu a jeho masek ve výsledné simulaci je ukázána na obrázku 15. Jsou zde vyobrazeny všechny kroky kromě čtvrtého. Průběh simulace je následující:

A. Svítí červená, žádný automobil nesmí jet. Světla semaforů jsou v kroku 5.
B. K červené se rozsvěcí ještě oranžová na silnicích 1 a 3. Žádný automobil zatím nesmí jet. Světla semaforů jsou v kroku 6.
C. Na silnicích 1 a 3 svítí zelená, spodní automobil smí jet, horní mu musí dát přednost. Světla semaforů jsou v kroku 7.
D. Semafory svítí stále stejně (krok 7), spodní automobil vyjíždí z křižovatky, horní bude moct projet.
E. Semafory jsou stále beze změny (v kroku 7), hornímu automobilu nic nebrání, smí projet.
F. Automobil projíždí křižovatkou. Žádná jiná změna.

G. Zprava se blíží automobil. Má však červenou, a tak zpomaluje. Automobil shora vyjíždí z křižovatky. Světla semaforů jsou stále v kroku 7.
H. Automobil shora odjíždí, automobil zprava stojí na červené a semafory se změnily na krok 8, čili svítí oranžová na silnicích 1 a 3.
J. Stále svítí oranžová na silnicích 1 a 3, shora přijíždí automobil, ale protože odbočuje doprava, již nesmí na oranžovou projet. Automobil vpravo stále stojí na červené.
K. Všude svítí červená, auta stojí. Světla semaforů jsou v kroku 1.
L. Na silnicích 0 a 2 se rozsvěcí současně s červenou i oranžová. Automobily však všechny stojí. Světla semaforů jsou v kroku 2.
M. Na silnicích 0 a 2 svítí zelená, automobil vpravo může vyjet. Světla semaforů jsou v kroku 3.



Obrázek 15 - Průběh simulace jednoduché světelné křižovatky v TRASI

5. MANUÁL K PROGRAMU TRASI

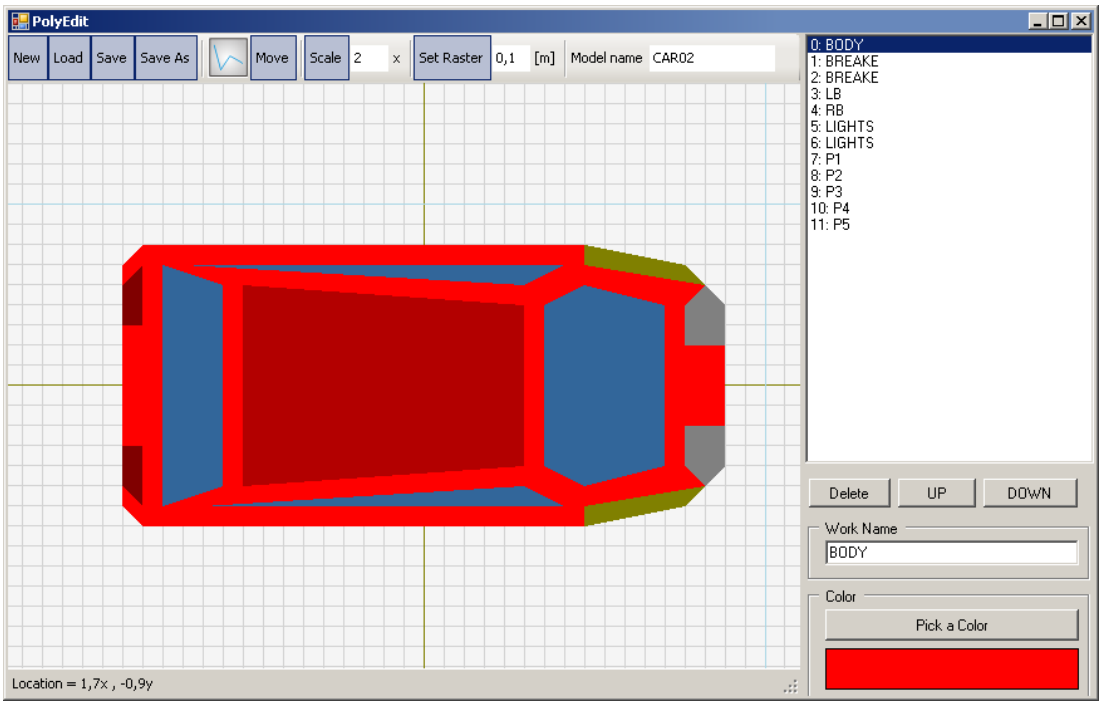
Kapitola je stručným přehledem základních funkcí simulátoru TRASI (TRAffic SImulator). Simulátor sestává ze dvou dílčích programů. PolyEdit je program pro vytváření grafického prostředí simulace (kreslení silnic a automobilů), který lze považovat za dokončený. SimCity je rozpracovaný simulační program, který řeší pohyby automobilů, udílení předností a další. Proto je u něho uveden i stručný popis struktury programu, pro ulehčení následných úprav.

5.1 EDITOR POLYGONŮ POLYEDIT

V programu PolyEdit se vytváří soubory s koncovkou *.model. Obsahem těchto souborů je název modelu a setříděný seznam objektů. Každý objekt má své jméno, seznam bodů polygonu a barvu jeho výplně. Objekty s vyšším indexem v seznamu překryjí ty s nižším.

5.1.1 Uživatelské prostředí PolyEdit

Uživatelské prostředí programu PolyEdit (obr. 16) lze rozdělit na tři části. Kreslicí plochu (vlevo dole), menu (nahoře) a panel pro správu objektů (vpravo).



Obrázek 16 - Uživatelské prostředí PolyEdit

Modely lze pojmenovat, načítat, ukládat, nebo vytvářet nové pomocí položek v menu. Seznam objektů (polygonů) je vypsán v panelu napravo. Zde lze dělat základní úkony se seznamem (mazání objektů a jejich posuv v seznamu), přejmenovávat objekty a měnit jejich barvu.

Pro vytvoření nového objektu je třeba kliknout v menu na ikonku s lomenou čarou a následně klikáním na kreslicí ploše určovat souřadnice bodů polygonu. Body se přichytávají k rastru nastavenému v menu. Pro uzavření polygonu slouží mezerník. Klávesa Esc zruší vytváření nového objektu. Velikost vytvořených polygonů lze měnit pomocí tlačítka Scale a jejich poloha lze měnit tlačítkem Move.

5.1.2 Požadavky na modely pro kompatibilitu

Každá simulace vyžaduje 5 základních modelů. Bez jejich vytvoření sice funguje, ale není vidět její průběh. Jedná se o modely mapy, automobilu, uzlu, křižovatky a generátoru automobilů. Pro každý typ modelu platí pravidla, jak ho pojmenovat, aby ho program SimCity rozpoznal. Model mapy se musí jmenovat *$Map*, model automobilu *CAR01*, model uzlu *$Node*, model křižovatky (ikonka umístěná v jejím středu) se musí jmenovat *$Crossing* a model generátoru automobilů *$CarGen*. Na názvech uložených souborů nezáleží, ale musí být všechny umístěny ve složce se simulací (soubor workspace.cim, který se vytváří v programu SimCity).

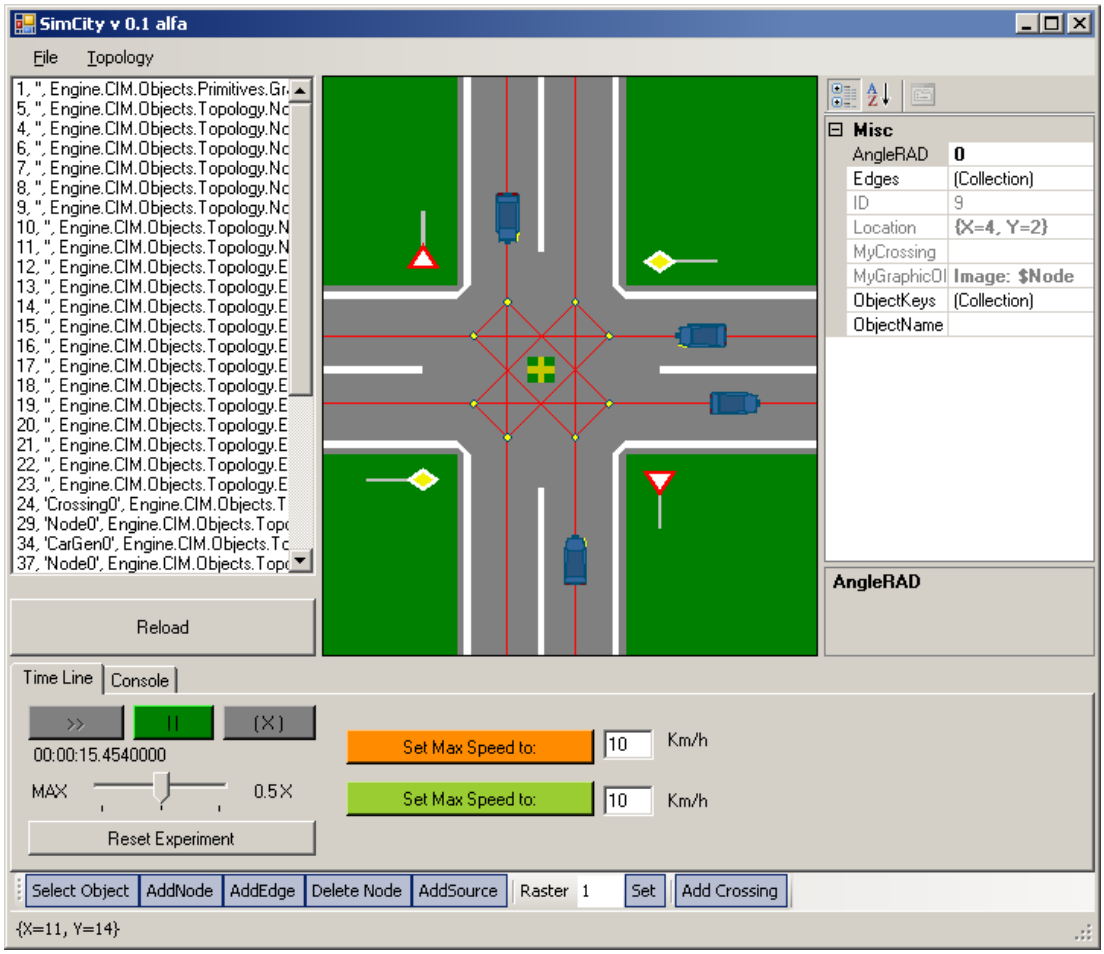
U modelů automobilu a mapy jsou ještě požadavky na pojmenování důležitých objektů. Pro automobil jsou to signální světla, jejichž barva se během simulace mění v závislosti na směru, kterým se automobil bude ubírat. Tato světla se musí jmenovat *LB* (levý blinkr) a *RB* (pravý blinkr), aby je simulátor dokázal vyhledat. Pro zjištění rozměrů automobilu je také třeba vytvořit kostru automobilu a pojmenovat ji *BODY*. Z jejích rozměrů simulátor vypočítá vzdálenosti, kdy dojde ke kolizím.

Model mapy má specifické požadavky pouze pokud se jedná o světelnou křižovatku. Ta musí mít vyobrazené semafory, jejichž barva se stejně jako u signálních světel mění v simulátoru. V současné verzi programu lze vytvářet pouze jednu světelnou křižovatku na mapu a ta musí mít čtyři vjezdy a čtyři výjezdy. To lze ale v programu SimCity pro budoucí verze změnit. Požadované názvy objektů jsou *SemG0, SemO0, SemR0* pro silnici s indexem nula, přičemž G reprezentuje zelené

světlo semaforu, O oranžové a R červené. Pro další silnice jsou semafory pojmenovány stejným způsobem, pouze mají jiné číslo, čili *SemG1*, *SemG2*, *SemG3*, *SemO1*, *SemO2*, atd.

5.2 SIMULAČNÍ PROGRAM SIMCITY

Program SimCity slouží pro simulace silničního provozu. Jeho hlavní funkce jsou vytvoření sítě silnic a následný běh simulace. Pro vizualizace jsou zde využity polygony vytvořené v editoru PolyEdit. Simulaci lze v jakékoliv fázi uložit do souboru workspace.cim. Jiný název souboru nelze použít. Každá simulace musí mít proto vlastní adresář.



Obrázek 17 - Uživatelské prostředí SimCity

5.2.1 Uživatelské prostředí SimCity

Uživatelské prostředí SimCity (viz obrázek 17) obsahuje hlavní menu (nahoře), seznam objektů v simulaci (vlevo), samotnou simulaci (workspace) a její grafické zobrazení (uprostřed), editor parametrů objektů (vpravo) a panel pro ovládání simulace s lištou na vkládání objektů (dole).

V menu programu v položce File lze zavřít simulátor, nebo načíst či ukládat workspace. V otevřeném okně se vždy vybere pouze adresář, ve kterém se nachází požadovaný workspace.cim. Položka Recent otevře poslední používaný workspace.

V menu programu v položce Topology lze pomocí Show Topology vypínat a zapínat zobrazení v reálném světě neviditelných objektů, jako jsou uzly, hrany a generátory automobilů. Položkou Show FPS lze zvolit zobrazení nebo skrytí počtu snímků za sekundu a položka Show Raster vypne či zobrazí rastr v malém okolí ukazatele myši.

5.2.2 Tvorba worskpace a simulace

Pro vložení objektů do workspace slouží tlačítka na liště dole. Tlačítko Add Crossing v současné verzi programu vkládá křižovatku se čtyřmi silnicemi. Jak nastavit její parametry, bude blíže popsáno v kapitole 5.2.4. Po zvolení vkládání křižovatky je třeba ji kliknutím umístit na pracovní plochu.

Rastr lze nastavit na dolní liště zadáním čísla za nápis Raster a stisknutím Set. Tlačítko Add Source slouží k vložení generátoru automobilů, tlačítko Add Node k vložení uzlu. Vložení se opět provede kliknutím na požadované umístění. Tlačítkem Delete Node lze kterýkoliv z těchto dvou objektů (uzel `Node` a generátor automobilů `CarGen`) z workspace vymazat kliknutím na něj nebo do jeho blízkého okolí.

Tlačítko Add Edge slouží k propojování uzlů s uzly nebo s generátory automobilů. Po jeho zvolení je třeba klikáním navolit body, přes které mají automobily projíždět, přičemž první kliknutí se musí nacházet v uzlu nebo generátoru, odkud auta vyjedou a poslední kliknutí musí být jedině v uzlu, kam budou auta směřovat. Po posledním kliknutí je třeba trasu potvrdit stisknutím

mezerníku. Pokud automobil dojede do uzlu, ze kterého nevede žádná hrana (`Egde`), zmizí (vyjede pryč ze simulace).

Tlačítka Set Max Speed To v současné verzi nemají žádnou funkci, ale byla ponechána. Tlačítkem >> lze spustit chod simulace. Rychlost běhu času lze nastavit na posuvníku na tři různé rychlosti. Simulaci lze kdykoliv pozastavit tlačítkem II. Tlačítko (X) vynuluje čas simulace. Po stisknutí Reset Experiment se odstraní veškeré vygenerované automobily. Přepnutím na záložku Console lze sledovat průběhy načítání a ukládání souborů pro workspace.

Parametry vložených objektů lze měnit v panelu pro editaci parametrů vpravo v programu. Objekt, jehož parametry budou upravovány, lze vybrat kliknutím do workspace nebo ho vyhledat v seznamu objektů nalevo od workspace. V případě hran (`Edge`) lze použít pouze výběr v seznamu. Není však vhodné většinu parametrů měnit. Například správný chod světelné křižovatky je závislý na pojmenování křižovatky Crossing0. Jediné, co má smysl měnit, jsou parametry `Attractivity` a `MaximalSpeed` u objektů `Edge`. `Attractivity` určuje šance, s jakými si automobil vybere právě tuto trasu a `MaximalSpeed` udává maximální povolenou rychlost, jakou se smí automobil v daném úseku pohybovat.

Při chodu simulace lze stisknutím tlačítka Refresh přidat do seznamu objektů i právě přítomné automobily. U nich však také není třeba měnit jakékoliv parametry.

5.2.3 Programová struktura

Program SimCity je členěn do tří základních částí. Okno programu (uživatelské prostředí) `MainForm`, systém zpracovávání událostí `GUI` a systém tříd a objektů `Engine`, který je pro úpravy zásadní, a tak zde bude podrobněji rozebrán.

`Engine` obsahuje několik podčástí:

`CIM` – základní třídy a nepohyblivé objekty simulace (*City Infrastructure Model*)

`Interfaces` – základní rozhraní

`MAM` – modely a AI pro automobily a agenty (*Multi Agent Model*)

`NGM` – třídy a funkce pro grafické zobrazení simulace (*Native Graphic Module*)

`SimulationModel` – řízení chodu simulace

`Utils` – pomocné třídy výpočtů a prací s konzolou

City Infrastructure Model obsahuje třídu Infrastructure (v souboru CIM.cs), která se stará o skupiny jednotlivých objektů v simulaci a o mapu. Podskupina `Objects`, v `CIM` obsažená, se dále dělí na `Agents`, `Primitives` a `Topology`. `Agents` se stará o základní vlastnosti agentů, `Primitives` o základní třídy použité při dědění. Součástí oblasti `Topology` jsou kromě třídy `Topology`, která řeší přidávání a mazání objektů, také objekty `CarGen` (generátor automobilů), `Crossing` (křižovatka – včetně masky předností a vyhodnocování přítomných aut v křižovatce), `Edge` (hrana) a `Node` (uzel).

Multi Agent Model je rozdělen na `AgentAI` a `AgentModel`. V druhém zmíněném se nachází třída `AgentHelper` (v souboru Agents.cs), která pracuje se skupinami agentů. `BasicAgentModel` přidává třídě `BasicAgent` z `CIM` závislost na čase, a tedy proměnlivost. `BasicCarModel` dědí po třídě `BasicAgentModel` a přidává parametry příslušející automobilům. V namespace `AgentAI` se nachází `BasicAgentAI` a `BasicCarAI`, která dědí od předešlé a zpracovává chování automobilů po celou dobu jejich jízdy. V této třídě je řešeno udílení předností ve spolupráci s objekty `Crossing` (z `CIM`).

Native Graphic Module obsahuje podoblast `PolygonModels`, která je obsažena i v editoru polygonů `PolyEdit` pro kompatibilitu. Dále `NGM` obsahuje třídu `NativeGraphicModule`, která se stará o zobrazování veškerých objektů a například také o změny barev signálních světel automobilů nebo světel semaforů.

Z ostatních částí stojí za zmínku už jenom `Utils`. Zde se ve třídě `Calc` nacházejí výpočty mocnin, odmocnin, vzdáleností, časů a převody jednotek úhlů. Ve třídě `Range` jsou funkce pro omezení úhlů (z důvodu jejich periodického opakování) a zamezení příliš velkých vzdáleností. Třída `TextConsole` a další mají na starosti psaní na konzolu.

Veškeré třídy dědí své vlastnosti ze základních rozhraní ze složky `Interfaces` a ze třídy `BasicObject`. Rozhraní (interface) upřesňuje, které vlastnosti musí mít nová vytvářená třída a `BasicObject` udává základní parametry, jako je pozice a jiné, které třída zdědí [3].

5.2.4 Detaily k vkládání křižovatky

V současné verzi programu nebylo implementováno uživatelsky příjemné vkládání křižovatek. Je to součástí již zmíněných navrhovaných rozšíření programu. Ve stávající verzi je třeba před vložením křižovatky do workspace nastavit veškeré parametry přímo v kódu programu, což je značně nepraktické.

Veškeré nastavení se nachází v `Engine.CIM.Objects.Topology` ve třídě `Crossing` v konstruktoru (`Crossing(PointF center, float width)`). Zde se jedná o první dva řádky. Na prvním se vyplňuje maska předností křižovatky pomocí funkce `FillMasks(4,0,2)`. První parametr určuje počet silnic v křižovatce a další dva určují, které silnice jsou hlavní (viz automatické generování masky v kapitole 3.4). Druhý řádek určuje proměnnou pro stav semaforu `SemaforStav`. Pokud je tato proměnná nastavena na -1, křižovatka není světelná a stav se samovolně nemůže změnit. Pokud se nastaví na 0 či výše (v současné verzi až 7), křižovatka je světelná a stav se po určitém čase změní o jednotku nahoru. Při překročení maxima se vrátí zpět na nulu a celý chod se opakuje.

Pro případné změny masek pro semafor je třeba zasáhnout přímo do funkce `FillMasks(int Silnic, int HlavniA, int HlavniB)` v dolní části, kde se vyplňuje přednastavený průběh světelné křižovatky o 4 silnicích do polí `SemaforMask` (maska předností), `SemaforLights` (pole určující, kdy bude které světlo svítit) a `SemaforTime` (určuje dobu, za kterou se příslušný stav semaforu změní na další). Při zadání záporné hodnoty do `SemaforTime` se při odpočtu času nastaví náhodná hodnota s maximem v kladné reprezentaci zadané záporné hodnoty. Čili zadá-li se -2000, potom se určí náhodně čas s maximální hodnotou 2000 ms.

6. ZÁVĚR

Prostudoval jsem kód programu a zjistil jsem, jaký je problém ve funkčnosti křižovatky. Následně jsem navrhnul nové řešení pomocí masky předností. Ta umožňuje simulovat libovolně velkou křižovatku, a to jak se statickým udílením předností, tak i s časově proměnným (světelná křižovatka). Dále byl navržen algoritmus pro automatické generování masky základních křižovatek (s hlavní silnicí a s předností zprava) s libovolným počtem silnic.

Systém s maskou předností byl úspěšně implementován do simulátoru včetně automatického generování masky. Problémy v křižovatce vyskytující se v převzaté verzi simulátoru byly implementací masky odstraněny a funkčnost nové verze byla ověřena na třech základních modelových situacích. Na křižovatce s hlavní silnicí, na křižovatce s přednostmi zprava a na jednoduché světelné křižovatce. Maska je však dostatečně obecná pro vyřešení libovolného počtu silnic s možností více pruhů.

7. SEZNAMY

7.1 SEZNAM POUŽITÉ LITERATURY

[1] HONZÍK, P. - SÁBLÍK, V. *Dopravní simulátor – fáze 2 (TRASI – TRAffic SImulator)*. Dokumentace projektu TRASI, 2008.

[2] Kolektiv autorů. *Autoškola – základní učebnice pravidel provozu a dalších předmětů předepsaných autoškolskými osnovami*. Dotisk 4. vydání, 2005, ISBN 80-86411-50-8.

[3] GUNNERSON, Eric. *Začínáme programovat v C#*. Praha: Computer Press, 2001, ISBN 80-7226-525-3.

[4] DYM, Clive L. *Principles of Mathematical Modeling*. Second edition, 2004, ISBN 0-12-226551-3.

[5] BELLOMO, Nicola. *Modeling Complex Living Systems*. First edition. 2008, ISBN-13 978-0-8176-4510-6.

7.2 SEZNAM OBRÁZKŮ

7.3 SEZNAM SOUBORŮ NA CD

- Nová verze TRASI: Workspace 1 – Hlavní silnice
  Workspace 2 – Přednost zprava
  Workspace 3 – Semafor
  lastworkspace.txt
  PolyEdit.exe
  SimCity.exe
- Nové zdrojové kódy: PolyEdit
  SimCity
  Vizualizace generátoru masky
- Původní verze TRASI: Workspace A
  Workspace B
  PolyEdit.exe
  SimCity.exe
- Původní zdrojové kódy: PolyEdit
  SimCity
- Vizualizace generátoru masky: Vizualizace generátoru masky.exe
- Elektronická verze bakalářské práce.pdf