Inteligentní Systémy

Dopravní telematika

11. prosince 2016

Autoři: Adam Jež, xjezad00@stud.fit.vutbr.cz
Tomáš Mlynarič, xmlyna06@stud.fit.vutbr.cz
Ivan Ševčík, xsevci50@stud.fit.vutbr.cz

Fakulta Informačních Technologií
Vysoké Učení Technické v Brně

1 Zadání

Cílem práce je vytvořit model křižovatky s okolím a realizovat adaptivní řízení křižovatky. Požadavkem je použít simulační framework pro realizaci inteligentního řízení a následně systém vizualizovat.

1.1 Upřesnění

K implementaci bude sloužit Java framework Jade, který poslouží jako simulační framework. Pro vizualizaci je možné, že bude použit vlastní jednoduchý systém. Jako model systému bude sloužit křižovatka, přes kterou mohou auta jezdit v libovolných směrech. Ozvláštněním by mohlo být přidání tramvají, které by měly před auty přednost, nebo generování více propojených křižovatek, případně použití jiných typů křižovatek (např. kruhový objezd).

2 Systém

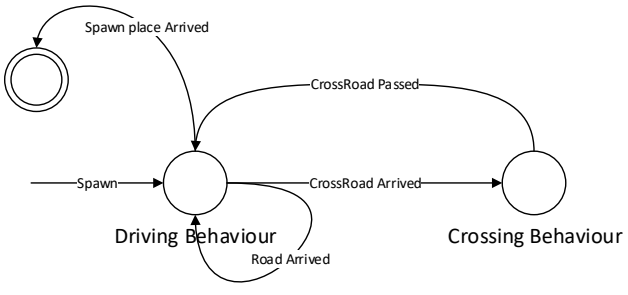
Systém se skládá ze dvou hlavních celků: simulátoru, tedy řídící logiky, a vizualizátoru. Simulace je založena na frameworku Jade, který umožňuje v programovacím jazyce Java ovládat systém agentním způsobem.

2.1 Svět

Svět je modelován jako mřížka bodů, kde v každém bodě je možné umístit objekt systému neboli křižovatku nebo spawn point[1]. Obě tyto místa jsou modelována jako jednotliví agenti, jelikož obsahují logiku pro komunikaci uvnitř systému. Dalším objektem světa je cesta, která však není osazována na specifické místo, nýbrž je napojena na výše zmíněná místa a její poloha je dopočítána z bodů připojených míst (viz sekce 2.4). Nestatickými prvky uvnitř systému jsou vozidla, která jsou modelována jako agenti a obsahují celou logiku chování podobně jako by tomu bylo v realitě (popis viz sekce 2.2).

2.2 Vozidlo

Vozidla reprezentována agenty jsou vytvořena v místech zvaná *SpawnPoint* a také na stejném typu míst jsou vozidla zrušena ze systému. Při vytvoření vozidla je specifikována celá cesta, kterou auto pojede. Cesta je vygenerována po každé náhodně. Agent ovládající vozidlo obsahuje v základu jednoduché chování, které je prezentováno na obrázku 1.

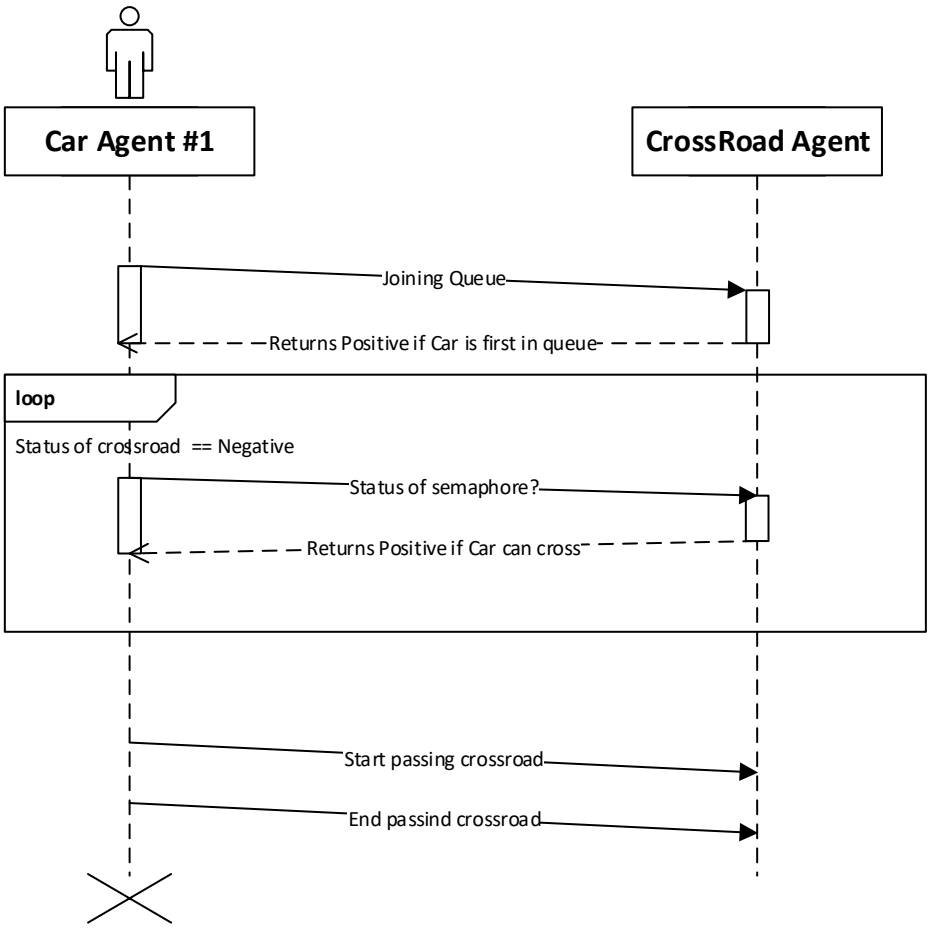


Obrázek 1: Chování agenta vozidla na nejvyšší úrovni

Vozidlo se může nacházet ve dvou stavech kromě stavu terminálního, do kterého se do stane po dojetí ke *SpawnPointu*. Při vytvoření vozidla agent zkontroluje další místo na svojí cestě a chováním

[1] Místo, ve kterém se generují a zanikají vozidla v systému

Driving se k dalšímu místu dostane, což je simulováno probuzením agenta za čas, který odpovídá délce jízdy na dané cestě. V případě dojetí ke křižovatce se spustí chování *Crossing*, které obstarává průjezd křižovatkou. Na sekvenčním diagramu 2 je demonstrována zjednodušená verze komunikace agenta ovládajícího auto a agenta ovládajícího křižovatku.

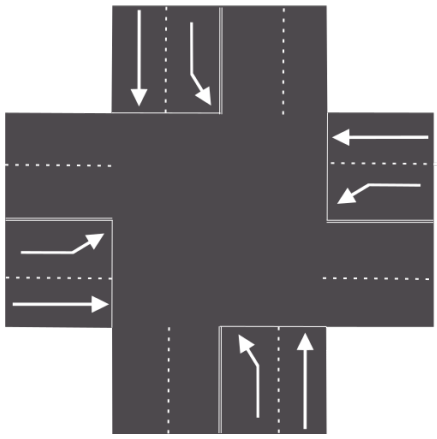


Obrázek 2: Zjednodušená ukázka komunikace při příjezdu vozidla ke křižovatce

Auto znázorněné na sekvenčním diagramu přijelo ke křižovatce, kterou chce projet rovně nebo doprava, chování by bylo identické. Nejprve zašle zprávu, že se chce zařadit do fronty před semafory. V Případě kladné odpovědi auto ví, že se nachází první ve frontě. V opačném případě vozidlo čeká, až se dostane na první místo ve frontě (tento případ není v diagramu znázorněn). Dále se opakovaně ptá křižovatky, jestli může projet, neboli jestli má auto na semaforu zelenou. V případě kladné odpovědi vozidlo začne křižovatkou projíždět (což je opět simulováno uspáním vozidla na určitý časový úsek). O začátku a konci projíždění notifikuje křižovatku, která těchto informací využívá pro umožnění projetí vozidel, které zatáčejí doleva a musí tedy dávat přednost protijedoucím vozidlům.

2.3 Křižovatka

Křižovatka představuje v systému místo, ve kterém se střetává a kříží vícero cest. Návrh systému umožňuje zavedení libovolného tvaru křižovatky, nutné je jenom implementovat odpovídající chování. V rámci projektu byl implementovaný tvar křižovatky "plus"(4 vjezdy, 2 a 2 naproti sobě), který je jedním z nejběžnějších při řízení dopravy a umožňuje ilustrovat dostatečně různorodé řízení křižovatky. Tvar křižovatky je dále specifický tím, že byl v každém směru zavedený jízdný pruh pro odbočení vlevo, který je běžnou součástí frekventovaných křižovatek. Tento jízdní pruh umožňuje lepší řízení křižovatky a plynulejší průjezd aut, jelikož ve velké míře eliminuje blokování aut z důvodu dodržení pravidel silničního provozu, neboli z důvodu nutnosti dávat přednost protijedoucím vozidlům. Obrázek 3 vizuálně představuje modelovanou křižovatku.



Obrázek 3: Ilustrace křizovatky ve tvaru plus

Samotná křižovatka je implementována jako agent s různým chováním zabezpečující komunikaci s okolím, signalizaci, ale i inteligentní řízení. Právě poslední chování umožňuje křižovatce na základě minulosti efektivněji řídit křižovatku a zvyšovat tím její propustnost.

Zmíněné inteligentní řízení je možné popsat následovně. Křižovatka je řízena v cyklu, kdy v jednotlivých, obecně libovolných, za sebou jdoucích fázích nastavuje na semaforu zelenou pro auta v určitých jízdních pruzích, zatímco ostatní semafory nastaví na červenou. Při řízení křižovatky ve tvaru plus byli zvolené následující fáze:

1. Zelená pro oba jízdní pruhy v západním a východním směru.
2. Zelená jenom pro odbočovací (vlevo) jízdní pruhy v západním a východním směru.
3. Zelená pro oba jízdní pruhy v severním a jižním směru.
4. Zelená jenom pro odbočovací (vlevo) jízdní pruhy v severním a jižním směru.

Dále platí, že křižovatka má na vykonání cyklu, tedy všech fází, pevně stanovený čas, tedy periodu s kterou se cyklus opakuje. Navíc, přechod mezi fázemi 2-3 a 4-1 je oddělen krátkým intervalem, ve kterém jsou všechny semafory přepnuté na červenou, což zabezpečuje vyprázdnění křižovatky a předchází tak potencionálním kolizím aut. Inteligentní řízení spočívá v přerozdělení času vyhrazeného pro cyklus mezi jednotlivé fázemi na principu frekventovanosti jízdních pruhů. To se zjišťuje na základě počtu aut v jednotlivých jízdních pruzích těsně před tím, než je semafor pro daný jízdní pruh přepnutý na zelenou. Časy pro jednotlivé fáze jsou alokované podle níže uvedeného vzorce. Platí však, že čas pro jednotlivou fázi nemůže klesnout pod určitou minimální dobu (v implementaci desetina periody), aby se zabezpečilo vyprazdňování i méně preferovaných směrů.

$$
\begin{aligned}
A_k &= \frac{n-1}{n} A_k + \frac{1}{n}\frac{T}{C} B_k \mid k \in \{1,2,3,4\} \\
B_0 &= \#S_A + \#S_C + 0.5(\#L_A + \#L_C) \\
B_1 &= \#L_A + \#L_C \\
B_2 &= \#S_B + \#S_D + 0.5(\#L_B + \#L_D) \\
B_3 &= \#L_B + \#L_D \\
C &= \sum_{i=1}^{4} B_i
\end{aligned}
$$

Kde L_x značí levý odbočující pruh, S_x značí pravý jízdní pruh, $\#Y$ značí zaznamenaný počet aut v jízdním pruhu Y, T značí periodu cyklu, n představuje váhu s jakou nový přepočet ovlivní aktuální alokaci časů, a A, B, C, D značí jednotlivé směry z kterých auta ke křižovatce přijíždějí, tedy západ, jih, východ, sever.

2.4 Cesta

Jak již bylo zmíněno, cesty propojují různá statická místa mapy a tudíž není potřeba specifikovat svou polohu. Poloha je odvozena od propojených míst a délka cesty mezi nimi je pak vypočítána z Euklidovské metriky prostoru na základě souřadnic těchto dvou bodů:

$$
d = (int)\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \tag{1}
$$

Kde d je vzdálenost mezi dvěma body $[x_1, y_1]$ a $[x_2, y_2]$, která je přetypována na datový typ *int*.

V cestě se pohybují vozidla, což je simulováno uspáním agenta na určitou dobu (chování agenta viz 2.2). Doba průjezdu cestou neboli uspání vozidla je pak vypočítána na základě rovnice:

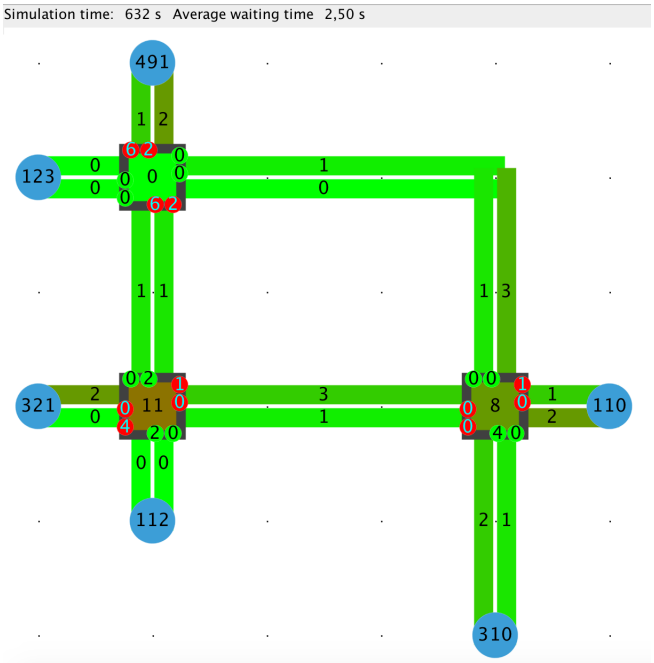
$$
t = \frac{d \times M}{v} \tag{2}
$$

Kde t je výsledný čas průjezdu v sekundách, M je délka jednoho bodu cesty v metrech[2] a v je rychlost vozidla v m/s.

3 Vizualizace a simulace

Vizualizace probíhá ve vlastním okně, které je inicializováno spouštěcím agentem *WorldAgent*. K vykreslení je použita *Java*, konkrétně framework *Swing* jež umožňuje vykreslovat různá primitiva do prvku UI. Na obrázku 4 je možné vidět výslednou vizualizaci systému. Modrými prvky jsou zaznačeny spawn pointy, šedá barva představuje křižovatku, uvnitř které jsou vloženy 2 semafory pro každý směr, tedy vlevo a rovně + doprava. Vizualizace umožňuje zobrazit vytíženost jednotlivých cest a křižovatek gradientní barvou mezi zelenou (vůbec vytížená) a červenou (velmi vytížená) a číselným popisem, kolik se aktuálně nachází aut na dané cestě, stojí ve frontě na daném semaforu či projíždí aktuálně křižovatkou. Čísla uvnitř spawn pointů představují kolik daný spawn point „vyprodukoval“ aut během doby simulace. Jelikož cílem projektu bylo umožnit generovat světy různých velikostí, je vizualizace přizpůsobena velikosti mřížky světa.

[2] Pro účely simulace bylo zvoleno $20m$ pro jeden bod



Obrázek 4: Výsledná vizualizace

3.1 Princip komunikace

Vizualizátor překresluje okno aplikace s danou překreslovací frekvencí (nastaveno na $100ms$). V každém cyklu se zeptá statických agentů (viz 2.1) na svůj stav a poté přerenderuje stav světa. Jelikož je agentů vozidel mnohonásobně více, systém získavání stavu probíhá jinak, protože by nebylo efektivní pollovat všechny a čekat na odpověď. Každé vozidlo proto při vykonáni akce (průjezd místem, zařazení se do fronty na křižovatce) pošle světu zprávu o svém stavu. Vizualizátor si následně tyto zprávy kumuluje a přenáší do vizuální podoby.

3.2 Struktura

Každý objekt, který je možné vykreslit na mapě musí dědit ze třídy *Renderable*. Tím je zaručené generické vykreslování různých objektů, kde každá poděděná třída rozhoduje o vlastním vykreslení. Existují tedy třídy *SpawnPointRenderable*, *CrossRoadRenderable* a *RoadRenderable*. Každý z těchto objektů musí převést souřadnice modelového světa na pixely na obrazovce. Pro to slouží přepočet viz rovnice 3.

$$p = i * \frac{s}{g} \tag{3}$$

Kde p je výsledná souřadnice na obrazovce (y nebo x), i je modelová souřadnice, s je velikost okna a g je velikost mřížky světa.

4 Překlad a spuštění

Pro získání potřebných knihoven k projektu slouží skript *install.sh*, který stáhne knihovny do složky *lib*. Pro překlad projektu slouží skript *compile.sh* a spuštění je možné provést skriptem *run.sh*. U spuštění je možné specifikovat velikost okna nepovinným parametrem. Výchozí velikost je $1000px$.

5 Závěr

Výsledný systém umožňuje generovat různé světy a simulovat dopravu v tomto světě. Byly vytvořené základní objekty pro dopravu, nicméně díky abstrakci je možné rozšířit objekty o různé typy a tvary.

Systém by bylo možné jednoduše rozšířit a dosáhnout tak lepší ovladatelnosti systému. Bylo by možné například parametrizovat různé principy (čas generování aut, rychlost jízdy aut, rychlost simulace atd.), dále by bylo možné přidat například systém uzavírek, kde auta by nemohla jezdit po nějaké cestě.