|  |
| --- |
| OSTRAVSKÁ UNIVERZITA  PŘÍRODOVĚDECKÁ FAKULTA  KATEDRA INFORMATIKY A POČÍTAČŮ |
| Systém pro simulaci pohybu v křižovatce s využitím fuzzy přístupů  Diplomová práce |
| Autor práce: Bc. Jan Mikulík  Vedoucí práce: RNDr. Marek Vajgl Ph.D. |
| 2020 |

|  |
| --- |
| UNIVERSITY OF OSTRAVA  FACULTY OF SCIENCE  [DEPARTMENT OF INFORMATICS AND COMPUTERS](http://prf.osu.eu/kip) |
| System for an intersection simulation using fuzzy approaches  DIPLOMA THESIS |
| Author: Bc. Jan Mikulík  Supervisor: RNDr. Marek Vajgl Ph.D. |
| 2020 |

(Zadání vysokoškolské kvalifikační práce)

ABSTRAKT

Český text abstraktu

*Klíčová slova:*

*(klíčová slova vypsaná na řádku, oddělená od sebe čárkami)*

**ABSTRACT**

The text of the abstract.

*Keywords:*

čestné prohlášení

Já, níže podepsaný/á student/ka, tímto čestně prohlašuji, že text mnou odevzdané závěrečné práce v písemné podobě je totožný s textem závěrečné práce vloženým v databázi DIPL2.

Ostrava dne

………………………………

podpis studenta/ky

|  |
| --- |
| Poděkování |
| Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval/a samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal/a, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.  V Ostravě dne . . . . . . . . . . . .  . . . . . . . . . . . . . . . . . .  (podpis) |

OBSAH

[ÚVOD 10](#_Toc58930460)

[1 CÍL PRÁCE 11](#_Toc58930461)

[2 SOUČASNÁ ŘEŠENÍ 12](#_Toc58930462)

[2.1 Simulátor dopravní sítě - HakChol Pak 12](#_Toc58930463)

[2.2 Simulace automobilového provozu – Ivan Gregor 12](#_Toc58930464)

[2.3 Traffic Lanes - ShadowTree 13](#_Toc58930465)

[2.4 Tvorba nástroje pro interaktivní řešení dopravních situací – Jan Mikulík 13](#_Toc58930466)

[2.5 Požadavky na funkci aplikace 14](#_Toc58930467)

[2.6 Seznam požadavků 14](#_Toc58930468)

[2.6.1 Fuzzy logika 14](#_Toc58930469)

[2.6.2 Určení přednosti v jízdě 14](#_Toc58930470)

[2.6.3 Návrh silnic 14](#_Toc58930471)

[2.6.4 Řízení dopravy semafory 14](#_Toc58930472)

[2.6.5 Uložení/načtení návrhu 14](#_Toc58930473)

[2.6.6 Editace celého návrhu 14](#_Toc58930474)

[2.6.7 Rozlišení trasy pro auta/tramvaje 15](#_Toc58930475)

[2.6.8 Editace pozadí 15](#_Toc58930476)

[2.6.9 Signalizace brzd 15](#_Toc58930477)

[2.6.10 Signalizace odbočení vozidel 15](#_Toc58930478)

[2.6.11 Detekce souběžných silnic 15](#_Toc58930479)

[2.6.12 Vyhlazení silnice 15](#_Toc58930480)

[2.6.13 Vypnutí a zapnutí semaforů 15](#_Toc58930481)

[2.6.14 Vytížení jednotlivých silnic 15](#_Toc58930482)

[2.6.15 Ovládání vlastního vozidla 16](#_Toc58930483)

[2.6.16 Statistické údaje 16](#_Toc58930484)

[2.6.17 Nastavení simulace 16](#_Toc58930485)

[2.6.18 Režimy aplikace 16](#_Toc58930486)

[2.1 Přidané funkce oproti bakalářské práci 16](#_Toc58930487)

[2.2 Kritéria pro splnění cílů práce 17](#_Toc58930488)

[2.3 Seznam funkcionálních požadavků 17](#_Toc58930489)

[2.4 Priorita dokončení požadavků 18](#_Toc58930490)

[3 Fuzzy logika 19](#_Toc58930491)

[3.1 Co je fuzzy logika 19](#_Toc58930492)

[3.1.1 Fuzzy proměnná 20](#_Toc58930493)

[3.1.2 Fuzzy výrazy 20](#_Toc58930494)

[3.1.3 Fuzzy pravidla 21](#_Toc58930495)

[3.1.4 Výpočet výsledné hodnoty 22](#_Toc58930496)

[3.2 Implementace v aplikaci 23](#_Toc58930497)

[3.2.1 LFLController 23](#_Toc58930498)

[3.2.2 Použití dat z LFLC v aplikaci 24](#_Toc58930499)

[3.2.3 Pohyb vozidel za sebou 26](#_Toc58930500)

[3.2.4 Rozhodnutí udělení přednosti 29](#_Toc58930501)

[4 představení vytvořeného řešení 32](#_Toc58930502)

[4.1 Režimy aplikace 32](#_Toc58930503)

[4.2 Uživatelské rozhraní 32](#_Toc58930504)

[4.2.1 Návrh 32](#_Toc58930505)

[4.2.2 Testování 32](#_Toc58930506)

[4.3 Funkční prvky aplikace 33](#_Toc58930507)

[4.3.1 Vložení pozadí 33](#_Toc58930508)

[4.3.2 Možnosti návrhu 33](#_Toc58930509)

[4.3.3 Tvorba silnic 33](#_Toc58930510)

[4.3.4 Určení předností 34](#_Toc58930511)

[4.3.5 Nastavení semaforů 34](#_Toc58930512)

[4.3.6 Nastavení vytížení silnic 34](#_Toc58930513)

[4.3.7 Uložení návrhu 35](#_Toc58930514)

[4.4 Architektura aplikace 35](#_Toc58930515)

[4.4.1 Přínosy práce 35](#_Toc58930516)

[4.5 Měření rozdílů 36](#_Toc58930517)

[4.5.1 Pohyb 36](#_Toc58930518)

[4.5.2 Změna směru 36](#_Toc58930519)

[4.6 Splnění požadavků 36](#_Toc58930520)

[4.7 Fuzzy logika v aplikaci 36](#_Toc58930521)

[5 metodika tvorby práce 37](#_Toc58930522)

[5.1 Vize 37](#_Toc58930523)

[5.2 Způsob práce 37](#_Toc58930524)

[5.3 Způsob zápisu požadavků 37](#_Toc58930525)

[5.4 Správa verzí aplikace 39](#_Toc58930526)

[5.5 Použité nástroje 39](#_Toc58930527)

[5.5.1 NetBeans 39](#_Toc58930528)

[5.5.2 GitHub 40](#_Toc58930529)

[5.5.3 Trello 40](#_Toc58930530)

[5.5.4 MS Teams 40](#_Toc58930531)

[ZÁVĚR 41](#_Toc58930532)

[RESUMÉ 42](#_Toc58930533)

[SUMMARY 43](#_Toc58930534)

[SEZNAM POUŽITÉ LITERATURY 44](#_Toc58930535)

[SEZNAM POUŽITÝCH SYMBOLŮ 45](#_Toc58930536)

[SEZNAM OBRÁZKŮ 46](#_Toc58930537)

[SEZNAM TABULEK 47](#_Toc58930538)

[SEZNAM PŘÍLOH 48](#_Toc58930539)

ÚVOD

Text

1. CÍL PRÁCE

Cílem práce je aplikace fuzzy přístupu do řízení pohybu vozidel v simulátoru křižovatky. Autor ve svém řešení pokračuje bakalářské práci. Do jím vytvořeného simulačního modelu začlení chování vozidel na základě fuzzy přístupu, čímž by měl navýšit subjektivní real-life pocit z chování simulace.

1. SOUČASNÁ ŘEŠENÍ
   1. Simulátor dopravní sítě - HakChol Pak

Aplikace byla vytvořena studentem na Karlově Univerzitě v Praze roku 2010, na které pracoval v rámci bakalářské práce.

Tato aplikace umožňuje uživateli editaci silničních situací, jež jsou vyznačeny úsečkami. Navrženou situaci lze během simulace libovolně upravovat.

Doprava je řízena pomocí semaforů, které zde jsou pro řízení provozu jediná možnost.

Během simulace lze upravovat její rychlost, aktuální počet vozidel na silnicích a jejich frekvenci generování. Při spuštění simulace se zaznamenávají statistické údaje, jako je celkový počet vozidel, ujetá vzdálenost či dobra jízdy po jednotlivých cestách.

Aplikace je vytvořena v programovacím jazyce C#. Samotnou aplikace ovšem nebylo možno otestovat, jelikož není dostupná ke stažení [[1]](#PakHakChol).

* 1. Simulace automobilového provozu – Ivan Gregor

Tato aplikace byla taktéž vytvořena v rámci bakalářské práce, na Karlově Univerzitě v Praze, roku 2006.

Práce obsahuje především popis programu pro simulaci silničního provozu, který byl vytvořen v jazyce Java.

Silniční situace jsou tvořeny úsečkami znázorňujícími cesty. Tyto situace lze do aplikace pouze vložit, nelze je vytvářet ani editovat.

Doprava je řízena pouze semafory a omezením rychlosti. Jednotlivé silnice zaznamenávají statistické údaje o jízdě vozidel jako je průměrná, maximální a minimální doba jízdy, aktuální a celkový počet vozidel na silnici.

Vozidla jedou po předem definovaných trasách a zároveň mají své fyzické vlastnosti, tedy délku vozidla, váhu, spotřebu, maximální rychlost či zrychlení. O jednotlivých vozidlech se také zaznamenávají údaje, které jsou závislé na jejich vlastnostech.

Aplikaci ovšem nebylo možno otestovat, není veřejně ke stažení. Informace byly získány pouze z dokumentace, a nelze ověřit jejich pravost [[2]](#GregorIvan).

* 1. Traffic Lanes - ShadowTree

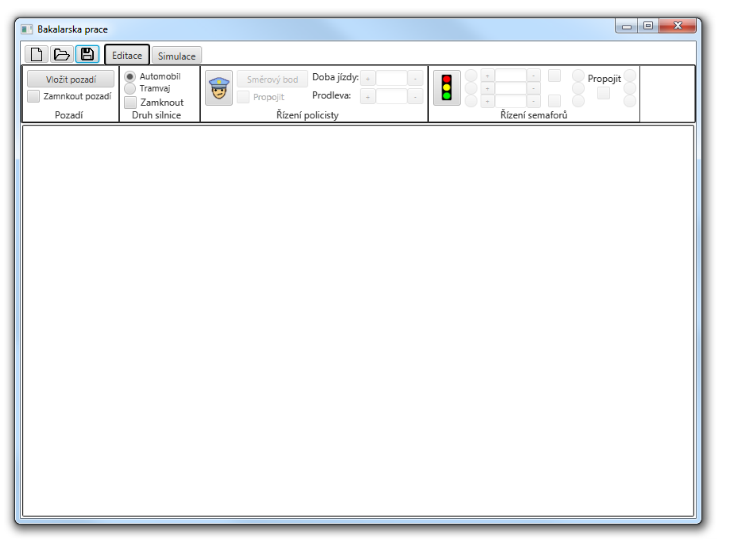
Aplikace je určena pro mobilní zařízení s operačním systémem Android, verze 2.3 a vyšší. Jedná se o hru, ve které probíhá simulace zvolené křižovatky, při níž uživatel ovládá vozidla a semafory. Účelem hry je zajistit plynulý provoz, bez způsobení dopravní nehody.

V aplikaci je k výběru několik desítek předem vytvořených křižovatek s různými obtížnostmi. V simulaci lze upravovat frekvenci generování vozidel. Mění se zde také počasí a ubíhá čas, mění se tedy i den a noc, což ovlivňuje způsob jízdy. V aplikaci jsou zobrazeny reálné křižovatky z pohledu shora, což umožňuje její přehledné ovládání [[3]](#ShadowTree).

Aplikace je možno stáhnout pomocí služby Google Play.

* 1. Tvorba nástroje pro interaktivní řešení dopravních situací – Jan Mikulík

Aplikace byla vytvořena v rámci bakalářské práce [[4]](#Mikulik), což je první verze aplikace vytvořené v rámci této diplomové práce. Práce se zabývá tvorbou aplikace pro simulaci dopravních situací, avšak pohyb vozidel a řízení provozu jsou tvořeny jednoduchými podmínkami, bez lepší logiky. V implementaci fuzzy logiky do aplikace diplomové práce je tedy největší rozdíl oproti první verzi aplikace.



Vzhled první verze aplikace

Vzhled aplikace a rozložení komponent je oproti diplomové práci méně přívětivý, navíc aplikace neobsahuje všechny funkce jako v této práci.

* 1. Požadavky na funkci aplikace

Před začátkem práce na aplikace byly stanoveny základní funkce, které aplikace bude nabízet. Seznam také zahrnuje několik požadavků, které již dříve byly obsaženy v bakalářské práci autora, na kterou tato práce navazuje.

* 1. Seznam požadavků
     1. Fuzzy logika

Hlavním požadavkem je implementace fuzzy logiky do aplikace. Pomocí této logiky se především určuje rychlost jedoucích vozidel podle dané situace. Účelem použití je plynulejší pohyb vozidel a jejich reakce na danou situaci.

* + 1. Určení přednosti v jízdě

Při spojení nebo křížení silnic je možno jednoduše zvolit, která ze silnic je hlavní a která vedlejší.

* + 1. Návrh silnic

Dopravní situaci lze vytvořit pomocí křivek znázorňujících silnice. Tyto křivky bude možno vytvořit, propojit či rozpojit nebo také smazat.

* + 1. Řízení dopravy semafory

Do návrhu lze vložit semafory a nastavit časovou linii přepnutí jejich přepnutí na určitou barvu.

* + 1. Uložení/načtení návrhu

Rozpracovaný návrh může být uložen do zvoleného souboru a poté opět načten do aplikace.

* + 1. Editace celého návrhu

S návrhem situace lze libovolně manipulovat, tedy jej posouvat, přiblížit či oddálit.

* + 1. Rozlišení trasy pro auta/tramvaje

V návrhu jsou typy vozidel, tramvaje a automobily. Trasa, po které vozidlo jede, je zvolena pro každý typ vozidla zvlášť.

* + 1. Editace pozadí

Do návrhu lze vložit vlastní pozadí, kterým je možno pohybovat, změnit jeho velikost, otáčet, nebo jej smazat.

* + 1. Signalizace brzd

Při zpomalení vozidla se rozsvítí zadní brzdová světla.

* + 1. Signalizace odbočení vozidel

Pokud vozidlo odbočuje, směr jeho jízdy je signalizován blinkrem.

* + 1. Detekce souběžných silnic

Při rozpojení nebo spojení jsou zobrazeny části cesty, které jsou blízko sebe. Na těchto úsecích nemohou vozidla jet vedle sebe.

* + 1. Vyhlazení silnice

Možnost vyhlazení klikaté části cesty do rovinky.

* + 1. Vypnutí a zapnutí semaforů

Přepnutí semaforů do režimu blikání oranžové, kdy začnou platit základní přednosti. Poté možnost opětovné zapnutí semaforů.

* + 1. Vytížení jednotlivých silnic

Pro označenou počáteční silnici lze nastavit frekvenci generování vozidel, tedy její vytížení.

* + 1. Ovládání vlastního vozidla

V testovacím režimu lze vložit vlastní vozidlo, které uživatel ovládá změnou jeho rychlosti.

* + 1. Statistické údaje

Během probíhající jízdy uživatele jsou zaznamenávány údaje o provozu, dopravních nehodách či přestupcích, kterých se dopustil. Po dokončení jízdy je uživatel ohodnocen.

* + 1. Nastavení simulace

Simulaci provozu lze nastavit dle potřeb. Může být nastavena frekvence generování vozidel, zapnutí pro každý typ vozidel zvlášť, nebo spuštění semaforů.

* + 1. Režimy aplikace

Aplikace je rozdělena do tří režimů. V editačním režimu se nejprve vytvoří návrh situace. Poté lze spustit testovací režim, ve kterém uživatel ovládá své vozidlo. Vytvořený návrh lze spustit jako spořič obrazovky, při čemž probíhá simulace provozu zobrazena přes celou obrazovku.

* 1. Přidané funkce oproti bakalářské práci

Vytvořená aplikace obsahuje mnoho funkčních částí z rozšiřované bakalářské práce, ovšem převzaté funkce jsou navíc vylepšeny, případně vytvořeny jiným způsobem.

K převzatým funkcím byly navíc vytvořeny další funkce, kvůli kterým je v aplikacích největší rozdíl. Zde jsou jednotlivé funkcionality rozšiřující původní aplikaci.

* Vyhlazení silnice
* Řízení rychlosti pomocí fuzzy logiky
* Povolení jízdy pomocí fuzzy logiky
* Vypnutí a zapnutí semaforů
* Nový systém semaforů
* Detekce souběžných silnic
* Vytížení jednotlivých směrů jízdy
* Úprava grafického rozhraní
  1. Kritéria pro splnění cílů práce
* Aplikace je funkční, lze spustit a používat bez chyb/pádů aplikace
* Jsou splněny požadavky na funkcionalitu aplikace označené jako základní (must-be v Trellu), hodnocení závislé také na splnění požadavků (should-be)
* Aplikace využívá pro řízení provozu fuzzy logiku, auta akcelerují a brzdí plynule s ohledem na okolní provoz
  1. Seznam funkcionálních požadavků

Seznam jednotlivých požadavků z aplikace Trello, které byly postupně implementovány do aplikace. Požadavky jsou rozděleny podle priority, tedy must-be (musí být) a should-be (může být).

|  |  |
| --- | --- |
| **Must-be** | **Should-be** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

* 1. Priorita dokončení požadavků

Jednotlivé požadavky jsou seřazeny podle priority. Čím větší číslo, tím požadavek důležitější a na jeho dokončení může záviset funkčnost jiných funkcí. U každého požadavku je také předpokládaná doba dokončení.

Předpoklad dokončení požadavků

|  |  |  |
| --- | --- | --- |
| **Priorita** | **Název** | **Předpoklad dokončení** |
| 5 | Fuzzy logika | 9. 2019 |
| 5 | Určení přednosti v jízdě | 10. 2019 |
| 4 | Návrh silnic | 10. 2019 |
| 4 | Řízení dopravy semafory | 11. 2019 |
| 3 | Uložení/načtení návrhu | 11. 2019 |
| 3 | Editace celého návrhu | 12. 2019 |
| 2 | Rozlišení trasy pro auta/tramvaje | 12. 2019 |
| 2 | Editace pozadí | 1. 2020 |
| 2 | Signalizace odbočení vozidel | 1. 2020 |
| 2 | Ovládání vlastního vozidla | 1. 2020 |
| 2 | Statistické údaje | 2. 2020 |
| 1 | Nastavení simulace | 2. 2020 |
| 1 | Režimy aplikace | 2. 2020 |

Ze začátku bylo cílem dokončit práci začátkem léta 2020, avšak kvůli nastalým vládním opatřením se práce na delší dobu přerušila. Od této doby se další termíny dokončení a záznamy nevedly, jelikož byla práce na projektu individuální a největší část byla dokončena až na podzim roku 2020.

1. Fuzzy logika

Práce se zabývá implementací fuzzy logiky do simulace silniční dopravy. V této kapitole jsou základní informace o tom, co je fuzzy logika, proč a jak je využita. Také je zde popsána samotná implementace v aplikaci ukázán rozdíl oproti původní aplikaci, kde fuzzy logika implementována nebyla.

* 1. Co je fuzzy logika

Fuzzy logika se zabývá, jak již název napovídá, prací s nepřesnými daty. Lze tedy na základě pouze několika hodnot vyvodit závěry, což by při jiném způsobu řešení bylo mnohem složitější, nebo nepřesné.

Pro příklad řekněme, že chceme automatizovaně na základě teploty určovat účinnost klimatizace. Při běžném přístupu by musely být určeny parametry nebo hranice pro jednotlivé teploty, na jejichž základě by se klimatizace spustila nebo vypnula. Díky fuzzy logice tyto hranice nemusí být přesně definovány a určeny budou jen základní podmínky.

K práci s fuzzy logikou jsou využity fuzzy proměnné, což jsou vstupní a výstupní parametry logiky. Tyto proměnné jsou rozděleny na fuzzy výrazy, které znázorňují rozdělení rozsahu vstupních a výstupních proměnné na „skupiny“, pomocí kterých se následně tvoří fuzzy pravidla. Fuzzy pravidla tvoří kombinace jednotlivých fuzzy výrazů pro dané fuzzy proměnné. Pro výpočet výsledné hodnoty je na vstupní hodnoty aplikována aproximace a následně defuzzyfikace. Detailněji jsou tyto pojmy popsány v odborných textech zabývajících se umělou inteligencí [[5]](#Novak)[[7]](#JanosekFarana).

* + 1. Fuzzy proměnná

Pro využití fuzzy logiky musí být definovány proměnné. Tyto proměnné obsahují hodnoty, se kterými se pracuje. Proměnné jsou rozděleny na vstupní, kterých může být více a jednu výstupní.

Proměnná má definovanou svou minimální a maximální hodnotu, a její střední hodnotu, které ovšem určuje rozložení proměnné, nikoliv její střed.

Každá proměnná má své výrazy, které definují jistou množinu, mezi minimální a maximální hodnotou.

* + 1. Fuzzy výrazy

Výrazy jsou parametry, které rozdělují proměnnou na menší množiny, a jednotlivým hodnotám přiřazuje stupeň příslušnosti.

Stupeň příslušnosti určuje, jak moc hodnota náleží do dané množiny. Tento stupeň má hodnotu mezi 0 a 1, při čemž 0 znamená, že do množiny nepatří a 1 že do množiny zcela náleží.

Pro jednodušší práci má každý výraz také svůj název, s nímž se pracuje při určování pravidel. Například na výše uvedeném obr 1. je označen fuzzy výraz „too close“, který označuje část fuzzy proměnné v jistém rozsahu. Namísto práce s hodnotami tohoto rozsahu je dále pracováno pouze s názvem.

Množiny a stupně příslušnosti jsou pro jednoduchou práci tvořeny grafy, které mají základní tvary. Tyto tvary jsou trojúhelník, čtyřhran a křivka.

Každý fuzzy výraz má několik základních parametrů. Základním parametrem je „Support“, tedy základna, která zahrnuje rozsah proměnní, kde je stupeň příslušnosti daného výrazu větší než 0. Tato základna má svou minimální (levou) a maximální (pravou) hranici. Na obrázku 1 to jsou sloupce „Left Supp“ a „Right Supp“.

Druhým parametrem je Kernel, což je jádro fuzzy výrazu. Jádro označuje rozsah, kde je stupeň příslušnosti 1. Stejně jako základna, i jádro je určeno dvěma parametry, tedy minimální a maximální hranicí. V LFLC jsou pojmenovány „Left Ker“ a „Right Ker“.



Fuzzy proměnné

* + 1. Fuzzy pravidla

Pravidla určují vztahy mezi vstupními a výstupními proměnnými, tedy vztahy jednotlivých výrazů. K tvorbě slouží základní podmiňovací zápis, což je IF podmínka THEN výsledek. Na obrázku 3 lze vidět dvě vstupní fuzzy proměnné a jednu výstupní. Podmínku pro každé pravidlo tedy tvoří kombinace obou vstupních proměnných. Výsledek je určen výstupní proměnnou.

****

3 Příklad fuzzy pravidel

* + 1. Výpočet výsledné hodnoty

Pro získání výsledné hodnoty je potřeba provézt dva základní kroky. Prvním krokem je transformace daných fuzzy pravidel do funkce vysvětlující dané závislosti. Pro vytvoření této funkce je použita aproximace. Každé pravidlo charakterizuje výslednou funkci lokálně v nějakém vhodném okolí, při čemž každému pravidlu je přiřazena fuzzy relace. V tomto případě je jako relace použita disjunktivní normální forma, dále jen DNF. V DNF je použita jedna ze tří t-norem, zde tedy t-norma minimum. Celá báze fuzzy pravidel je poté interpretována fuzzy relací dle rovnice 1.

. (1)

Druhým krokem výpočtu je aplikace defuzzifikace na výslednou fuzzy relaci. Společně s aproximací DNF je nejčastěji použita defuzzifikace center of gravity (COG). Výsledná hodnota je získána jako těžiště fuzzy množiny. Pro výpočet defuzzifikace COG je použita rovnice 2.

(2)

Rovnice pro tyto výpočty jsou obecně dostupné na internetu, avšak pro zajištění jejich správnosti byly ověřeny také z jiných zdrojů [[4]](#Novak). Zde jsou rovnice také více vysvětleny pro neznalé čtenáře.

Výsledkem defuzzifikace je hledaná hodnota, spočítána na základě vstupních hodnot.

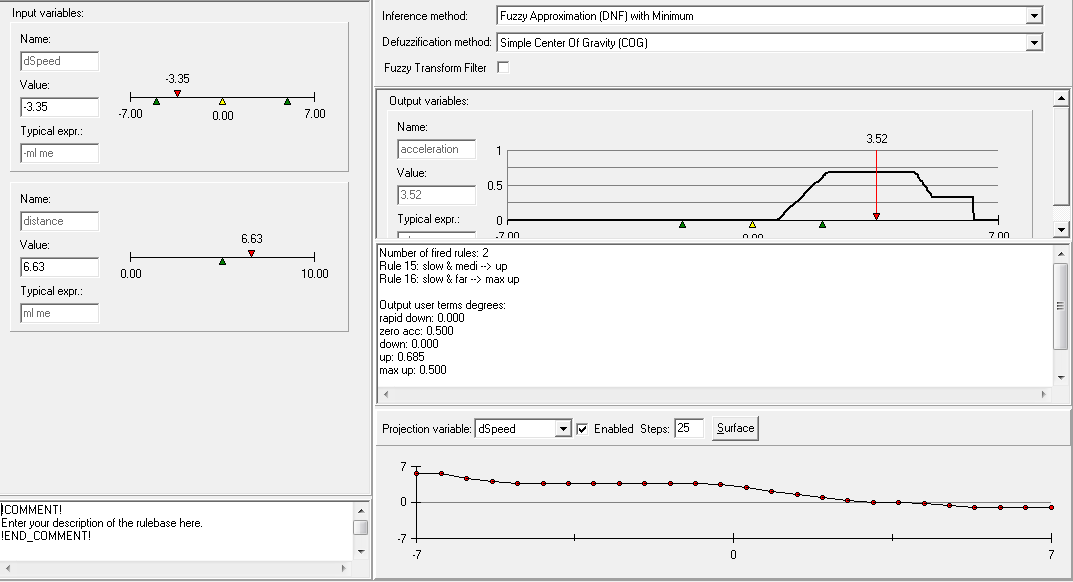
* 1. Implementace v aplikaci

Fuzzy logika v aplikaci je využita pro dvě základní části, a to jsou jízda vozidel za sebou a rozhodování, zda dát přednost v jízdě či nikoliv.

K vytvoření dané logiky byl použit nástroj LFLC, který nabízí mnoho možností při práci s fuzzy logikou.

* + 1. LFLController

Linguistic Fuzzy Logic Controller je specializovaný nástroj založen na teorii fuzzy logiky. Slouží pro práci s fuzzy logikou, modelování a testování pravidel.



4 Testování fuzzy logiky

Nástroj byl vytvořen na Ostravské univerzitě, v institutu pro výzkum a aplikaci fuzzy modelování – IRAFM [[4]](#IRAFM).

* + 1. Použití dat z LFLC v aplikaci

Pro modelování fuzzy logiky, jak je výše zmíněno, byla použita aplikace LFLC. Vytvořený model lze uložit do souboru s koncovkou \*.rb, který je možno zobrazit v klasických textových editorech, díky čemuž jej lze jednoduše použít i v jiných aplikacích.

Data uložená v tomto souboru určují všechny parametry modelované logiky. Pro vlastní použití jsou ovšem nejdůležitější data, která určují vstupní a výstupní proměnné a také pravidla, která tyto proměnné používají.

Formát uložení proměnných v souboru lze vidět na obrázku č. 4, seznam pravidel dále na obrázku č. 5.



5 Příklad struktury proměnné



6 Příklad fuzzy pravidel

O načtení dat do aplikace se stará třída RuleBaseReader, kde jsou data načítány jednoduchým procházením řádků souboru. Díky neměnné struktuře dat je poté podle klíčových slov rozpoznáno, o které parametry modelu se jedná.

Jakmile jsou data načtena v aplikaci, je vytvořen objekt třídy RuleBaseCalculator , který tvoří celou logiku, tedy výše zmíněnou aproximaci DNF a defuzzifikacie COG. Parametry této třídy jsou výstupní fuzzy proměnná, a fuzzy pravidla, jež byly již dříve načteny třídou RuleBaseReader.

Požadovaná hodnota se z vytvořeného objektu získá zavoláním metody „calculateByValues“ s parametrem všech vstupních hodnot. Jakmile je tato metoda použita, vypočte se výsledná hodnota výpočtem defuzzifikace COG

* + 1. Pohyb vozidel za sebou

Jak již název napovídá, jedna z vytvořených logik se stará o určování rychlosti jízdy jedoucího vozidla, které jede za vozidlem druhým. Vozidlo přizpůsobuje svou rychlost podle dané situace, pokud hrozí kolize vozidlo začne brzdit, naopak pokud je cesta volná, může jet rychlostí stanovenou parametry daného vozidla.

Rychlost vozidla je měněna zrychlením, což je výsledek výpočtu. Zrychlení může být kladné, při kterém vozidlo zrychluje, až záporné, kdy naopak zpomaluje. Pokud je zrychlení rovno nule, vozidlo rychlost nemění. Vstupními hodnotami jsou vzdálenost od druhého vozidla a rozdíl jejich rychlostí. Na obrázku 7 lze vidět vstupní fuzzy proměnnou vzdálenost, a její rozdělení na fuzzy výrazy, což je rozdělení na vzdálenosti „to close“, „close“, „medi“, „far“. Tato fuzzy proměnná pro další použití byla označena jako distance.



7 Vstupní proměnná vzdálenost

Stejně jako předchozí, i druhá vstupní fuzzy proměnná určující rychlost je rozdělena na fuzzy výrazy rozdělující rychlost. Pro tuto fuzzy proměnnou bylo zvoleno označení dSpeed. Fuzzy výrazy použity u této fuzzy proměnné jsou „very slow“, „slow“, „equal“, „fast“, „very fast“. Samotné rozdělení lze vidět na obrázku 8.



Vstupní proměnná rychlost

Výstupní fuzzy proměnná, určující výsledné zrychlení, stejně jako vstupní fuzzy proměnné, je rozdělena na několik fuzzy výrazů. Na obrázku 9 je znázorněno toto rozdělení a použité výrazy, což jsou „rapid down“, „zero acc“, „down“, „up“, „max up“, „light down“. Pro tuto fuzzy proměnnou bylo použito označení acceleration.



9 Výstupní fuzzy proměnná zrychlení

Tento způsob určení rychlosti je také použit při zastavení před křižovatkou, pouze s úpravou, kde místo vozidla jedoucího vepředu je „pevný bod“, který nemá žádnou rychlost.

V tabulce 1 lze vidět výsledné fuzzy pravidla pro jízdu vozidel za sebou, zahrnující obě vstupní fuzzy proměnné, dspeed a distance. Výstupem těchto pravidel je fuzzy proměnná acceleration.



Pravidla pro jízdu za sebou

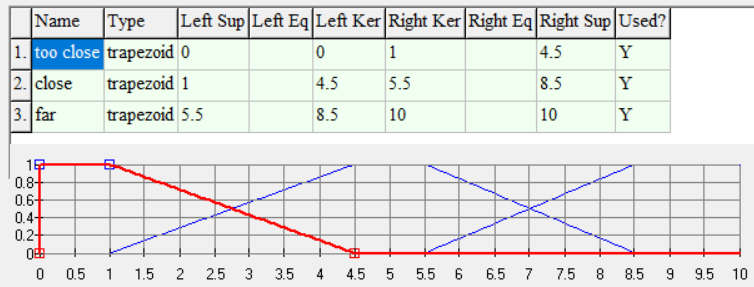
* + 1. Rozhodnutí udělení přednosti

Druhá vytvořená logika zajišťuje rozhodování, zda při jízdě přes křižovatku vozidlo má zastavit, nebo stihne křižovatkou projet, aniž by došlo ke kolizi.

Ve srovnání s předchozí logikou je tato složitější, neboť pracuje se čtyřmi vstupními proměnnými. Parametry jsou podobné, tedy vzdálenost od křižovatky a rychlost vozidla, avšak dvojnásobné, neboť musí být zvlášť pro vozidlo které je na hlavní silnici i pro vozidlo jedoucí po silnici vedlejší. Výsledkem je naopak pouze hodnota 0 nebo 1, tedy zda jet, nebo ne.

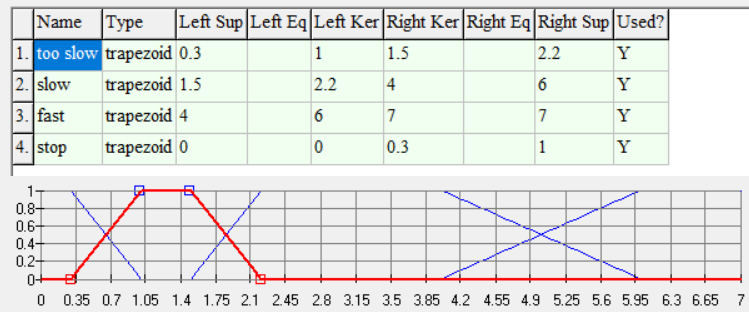
Tato fuzzy logika je řízena parametry dvou vozidel. Pro práci byly zvoleny označení vozidel vozidlo A a B, při čemž vozidlo B je aktuální čili jedoucí po vedlejší silnici, pro které se určuje, zda zpomalí, či projede. Vozidlo A je tedy vozidlo jedoucí na hlavní silnici.

První použitou fuzzy proměnnou zvolenou v této logice je distanceB, tedy vzdálenost vozidla B od křižovatky. Proměnná je rozdělena na tři fuzzy výrazy, což jsou „to close“, „close“ a „far“. Přesné hodnoty této fuzzy proměnné lze vidět na obrázku 10.



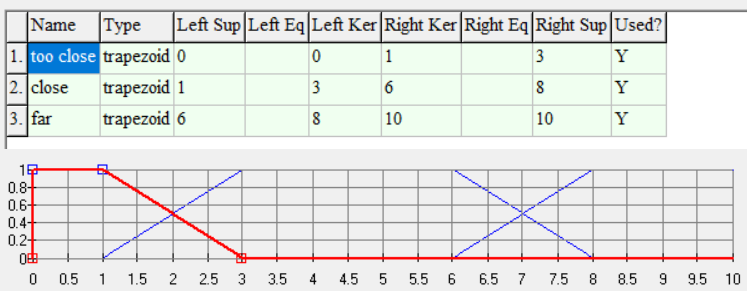
Vstupní fuzzy proměnné distanceB

Druhou fuzzy proměnnou, která odkazuje na vozidlo B, je proměnná speedB. Je rozdělena na čtyři fuzzy výrazy, což jsou „too slow“, „slow“, „fast“, „stop“. Hodnoty zvolené pro fuzzy proměnnou speedB lze vidět na obrázku 11. Této fuzzy proměnné se přiřazuje aktuální rychlost vozidla B.



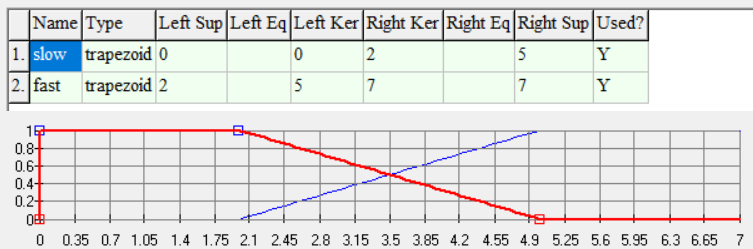
Vstupní fuzzy proměnné speed B

Další dvě fuzzy proměnné se zabývají hodnotami druhého vozidla A. Fuzzy proměnné distanceA, která určuje taktéž vzdálenost od křižovatky, je rozdělena na fuzzy výrazy „too close“, „close“, „far“. Pojmenování fuzzy výrazů je stejné jako u fuzzy proměnné distanceA, avšak, jak lze vidět na obrázku 12, rozdělení hodnot jednotlivých výrazů je odlišné.



Vstupní fuzzy proměnné distanceA

Poslední vstupní fuzzy proměnnou použitou v této logice je speedA, což je rychlost vozidla A. Zde byly zvoleny pouze dva fuzzy výrazy určující rychlost, což jsou „slow“ a „fast“. Na obrázku 13 lze vidět, že je toto rozdělení celkem jednoduché, avšak ve výsledku účinné.



Vstupní fuzzy proměnné speedA

Výstupní fuzzy proměnnou, jak již bylo na začátku této podkapitoly řečeno, je rozhodnutí, zda vozidlo A pojede, či zastaví. Pro toto rozhodnutí, jak lze vidět na obrázku 14, byly použity dva fuzzy výrazy, což jsou „stop“ a „run“. V samotné aplikaci se poté jednoduše rozhoduje, zda je výstupní hodnota větší či menší než 1.



14 Výstup určení přednosti

Následující tabulka obsahuje fuzzy pravidla vytvořená pro danou logiku. Vstupními hodnotami jsou všechny výše popsané proměnné, tedy rychlost a vzdálenost od křižovatky pro vozidlo jedoucí po vedlejší i hlavní silnici



Pravidla pro určení přednosti

1. představení vytvořeného řešení

V následující kapitole je představena samotná aplikace. Obsahuje popis základních funkcí aplikace z hlediska jejich využití, zda byly splněny všechny požadavky a stručnou architekturu aplikace.

* 1. Režimy aplikace

Aplikace je vytvořena pro fungování ve třech režimech použití.

Nejsložitějším režimem je režim návrhu, ve kterém uživatel může vytvořit vlastní silniční situaci. Funkce zde používané jsou popsány v další bodech funkcí.

Druhým režimem, pro který je původně aplikace tvořena, je režim testování uživatele. V tomto režimu si uživatel zvolí předem vytvořenou silniční situaci, na které poté ovládá vlastní vozidlo. Vozidlo jede po své trase, uživatel může ovládat pouze rychlost. Ovládání rychlosti bohatě stačí, neboť donutí uživatele reagovat podle aktuální situace, tedy zda hrozí srážka s jiným vozidlem či podle barvy na semaforech.

Třetím režimem aplikace je spořič obrazovky. Aplikace se spustí v režimu full-screen, tedy přes celou obrazovku. Z předem vytvořených situací se vybere jedna, která se spustí a simulovaná doprava běží bez možnosti jakéhokoliv zásahu uživatele.

* 1. Uživatelské rozhraní
     1. Návrh
     2. Testování
  2. Funkční prvky aplikace

Jsou zde popsány jednotlivé vlastnosti z funkčního hlediska, tedy jak je lze využít.

* + 1. Vložení pozadí

Vložení pozadí by mělo být jedním z prvních kroků při tvorbě návrhu situace, neboť nejlépe podle obrázku lze navrhnout například reálnou křižovatku.

Jako pozadí lze vybrat pomocí průzkumníku obrázek uložený v počítači. Po vložení lze obrázek posouvat, otáčet či měnit jeho velikost. Jakmile je uživatel spokojen s pozadím, potvrdí jej a může dále pracovat na návrhu. Pozadí lze případně znovu změnit či upravit.

* + 1. Možnosti návrhu

Vzhled aplikace je rozložen do několika skupin menu. Menu simulace je zobrazena stále, zde lze spustit či zastavit generování vozidel či vypnout a zapnout semafory. Dále je zde možnost nastavit vytížení silnic.

Kromě nabídky simulace lze přepínat mezi třemi režimy. V režimu „silnic“ lze manipulovat se samotnými silnicemi a nastavovat přednosti v jízdě. Režim „semafory“ slouží ke vložení semaforů, nastavení jejich barvy a tvorbě „časové linie“. Tato časová linie tvoří hlavní část systému semaforů, neboť je zde nastavena posloupnost přepínání semaforů na určitou barvu v určitém čase. Při spuštění simulace semafory mění barvu podle této posloupnosti.

S celým návrhem lze manipulovat. K tomuto slouží režim „mapa“, kde nelze nic jiného než celý návrh posouvat a přiblížit či oddálit.

* + 1. Tvorba silnic

Silnice lze tvořit kliknutím na „kreslící plátno“. Při prvním kliknutí se vytvoří spojovací bod, ze kterého se při dalším kliknutí vytvoří křivka. Ke spojovacím bodům lze přidávat nové křivky či připojit již existující. Kliknutím pravým tlačítkem na tento bod se zobrazí nabídka pro odstranění bodu nebo rozpojení křivek.

Silnice jsou tvořeny kubickými Bézierovými křivkami, které mají čtyři kontrolní body. Těmito body lze jednotlivé části silnic upravovat dle libosti.

Aplikace umožňuje vytvořit trasy pro automobily nebo tramvaje. Mezi typy silnice lze přepínat v menu nabídce.

* + 1. Určení předností

Přednosti v jízdě lze vytvořit kliknutím na určitý úsek silnice. První označený úsek je úsek, před kterým vozidlo zastaví, pokud hrozí srážka s vozidlem, které míří k úseku druhému.

Jakmile je označen úsek silnice, lze vybrat mezi typem přednosti. Typ „CP“ – check-point určuje přednost v jízdě, typ „WP“ – watch-point naopak slouží ke kontrole, zda je daný úsek cesty volný. Ke každé z předností lze navíc nastavit vzdálenost (v úsecích), o jakou vozidlo zastaví dříve před označeným úsekem.

* + 1. Nastavení semaforů

Semafory lze vkládat jednoduchým kliknutím, poté je možno je přesouvat, případně odstranit. Jak již bylo dříve zmíněno, semafory se tvoří ve vlastním režimu.

Každému semaforu může uživatel změnit počáteční barvu.

K jednoduchému nastavení přepínání semaforů je vytvořena časová linie, ve které jsou poté skupiny pro přepínání v určitých časech. Do těchto skupin jsou přidány semafory, jež se mají aktivovat v daný čas a přepnou se na zvolenou barvu.

Jakmile se postupně aktivují všechny časové skupiny, semafory se přepnou na jejich počáteční stavy a tato posloupnost se znovu opakuje.

* + 1. Nastavení vytížení silnic

V hlavním menu se nachází dva posuvníky, kterými je možno nastavit vytížení silnic. Jeden posuvník je určen pro vozidla, druhý pro tramvaje. Vytížení je určeno frekvencí generování vozidel, tedy počtem za minutu.

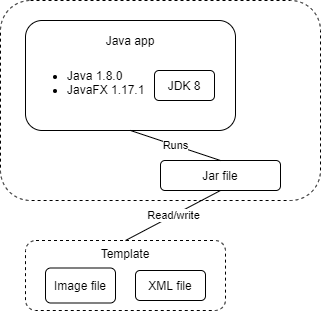
Vytížení silnic je možno nastavit globálně, tedy pro všechny silnice najednou, nebo pro určitou silnici. Pro nastavení vytížení určité cesty stačí označit počáteční křivku cesty, na které se budou objevovat vozidla.

* + 1. Uložení návrhu

Vytvořený návrh uživatel může uložit do souboru typu XML. V případě, že je vybráno pozadí, se na zvolené místo také uloží obrázek se stejným názvem. Uložený návrh lze opětovně nahrát do aplikace a dále jej upravovat.

* 1. Architektura aplikace

Základní architektura aplikace, společně s verzemi komponent, viz obrázek 15. Aplikace je vytvořena v jazyce Java, využívající rozšíření o grafické rozhraní JavaFX. Spouštěcí soubor je typu JAR. Pro uložení návrhu jsou využity soubory typu XML k zápisu informací a soubory ve formátu obrázku k využití pozadí.



Architektura aplikace

* + 1. Přínosy práce

Aplikace umožňuje otestování znalostí uživatele v silničních pravidlech. Uživatel se zde může připravit na nastání možných situací a vyzkouší zde své reakce.

Díky přidané fuzzy logice se vozidla pohybují reálně a jejich pohyby jsou plynulé.

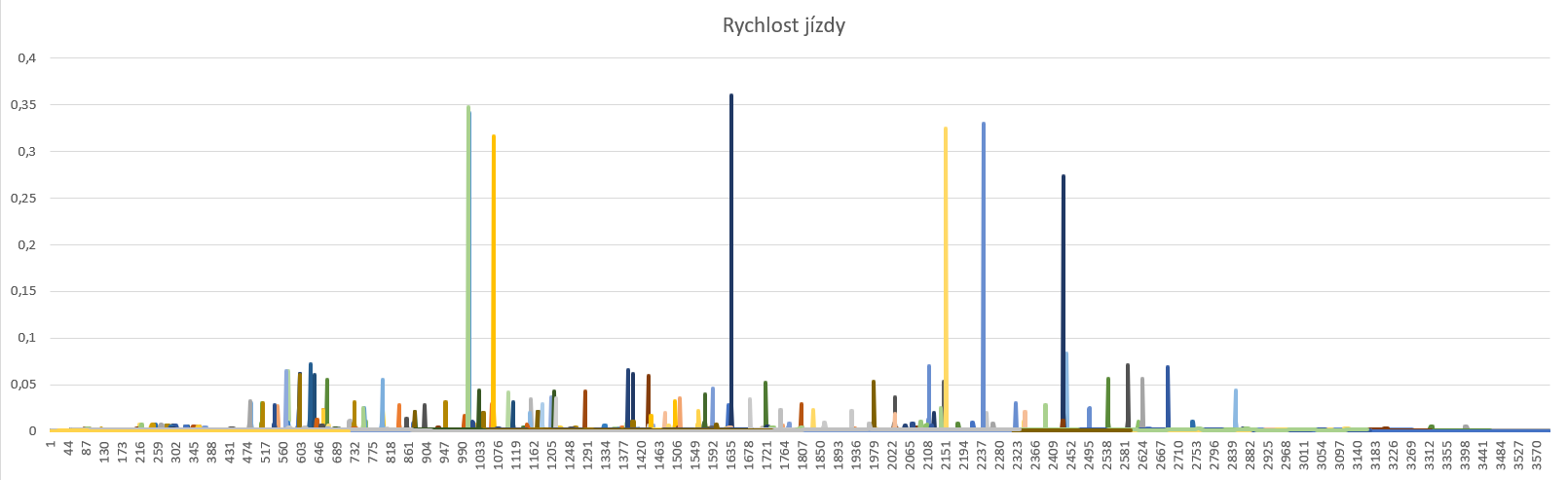
Aplikace také může sloužit pro tvorbu nových, ještě neexistujících silničních situací, křižovatek, či kruhových objezdů, nebo případně pouze systému semaforů.

* 1. Měření rozdílů

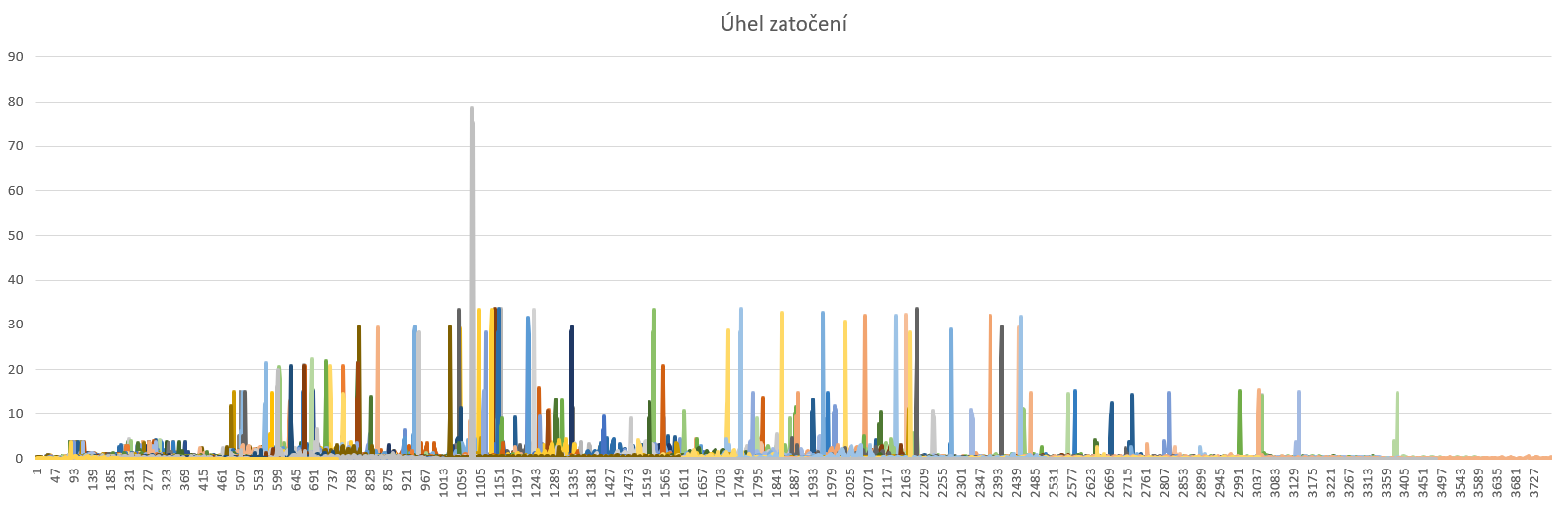
Jak již bylo dříve řečeno, tato práce rozšiřuje bakalářskou práci o fuzzy logiku. Pro porovnání těchto dvou prací bylo provedeno měření ve vytvořených aplikacích.

Měření bylo zaměřeno na plynulý pohyb vozidel, tedy na zrychlení, zpomalení a zatáčení.

* + 1. Pohyb



* + 1. Změna směru



* 1. Splnění požadavků
  2. Fuzzy logika v aplikaci
  3. Problémy při vývoji aplikace

1. metodika tvorby práce

V metodice práce je popsán postup při vývoji aplikace a tvorbě textu práce. Je zde zaznamenán způsob komunikace s vedoucím práce, použité nástroje, vize aplikace a způsob, jakým byla řízena implementace požadavků.

* 1. Vize

Hlavní vizí práce je rozšíření původní aplikace vytvořené v rámci bakalářské práci o fuzzy logiku. Fuzzy logika slouží k řízení rychlosti jedoucích vozidel a k určení, zda vozidlo na křižovatce zastaví, či ne.

V aplikaci je uživateli umožněno navrhnout funkční silniční situaci, buďto reálnou, nebo vymyšlenou a nastavit pravidla provozu tak, aby provoz probíhal plynule bez srážek vozidel. O tuto funkci se stará především dříve zmíněná fuzzy logika implementovaná do aplikace.

* 1. Způsob práce

Během práce probíhaly ze začátku jednou týdně konzultace s vedoucím práce. Při těchto konzultacích se řešily otázky ohledně nových požadavků, možnosti řešení implementace funkcí a jejich úpravy. Na každé konzultaci se také zhodnotily přírůstky na práci od poslední schůzky.

Problém ovšem nastal po příchodu vládních nařízení kvůli koronaviru COVID-19. Jelikož již konzultace nebyly možné, zbyla pouze elektronická komunikace, avšak tímto způsobem bylo velmi těžké pracovat na aplikaci. Vývoj aplikace byl tedy na delší dobu zastaven a čekalo se na znovu-povolení osobních konzultací, což ale nenastalo. Poté byla komunikace řešena prostřednictvím Microsoft Teams, avšak ani tímto způsobem to nebylo zcela ideální, bylo zde mnoho nedorozumění, a proto práce není zcela dokonalá.

* 1. Způsob zápisu požadavků

Při práci byla použita webová aplikace Trello, která pomáhá řídit práci na požadavcích projektu.

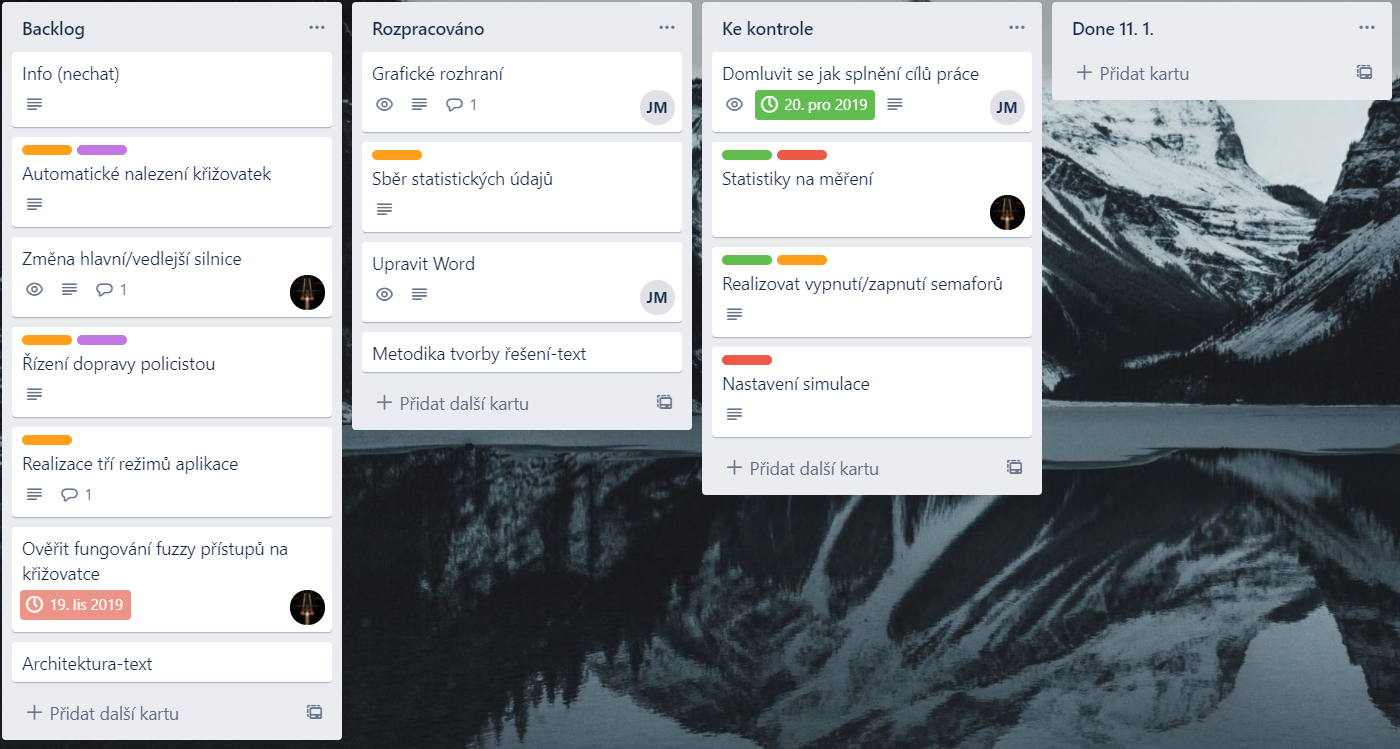
Do sloupečku „backlog“ byly přidávány po konzultacích s vedoucím práce nové požadavky, kterým byla nastavena priorita implementace. V tomto sloupečku se nacházely požadavky, které čekaly na implementaci.

Z backlogu byly poté požadavky po jednom přesouvány do sloupečku „rozpracováno“, což znamená, že se na daném požadavku pracuje.

Po zhotovení daného požadavku byl v Trellu přesunut do sloupečku „ke kontrole“. Při další konzultaci byly všechny požadavky z tohoto seznamu zkontrolovány vedoucím práce. Pokud měl vedoucí práce připomínky, byl požadavek přesunut zpátky do sloupečku rozpracováno. Požadavky, které vedoucí práce schválil, byly přesunuty do sloupečku „done“ a samotný sloupeček byl poté uložen a vytvořen nový, prázdný.

Následně se pokračovalo s dalšími požadavky z backlogu.

Na obrázku 15 lze vidět nástěnku v Trellu a příklad rozložení požadavků.



Seznam požadavků v Trellu

* 1. Správa verzí aplikace

Ke správě verzí aplikace byl použit systém Git. Tento systém nabízí možnost vytvoření repositáře s vyvíjenou aplikací. Repositář je potom možno dělit na větve, což umožňuje mít několik rozdílných verzí aplikace.

Při implementaci jednotlivých funkcí aplikace byla vždy vytvořena nová větev nazvána podle dané funkce. Během vývoje dané funkce se všechny změny ukládaly do této větve. Jakmile byla funkce hotova a schválena, tato větev byla připojena k hlavní větvi repositáře.

Projekt se nachází na soukromém repositáři, tudíž k němu kromě určených spolupracovníků nikdo nemá přístup.

* 1. Použité nástroje

Během práce na vývoji aplikace a tvorbě textu byly použity různé softwarové nástroje pro usnadnění práce. Nástroje slouží především k usnadnění práce, avšak také ke komunikaci.

* + 1. NetBeans

K vývoji aplikaci bylo použito vývojové prostředí NetBeans IDE 8.2 od společnosti Oracle Corporation. Prioritním jazykem pro práci v tomto prostředí je Java, avšak podporuje i další programovací jazyky. Prostředí NetBeans nabízí mnoho frameworků, kterými lze usnadnit práci na aplikaci. V rámci tohoto prostředí byl pro vývoj aplikace použit jazyk JavaFX.

Samotné prostředí a mnoho komponent se nachází na oficiálních webových stránkách <www.netbeans.apache.org>.

Prostřednictvím tohoto prostředí také probíhalo ukládání aplikace do systému Git.

* + 1. GitHub

Verzování aplikace je vzhledem k počtu funkcí velmi obtížné, proto existují systému typu Git. Systém slouží pro správu verzí projektu. Nejvíce je uplatněn, pokud na projektu pracuje více vývojářů zároveň. Jednotlivé verze je nutno poté dát nějak dohromady, aniž by se celá aplikace nerozbila.

Při práci na tomto projektu byla použita webová služba GitHub, nabízející ukládat bezplatně i soukromé repositáře.

* + 1. Trello

Trello je webová aplikace sloužící ke správě projektů. Aplikace, kterou vytvořila firma Fog Creek Software, je dostupná na webové stránce <www.trello.com>. Trello používá Kanban, což je způsob zastoupení projektu nástěnkou, které obsahují seznamy úkolů.

* + 1. MS Teams

Jakmile začaly vládní nařízení kvůli pandemii coronaviru, muselo se přejít na elektronickou komunikaci. Pro komunikaci byla zvolena služba Microsoft Teams, která umožňuje nejen textovou komunikaci, ale také videohovory s možností sdílení obrazovky.

ZÁVĚR

RESUMÉ

SUMMARY

SEZNAM POUŽITÉ LITERATURY

1. **Pak**, **HakChol**. *Simulátor dopravní sítě.* [online]. Praha, 2010. Bakalářská práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra softwarového inženýrství, 2010-05-28. [cit. 2018-01-28]. Dostupné z: <<https://is.cuni.cz/webapps/zzp/detail/49977>/>.
2. **Gregor, Ivan**. *Simulace automobilového provozu*. [online]. Praha, 2006. Bakalářská práce. Univerzita Karlova v Praze, Matematicko-fyzikální fakulta, Katedra teoretické informatiky a matematické logiky. 2006-09-18. [cit. 2018-01-28]. Dostupné z: <<https://is.cuni.cz/webapps/zzp/detail/44070/>>.
3. **ShadowTree.** *Traffic Lanes 2* [online]. Dostupné z: <<http://shadowtree-software.se/trafficlanes2.html>>
4. **Mikulík, Jan**. *Tvorba nástroje pro interaktivní řešení dopravních situací.* Ostrava, 2018. Bakalářská práce.
5. **NOVÁK, Vilém**. *Základy fuzzy modelování*. BEN, Praha 2000.
6. **Štěpnička, M**.: *Matematické metody pro umělou inteligenci*. 2013.
7. **Janošek, M., Farana, R**.: *Fuzzy modelování a řízení*. Ostrava, 2014.
8. **Štěpnička, M., Vavříková, L**.: *Úvod do soft computingu*. Ostravská univerzita, 2010.
9. **Novák, V., Knybel, J**.: *Fuzzy modelování.* Studijní opora OU, Ostrava, 2005.

SEZNAM POUŽITÝCH SYMBOLŮ

|  |  |  |
| --- | --- | --- |
| ABC |  | Význam první zkratky. |
| B |  | Význam druhé zkratky. |
| C |  | Význam třetí zkratky. |
|  |  |  |

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH