# Modelovanie a riadenie systému automaticky navádzaných vozidiel pre dopravu vo výrobných procesoch (Tímový projekt)

Štefan Krištofík, Filip Lörinc, Igor Sečanský,
Stanislav Panák

25. mája 2008

# Znenie zadania

Systém automaticky navádzaných vozidiel (automatic guided vehicles — $AGV$) je dôležitou súčasťou komplexne automatizovaných výrobných procesov. Bezkolízne a optimalizované plnenie dopravných úloh napomáha ku zefektívneniu výroby.

V rámci tímového projektu riešte nasledujúce úlohy:

- analyzujte a vyhodnoťte dopravné systémy používané vo výrobe z funkčného hľadiska
- naštudujte a urobte rozbor modelovacích prostriedkov, ako sú Petriho siete, časové okná a iné, z hľadiska ich vhodnosti pre triedu zónovo riadených dopravných prostriedkov
- pre vybranú triedu dopravných systémov spracujte metódu ich modelovania zvolenými prostriedkami
- navrhnite riadenie pre riešenú triedu dopravných systémov
- pre modelovanie a riadenie riešenej triedy dopravných systémov vypracujte a odskúšajte programový systém na modelovanie a riadenie

# Účel a rozsah dokumentu

Predkladaný dokument slúži ako dokumentácia ku:

- analýze problému
- špecifikácií požiadaviek riešenia
- hrubému návrhu riešenia

programového systému na modelovanie a riadenie systému AGV pre dopravu vo výrobných procesoch vytváraného v rámci predmetu *Tímový projekt I.*

V rámci predmetu *Tímový projekt II* sú tu dokumentované moduly pre simuláciu systému, modul pre vytvorenie grafu a modul na realizáciu algoritmu *free windows*. Každý modul bol zdokumentovaný členom tímu, ktorý sa podieľal na jeho implementácii:

- Modul pre simuláciu: (Stanislav Panák)
- Modul pre tvorbu grafu: (Igor Sečanský)
- Modul pre realizáciu algoritmu free windows: (Štefan Krištofík)

## Použité skratky

**AGV** – Automated Guided Vehicles

**FMS** – Flexible manufacturing system

**CNC** – Computer controlled machines

**PS** – Petriho siete

# Obsah

# Zoznam obrázkov

# Časť I

# Zimný semester

# Kapitola 1

# Analýza problému

## 1.1 Úvod

Modelovanie výrobných procesov, pracovných postupov, komunikácie v sieti, dopravnej situácie a mnoho ďalších vecí sa dá realizovať spoločnými prostriedkami, ktoré zvyčajne vyúsťujú do reprezentácie pomocou rôznych orientovaných grafov, ktoré sa navyše môžu dynamicky meniť, či už v závislosti od času, alebo od výskytu rozličných udalostí. Na vizualizáciu sú vhodné napríklad grafy udalostí, s ktorými sme sa mali možnosť stretnúť na predmete *modelovanie a simulácia* v podobe nástroja *EGRET*, alebo Petriho siete, ktoré boli spomenuté v záverečných prednáškach zo *špecifikačných a opisných jazykov*. Petriho siete, okrem toho že problém vizualizujú, slúžia napríklad aj na dokazovanie rôznych zacyklení, uviaznutí, alebo na dokázanie korektnosti fungovania systému. Táto problematika je však veľmi náročná. Uplatnenie tu nachádzajú aj grafové algoritmy. Napríklad na obrázku 1.1 vidno špičkový automaticky navádzaný laboratórny vozík. Vozíky sú použiteľné v laboratóriách alebo v nemocniciach na rôznych oddeleniach (biochémia, mikrobiológia, imunológia, hematológia, farmácia, rádiológia, virológia, histológia, ...), bezpečne prechádzajú popri personále, dokážu manévrovať v úzkych priestoroch, samé dokážu otvárať a zatvárať dvere (obrázok 1.2), pohybujú sa po poschodiach využívajúc existujúce výťahy, sú v prevádzke 24 hodín den-

Obrázok 1.1: AGV

Obrázok 1.2: Špičkové laboratórne vozíky

ne, 7 dní v týždni a majú samodobíjanie batérií. Ľahko sa programujú a jednoducho nasadzujú.

Obrázok 1.3: Vozíky v tlačiarenskom priemysle

Ďalšími aplikáciami AGV (Automated Guided Vehicles) sú napríklad vysokozdvižné vozíky používané v tlačiarenskom priemysle na prepravu papierových kotúčov (obrázok 1.3) zo skladiskových priestorov ku tlačiarenskému stroju (vozík papierový kotúč dopraví, aj nasadí), alebo automatické vysokozdvižné vozíky požívané vo výrobných halách prípadne skladoch na prekladanie paliet, zabaleného tovaru a pod. (obrázok 1.4). Vozíky sa dokážu orientovať v priestore (použitie senzorov, resp. kamerového systému s rozpoznávaním), vyhýbajú sa prekážkam, sú schopné identifikovať prepravovanú vec, prípadne vzájomne komunikujú (riešenie uviaznutí a zacyklení). Už zostrojenie a naprogramovanie reálneho modelu (obrázok 1.5) je veľmi náročné, preto sa obmedzíme na jednoduchý počítačový model, v ktorom budú vozidlá, resp. prepravované objekty reprezentované jednoduchými geometrickými

Obrázok 1.4: Vysokozdvižné vozíky vo výrobných halách

objektami (krúžky, guľôčky, prípadne ich kompozícia propomínajúca autíčko a pod.). Trajektórie dráh, po ktorých budú vozidlá chodiť, budú reprezentované pomocou orientovaného grafu, ktorého vrcholy budú zastávky pre vozidlá (prekladiská, parkovacie miesta, čerpacie stanice a pod.) Program bude zobrazovať aktuálnu polohu vozidiel v čase a bude predstavovať akýsi velín, prípadne bude obdobou navigátorského systému. Ďalšou jeho fukciou bude optimalizácia. Bude sa jednať o nachádzanie najkratších ciest a podobne.
Na nižšie uvedených linkách sa nachádzajú krátke videá, na ktorých možno vidieť fungovanie automaticky navádzaných vozidiel, o ktorých sme hovorili:

**Laboratórne vozíky** :

`http://www.youtube.com/watch?v=mgp1cvkDPTI`

**Vozíky v tlačiarenskom priemysle** :

`http://www.youtube.com/watch?v=iywjAVshnWg`

**Vysokozdvižné vozíky vo výrobnej hale** :

Obrázok 1.5: Model autíčok a robotického ramena

`http://www.youtube.com/watch?v=hLwI-pHUvRQ`

**Model s robotickým ramenom** :

`http://www.youtube.com/watch?v=KnJljQO8LHo`

## 1.2 Flexibilné výrobné systémy

Flexibilný výrobný systém – *Flexible Manufacturing System* (FMS) – ide vlastne o viac intuitívny pojem ako o niečo hmatateľné. FMS je v podstate myšlienka, že rýchle je lepšie a používa stroje na výrobu produktov. Namiesto využívania ľudských síl na vykonávanie opakujúcich sa úloh je použitý stroj, ktorý danú úlohu vykonáva 24 hodín denne. FMS využíva tzv. *computer numerical controlled machines* (CNC), čo sú počítačom riadené stroje, na vytvorenie tzv. *pracovných buniek*. Každá takáto bunka potom vykonáva špecifickú úlohu, čím napomáha svojím dielom k vyrobeniu produktu. FMS sú rýchle a efektívne, na druhej strane však nie sú veľmi lacné a je potrebné mať veľa drahých strojov pre začatie výroby. Z tohto dôvodu veľa spoločností volí radšej cestu čiastočnej náhrady výroby systémom FMS. V takomto prípade hovoríme o vytváraní tzv. flexibilných výrobných buniek. Táto bunka(y) vyrobí časť finálneho produktu, zatiaľ čo ostatné časti sú vyrobené inými metódami. Často nastáva prípad, že niekoľko vozidiel typu AGV (vysvetlené nižšie) je vyhradených za účelom prepojenia jednotlivých výrobných buniek [4].

## 1.3 Automaticky navádzané vozidlá

*Automatic Guided Vehicle* (AGV) - automaticky navádzané vozidlo je mobilné robotické vozidlo ovládané počítačom používané v priemysle, ktoré slúži na premiestňovanie rôznych predmetov z jedného miesta na iné vo výrobných halách alebo v skladoch a pod. Takéto vozidlá sú vybavené zvyčajne elektromagnetickými alebo optickými zariadeniami, ktoré umožňujú ich automatické navádzanie. Takto vybavené vozidlá sú potom schopné pohybovať sa po predurčených trasách, zastavovať na určených miestach, a plniť ďalšie rozličné funkcie, ktoré sú na ne kladené počas prevádzky [4].

AGV pomáhajú znižovať náklady na výrobu a zvyšovať efektivitu výrobného systému. Sú napríklad schopné ťahať predmety za sebou v malých vlečkách, na ktoré sa dokážu samé autonómne zahákovať. Na týchto vlečkách je možné prenášať surový materiál do linky, kde je už pripravený na výrobu. AGV dokážu taktiež skladovať objekty na vrstvách alebo ich umiestňovať na poličky. Niektoré AGV dokonca používajú technológiu ako vysokozdvižné vozíky, kde sa pod predmety podsunú kovové vidlice, ktoré sa potom zdvíhajú do príslušnej výšky. Tento prístup nájde uplatnenie najmä pri práci v sklade. AGV môžu slúžiť aj na prepravu medicínskeho materiálu v urgentných prípadoch. Ich použitie je široké [3].

AGV sa stávajú stále populárnejšími v automaticky riadených systémoch, vo flexibilných výrobných systémoch a dokonca v systémoch s prepravou kontajnerov, napríklad v prístavoch. V predošlých desaťročiach sa technológii AGV venovalo množstvo výskumov a štúdií a bol zaznamenaný významný progres. Napríklad technológii plánovania (*scheduling*) a smerovania (*routing*) AGV sa v poslednom období venovalo viacero autorov. Navrhnutých bolo niekoľko algoritmov riešiacich tieto problémy, avšak väčšina existujúcich riešení je vhodná len pre systémy s malým počtom AGV. V súčasných systémoch, kde počet vozidiel AGV sa môže pohybovať až okolo sto a viac, je nutné zaviesť efektívne algoritmy plánovania a smerovania vozidiel na riešenie zvýšeného počtu prípadov, kedy dochádza k súpereniu pri využívaní zdrojov medzi AGV (napríklad konflikty typu keď viacero vozidiel chce využiť tú istú cestu).

Takisto je potrebné zvyšovať celkový výkon takýchto systémov, čo môže byť dosiahnuté len inovatívnymi myšlienkami, postupmi a stratégiami ako napríklad paralelné spracovanie a podobné inovatívne prístupy. Na tomto poli sú teda v súčasnosti možnosti otvorené pre budúce výskumy a štúdie [3].

AGV v systémoch FMS sú používané na premiestenie objektu z bodu A do bodu B. AGV sa orientuje vo výrobných priestoroch použitím senzorov. Použiť sa môžu dva typy senzorov: drôtové a bezdrôtové (napríklad navádzanie laserom, gyroskopické navádzanie, ...) [4]. Rozoznávame viaceré druhy

AGV:

- ťažné – najstarší typ, stále však používaný
- nakladacie – majú plošiny na nakladanie materiálu, môžu byť zdvíhané, znižované, sú rôzne typy
- paletizované – na prepravu materiálov na paletách
- vidlicové – zdvíhanie materiálov, ukladanie materiálov do poličiek
- nízkonákladové – na transport menších, ľahších častí, používajú sa v prostrediach s obmedzenými priestorovými možnosťami
- výrobné – nízkonákladové, používajú sa pri procesoch sériovej výroby pri linkách [4]

Ak FMS má viacero AGV, je nutné nejakým spôsobom zabezpečiť kontrolu dopravného toku, aby napríklad AGV navzájom do seba nenarážali a nevznikali podobné problémy. Na riešenie kontroly dopravy [4] existujú nasledovné metódy:

- zónová kontrola
- predvídavá kontrola
- kombinovaná kontrola

### 1.3.1 Zónová kontrola

Táto metóda kontroly dopravy v FMS je obľúbená vo väčšine prostredí nakoľko sa ľahko inštaluje a rozširuje. Zónová kontrola používa bezdrôtový vysielač na prenos signálu v nemennom priestore. Každé AGV má zariadenie na príjem signálu a jeho spätné vysielanie k vysielaču. Ak oblasť je voľná, AGV obdržia signál *clear* (voľný) a je im umožnený vstup do oblasti. Ak v oblasti je nejaké AGV, potom ostatné AGV, ktoré sa snažia do tejto oblasti vstúpiť, obdržia signál *stop* (zastaviť) a musia počkať. Po východe AGV z oblasti obdrží jedno z čakajúcich AGV signál *clear*. Inou možnosťou ako zaviesť zónovú kontrolu je vybaviť každé AGV svojím vlastným vysielačom a prijímačom. Potom takto vybavené vozidlo je schopné po vstupe do oblasti rozoslať ostatným vozidlám v blízkom okolí signál typu *do not enter* (nevstupovať) [4].

### 1.3.2 Predvídavá kontrola

Tento spôsob používa špeciálne senzory na predchádzanie kolíziám s inými AGV v oblasti. (zvukové, optické senzory, senzory fyzického kontaktu). Senzory fyzického kontaktu sú namontované na AGV ako záloha. Takéto riešenie je však náročné čo sa týka inštalácie aj práce s ním [4].

### 1.3.3 Kombinovaná kontrola

Ide o prípad, kedy je použitá kombinácia oboch vyššie spomenutých metód. AGV majú senzory predchádzania kolíziám aj senzory zónovej kontroly. Kombináciou týchto sa predíde kolíziám v každom prípade. Pri bežnej prevádzke sú použité senzory zónovej kontroly s využitím senzorov predchádzania kolízie ako poistka. Napríklad v prípade výpadku zónovej kontroly potom predvídavá kontrola predíde prípadným kolíziám [4].

## 1.4 Riadenie systému

Výroba s využitím AGV musí mať určitý stupeň kontroly nad týmito vozidlami. Toto je možné zabezpečiť troma metódami:

- panel lokátora – jednoduchý panel, kde vidno, kde sa AGV nachádza
- farebný displej CRT – v reálnom čase zobrazuje pozície všetkých AGV a ich stav (napätie batérie, ID, a pod.)
- centralizované logovanie – použije sa na uchovávanie histórie o pohybe všetkých AGV v systéme (kde boli, kadiaľ išli a podobne). Takto nazbierané údaje sa uchovávajú v nejakom centrálnom bode a je možné ich napríklad vytlačiť za účelom nejakej kontroly príp. inými účelmi [4].

## 1.5 Systémy s AGV

*Automated Guided Vehicle System* (AGVS) – systémy s automaticky navádzanými vozidlami sú také priemyselné systémy, v ktorých automaticky navádzané vozidlá zohrávajú dôležitú úlohu ako hlavné prostriedky (nástroje) na transport materiálov. AGVS môžeme definovať aj ako počítačom riadený systém, ktorý obsahuje navzájom nezávisle adresovateľné vozidlá bez vodiča (to sú AGV) pohybujúce sa po dopredu zadefinovanej sieti transportných ciest [3]. Presne tak, ako počítačový systém, aj systémy s automaticky navádzanými vozidlami sú typicky zložené z dvoch hlavných kooperujúcich

podsystémov - hardvér a softvér. Hardvér zahŕňa fyzické komponenty ako samotné vozidlá, cesty, senzory, navádzacie systémy, ovládacie prvky a pod. Softvér zahŕňa rôzne prístupy a algoritmy na systematické manažovanie hardvérových prostriedkov celého AGVS. Tieto prístupy a algoritmy napomáhajú harmonickému fungovaniu AGVS a k ich vysokej efektívnosti. V posledných desaťročiach však je pokrok v oblasti hardvérového vybavenia AGVS oveľa výraznejší ako pokrok v oblasti vývoja softvéru. Toto zaostávanie softvérového vybavenia začalo byť problémom až v posledných rokoch pri zvyšovaní počtu AGV v systémoch AGVS. Taktiež problémy plánovania a smerovania AGV, ktoré boli v minulosti považované za triviálne, sa stali dôležitými len v poslednom období [3].

## 1.6 Plánovanie AGV

Pod plánovaním (*scheduling*) AGV môžeme chápať činnosť, keď pošleme skupinu AGV vykonať sériu jednoduchých úloh typu napríklad zdvihnutie/položenie predmetu za cieľom dosiahnuť nejaký konkrétny cieľ pri daných obmedzeniach [3]. Príklady cieľov plánovania:

- ukončiť úlohu do určitej doby (deadline)
- ukončiť úlohu za minimálny možný čas
- minimálna prejdená vzdialenosť všetkých AGV
- minimálna doba nečinnosti (idle time) všetkých AGV
- plnenie úlohy (napr. produkcia výrobkov) s čo najnižším možným počtom AGV

Plánovanie zahŕňa aj nasledovné problémy:

- vehicle selection problem (vehicle dispatching problem) – z množiny momentálne voľných vozidiel vyberáme jedno, ktorému úlohu pridelíme
- task selection problem – z množiny úloh vyberieme jednu, ktorú budeme niektorým vozidlom riešiť

## 1.7 Smerovanie AGV

Pod smerovaním (*routing*) AGV rozumieme hľadanie optimálnej a uskutočniteľnej cesty pre každé AGV. Tento proces [3] zahŕňa tri aspekty:

1. Musí sa zistiť, či vôbec nejaká cesta medzi zdrojom a cieľom cesty existuje.

2. Cesta musí byť uskutočniteľná, to znamená že nesmie dôjsť ku upchaniu cesty ani ku konfliktom medzi vozidlami.

3. Cesta musí byť optimálna, alebo aspoň čiastočne optimálna (napr. minimalizovať čas nečinnosti vozidla).

## 1.8 Nežiadúce okolnosti

Napriek tomu, že problémom plánovania a smerovania AGV bolo venované množstvo výskumného úsilia, stále chýbajú prijateľné všeobecné riešenia a to kvôli rôznorodosti aplikácií. Navrhnuté postupy a algoritmy sú zvyčajne vhodné len pre špecifické prostredia, alebo sa venujú riešeniu problémov, ktoré sa objavujú len v osobitých prípadoch. Pri plánovaní a smerovaní AGV sa bežne stretávame s nasledovnými problémami (tzv. *fenoménmi*) [3]:

- *kolízie* – ak sa viacero AGV pokúša využiť ten istý segment cesty v tom istom čase. Príklad kolízie je na obr. 1.6 a).

- *zahltenie* – nastáva tam, kde počet príchodzích vozidiel je väčší ako počet požiadaviek. Miera zahltenia musí byť znížená alebo musí byť odstránené úplne, nakoľko znižuje celkovú efektívnosť systému a môže viesť k ďalším problémom. Príklad zahltenia je na obr. 1.6 b).

- *livelock* – špeciálny prípad, ak priorita premávky na jednej ceste na križovatke je neustále vyššia ako priorita premávky na inej ceste. Príklad takejto situácie je znázornený na obr. 1.6 c).

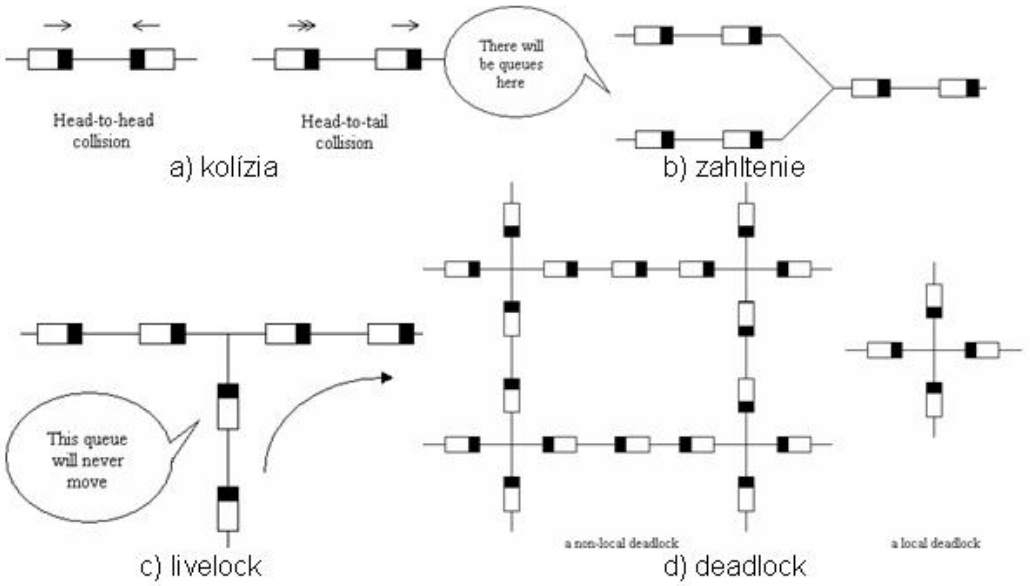
- *deadlock* – tento prípad nastane, ak viacero AGV čaká na uvoľnenie zdroja (ktoré nikdy nenastane), ktorý je okupovaný inými AGV. Dva príklady takejto situácie sú na obr. 1.6 d).

## 1.9 Smerovacie algoritmy

Existujúce práce v tejto oblasti môžeme rozdeliť do troch kategórií [3]:

- smerovacie algoritmy pre topológie so všeobecnými cestami

- optimalizácia ciest




Obrázok 1.6: Fenomény pri plánovaní a smerovaní AGV

- smerovacie algoritmy pre topológie so špecifickými cestami

Práce v prvej kategórii zvyčajne transformujú problém smerovania na teoretický problém z oblasti grafov a používajú prístupy ako *Dijkstrov algoritmus hľadania najkratšej cesty* na nájdenie optimálnych ciest pre AGV. Takéto algoritmy sú obyčajne veľmi náročné na výpočtový čas v prípade smerovania viacerých vozidiel. Práce v druhej kategórii sa zaoberajú použitím na mieru šitých optimalizačných techník ako napr. integerové programovanie na optimalizovanie sietí ciest namiesto hľadania efektívnejších smerovacích algoritmov. Pretože však v týchto prípadoch je výpočtová náročnosť veľmi vysoká, veľkosť optimalizovanej siete je obyčajne malá a umožňuje obsiahnuť zhruba do 20 AGV. Čo sa týka prác v tretej kategórii, tu sieť tratí má špecifickú topológiu, v ktorej sa nachádzajú napríklad viacnásobné slučky, preto na smerovanie AGV sú potrebné vysoko špecializované algoritmy [3].

Bližšie sa pozrieme len na prvú kategóriu smerovacích algoritmov, pretože pre triedu zónovo riadených vozidiel budeme používať jeden z algoritmov takéhoto typu.

## 1.10 Smerovacie algoritmy pre topológie so všeobecnými cestami

Algoritmy v tejto kategórii sa zameriavajú na nájdenie uskutočniteľných ciest pre AGV pričom do úvahy sa neberie topológia siete dráh. Snažia sa nájsť univerzálne riešenie smerovacieho problému. Základnou úvahou je poskytnúť bezkonfliktné časovo najkratšie možné riešenia smerovania pre vozidlá AGV [3].

### 1.10.1 Statické smerovacie algoritmy

Prvé výsledky (*Broadbent* 1985) v tejto oblasti využívali Dijkstrov algoritmus na nájdenie najkratšej cesty v grafe na vygenerovanie matice, ktorá popisovala okupáciu ciest vozidlami v čase. Potenciálne konflikty medzi vozidlami, typu *head-to-head*, *head-to-tail* a kolízie na križovatkách boli detegované porovnaním časov okupácie dráh. Head-to-head konflikty boli vyriešené nájdením inej najkratšej cesty pričom konfliktný segment sa vynechal. Konflikty na križovatke a head-to-tail boli vyriešené spomalením alebo pozastavením vozidla, aby mohlo skôr naplánované vozidlo prejsť inkriminovaný úsek ako prvé. Ďalším výrazným posunom bolo nahradiť doteraz používané jednosmerné cesty pri smerovaní AGV obojsmernými. Výhody takéhoto riešenia čo sa týka využitia vozidiel a produktivity celého systému sú zrejmé. *Egbelu* a *Tanchoco* v roku 1986 boli schopní takýmto spôsobom zmenšiť počet potrebných AGV v systéme pričom sa zároveň zvýšila produktivita. O takýchto systémoch ale platí, že môžu byť veľmi komplexné, pretože môže dôjsť k sporom o určité obojsmerné úseky. Smerovacie algoritmy musia byť schopné eliminovať potenciálne konflikty, kolízie aj deadlocky. Spomínaní Egbelu a Tanchoco ako prví v r. 1986 navrhli použitie obojsmerných ciest pri smerovaní AGV. V ich práci však nebol navrhnutý žiaden algoritmus na optimalizáciu smerovania po obojsmerných cestách. Prvým, kto predložil algoritmus na smerovanie AGV po obojsmerných cestách bol Daniels v r. 1988. Bol to PSP algoritmus (*partitioning shortest-path algorithm*). Realizovateľnosť a správnosť algoritmu bola v tejto práci matematicky dokázaná. Algoritmus bol schopný nájsť bezkonfliktnú časovo najkratšiu cestu pre novo pridané AGV bez toho, aby sa cesty ostatných vozidiel menili. Ak však bol niektorý segment rezervovaný niektorým vozidlom, potom novo príchodzie vozidlo tento úsek nemohlo vôbec využiť hoci tento úsek bol nedostupný len po určitú dobu (čiže kým to vozidlo tadiaľ prechádzalo). Spomínané problémy s Danielsovým algoritmom sa snažil vyriešiť *Huang* a v r. 1989 navrhol značkovací algoritmus pre smerovanie jediného AGV v systéme s obojsmer-

nými cestami. Algoritmus skonvertoval každú hranu (fyzickú) z originálnej siete na uzol s dvoma hranami spájajúcimi originálne 2 uzly v neskonvertovanej sieti. Takto je získaná skonvertovaná sieť a pridelia sa značky voľným časovým úsekom (oknám) definovaným pre každý uzol. Porovnávaním týchto značiek každého uzla bolo možné získať časovo najkratšiu cestu, ak taká existovala. Ďalším vylepšením metód navrhnutých *Danielsom* a *Huangom* bola práca *Kima* a *Tanchoca*, ktorí v r. 1991 prezentovali algoritmus nájdenia bezkonfliktnej časovo najkratšej cesty pre AGV v sieti s obojsmernými cestami. Ich algoritmus bol založený na Dijkstrovom algoritme hľadania najkratšej cesty. Pre každý uzol sa uchovával zoznam časových okien rezervovaných naplánovanými vozidlami a zoznam voľných časových okien, ktoré boli k dispozícií pre plánovanie ďalších vozidiel. Predstavený tu bol koncept grafu časových okien, v ktorom množina uzlov reprezentovala voľné časové okná a množina hrán reprezentovala dosiahnuteľnosť medzi voľnými časovými oknami. Potom algoritmus vlastne smeroval vozidlá cez jednotlivé voľné časové okná v grafe časových okien namiesto skutočných uzlov v sieti (podrobnejšie v metóde *Free windows*) *Kim* a *Tanchoco* svoju metódu vylepšili v r. 1993 o princíp tzv. konzervatívnej krátkozrakej stratégie, kde je uvažované naraz len jedno vozidlo v čase a všetky predošlé naplánovania sú striktne rešpektované a následné cestovné plány sú vozidlám prideľované len až keď vozidlo sa stane nečinným (*idle*) [3]. Na obrázku 1.7 je tabuľka s porovnaním vyššie spomínaných algoritmov.
Pre riešenie nášho projektu sme vybrali metódu *Free Windows* (pozri dodatok) popísanú *Kimom* a *Tanchocom* z r.1991 [1], ktorá je založená na Dijkstrovom algoritme hľadania najkratšej cesty v orientovanom grafe a využíva princíp časových okien na smerovanie AGV pohybujúcich sa po obojsmerných cestách v sieti. Pri analýze danej problematiky sme sa zaoberali *Petriho sieťami* a *udalostnými grafmi.*

## 1.11 Petriho siete

Petriho sieť je grafickým a matematicky opísateľným nástrojom, vhodným pre modelovanie a analýzu systémov diskrétnych udalostí. Ako grafický nástroj sú využívané ako pomôcka pre vizuálnu komunikáciu, podobne ako blokové diagramy, diagramy tokov a sietí. Navyše pomocou značiek dokážeme pomerne ľahko simulovať dynamické a súbežné udalosti v systéme. Sú tiež efektívny matematický nástroj, keďže ich dokážeme popísať pomocou stavových rovníc, algebraických rovníc, či inými matematickými modelmi, ktoré pokrývajú správanie sa systémov. Skrátene ich môžme použiť pre všetky systémy, ktoré môžme charakterizovať ako paralelné, distribuované, asynch-

| | Broadbent 1985 | Daniels 1988 | Huang 1989 | Kim a Tanchoco 1991, 1993 |
|---|---|---|---|---|
| Riešené problémy | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV | Hľadanie bezkonfliktnej časovo najkratšej cesty pre AGV |
| Základný algoritmus | Dijkstrov algoritmus hľadania najkratšej cesty | Deliaci algoritmus hľadania najkratšej cesty (PSP) | – | Dijkstrov algoritmus hľadania najtratšej cesty, konzervatívna krátkozraká stratégia |
| Zložitosť výpočtu | $O(N^2)$ (priemerný prípad) | O(N x A) (priemerný prípad) | $O((N+A)^2 \log(N+A))$ (priemerný prípad) | $O(v^4N^2)$ (najhorší prípad) |
| Smer ciest | obojsmerné | obojsmerné | obojsmerné | obojsmerné |
| Výhody | Ľahko realizovateľné | Ľahko realizovateľné a rýchlejšie ako Broadbentovo | Časové okná sú použité pre každý uzol, využitie úsekov ciest je zvýšené | Ľahko realizovateľné a kontrolovateľné, rýchle |
| Nevýhody | Zložitosť výpočtu, slabé využitie úsekov ciest | Zložitosť výpočtu, slabé využitie úsekov ciest, môže spôsobiť zlyhanie pri hľadaní ciest, ktoré existujú | Zložitosť výpočtu, veľké množstvo dát o konvertovanej sieti na udržiavanie | Zložitosť výpočtu, veľké množstvo dát o sieti na udržiavanie |

Legenda:

N – počet uzlov v sieti ciest A – počet hrán v sieti ciest v – počet vozidiel

Obrázok 1.7: Porovnanie algoritmov

rónne, nedeterministické, stochastické, súbežné. Majú nespočetné aplikácie v oblasti spracovávania dát, distribuovaných systémov, vyhodnocovania výkonu alebo riadenia zložitých procesov.

Petriho sieť pozostáva z miest, prechodov a hrán, ktoré ich ako orientované úsečky spájajú. Vstupné hrany spájajú jednotlivé miesta s prechodmi, zatiaľ čo výstupné hrany začínajú v prechodoch a končia v miestach. Podobne môžme rozdeliť aj jednotlivé miesta vzhľadom na konkrétny prechod na vstupné (s prechodom ho spája vstupná hrana) a výstupné (s prechodom ho spája výstupná hrana). Miesta môžu obsahovať značky, pomocou ktorých vzniká značkovanie ktorým môžme popísať konkrétny stav modelovaného systému. Prechody sú dynamické entity pomocou ktorých dokážeme modelovať akcie aké môžu nastať v reálnom systéme a tak meniť stav v ktorom sa Petriho sieť (modelovaný systém) nachádza. Prechody sa spúšťajú

len za špecifických podmienok, keď sú splnené všetky podmienky pre ich aktiváciu. Prechod je aktivovaný vtedy, ak sa vo všetkých vstupných miestach nachádza aspoň minimálny počet značiek, ktorý je určený príslušnou vstupnou hranou (jej kardinalitou).

Keď je daný prechod spustený, daný počet značiek vo vstupných miestach je odobraný a následne sa pridávajú značky do všetkých výstupných miest. Ich množstvo zase závisí od kardinality výstupných hrán. Základná hodnota hrán je 1 a zvyčajne sa číselne nezobrazuje. Pomocou spúšťania prechodov v určitom poradí za určitého počiatočného značkovania môžme skúmať danú Petriho sieť a určovať konkrétne vlastnosti.

Všeobecné Petriho siete neumožňujú modelovať to, že nejaká operácia trvá určitý čas, či hierarchiu systému. Takisto, počas simulácie reálnych systémov, na označenie miest vo viacerých prípadoch prirodzené čísla nepostačujú. Práve z tohoto dôvodu existuje viacero modifikácií, využívajúcich obohacovanie všeobecných vlastností o nové za účelom efektívnejšieho modelovania. Medzi tie známejšie patria:

- Deterministické časované PS – umožňujú popísať systém závislý od času, pričom berú do úvahy skutočnosť, že medzi začiatkom a koncom operácie prejde určitý čas.
- Stochastické PS – definujú stochastické časové oneskorenia prechodov.
- Farebné PS – siete tohto typu obsahujú značky rôzneho typu, ktoré sú navzájom rozlíšiteľné (graficky pomocou farby). Jednotlivým prechodom je priradená množina farieb, vzhľadom na ktoré môže byť aktivovaný. Hrany zase obsahujú funkciou, ako sa mení farba značky počas manipulácie s ňou. Značkami tohoto typu môžme modelovať rozličné dátové typy a podobne.
- Spojité PS – sú realizované aplikáciou reálnych čísiel na označovanie miest.
- Hybridné PS a ďalšie ...

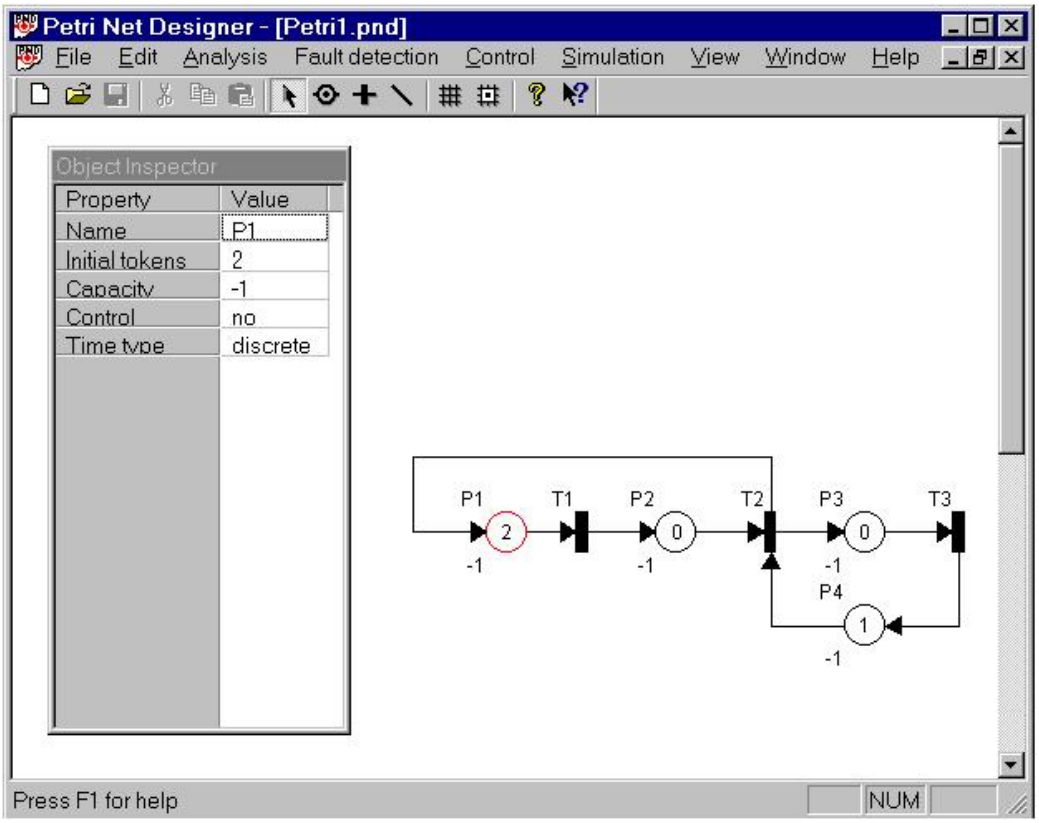
Značnou výhodou Petriho sietí je ich priama podpora pre analýzu rozličných vlastností modelovaného systému. Tieto vlastnosti sa vo všeobecnosti dajú rozdeliť do dvoch hlavných skupín a to na podľa toho, či sú závislé od počiatočného značenia, alebo nie. Medzi základné vlastnosti správania sa systému, ktoré závisia od počiatočného značkovania patria: dosiahnuteľnosť, ohraničenosť, živosť, reverzibilnosť, pokryteľnosť, stálosť, korektnosť. Ďalšie štrukturálne vlastnosti, ktoré nezávisia od počiatočného značkovania: štrukturálna živosť, kontrolovateľnosť, štrukturálna ohraničenosť, konzervatívnosť, opakovateľnosť, konzistentnosť.

### 1.11.1 Analýza Petriho sietí

Existuje viacero možností, ako analyzovať konkrétnu sieť. Medzi ne patrí aj graf dosiahnuteľnosti, vychádzajúci z grafovej reprezentácie stavového priestoru Petriho siete, maticový prístup, dekompozičné a redukčné techniky, teória formálnych jazykov, či ďalšie.

### 1.11.2 PNDesigner

Petri Nets Designer je aplikačný softvér vyvíjaný na Fakulte informatiky a informačných technológií STU v Bratislave a primárne je určený na pedagogické a výskumné účely. Jeho účelom je modelovanie Petriho sietí a




Obrázok 1.8: Petri Net Designer

následná analýza pomocou ktorej sa dajú zistiť a skúmať niektoré základné vlastnosti siete: ohraničenosť, živosť, reverzibilnosť, či analyzovať danú sieť pomocou stromu pokrytia a P, T invariantov. Umožňuje modelovanie jednoduchých, alebo aj zložitejších systémov pomocou intuitívneho rozhrania

a grafických prvkov. Ovláda sa štandardným systémom menu, charakteristickým pre okenné aplikácie. Implementačným prostredím programu je Microsoft Visual Studio a je spustiteľný na platformách operačného systému Microsoft Windows. Na obrázku 1.8 možno vidieť obrazovku počas behu programu.

### 1.11.3 CitectHMI

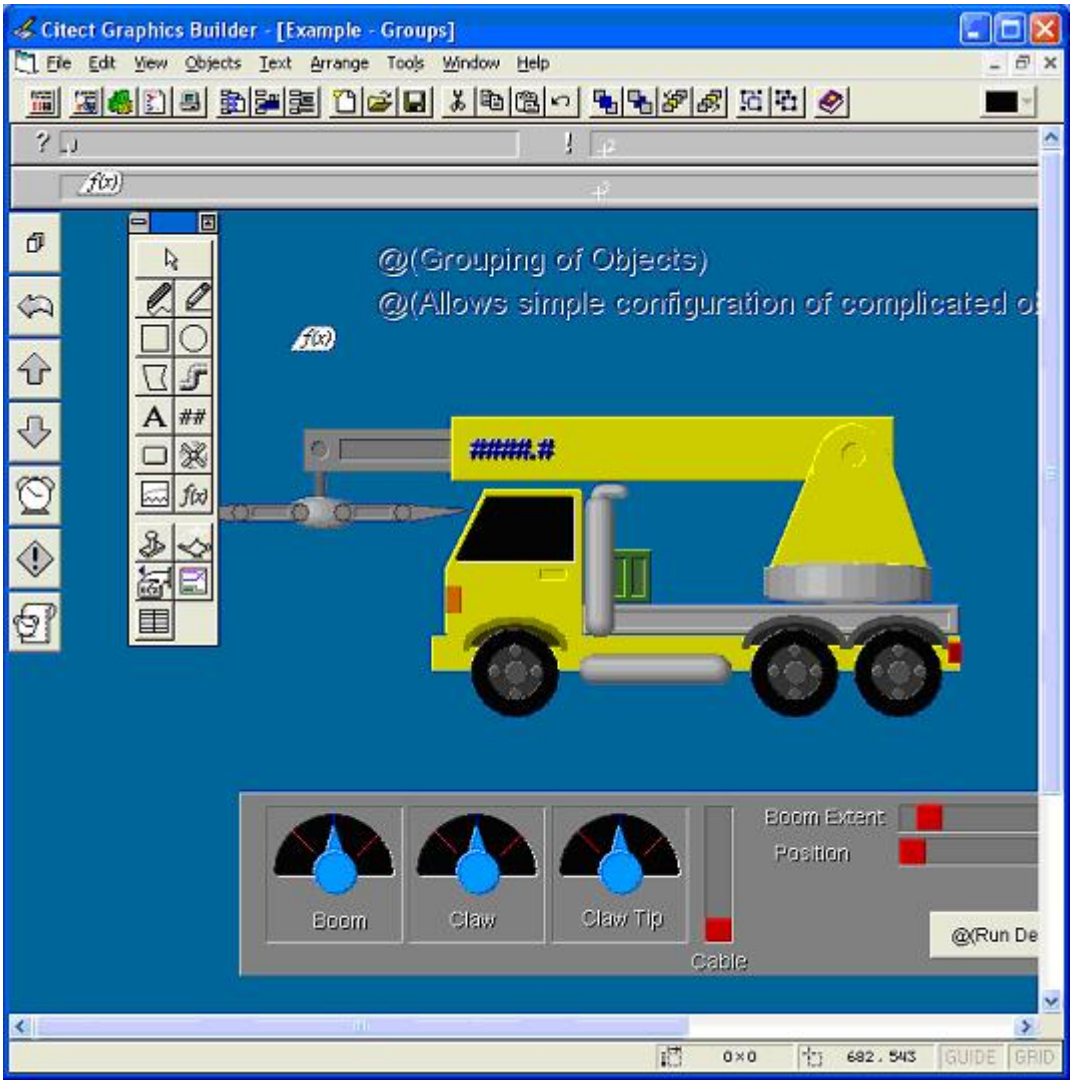
Citect je spoločnosť, ktorá už viac ako desiatku rokov vyvíja softvér špecializujúci sa na automatizáciu a ovládanie procesov. Medzi jej hlavné produkty patria CitectSCADA, či CitectHMI (obrázok 1.9). Citect HMI je programový balík určený pre vývoj a tvorbu vizulizačných systémov na platforme Microsoft Windows, vychádzajúci zo systému CitectSCADA. Medzi jeho najcharakteristickejšie vlastnosti patrí:

- Pružnosť – jednotlivé stupne vizualizačného systému je možné meniť počas života aplikácie pomocou sprievodcov a zmenou parametrov vo formulároch editora projektu.
- Redundancia – zabudovaná podpora pre redudantnú LAN, či redundanciu súborového serveru.
- Otvorenosť – z tohto hľadiska patrí Citect k naotvorenejším produktom v oblasti SCADA systémov na trhu.
- Cicode – vlastný programovací jazyk používaný v CitectSCADA. Štruktúra a syntax tohto jazyka je veľmi podobná programovaciemu jazyku Pascal. Medzi hlavné rozdiely patrí absencia smerníkov. Pomocou vstavaných funkcií je možné jednoducho implementovať niektoré zložitejšie jazykové konštrukcie ako súbeh procesov a podobne.

Samotné grafické rozhranie sa vyznačuje svojou rýchlosťou, podporou bežných grafických formátov či šablón a systém umožnuje použitie neobmedzeného počtu okien a veľký počet animačných prvkov v jednom okne. Obsahuje tiež preddefinovanú knižnicu symbolov parametrizovateľných grafických objektov a okien s možnosťou tvorby vlastných knižníc.

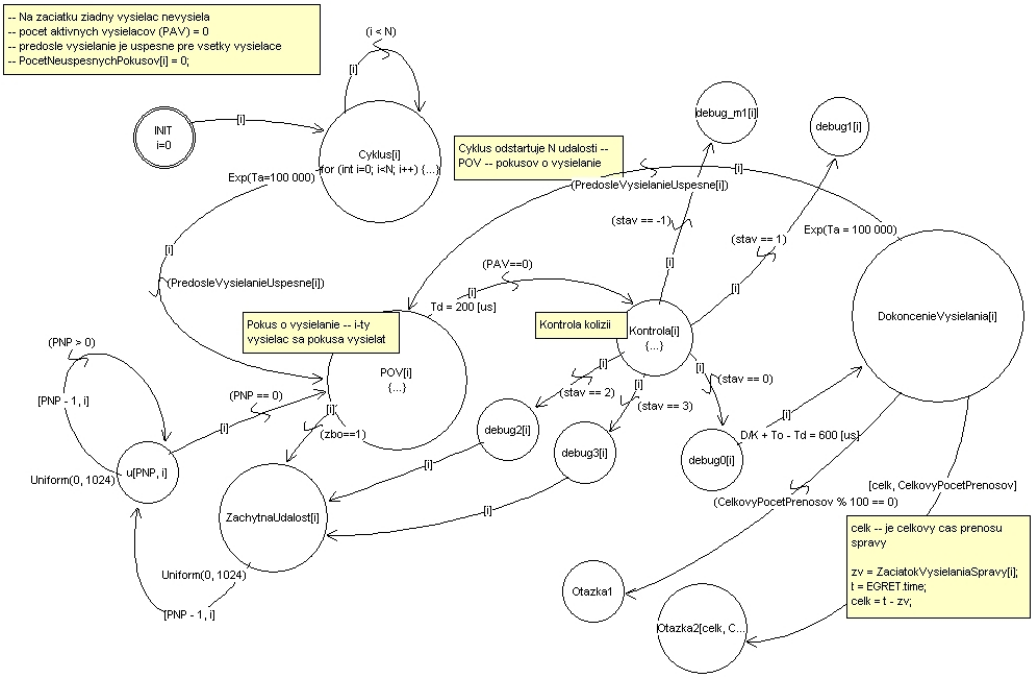
## 1.12 Grafy udalostí

Ďalším formalizmom, ktorý sme skúmali popri Petriho sieťach, sú grafy udalostí v podobe nástroja *EGRET* obrázok 8.1, C# knižnice *SimuLib* a C




Obrázok 1.9: Časť vývojového prostredia Citect

knižnice *tiny*. Jedná sa o orientované grafy, kde vrcholy (krúžky) predstavujú udalosti a hrany predstavujú ich plánovanie v čase. Plánovanie ďalších udalostí môže byť podmienené, t.j. aby sa vykonala ďalšia udalosť musí byť splnená daná podmienka. Označuje sa to vlnovkou pri príslušnej hrane (spolu s podmienkou) a ďalej, udalosti môžu byť parametrizovateľné (nad šípkou je v hranatých zátvorkách zoznam parametrov, ktoré sa predávajú udalosti). Keď nastane udalosť, vykoná sa zadaný JavaScriptový kód príslušnej udalosti. Po jeho dobehnutí sa plánujú udalosti do ktorých smerujú




Obrázok 1.10: EGRET – Graf udalostí

šípky. Plánovanie spočíva v tom, že sa nová udalosť zaradí do kalendára udalostí, podľa zadefinovaného časového oneskorenia, ktoré môže byť dané nahodným rozdelením (s parametrami) alebo s konštantným časovým oneskorením. Pridanie udalosti do kalendára spôsobí jeho usporiadanie podľa času. Ak sú udalosti plánované na rovnaký čas, rozhoduje sa o ich poradí podľa priority. Podobnú výpočtovú silu ako *EGRET* majú knižnice *SimuLib* a *tiny*, avšak bez názornej vizualizácie.

# Kapitola 2

# Špecifikácia požiadaviek

Základnou požiadavkou zadania je vytvorenie softvérového produktu implementujúceho niektoré špecifické algoritmy pre riadenie dopravy a ich následná vizualizácia. U nej je vhodné dbať najmä na jednoduchosť použitia, pri zachovaní intuitívnosti ovládnia. V tejto časti uvedieme špecifikáciu požiadaviek z hľadiska vizualizácie AGV.

Systém by mal umožniť:

1. Navrhnúť trajektórie, dráhy, ciesty a uzly po ktorých sa budú vozidlá pohybovať.
2. Uzly predstavujú zastávky, v ktorých sa vozíky určitý čas zdržia, pričom tento čas bude možné zadávať (konfigurovať, meniť dynamicky v závislosti od typu činnosti).
3. Zastávky môžu byť napríklad prekladiská, parkovacie miesta, čerpacie stanice a pod. K zastávke je možné priradiť textový popis podľa jej účelu.
4. Dráhy budú mať tvar orientovaného grafu s možnosťou zakrivenia hrán ako aj pomenovaním hrán (napríklad názvami ulíc) a ohodnotením hrán podľa ich dĺžky.
5. Bolo by vhodné, keby graf mohol byť priestorový (nie len rovinný), čo by umožnilo napríklad vizualizáciu spomínaných laboratórnych vozíkov pohybujúcich sa vo viacpodlažnej budove.
6. S tretím rozmerom prichádza prirodzene požiadavka približovania a natáčania scény (prototyp nebude obsahovať trojrozmerné scény).

7. Vozíky budú reprezentované jednoduchými geometrickými objektami (kruh, štvorec, resp. ich kompozícia), odlíšiteľných farebne, s možnosťou priradenia identifikátoru k vozíku.

8. K vlastnostiam vozíka bude ďalej patriť aktuálna rýchlosť a priestorová pozícia (poprípade zrýchlenie, alebo kód dynamicky upravujúci rýchlosť na základe vonkajších podnetov). Vozík bude vedieť svoju trasu (t.j. kadiaľ má ísť) a bude obsahovať kód, ako túto trasu dynamicky meniť na základe podnetov zvonka. Trasu vozíkom bude vypočítavať aj optimalizačné jadro programu na základe algoritmu *Free Windows*.

9. Spracovanie vonkajších podnetov môže byť realizované preposielaním správ.

## 2.1 Špecifikácia funkcií systému

Na obrázku 2.1 je zobrazený diagram prípadov použitia, ktorý predstavuje prehľadný spôsob zobrazenia hlavnej funkcionality navrhovaného programového systému vzhľadom na koncového používateľa v grafickej podobe. Po ňom nasleduje slovný popis a podrobnejšie vysvetlenie jednolivých funkcií. Niektoré triviálne funkcie a činnosti kvôli väčšej prehľadnosti do diagramu zahrnuté neboli.

#### Funkcia 1: Výber modelu

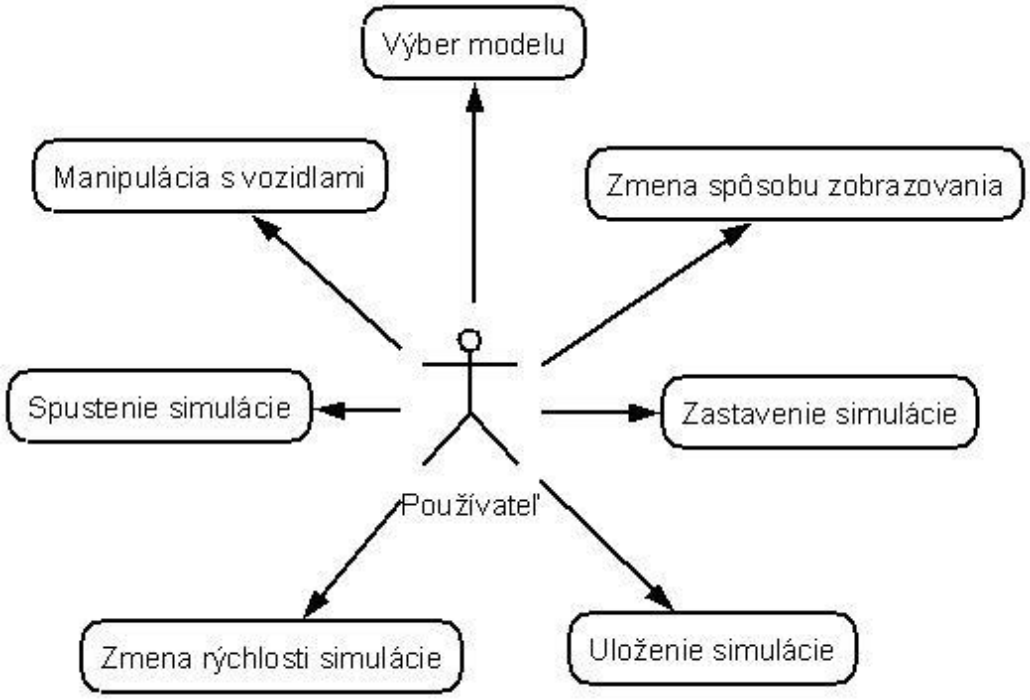
Používateľ si po spustení simulátora, alebo aj počas jeho behu, zvolí model pozostávajúci z uzlov, dráh a vozidiel s už definovaným cestovným poriadkom. S týmto modelom môže ďalej pracovať - manipulovať s vozidlami, spúšťať a pozastavovať simuláciu...

#### Funkcia 2: Spustenie simulácie

Tesne po výbere modelu a po prípadnej manipulácii s vozidlami by mal používateľ po aktivovaní príslušného ovládacieho prvku spustiť simuláciu. Tá pozostáva zo zobrazovania pohybu vozidiel po existujúcich cestách v čase.

#### Funkcia 3: Zastavenie simulácie

Počas behu simulácie je niekedy nutné nielen spomaliť simulačný čas, ale ho aj úplne zastaviť. Typickým príkladom takejto situácie je okamih, kedy používateľ zadáva cieľ cesty viacerým vozidlám súčasne a je podstatný je práve súbežný štart z príslušných zdrojových uzlov.




Obrázok 2.1: Model prípadov použitia

**Funkcia 4: Zmena rýchlosti simulácie**

Simulácia vozidiel v modelovanom systéme môže prebiehať v reálnom čase, ale pre účely ďalšieho skúmania je vhodné umožniť zrýchľovanie alebo spomaľovanie simulačného času. Táto zmena neovplyvní rýchlosť samotných vozidiel.

**Funkcia 5: Manipulácia s vozidlami**

V modelovanom systéme sa pohybuje viacero rozlíšiteľných vozidiel podľa svojich cestovných poriadkov. Používateľ má mať možnosť ľubovolné z týchto vozidiel odstrániť, či pridať nové. Takisto by mala existovať možnosť meniť cieľ cesty, pre jednotlivé vozidlá.

**Funkcia 6: Zmena spôsobu zobrazovania**

Vizualizácia modelu a celej simulácie musí umožniť používateľovi zvoliť si úroveň zobrazovaných podrobností a taktiež zmeniť zobrazenie modelu medzi klasickým orientovaným grafom a zobrazením časových okien.

**Funkcia 7: Uloženie simulácie**

Keďže simulácia môže pozostávať z velkého počtu vozidiel, ktorých počiatočné a cieľové stanice treba zadávať ručne, je vhodné umožniť používateľovi uložiť tieto parametre pre neskoršie znovupoužitie.

# Kapitola 3

# Hrubý návrh riešenia

Programový systém by mal pozostávať z dvoch častí. Tou prvou je program implementujúci samotné algoritmy na nájdenie najkratšej možnej cesty v grafe bez narušenia už existujúcich cestovných plánov. Algoritmom na to použitým bude metóda *Free Windows*, prípadne jej vylepšujúce modifikácie. Tento program nebude obsahovať žiadne používateľské rozhranie, na to bude slúžiť druhý program zabezpečujúci celú komunikáciu s používateľom, zobrazovanie simulácie, postaveným na technológií OpengGL.

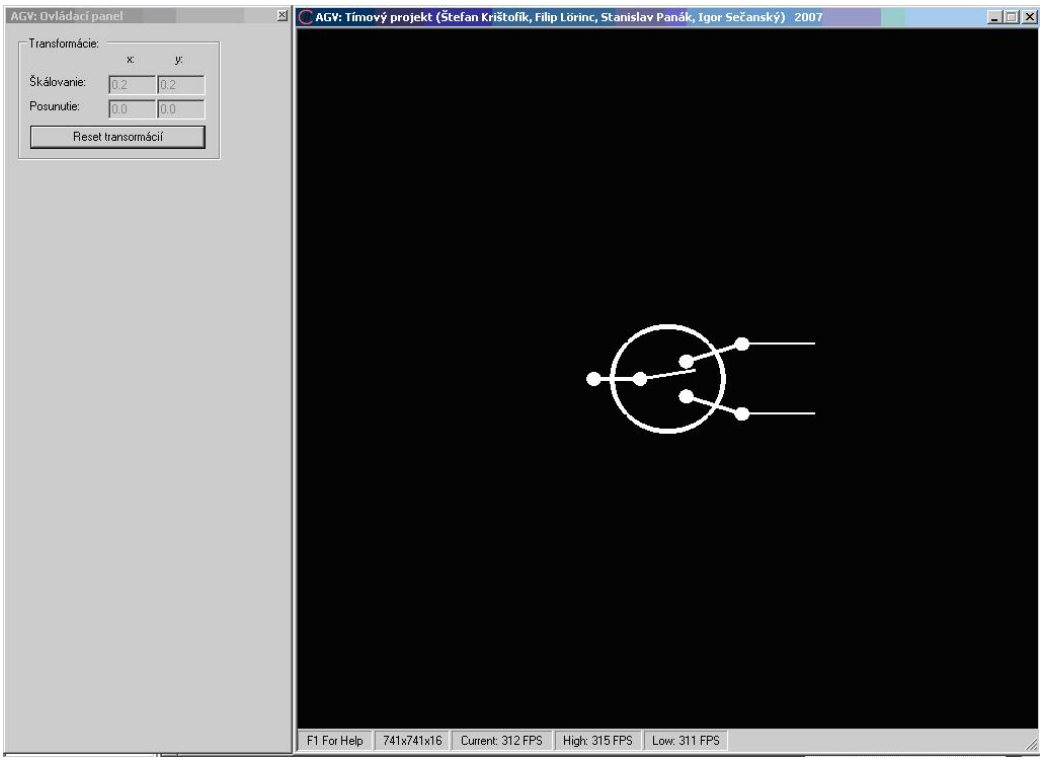
## 3.1 Cyklus programu

Postupnosť ako budú vykonávané jednotlivé činnosti kombinované s interakciou samotného používateľa s rozhraním znázorňuje diagram na obrázku 3.3. Zo systémového pohľadu sú toto hlavné funkcie pre implementáciu.

## 3.2 Vizualizácia

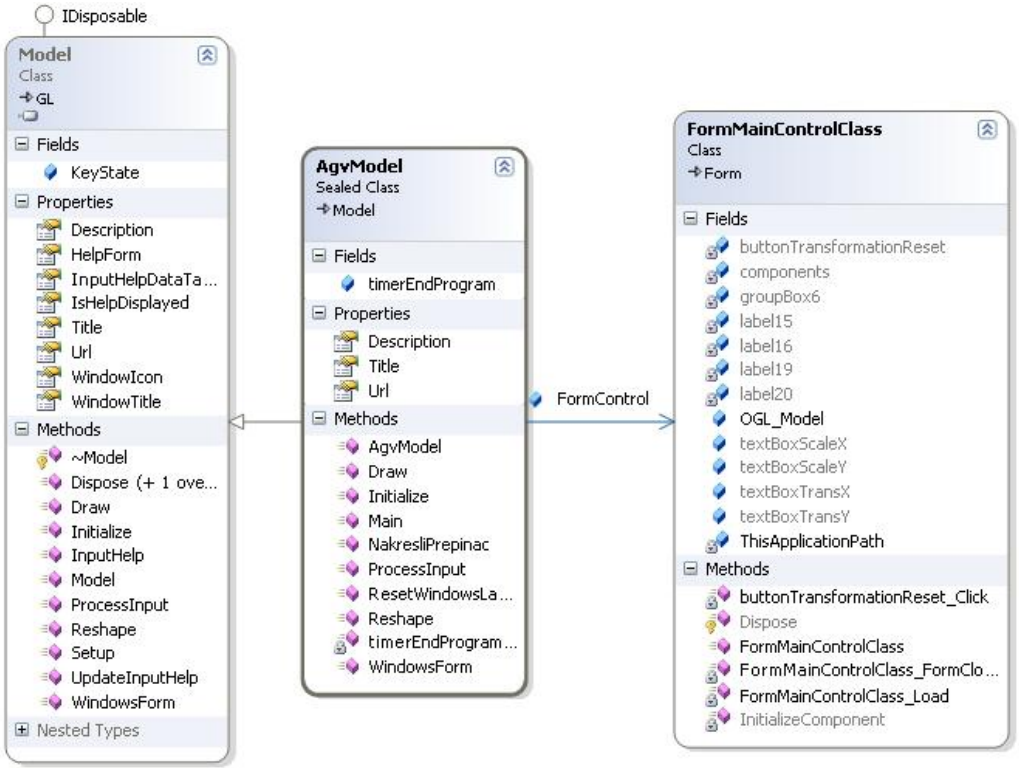
Prototyp je v počiatočných fázach vývoja. Zatiaľ je hotový vizualizačný framework 3.1, kde je panel, v ktorom budú umiestnené ovládacie prvky aplikácie, prepojený s CsGL oknom, v ktorom bude prebiehať vykreslenie scény. Scéna reaguje na zoom a posunutie. Nástroj *Egret* môže byť použitý dvomi spôsobmi: na modelovanie udalostí, aj na nakreslenie topológie trajektórií, kadiaľ budú chodiť vozidlá. Jeho výstupom je XML súbor, ktorý môže byť načítaný do našej aplikácie, ktorá zobrazí identický orientovaný graf. Na uložený súbor z *Egretu* by mohol byť zavesený objekt *FileSystemWatcher*, pomocou ktorého by sa dala udržiavať jeho aktuálnu verziu. Nevýhodou *Egretu* je, že nedokáže zobrazovať geometrické objekty, ktoré by sa spojito

presúvali po grafe. Preto si túto funkcionalitu musíme doprogramovať sami. Navrhujeme využiť plánovacie funkcie knižnice *SimuLib*, alebo využiť

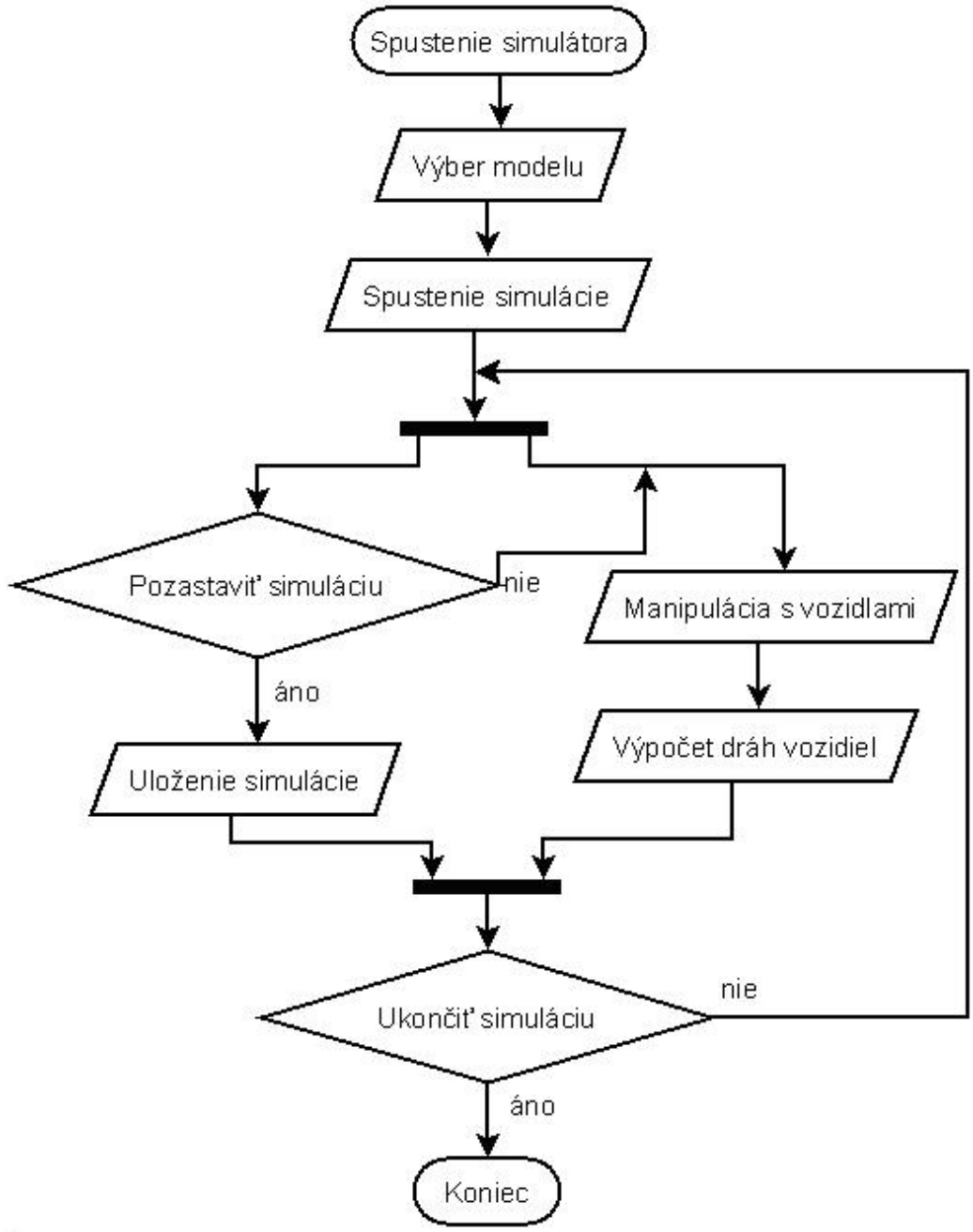



Obrázok 3.1: Vizualizačný framework

metódu *Free Windows* opísanú ďalej. UML diagram tried vizualizačného frameworku vidno na obrázku 3.2




Obrázok 3.2: Diagram tried vizualizačného frameworku




Obrázok 3.3: Životný cyklus programu

# Časť II

# Letný semester

# Kapitola 4

# Modul pre simuláciu

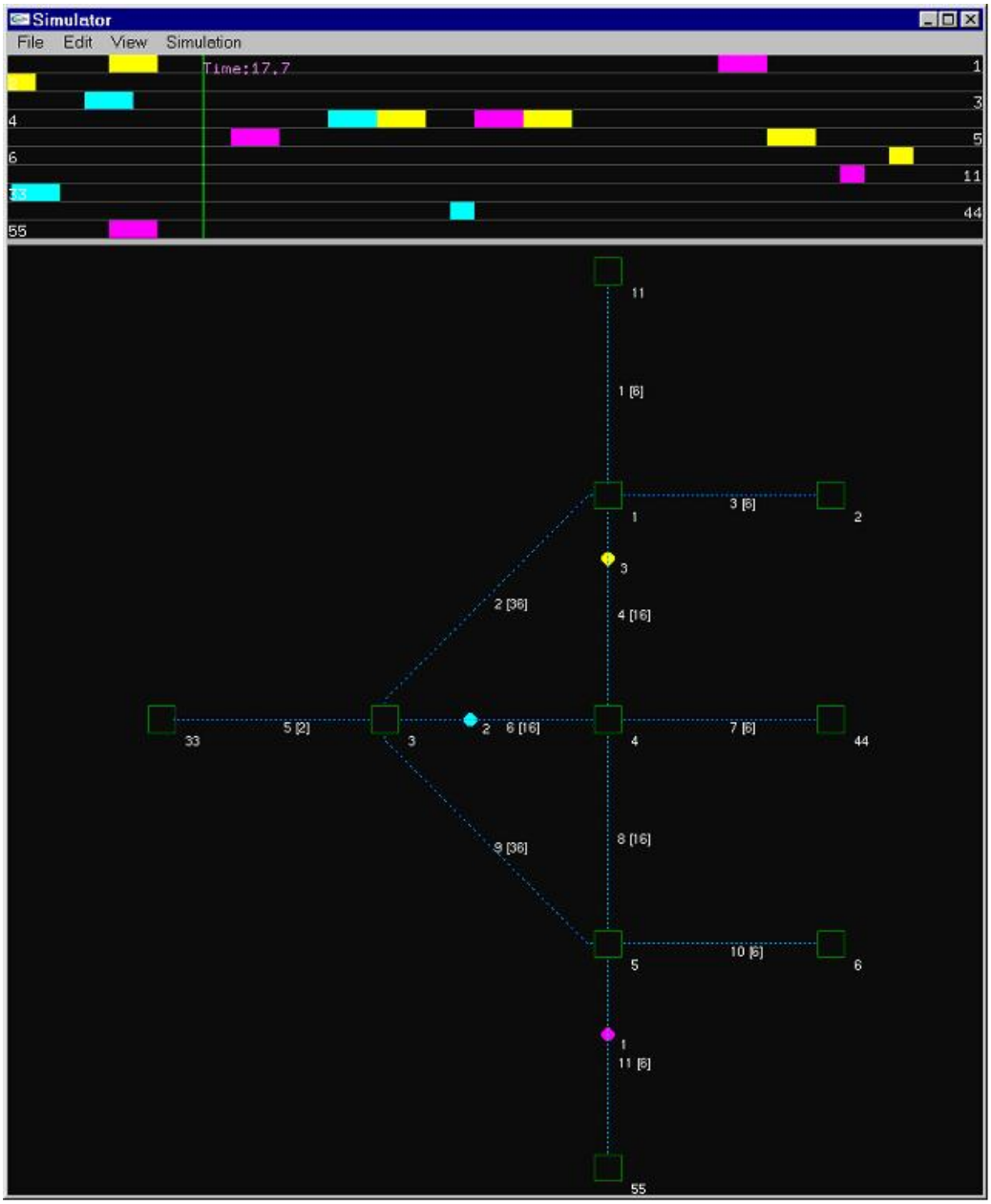
Táto kapitola obsahuje používateľskú príručku a ďalšie podrobnoti ohľadom vizualizačného modulu.

## 4.1 Inštalácia programu

Samotná inštalácia pozostáva v nakopírovaní vykonateľného súboru do ľubovoľného adresára. Vzhľadom na fakt, že program využíva nadstavbovú grafickú knižnicu *Glut*, je potrebné aby daný systém vo svojom systémovom adresári, alebo v adresári so spustiteľným súborom obsahoval aj súbor `glut32.dll`. V tomto stave je program možné spustiť, avšak pre využitie jeho plnej funkcionality je potrebný aj spustiteľný súbor zabezpečujúci dodanie cestovného poriadku pre konkrétne vozidlo. Program je spustiteľný pod operačných systémom MS Windows a vzhľadom na použité statické linkovanie nevyžaduje žiadne ďalšie externé knižnice.

## 4.2 Používateľské rozhranie

Vzhľadom na požiadavku čo najjednoduchšieho ovládania program využíva oknový systém a jeho charakteristické prvky - tlačidlá, textové, či výberové polia. Pomocou rozličných typov dialógov sa snaží chrániť používateľa pred zadávaním nevhodných parametrov, vykonaním nepovolenej úlohy, ako aj priebežne informuje o nežiadúcom stave aplikácie.



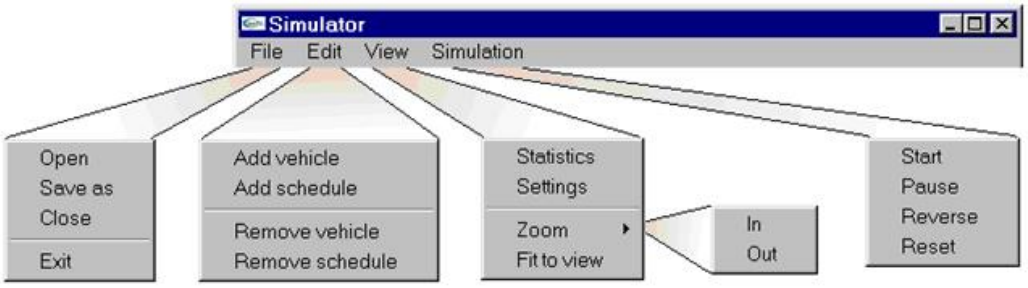

Obrázok 4.1: Hlavné okno aplikácie

## 4.3 Práca s programom

Hlavné okno aplikácie možno vidieť na obrázku 4.1. Je rozdelené na dve časti. Vrchná časť obsahuje zobrazenie jednotlivých cestovných plánov vozidiel

v čase, vzhľadom na ich polohu v niektorej z križovatiek či koncovom uzle. Vždy keď sa vozidlo nachádza v niektorom z uzlov, objaví sa v danom riadku obdĺžnik príslušnej farby s adekvátnou veľkosťou. Takisto je tu lokalizovaný aj statický bežec reprezentujúci čas, v ktorom sa celá vizualizácia nachádza. Intekacia s týmto podoknom je možná pomocou kolieska na myši, ktoré spôsobuje škálovanie horizontálnej časovej osi. Súčasným stlačením ľavého tlačidla a posúvaním myši na niektorú zo strán dochádza k zmene aktuálneho času.

V spodnej časti sa zobrazuje samotný model reprezentovaný sieťou poprepájaných uzlov a pohybujúcich sa vozidiel. Znova je možné pomocou kolieska na myši zväčšovať alebo zmenšovať zobrazovanú oblasť, alebo sa v nej presúvať pomocou ľavého tlačidla.



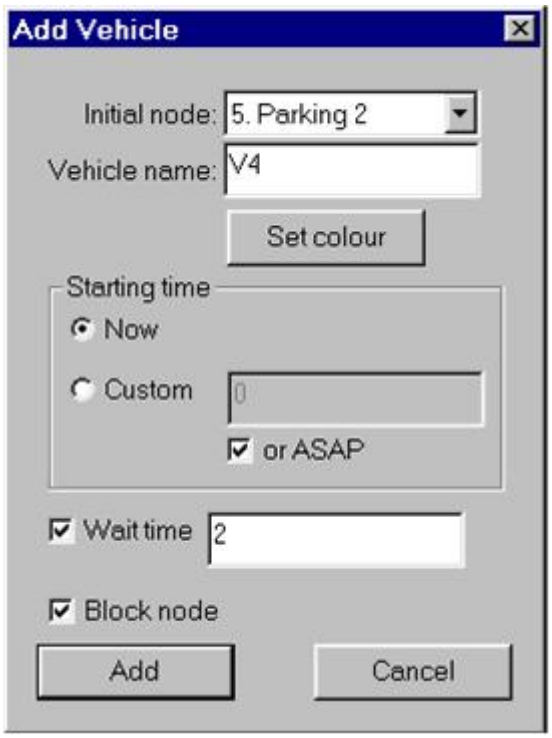

Obrázok 4.2: Systém menu

Systém menu možno podrobnejšie vidieť na obrázku 4.2. Voľbou *Open* alebo *Save as* sa otvorí systémový dialóg pre výber súboru. Následne je daný model obnovený, alebo uložený do zvoleného súboru. Voľba *Close* slúži na ukončenie práce s modelom a znovunastavením programu do jeho východiskového stavu. *Exit* ukončí aplikáciu.

Všetky položky, ktoré možno nájsť v podmenu *Edit* slúžia na manipuláciu s aktuálne zobrazeným modelom:

- *Add vehicle* – vytvorí dialóg pre pridanie nového vozidla.
- *Add schedule* – dialóg pre naplánovanie ďalšej cesty pre vozidlo.
- *Remove vehicle* – dialóg pre odstránenie vozidiel.
- *Remove schedule* – dialóg pre odstránenie položiek cestovného plánu konkrétneho vozidla.

Dialóg pre pridávanie nových vozidiel možno vidieť na obrázku 4.3. Používateľ si zvolí počiatočný uzol, zvolí si popis a farbu vozidla, ako aj štartovací



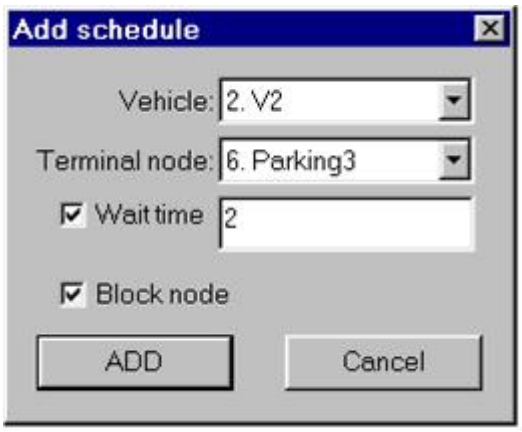

Obrázok 4.3: Dialóg pre pridanie vozidla

čas, v ktorom sa má vozidlo v danom uzle objaviť. *Now* symbolizuje aktuálny vizualizačný čas, *Custom* umožní zadať ľubovoľné nezáporné číslo. V prípade ak je uzol momentálne blokovaný, program nedovolí pridanie vozidla. Zaškrtnutím voľby *or ASAP* necháme program vyhľadať najbližší možný čas, kedy je uzol voľný.

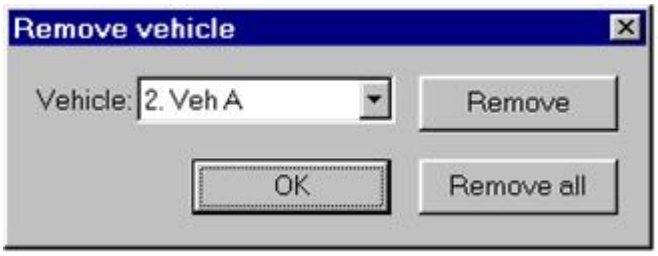
Ďalšie dve zaškrtávacie polia, *Wait time* a *Block node* určujú, aké bude nasledujúce správanie sa vozidla. Zaškrtnutím prvej voľby vozidlo bude používateľom zadaný čas čakať v strede uzla, zatiaľ čo zaškrtnutím tej druhej ho bude blokovať na neurčito dlhú dobu. Je možné nechať zašktnuté obe voľby, avšak nie žiadnu z nich. Samotné pridanie vozidla je realizované až po stlačení tlačidla *Add*.

Na zadávanie príkazov pre existujúce vozidlá slúži dialóg na obrázku 4.4. Používateľ si zvolí cieľový uzol, do ktorého sa má dané vozidlo dostať. Pre zaškrtávacie polia platia podobné pravidlá, ako v predchádzajúcom dialógu. Po stlačení tlačidla *Add* sa v pozadí spustí externý program zabezpečujúci samotné naplánovanie trasy, ak takáto existuje.

Na obrázku 4.5 možno vidieť dialóg pre odobratie vozidla. Používateľ




Obrázok 4.4: Dialóg pre pridanie cestovného plánu



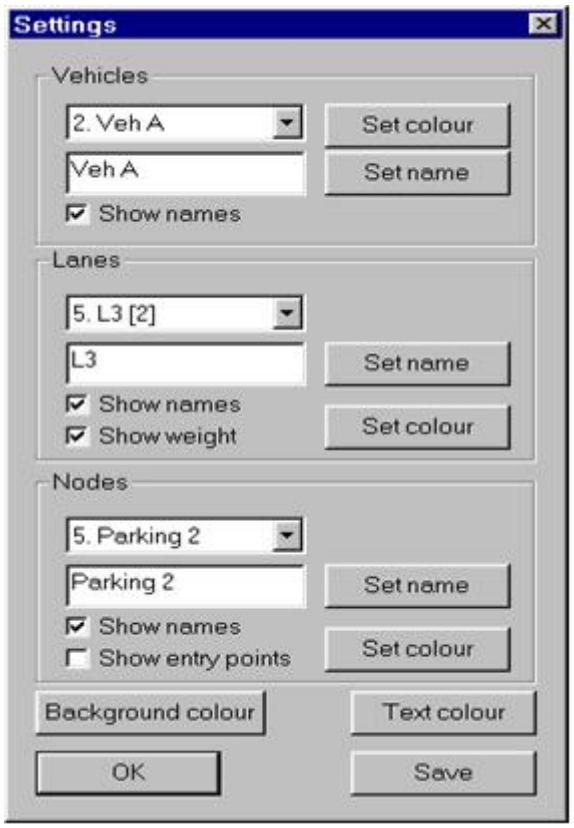

Obrázok 4.5: Dialóg pre odobratie vozidla

si môže pomocou rolovacieho menu vybrať konkrétne vozidlo a pomocou tlačidla *Remove* ho z modelu odobrať. Spolu s vozidlom bude odstránený aj jeho cestovný poriadok. Tlačidlom *Remove all* je možné z modelu odstrániť všetky vozidlá súčasne.

Podmenu *View* obsahuje nasledujúce položky:

- *Statistics* – zobrazenie dialógu s podrobnosťami v číslach.
- *Settings* – zobrazenie dialógu rozličnými s nastaveniami.
- *Zoom* – obsahuje ďalšie podmenu a slúži na zväčšovanie a zmenšovanie zobrazovaného podokna s modelom.
- *Fit to view* – znovunastaví škálovací pomer tak, aby sa celý model zmestil do okna.

Na obrázku 4.6 možno vidieť dialóg nastavení. V ňom je možné konfigurovať jednak podrobnosti daného modelu (farbu a mená vozidiel, uzlov a ciest),




Obrázok 4.6: Dialóg nastavení

ako aj úroveň zobrazovaných detailov, farbu pozadia, či textu. Tieto aplikačné nastavenia sú uložené v konfiguračnom súbore `Settings.conf`, ktorý je ukladaný v aktuálnom pracovnom adresári a ak je tento súbor prítomný pri štarte programu, sú z neho tieto nastavenia automaticky obnovené (v opačnom prípade sa použije defaultné nastavenie).

Podmenu *Simulation* obsahuje nasledujúce položky:

- *Start* – spustí inkrementovanie vizualizačného času.
- *Pause* – zastaví zmenu času.
- *Reverse* – spustí dekrementovanie vizualizačného času.
- *Reset* – nastaví čas na počiatok časovej osi (nulový čas).

Aplikácia ďalej umožňuje aj obmedzené ovládanie pomocou klávesnice, ktoré je však v niektorých prípadoch efektívnejšie a rýchlejšie:

**s** : pozastavenie simulačného času

**p** : pokračovanie simulácie

**r** : spätné spustenie simulácie

**+** : posunutie simulačného času dopredu

**-** : posunutie simulačného času dozadu

***** : skokové posunutie simulačného času dopredu

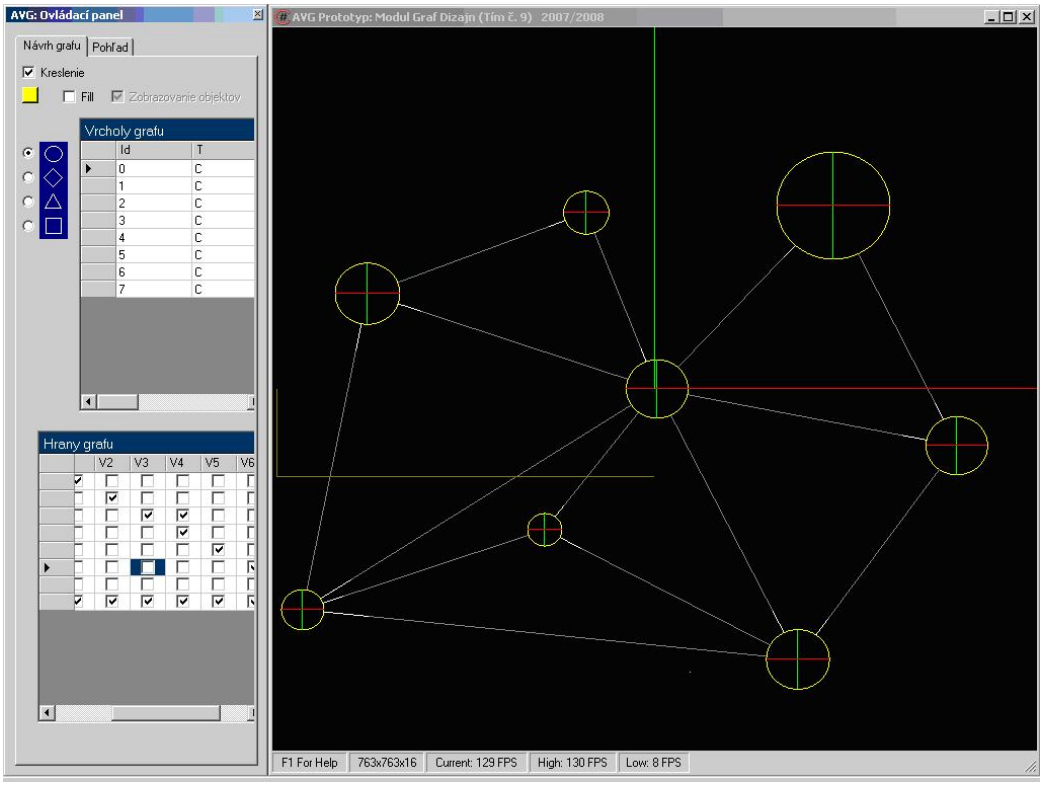
**/** : skokové posunutie simulačného času dozadu

**F5** : vycentrovanie obrazovky

# Kapitola 5

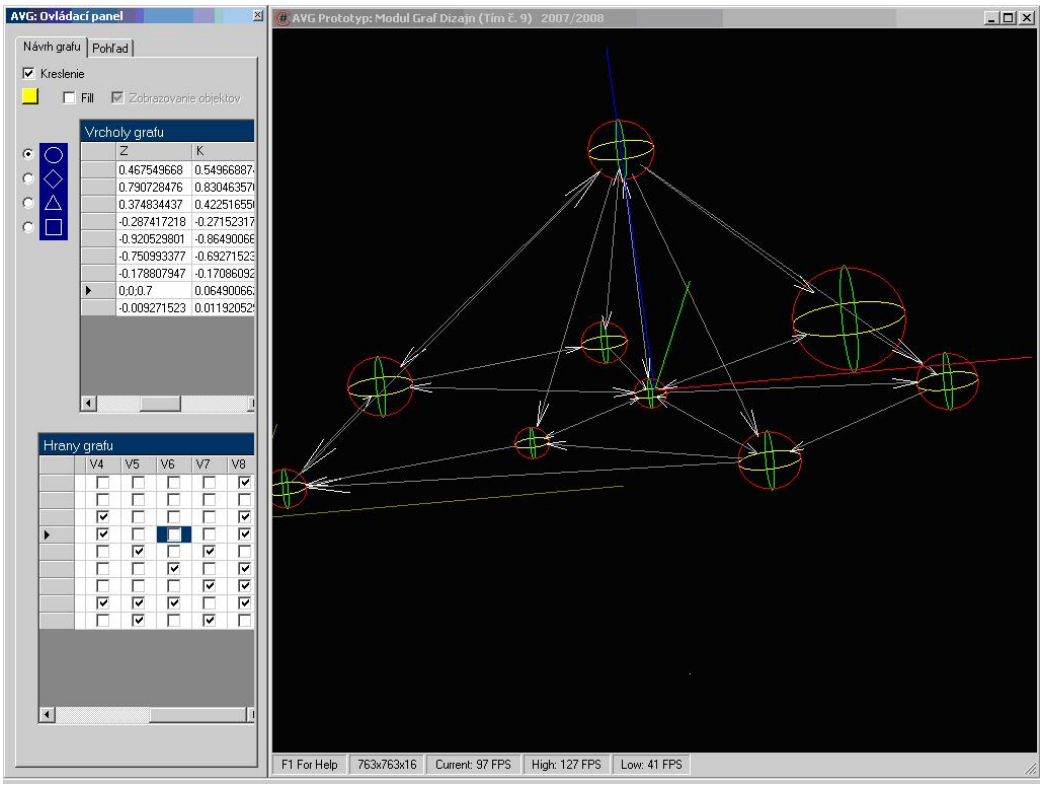
# Modul pre tvorbu grafu

V tejto časti je popísaný modul pre návrh orientovaného grafu, pomocou ktorého sa definujú trajektórie a dráhy, po ktorých sa budú vozidlá pohybovať. Výstupom tohto modulu je textový popis grafu s ohodnotením hrán podľa ich dĺžky, ktorý slúži ako vstup pre simulátor popísaný v predošlej kapitole.




Obrázok 5.1: Kreslenie vrcholov grafu

Cestovný plán vozidiel, ktorý je ďalšou súčasťou vstupu pre simulátor sa zatiaľ zadáva ručne v textovej forme. Samotná simulácia prebieha v 2D. Pre potreby simulátora sa ku každej orientovanej hrane automaticky pridáva aj opačná orientácia.



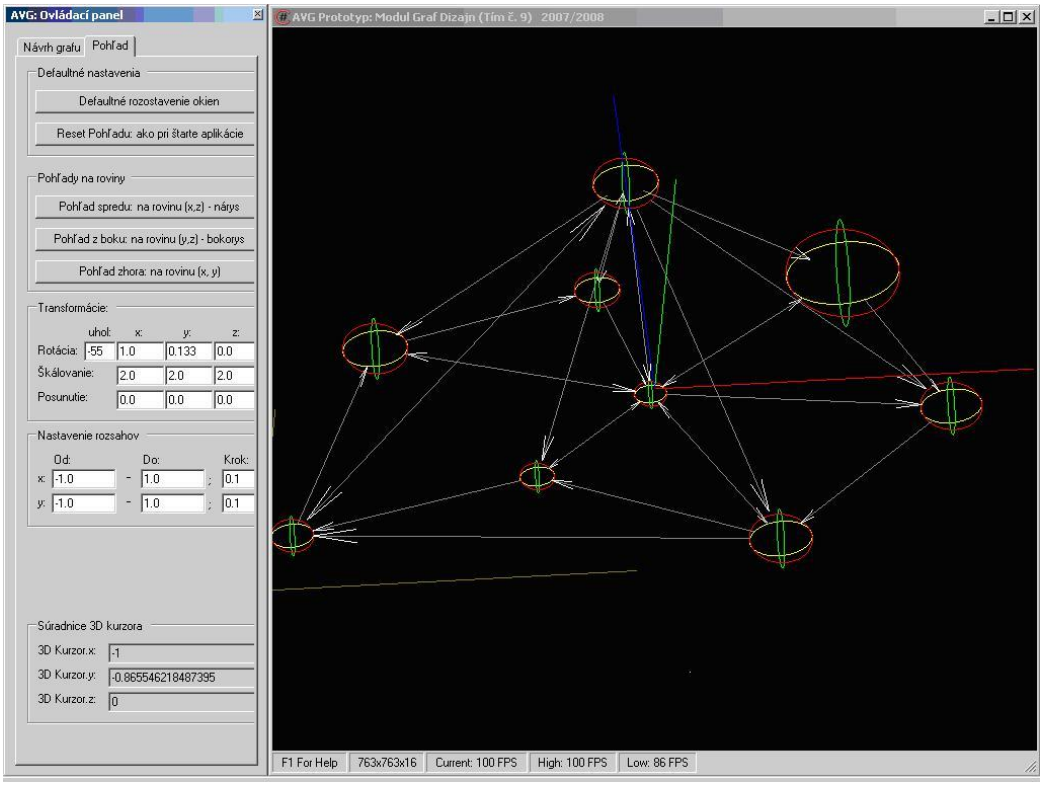

Obrázok 5.2: Definovanie hrán

## 5.1 Funkcionalita modulu

- Návrh orientovaného grafu v priestore
- Pridávanie a odoberanie vrcholov
- Rôzne tvary, farby a veľkosti vrcholov
- Pridávanie a odoberanie hrán
- Konzistencia pri odoberaní

- 3D transformácie priestoru
- Rôzne pohľady na orientovaný graf
- Uloženie do súboru, výber zo súboru

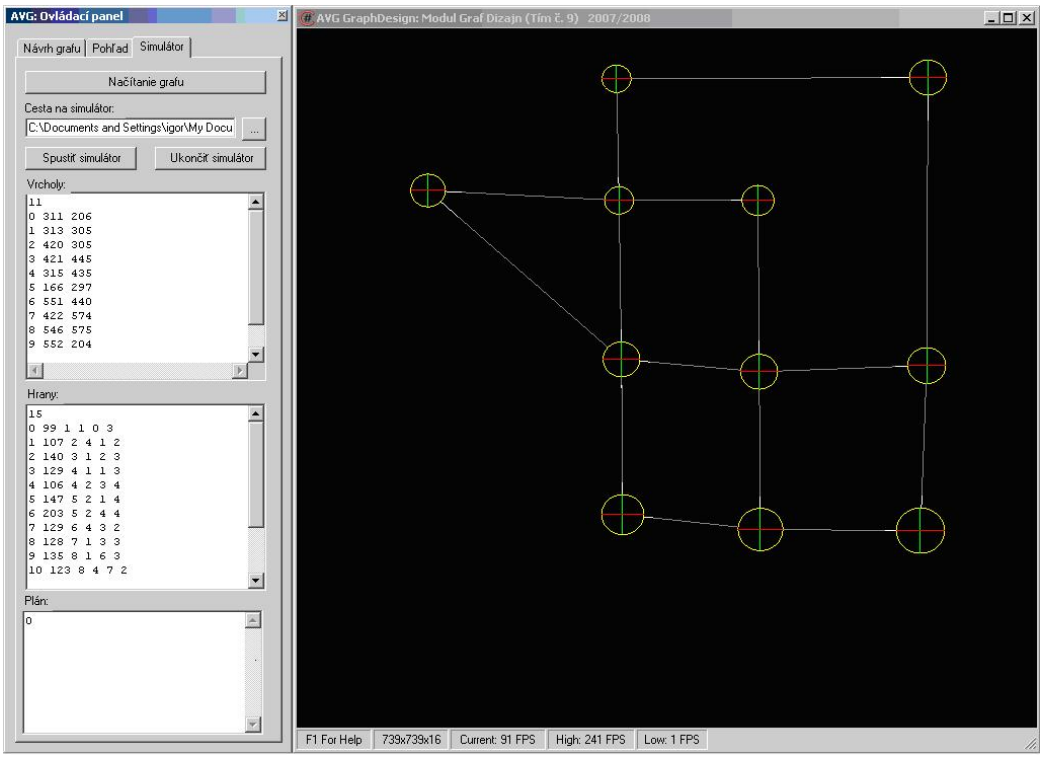
Program sa skladá z dvoch hlavných okien (obr. 5.1): z ovládacieho panelu, na ktorom sú umiestnené ovládacie prvky a z okna pre vykreslenie 3D scény. Ovládací panel má dve záložky. Prvá má prvky pre kreslenie grafu a druhá slúži na transformovanie scény (škálovanie, posúvanie, otáčanie, nastavovanie rozsahov a pohľadov).




Obrázok 5.3: Transformácie a pohľady na graf

Používateľ kreslí vrcholy grafu do scény, pričom volí veľkosti, tvary, farbu a výplň vrcholov. Z každým pridaným vrcholom sa pridá záznam do tabuľky vrcholov grafu. Tá obsahuje vlastnosti vrcholu ako identifikátor, tvar, súradnice začiatočného a koncového bodu vrcholu pri kreslení, polomer, farbu a výplň. Podobne sa mení aj tabuľka hrán. S pridaným vrcholom sa do nej

pridá zodpovedajúci riadok a stĺpec prislúchajúci pridanému vrcholu. Implicitne je nastavené, aby hrana medzi vrcholmi nebola. Tú pridá až používateľ odkliknutím checkboxu, ktorý určuje začiatok a koniec orientovanej hrany (obr. 5.2).



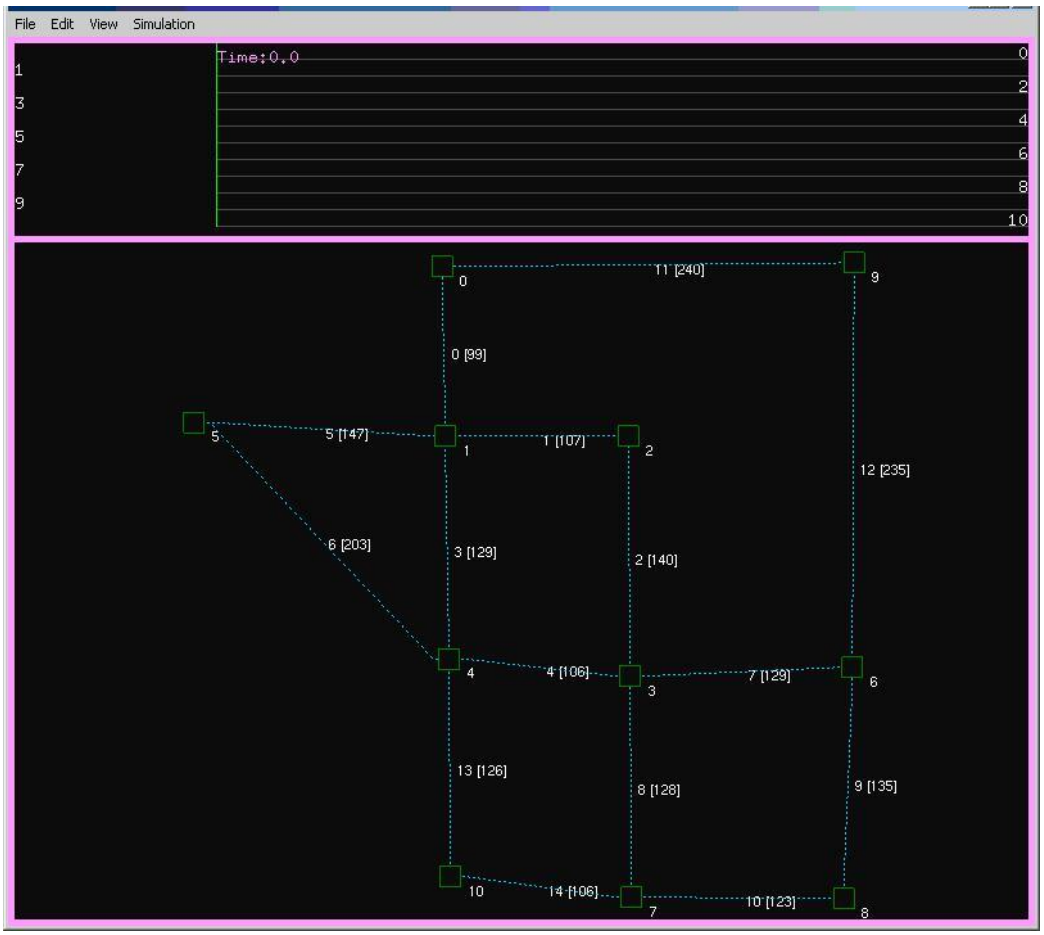

Obrázok 5.4: Návrh grafu pre simulátor

Program umožňuje interaktívne pridávanie a odoberanie hrán a vrcholov. Odobratím vrcholu sa vymažú všetky hrany, ktoré sú s ním incidentné. Rovnako je možné interaktívne meniť vlastnosti vrcholov. Na obrázku (obr. 5.2) sme takto vrchol posunuli z roviny xy (kde sú ostatné vrcholy) do priestoru. V tabuľke hrán grafu nie je možné riadky odoberať ako v tabuľke vrcholov. Riadky (aj stĺpce) pre odobraté vrcholy sa vymažú automaticky.

V záložke *Pohľad* ovládacieho panelu sú ovládacie prvky pre transformáciu scény. Približovanie a vzďaľovanie scény funguje aj pomocou myši a ľavého tlačítka. Posunutie funguje tiež pomocou pravého tlačítka a myši v okne scény. Natáčanie grafu (obr. 5.3) sa ovláda iba v ovládacom paneli a to tak, že buď parametre pre rotáciu zadáme do príslušných polí, alebo na políčko príslušného parametra klikneme a pohybujeme myšou so stlačeným ľavým tlačítkom. Hodnota v políčku sa automaticky zväčšuje alebo zmenšuje

podľa toho, do akej strany pohybujeme myšou.

Na obrázku 5.4 je znázornená záložka *Simulátor*, ktorá obsahuje tlačítko *Načítanie grafu*, ktoré graf preloží do textovej formy, ktorej rozumie simulátor. Po jeho stlačení sa z grafu aktualizujú textové polia pre vrcholy a hrany. Textové pole pre plán je treba zatiaľ vyplniť ručne. Pre prázdny plán je potrebné napísať 0 ako je vidno na obrázku 5.4.

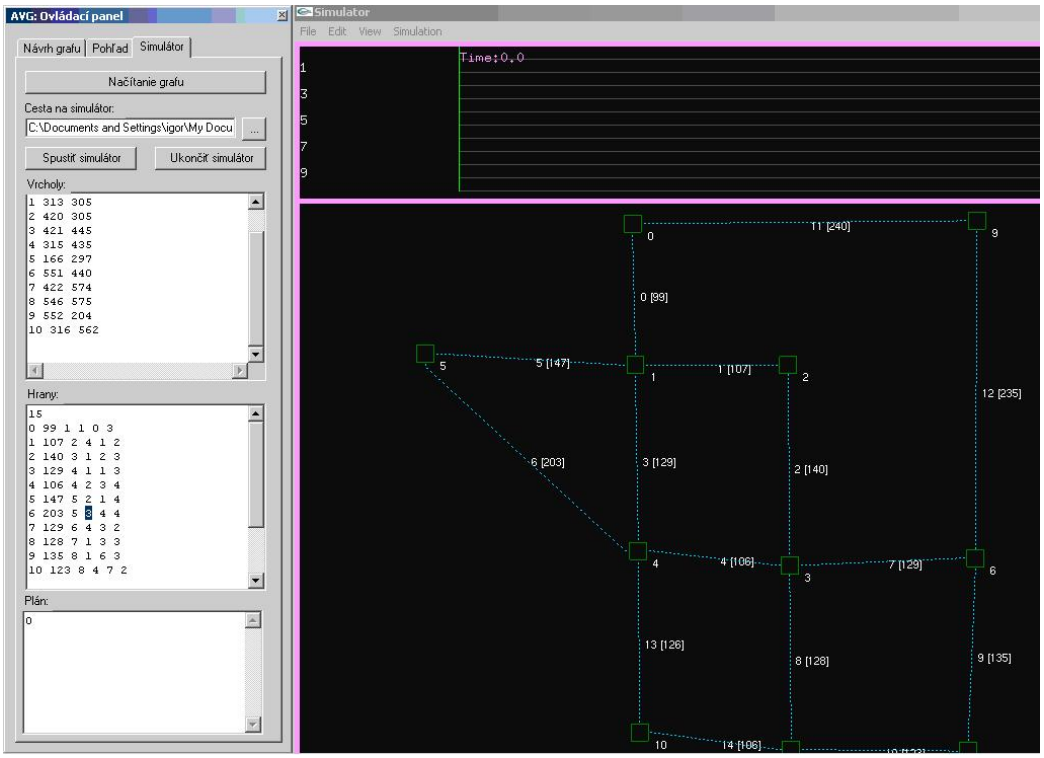



Obrázok 5.5: Načítaný graf v simulátore

Cesta na simulátor je vyplnená automaticky. Simulátor sa spúšťa tlačítkom *Spustiť simulátor* a ukončiť sa môže tlačítkom *Ukončiť simulátor*. Ak algoritmus načítania grafu vytvorí simulátoru textovú reprezentáciu grafu, ktorá hrany nespája so štvorcovými vrcholmi v miestach, v ktorých chceme, môžeme simulátor zavrieť, ručne zmeniť textovú reprezentáciu a znovu spustiť simulátor. Napríklad hranu 6 vrcholu 5 s váhou 203 (obr. 5.5) by sme

mohli napojiť na vrchol zo spodu (nie z prava).

Táto úprava je znázornená na obrázku 5.6. Simulátor sme najskôr ukončili, zmenili zvýraznenú hodnotu 2 pre hranu 6 na 3, čo znamená napojenie hrany na spodnú hranu štvorca. Nakoniec simulátor spustíme znovu. Ak sa




Obrázok 5.6: Zmena napojenia hrany na vrchol

uzly grafu volia približne do štvorcovej mriežky, algoritmus spravidla správne vypočíta pripojenie hrany na štvorcový uzol. Ciele priestorového zobrazovania chodu vozidiel, 3D simulovania a návrhu, ako aj návrhu nelineárnych trajektórií sa nám nepodarilo naplniť.

# Kapitola 6

# Modul algoritmu Free Windows

V tejto časti je stručne popísaný modul, ktorý realizuje plánovanie a smerovanie vozidiel v navrhnutom grafe. Je vysvetlené, ako sa zmenila predstava o jeho funkcii oproti tomu, čo sme uviedli v dokumentácii prototypu a dokumentácii súčasného stavu. Na konkrétnej úlohe je opäť predstavený samotný algoritmus a jeho činnosť.

## 6.1 Konečná podoba algoritmu

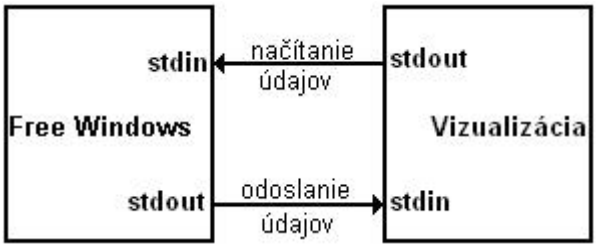
V bežiacom systéme máme nejakú aktuálnu situáciu, vozidlá sú v ňom nejakým spôsobom naplánované. Z modulu používateľského rozhrania môžu prichádzať príkazy a požiadavky od používateľa. Ak príde požiadavka na naplánovanie nového vozidla ako je uvedené v kapitole 4, potom sa aktuálna situácia odovzdá algoritmu. Bol navrhnutý určitý spôsob komunikácie medzi vizualizačným modulom a modulom pre realizáciu metódy *free windows*.

Medzi procesmi vykonávajúcimi tieto dva komunikujúce moduly sa vytvorí tzv. *rúra* (angl. *pipe*, pojem z teórie operačných systémov). Táto rúra prepojí štandardné vstupy a výstupy oboch modulov tak, ako je to znázornené na obrázku 6.1.

Po načítaní údajov o aktuálnom stave v systéme algoritmus naplánuje pre vybraté vozidlo optimálnu trasu a na svojom štandardnom výstupe poskytne informácie o tejto trase do vizualizačného modulu. Tieto informácie sa použijú v module vizualizácie, kde používateľ následne bude mať možnosť uvidieť pohyb novo naplánovaného vozidla po grafe.

## 6.2 Funkčnosť modulu

V tejto časti stručne zhrnieme, aká je súčasná funkcionalita modulu.




Obrázok 6.1: Prepojenie modulov

- načítanie a spracovanie vstupných informácií zo štandardného vstupu
- naplánovanie vozidla, ktoré sa nachádza v štartovacom uzle
- vytvorenie výstupných údajov a ich transformácia do cestovného plánu
- odoslanie údajov o cestovnom pláne pre novo naplánované vozidlo na štandardný výstup

## 6.3 Riešenie nedokončených častí

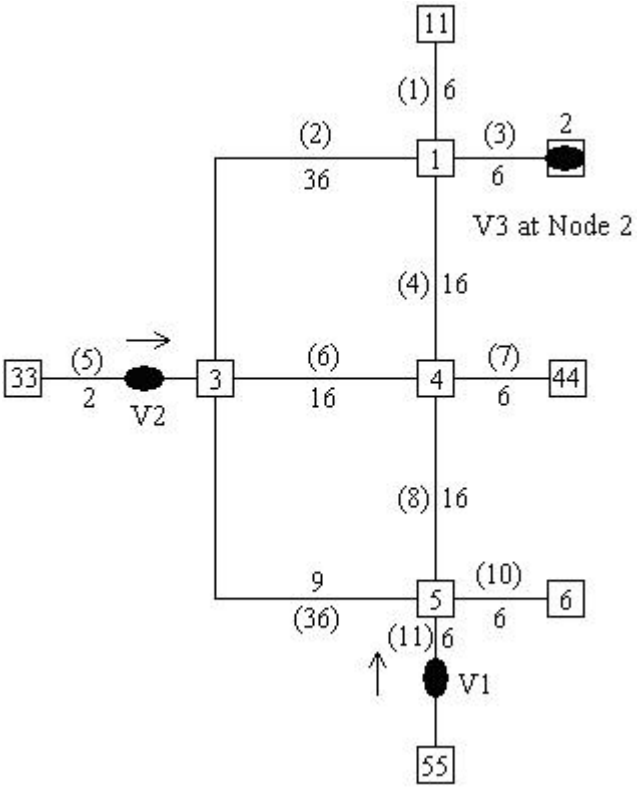
Teraz sa pozrime, ako sa podarilo vyriešiť problémy s nedokončenými časťami algoritmu uvedenými v dokumentácii k priebežnému stavu projektu.

- Problémy s reprezentáciou *nekonečna* – príklad: vozidlo je v uzle č.2 od času 0 do času nekonečno.
  - nekonečno je reprezentované ako hodnota -1
  - časti algoritmu, ktoré vyhľadávajú minimum alebo maximum z nejakého súboru hodnôt sú príslušne upravené, aby hodnotu -1 považovali za nekonečne veľkú
- Testovanie
  - boli vymyslené rôzne testovacie siete a testovali sa rôzne krajné extrémne prípady, podrobnejšie viď kapitolu o testovaní systému

## 6.4 Príklad

Tu uvedieme, ako sa algoritmus v konečnom stave vysporiada s konkrétnou úlohou. Jednať sa bude opäť o vzorový príklad z [1]. Schému vo vybratom príklade však bolo potrebné doplniť o ďalšie uzly tak, aby schéma bola

kompletná, t.j. aby sa nejednalo len o časť nejakého väčšieho systému ale o celistvý uzavretý systém. Inak povedané, v schéme sa nesmú nachádzať hrany, ktoré nevedú do žiadneho uzla. Schému sme upravili do podoby na obrázku 6.2.

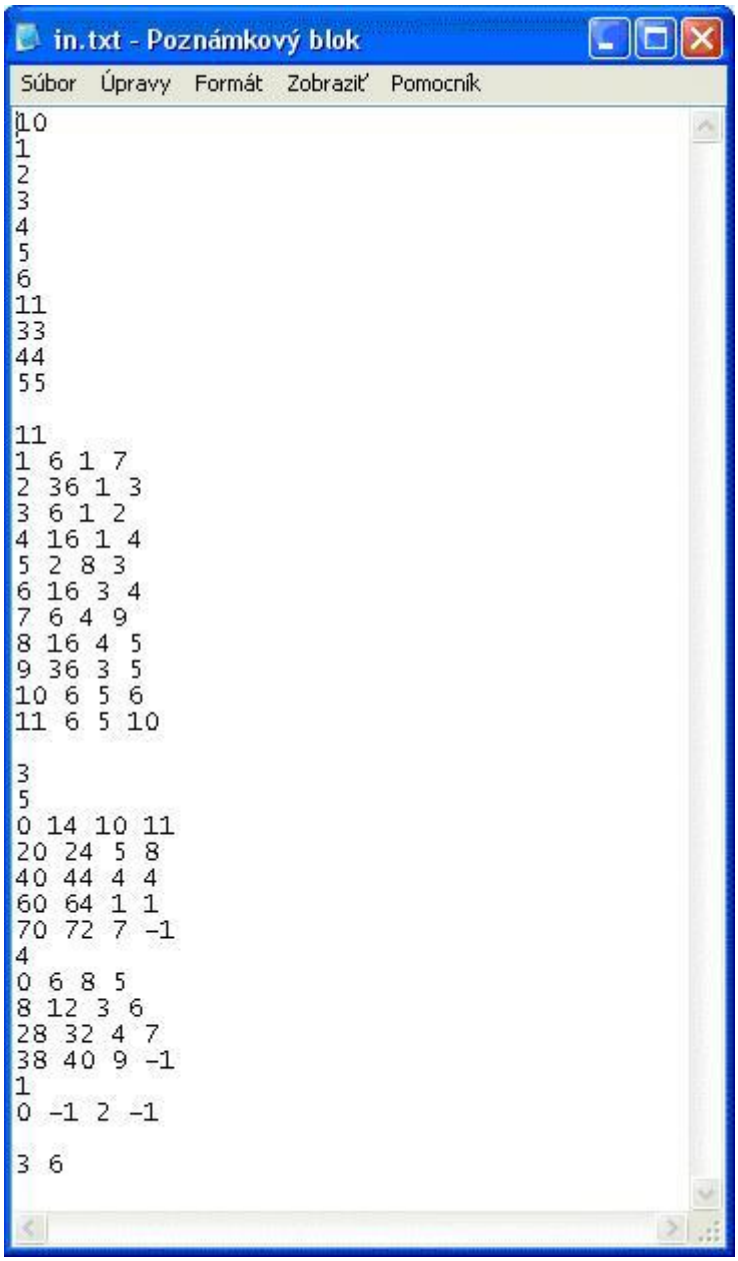



Obrázok 6.2: Upravená schéma

Pridané boli uzly 11, 33, 44 a 55. Vozidlo $V_1$ teda začne z uzla 55 a pokračuje cez uzly 5, 4, 1 a skončí v uzle 11. Vozidlo $V_2$ začne v uzle 33 a prejde cez uzly 3, 4 a dostane sa do uzla 44. Vozidlo $V_3$ blokuje uzol 2 a chceme ho naplánovať do uzla č. 6.

Povedzme, že vstupné údaje budú zadané ako cestovné plány pre vozidlá v systéme presne ako je to v [1] na strane 2. Vstupné údaje sa potom premietnu do podoby na obrázku 6.3.

Tento formát údajov bol dohodnutý interne v rámci tímu. Samozrejme, tu sú údaje zapísané vo forme súboru kvôli názornosti, aby bolo čitateľovi zrejmé, aké postupnosti údajov program cez svoj štandardný vstup načítava. Na načítavanie údajov sa používa príkaz `scanf("format", &premenna)`. Parameter `"format"` bude typu `"%d"` , pretože načítavané údaje sú typu in-




Obrázok 6.3: Vstupné údaje

teger. Uvedená reprezentácia vstupných údajov slúži iba na testovacie účely pre samotný algoritmus free windows bez jeho prepojenia s ostatnými časťami systému.

Prvý údaj znamená počet uzlov v systéme. Nasleduje číslovanie uzlov,

aké je použité v schéme. Prvý uzol sa volá '1', druhý '2' a tak ďalej, až desiaty uzol sa volá '55'. Táto transformácia je dôležitá pre zefektívnenie chodu algoritmu. Nasleduje číslo udávajúce počet hrán v systéme a následne údaje o jednotlivých hranách:

- číslo hrany
- váha hrany
- označenie dvoch uzlov, ktoré sú hranou prepojené

Ďalej pokračujeme udaním počtu cestovných plánov pre vozidlá. Plány sú asociované s vozidlami po poradí, t.j. prvý plán je automaticky pre prvé vozidlo, druhý pre druhé vozidlo atď.

Nasledujú samotné cestovné plány:

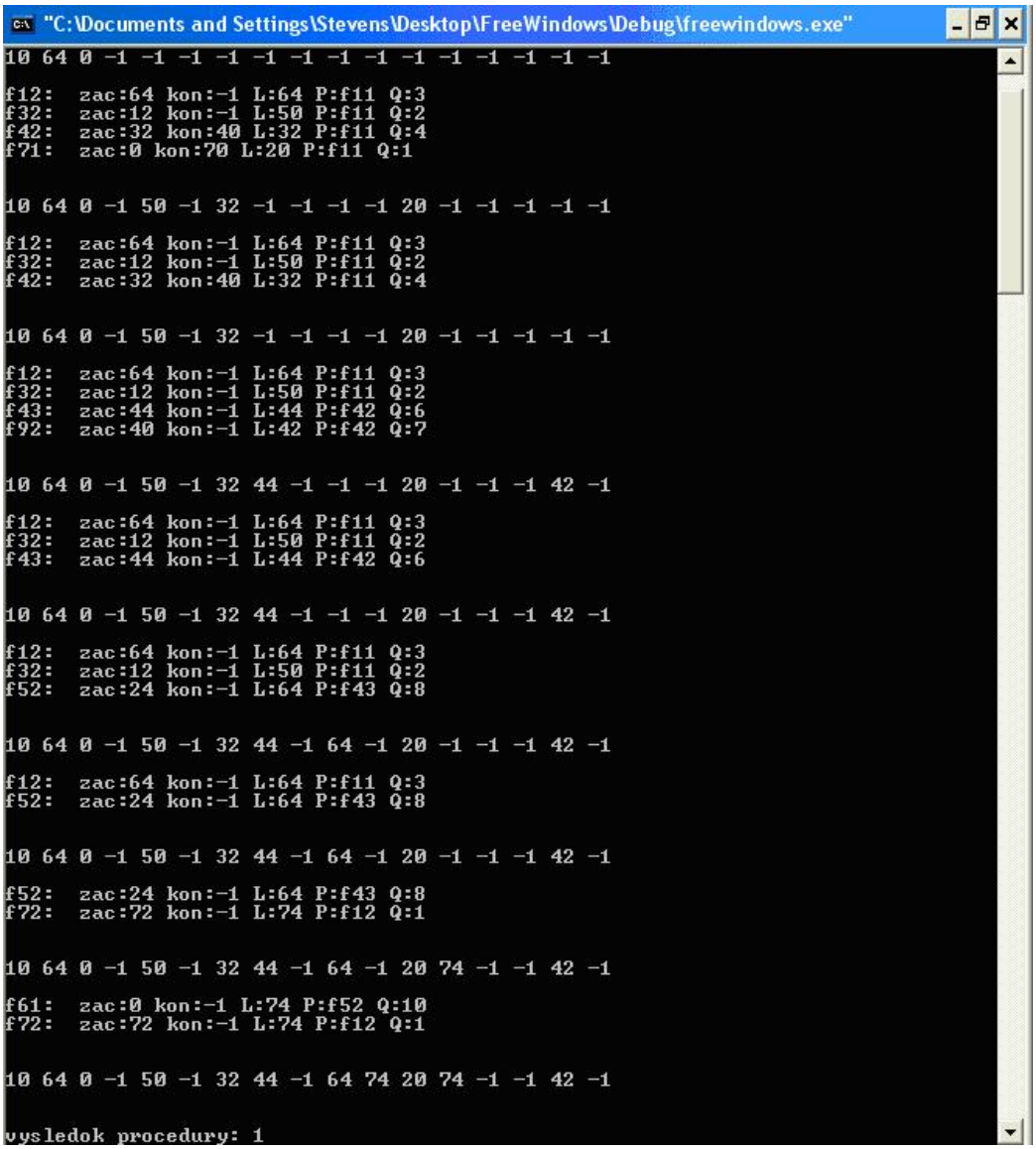
- počet riadkov plánu
- čas príchodu vozidla do uzla
- čas odchodu vozidla z uzla (ak sa tento údaj rovná -1, potom to znamená, že vozidlo má tento uzol obsadený pre seba donekonečna)
- číslo uzla
- číslo hrany, po ktorej sa vozidlo bude pohybovať ďalej (ak sa tento údaj rovná -1, potom to znamená, že vozidlo sa už nebude ďalej v systéme pohybovať, čiže je to jeho cieľový uzol, a v ňom zmizne zo systému, čo môžeme v reálnom výrobnom systéme chápať ako jeho odstavenie na pomocnú koľaj na dobýjanie batérie, servis a pod.)

Posledné dva údaje znamenajú číslo vozidla, ktoré je potrebné naplánovať a cieľový uzol, kam sa má vozidlo dostaviť.

Presne podľa príkladu v [1], chceme naplánovať vozidlo $V_3$, ktoré čaká v uzle 2 a chce ísť do uzla 6. Výstup algoritmu na konzolu je na obrázku 6.4.

Na obrázku vidíme postup algoritmu od iterácie č. 2 až po poslednú iteráciu č. 9. Vo výpise máme možnosť vidieť, ako algoritmus zisťuje o príslušných voľných časových oknách ich labely a hodnoty P a Q a konštruuje tak postupne minimálnu cestu pre plánované vozidlo. Vo výpise tiež vidíme po každej iterácii aj príslušný riadok tabuľky podobnej, ako sa konštruuje v [1].

Pridaním štyroch nových uzlov do schémy sa nám ale objavili nové voľné časové okná (konkrétne $f_7^1, f_7^2, f_8^1, f_9^1, f_9^2, f_{10}^1$). Ich labely sú posledné šesťčíslie vo výpisoch. Napríklad v iterácii č. 2 vidíme, že algoritmus zvažuje aj



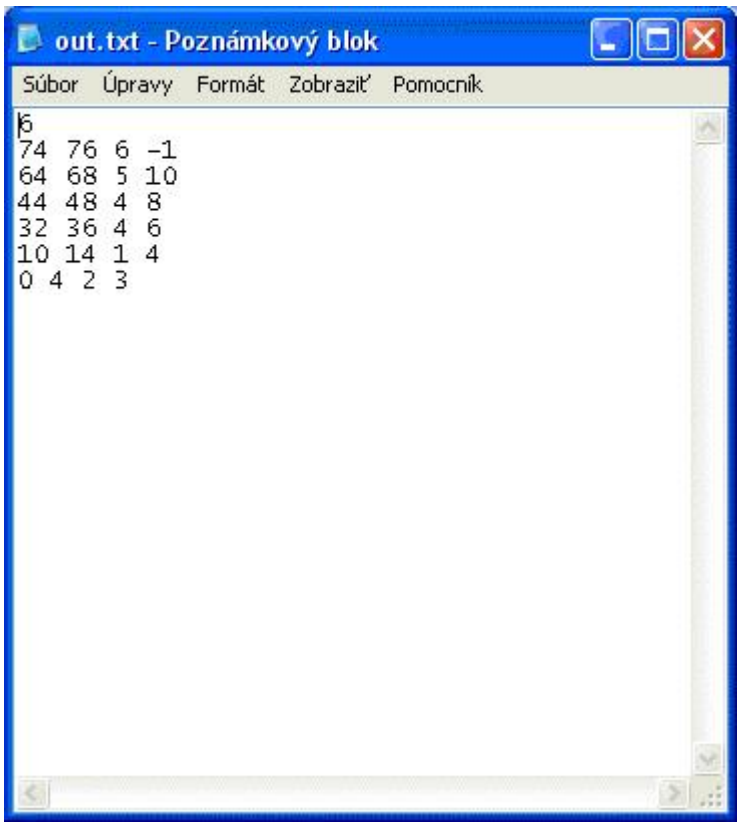

Obrázok 6.4: Výstup algoritmu

možnosť ísť cez uzol č. 7 (voľné okno sa vo výpise volá $f_7^1$) – jeho označenie je 11 a vypočíta mu label s hodnotou 20. Podobne v ďalších krokoch algoritmu získali labely aj niektoré z ďalších pridaných okien (napr. $f_9^2$ má label 42 a podobne). Taktiež vidíme, že pridanie nových uzlov nám spôsobilo predĺženie vykonávania algoritmu o 2 iterácie. Symbol nekonečna nám reprezentuje číslo -1.

Porovnaním výstupov v jednotlivých krokoch podľa riešenia v [1] vidíme,

že navrhnutý algoritmus dosahuje totožné výsledky plus navyše zahŕňa aj nové vzniknuté voľné časové okná.

Na poslednom riadku výpisu vidíme informáciu, že algoritmus skončil s výsledkom 1, čo znamená, že sa našla cesta pre vozidlo $V_3$. Táto cesta (plán) má nasledovnú podobu (obr. 6.5) zapísanú do formy výstupného súboru pre účely kontroly:

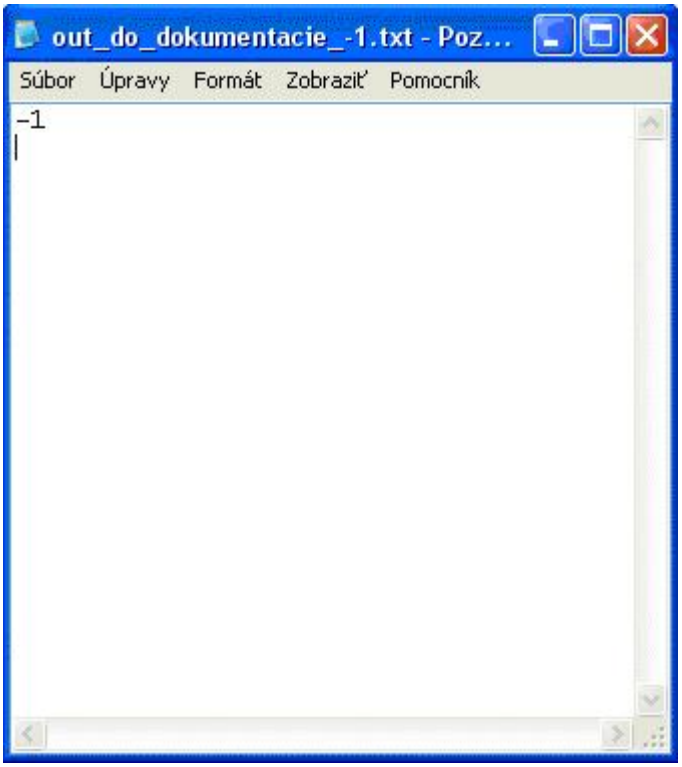



Obrázok 6.5: Plán

Na prvom mieste je číslo udávajúce počet riadkov výsledného cestovného plánu. Na ďalších riadkoch sú potom v takom istom formáte ako vo vstupných dátach zaznamenané jednotlivé zastávky vozidla na jeho ceste. To, že položky plánu sú chronologicky zotriedené opačne, vizualizačnému modulu nijako neprekáža. Zapisovanie na štandardný výstup sa bude realizovať použitím príkazu `printf("format", premenna)`. Parameter `"format"` bude zrejme `"%d"`, nakoľko zapisujeme celé čísla typu integer. V každom takomto pláne bude posledný údaj o ďalšej hrane v tvare -1. Toto bolo dohodnuté v rámci tímu pre zjednodušenie samotného algoritmu, jeho odbremenenie.

Totiž: keďže, ako bolo uvedené v kapitole 4, vizualizačný modul umožňuje zadávať rôzne modely správania sa vozidla po príchode do cieľového uzla, bude jeho úlohou správne zobrazenie tohto správania sa v zobrazovanom modeli na obrazovke. Algoritmus *free windows* sa do tohto problému nebude 'miešať'.

Ak sa vozidlo nepodarí naplánovať, výstup bude vyzerať nasledovne (zapísaný opäť v tvare súboru pre účely kontroly obr. 6.6):




Obrázok 6.6: Naplánovanie sa nepodarilo

Je zrejmé, že takýto výstup dosiahneme použitím príkazu `printf("-1\n");` kde symbol `"\n"` znamená zápis nového riadka.

Algoritmus komunikuje s vizualizačným modulom cez svoj štandardný vstup a výstup, ale vzhľadom na potreby tohto dokumentu bolo vhodné a potrebné uviesť názorný výstup do súborov resp. na konzolu.

# Kapitola 7

# Zhodnotenie

Tu uvedieme celkové zhodnotenie výsledkov dosiahnutých v rámci dvoch semestrov na predmetoch *Tímový projekt I* a *II*, poukážeme na prípadné nedostatky vytvoreného produktu a načrtneme možné cesty, po ktorých by sa mohli vydať prípadní záujemcovia o vylepšenie tohoto systému.

Cieľom predmetov *Tímový projekt I* a *II* bolo získať skúsenosti s tímovou prácou na projektoch väčšieho rozsahu z oblasti počítačových systémov a sietí. Ďalej bolo cieľom naučiť sa pracovať v tíme, dorozumieť sa v rámci tímu s ostatnými kolegami a zadeliť si úlohy medzi jednotlivých členov tak, aby každý vytvoril jednu časť celého systému, ktorá by bola porozumiteľná a ľahko modifikovateľná pre ostatných členov tímu. Nakoniec globálnym cieľom bolo vytvorenie produktu – počítačového programu – aplikácie, ktorá spĺňa požiadavky zadania projektu.

Zhodnoťme teraz, či v rámci nášho tímu došlo k splneniu spomínaných cieľov. Je zrejmé, že rozsah daného projektu bol značný. Preto bolo nutné rozumne rozdeliť celý problém na menšie časti, ktoré sme nazvali modulmi. Každý člen tímu začal s riešením odlišnej úlohy a postupne sme sa pokúšali jednotlivé moduly pospájať do funkčného celku, pričom vždy bolo nutné riešiť aj pomocné úlohy ako tvorbu dokumentácie a udržovanie aktuálnej web stránky o stave riešenia projektu. Z uvedených dôvodov môžme tvrdiť, že členovia nášho tímu počas práce na projekte získali určité skúsenosti s prácou v tíme, najmä pri vzájomnej komunikácii vo fáze spájania modulov. Komunikácia v tíme ani s odborným vedúcim projektu pánom doc. Hrúzom nepredstavovala problém, nakoľko vzťahy v tíme boli priateľské a bezproblémové. Zadelenie úloh taktiež nebolo problémom, pretože vyplynulo z preferencií jednotlivých členov. Inými slovami, každý sa pustil do toho, v čom je dobrý. Nie všetky vytvorené moduly však boli zrozumiteľné iným členom tímu. Nevznikla totiž potreba, aby členovia tímu 'videli' do modulov, ktorých neboli autormi. Pri náročnosti absolvovaných semestrov na to ani

nebol čas. Zhodli sme sa, že prioritou je vytvorenie funkčného programu. Tento globálny cieľ sa nám v požadovanom dohodnutom rozsahu podarilo splniť, čo je najdôležitejším výsledkom projektu.

Teraz zhodnotíme, či boli dosiahnuté ciele uvedené v zadaní projektu.

Bod č. 1: analýza dopravných systémov používaných vo výrobe
Bod č. 2: rozbor modelovacích prostriedkov

Týmto problémom sme sa venovali v prvom semestri riešenia projektu a výsledky sú zdokumentované v kapitolách 1 až 3.

Bod č. 3: spracovať metódu modelovania dopravných systémov

Po splnení bodov 1 a 2 sme rozhodli o spôsobe, akým budeme systémy s AGV modelovať. Podrobnejšie v kapitolách 4 až 6.

Bod č. 4: navrhnúť riadenie

Už v začiatkoch riešenia projektu bola zvolená metóda free windows. Bližšie viď prílohu.

Bod č. 5: vytvoriť programový systém na modelovanie a riadenie

Cieľ nášho snaženia, systém je zdokumentovaný v kap. 4 až 6 a nachádza sa v spustiteľnej podobe na priloženom CD médiu.

Za nedostatok vytvoreného systému môžeme považovať jednoduchosť použitého algoritmu na plánovanie vozidiel, ktorý bol použitý vo svojej základnej podobe, ktorá má isté obmedzenia. Bližšie v [1]. Keďže však išlo o úplne nový problém, ktorý ešte v rámci tímových projektov nebol doteraz riešený, môžeme dosiahntý výsledok považovať za veľmi uspokojivý. Grafické rozhranie modulu na tvorbu grafu a rozhranie samotnej aplikácie na modelovanie systému sú na dobrej úrovni a poskytujú používateľovi istý komfort pri používaní.

Ako možným vylepšením systému sa samozrejme javí prepracovanie smerovacieho algoritmu, zapracovanie vylepšení doňho. V rôznej dostupnej literatúre a odborných článkoch sa nachádzajú štúdie o pokročilých smerovacích algoritmoch, ktoré istými spôsobmi vylepšujú vlastnosti použitej metódy, čím zlepšujú možnosti riadenia a kontroly nad systémami s AGV.

# Kapitola 8

# Riadenie projektu

Táto časť dokumentácie obsahuje informácie o riadení projektu, o zdokumentovaní postupného vývoja softvérového systému spolu so sprievodnou dokumentáciou počnúc ponukou, rozdelením úloh v tíme a záznamami zo stretnutí. Kapitola končí obojstrannými posudkami našeho a konkurenčného tímu na časti hrubého návrhu a analýzy, prototypu a dokumentácie, ktoré sú doložené preberacími protokolmi.

## 8.1 Ponuka

### 8.1.1 Predstavenie tímu

Členovia tímu majú základy Petriho sietí, s ktorými sa stretli na predmete *špecifikačné a opisné jazyky*.

**Bc. Štefan Krištofík** – Absolvoval bakalárske štúdium na FIIT STU v Bratislave v odbore Počítačové systémy a siete. Pred štúdiom na vysokej škole získal skúsenosti s programovacími jazykmi Pascal a assembler, kde sa zaoberal riešením jednoduchých úloh z oblasti automatizácie. Počas doterajšieho štúdia na vysokej škole získal zručnosti s programovacími jazykmi C a C++ s použitím knižnice MFC. Ďalej pochopil základy modelovania a simulácie jednoduchých systémov pomocou nástroja EGRET a základy navrhovania jednoduchých digitálnych obvodov a systémov. V rámci svojho bakalárskeho projektu vytvoril systém na automatický návrh sériovo-paralelných kontaktovo-diódových (1,m)-pólov, čo sú špeciálne druhy hradlových logických obvodov. V ďalšom štúdiu sa bude zaoberať návrhom pokročilých digitálnych systémov a udalostnými systémami.

Kontakt: `zastaph@pobox.sk`

**Bc. Stanislav Panák** – Absolvoval bakalárske štúdium na Fakulte informatiky a informačných technológií STU v Bratislave v odbore Počítačové inžinierstvo, študijný program Počítačové systémy a siete. Programovaniu sa venoval už na strednej škole pri práci s štruktúrovaným programovacím jazykom Pascal, neskôr objektovým Delphi. Štyri roky programuje v jazyku C/C++ a tieto skúsenosti využil pri pri implementácií zadaní, ako aj bakalárskeho projektu. Počas štúdia nadobudol skúsenosti s programovaním v ďalších jazykoch ako assembler a Java. Má tiež skúsenosti s tvorbou web stránok a web aplikácií využívajúcich html, css, php a javascript.

Kontakt: `stanley001@zoznam.sk`

**Bc. Filip Lörinc** – Absolvoval bakalárske štúdium na FIIT STU v Bratislave v odbore Počítačové systémy a siete. Počas žiackych čias sa oboznámil s technológiami HTML, CSS, JavaScript a PHP. Tieto vedomosti rokmi zlepšoval, čoho výsledkom bolo aj niekoľko menších webových projektov. Taktiež má základne vedomosti z jazykov C++ a Java, tvorivý prístup, neobvyklé myšlienky či postupy, ktoré s horeuvedenými informáciami majú potenciál uplatniť sa pri riešení daného zadania.

Kontakt: `filip.lorinc@gmail.com`

**Bc. Igor Sečanský** – Absolvoval bakalárske štúdium na FIIT STU v Bratislave v odbore Počítačové systémy a siete. Absolvoval predmet Modelovanie a simulácia, kde pomocou nástroja EGRET a C# knižnice Simulib modeloval posielanie správ v prostredí zjednodušeného Ethernetu metodou CSMA/CD a naprogramoval jednoduchý model dopravného prúdu v jazyku C#.

Kontakt: `igor.secansky@gmail.com`

### 8.1.2 Motivácia

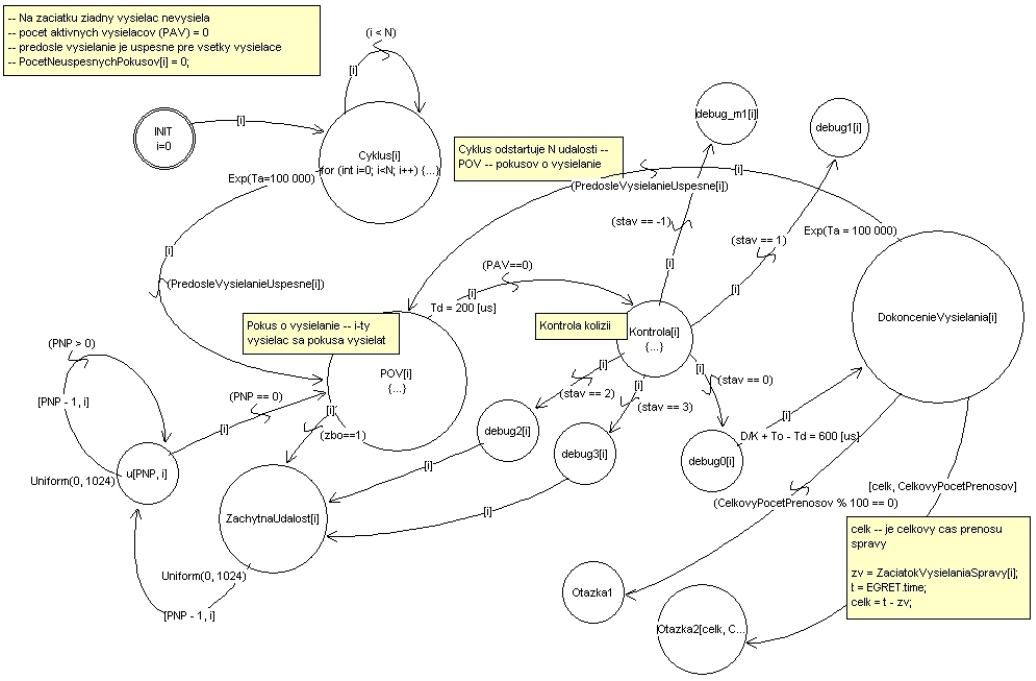
Motiváciou pre výber tejto témy tímového projektu boli hlavne nadobudnuté skúsenosti niektorých členov tímu z predmetu Modelovanie a simulácia v predošlom štúdiu a vízia toho, že nami vytvorený systém sa v prípade veľmi dobrej kvality môže aj prakticky uplatniť vo výrobe. Ďalším faktorom bola aj blízkosť témy modelovania niektorým členom tímu, ktorí majú záujem sa odborne ďalej profilovať v tejto oblasti. Pre ostatných členov je to zároveň príležitosť vniknúť do novej, nepoznanej odbornej témy a rozšíriť si tak obzory.

Bude potrebné sa oboznámiť s rôznymi novými témami, medzi ktoré patria napríklad úloha vozidiel AGV vo výrobných procesoch, metódy modelovania systémov so zónovo riadenými dopravnými prostriedkami, riadenie systémov s využitím takýchto dopravných prostriedkov a pod. Bude to iste proces veľmi náročný na čas, ale pokúsime sa vniknúť do problematiky čo najrýchlejšie, aby sme mohli pristúpiť k tvorbe samotného systému na modelovanie a riadenie spomínaných vozidiel.

S ohľadom na vyššie uvedené si za cieľ projektu stanovujeme jednoznačne vytvoriť originálne riešenie systému na modelovanie a riadenie vozidiel AGV.

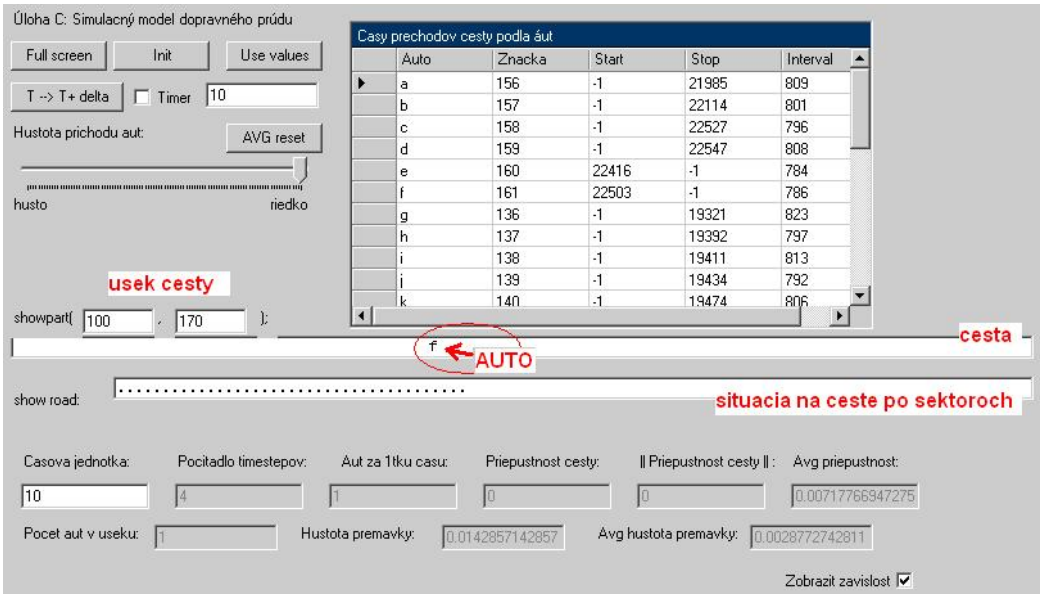
### 8.1.3 Čo môžeme poskytnúť

Aj keď problematika automaticky navádzaných vozidiel je pre nás nová a s Petriho sieťami sme sa stretli len veľmi okrajovo, s modelovaním pomocou udalostí máme isté skúsenosti, ktoré sme získali napríklad pri implementácii modelu lokálnej siete, kde sme pomocou nástroja *EGRET* a knižnice *SimuLib* modelovali fungovanie zjednodušeného Ethernetu s preposielaním správ podľa metódy CSMA/CD obr. 8.1. Ďalšie skúsenosti sme získali pri modelovaní




Obrázok 8.1: EGRET – Graf udalostí modelu lokálnej siete

dopravného prúdu obr. 8.2.




Obrázok 8.2: Model dopravného prúdu

### 8.1.4 Predpokladané zdroje

Členovia tímu predpokladajú, že na riešenie bude postačovať štandardné hardvérové a softvérové vybavenie dostupné napríklad z *MSDNA* ako je napríkad Visual Studio .Net 2005. Ďalej predpokladáme použiť modelovací nástroj EGRET, knižnicu SimuLib, prípadne ďalšie nástroje, ktoré budú uvedené na tomto mieste.

## 8.2 Plán projektu

- Do 4. týždňa – podrobné oboznámenie sa s témou a naštudovanie príslušných materiálov, vytvorenie web stránky tímu, názvu (prípadne iných symbolov) tímu, vytvorenie podrobnejšieho plánu projektu na zimný semester a následné rozdelenie úloh v tíme.

- Od 5. týždňa do konca zimného semestra – analýza a špecifikácia problému a požiadaviek, postupná tvorba požadovanej dokumentácie a ďalšie úlohy súvisiace s tvorbou prototypu, prezentácia.

- Letný semester – zdokonalenie prototypu, ladienie programu, odstaňovanie chýb, dopĺňanie modulov, údržba web stránky s aktuálnymi

dokumentami, prepojenie jednotlivých častí do funkčného celku, testovanie, vytvorenie dokumentácie.

## 8.3 Úlohy členov tímu

**Štefan Krištofík** – Vedúci projektu. Metóda Free Windows. Plánovanie vozidiel.

**Stanislav Panák** – Vizualizácia. Využitie OpenGL, C, C++, MFC. Preskúmanie možností nástroja Citect. Vytvorenie modulu pre simuláciu vozidiel.

**Filip Lörinc** – Webmajster. Má na starosti webovú stránku projeku. Dopĺňa aktuálne informácie o stave riešenia projektu a zápisy zo stretnutí.

**Igor Sečanský** – Vizualizácia. Preskúmanie možnosti využitia nástrojov a technológií Egret, SimuLib, C# a CsGL. Dokumentácia pomocou systému LaTeX. Vytvorenie modulu pre návrh grafu trajektórií.

## 8.4 Záznamy zo stretnutí

Stretnutia sa uskutočňujú v pondelok o 16:00 v softvérovom štúdiu. Členovia tímu spravidla referujú o tom, kto čo spravil a aké mal prípadné problémy. Ďalej vedúci tímu koordinuje, čo kto bude robiť do budúceho stretnutia.

1. stretnutie [15.10.2007 16:00] – Prítomní všetci okrem Štefana Krištofíka - skolila ho choroba. Pán docent Hrúz nás voviedol do problematiky AGV. Porozprávali sme sa o našom probléme z celkového pohľadu, pričom sme si uvedomili, že bude nutné tento veľmi obsiahly problém orezať a vybrať si z neho určitú časť, ktorú si bližšie ešte špecifikujeme a budeme ju nasledovne riešiť. Definovali sme si určité pravidlá, čo sa týka výrobného vozíkového problému. Budeme predpokladať induktívne trasy s navázdaním, pričom tieto trasy sa budú deliť do zón. Jednotlivé časti trasy budú mať 2 senzory. To znamená, že na začiatku každej časti trasy bude jeden senzor a na konci druhý senzor, ktorý nám bude indikovať aktuálnu pozíciu vozíka. Uvažovali sme aj o lokalizácii vozíkov skrz kamerový systém, avšak vzhľadom na zložitosť problému sme sa na 99% rozhodli, že senzory sú predsa len výhodnejšie.

   Povedali sme si aj to, že do systému budú v priebehu času prichádzať požiadavky (napr. nech sa posunie vozík atď.), ktoré sa budú vykonávať tak ako prišli (FIFO) s tým, že ak sa nepodarí vybaviť nejakú

požiadavku (v rozumnom čase), tak sa požiadavka presunie na koniec radu.

Dohodli sme sa, že budeme uvažovať 3 typy uzlov: parkovací uzol, spracovateľský uzol a ostatný uzol. Všetky tieto množiny uzlov sú vzájomne disjunktné. Detaily budú upersnené neskôr.

2. stretnutie [22.10.2007] – neuskutočnilo sa – mali sme všetci, vrátane odborného vedúceho, dovolenku. Dohodli sme sa, že náhradné stretnutie sa uskutoční v niektorý pondelok, predĺžením plánovaného stretnutia.

3. stretnutie [29.10.2007 16:00] – Prítomní: všetci okrem Filipa Lörinca - choroba. Na tomto stretnutí nám Stanislav ukázal a zhruba vysvetlil Petriho siete. Následne nám aj ukázal program PND (Petri Net Designer). Igor nám ukázal program Egret a načrtol možnosti vizualizácie systému v OpenGL. Štefan stručne analyzoval metódu Free windows a je predpoklad, že do ďalšieho stretutia dokončí podrobne analýzu v spolupráci s Filipom. Celkovo sme rozoberali rôzne otázky ohľadne systému ako celku, ako si ho predstavujeme. Rozmýšlali sme nad použitím predvedených programov a uplatnení metódy Free Windows. Následné úvahy viedli k forme a obsahu dokumentácie. Do dokumentácie zahrnieme popis metódy Free Windows, ktorú sme sa rozhodli použiť, analýza problému bude obsahovať popis riešeného systému a opis problémov, ktoré tu vystupujú (topológia, vozidlá, plánovanie...) a v časti špecifikácia požiadaviek rozoberieme podrobné požiadavky, ktoré budú na náš systém kladené.

4. stretnutie [5.11.2007 16:00 - 16:15] – Prítomní všetci. Toto stretnutie sme odložili na stredu 17.11. vzhľadom na fakt, že väčšina tímu mala nasledujúci deň veľmi dôležitú písomku. Napriek tomu sme si v krátkej chvíli stihli pozriet dva návrhy vizualizácie pomocou OpenGL (Stanislav, Igor). Stanislavov návrh (C/C++, MFC, OpenGL) obsahoval model topológie dráh reprezentovaných grafom, po ktorých sa pohybovali vozidlá reprezentované bodmi. Igorov návrh (C#, CsGL, Egret, SimuLib) obsahoval základný návrh frameworku pre vizualizáciu vozidiel so zoomom a transláciou scény. Navrhoval využiť existujúci modelovací nástroj Egret na namodelovanie topológie dráh pomocou grafu udalostí a pomocou C#, CsGL, SimuLib rozšíriť Egret o špecifické črty, ktoré potrebujeme pri vizualizácii premiestňovania vozidiel (napríklad spojité presúvanie viacerých vozidiel naraz, pohyb po krivkách).

5. stretnutie [7.11.2007 14:14 - 15:40] – Náhradné stretnutie z 5.11. Prítomní všetci okrem Igora, ktorému nevyhovoval daný čas. Na tom-

to stretnutí nám Štefan odprednášal polovicu metódy Free Windows. Druhá časť prednášky bude v pondelok.

6. stretnutie [12.11.2007 16:00] – Štefan dokončil druhú polovicu prezentácie metódy Free Windows, Stanislav prezentoval použitie a možnosti modelovacieho nástroja Citect, Filip spravil webovú stránku tímového projektu, umiestnil na ňu aktuálne informácie o stave projektu so zápismi zo stretnutí. Igor spracoval existujúce dokumentačné výstupy pomocou typografického vysádzacieho systému LaTeX.

7. stretnutie [22.11.2007 16:02 - 16:46] – Prítomní všetci. Na tomto stretnutí sme boli pochválnení našim odborným dozorom za zaujímavé čítanie v rámci priebežnej správy, ktorú sme odovzdali. Následne na to prebiehali interné diskusie na tému posudok. Zjednotili sme naše množiny výčitiek a následne sme sa rozišli s tým, že finálnu verziu posudku dokonzultujeme virtuálne. Výsledok si môžete prezriet v dokumentoch.

8. stretnutie [26.11.2007] – Prítomní nikto. Pán doc. Hrúz bol chorý, tak sa naše stretnutie neuskutočnilo.

9. stretnutie [3.12.2007 16:02 - 16:45] – Prítomní všetci. Zhovárali sme sa o úrovni nepochopenia vlastnej práce konkurenčného tímu. Co do práce napísali a evidentne čomu neporozumeli, kedže niektoré matematické zápisy neboli správne, respektíve boli nekompletné. Následne sme si ukázali Stanislavov pokrok v rámci vizualizácie. Potom sme sa rozišli.

10. stretnutie [10.12.2007] – Prítomní nikto. Vzdhľadom na fakt, že sme mali tažkú písomku, naše stretnutie bolo preložené na 17.12 - 17:00

11. stretnutie [17.12.2007 17:04] Prítomní všetci. Naše stretnutie trvalo krátko. Pozreli sme si naše výtvory, porozprávali sme sa a dohodli si zbežne termín prezentácie prototypu.

12. Prezentacné stretnutie [29.1.2008 10:33 - 11:40] – Prítomní všetci (aj tí druhí). Zišli sme sa spolu všetci v kruhu rodiny ako jeden tím, aj ked sme tam boli vlastne ako dva tímy. Ale čo na tom záleží, my spolu nesúperíme. Naše vzťahy sú kolegiálne aj keď sa vlastne nepoznáme. A aj o tom bolo naše prezentacné stretnutie. Spoznať sa a najmä spoznať naše potencionálne produkty. Nevedeli sme sa dohodnúť, že kto bude druhý prezentovať, tak sme si hodili mincou a vyhrali sme. Čiže sme šli prezentovat ako prví. Svoje sme si odprednášali a následne aj tím č. 5. Potom sme ešte trochu diskutovali a následne sme sa rozišli do rôznych

kútov sveta. Našu prezentáciu ako aj dokumentáciu k prototypu si je možné stiahnut v položke *Dokumenty*.

13. stretnutie [25.2.2008 17:03 - 17:28] – Prvé letné stretnutie. Prítomní všetci. Naše prvé stretnutie v letnom semestri sa nieslo v krátkom duchu. Naše úlohy na jednotlivých častiach ostali nezmenené a preto sme sa po krátkej konverzácií rozišli.

14. letné stretnutie [3.3.2008 17:05 - 17:55] – Prítomní všetci. Na tomto stretnutí sme začali vážne diskutovat o prepojení už existujúcich modulov celej aplikácie. Riešili sme aj to, akou formou bude interpretovaný graf dráhy a podobne. Ku konkrétnym záverom sme zatiaľ neprišli. Budeme vás i naďalej informovať o ďalšom progrese.

15. stretnutie [10.3.2008 17:00 - 17:33] – Prítomní všetci. Na tomto stretnutí, tak ako aj na ďalších, prebieha interná diskusia ohľadom dokončenia jednotlivých modulov.

16. stretnutie [17.3.2008] – Prítomní nikto. Vzhľadom na študijný nátlak sa toto stretnutie neuskutočnilo.

17. stretnutie [24.3.2008 17:11 - 17:49] – Prítomní všetci. Vzhľadom na pokročilý stav projektu, konzultácie s odborným vedúcim sa nesú už len v duchu informovania o aktuálnom stave, pretože už usmernenie v rámci projektu nie je nutné.

18. stretnutie [24.3.2008 17:01 - 17:33] – Prítomní všetci. Prebiehala interná diskusia tak ako po minulé stretnutia. Na zadaní pracujeme.

19. stretnutie [24.3.2008 17:02 - 17:22] – Prítomní všetci. Po krátkej internej rozprave sme sa rozhodli, že pôjdeme radšej domov robiť niečo užitočné.

20. stretnutie [31.3.2008 17:07 - 17:50] – Prítomní všetci. Prebiehala interná diskusia tak ako po minulé stretnutia. Intenzívne sa sme sa orientovali na implementáciu projektu.

21. stretnutie [7.4.2008 17:07 - 17:50] – Prítomní všetci okrem Filipa. Čas odovzdania sa blíži a preto sa tlak na naše mozgové membrány stupnuje a tým aj naše odhodlanie dokončiť projekt intenzívnymi internými konzultáciami. Nič nové, ale tak čo viac k tomu povedať.

22. stretnutie [14.4.2008] – Prítomní nikto. Odborný vedúci nemohol prísť, preto sme sa nestretli a riešili sezónne záležitosti.

23. stretnutie [21.4.2008 17:01 - 17:55] – Prítomní všetci. Náš výsledný produk už navonok vyzerá pekne, stále však pracujeme na jeho internom dokončení.

24. stretnutie [28.4.2008 17:00 - 17:37] – Prítomní všetci. V sekcii dokumenty je náš aktúalny stav bádania. Dokončujeme implementáciu a bavíme sa o veciach s tým súviasiacimi.

25. stretnutie [5.5.2008 17:09 - 17:47] – Prítomní všetci. Na tomto poslednom stretnutí sme doťahovali nedotiahnuté, celkovo toto stretnutie sa ponášalo na tie minulé takže nie je o com už velmi referovať.

# Dodatok

# A Metóda Free Windows

# Preberacie protokoly

# Posudky

# Vyjadrenia k posudkom

# Literatúra


[1] Chang W. Kim, J. M. A. Tanchoco: *Conflict-free shortest-time bidirectional AGV routing*, Int. journal Prod. Res. 1991, Vol. 29, No. 12 2377-2391

[2] Branislav Hrúz: *Solution of the Manufacturing Transport Control Using Petri Nets*, IEEE Conference on Systems, Man and Cybernetics, Washington DC, October 2003

[3] Qiu Ling, Hsu Wen-Jing: *Scheduling and Routing Algorithms for AGVs: a Survey*, 12 October 1999, Technical Report: CAIS-TR-99-26, Centre for Advanced Information Systems, Nanyang Technological University

[4] `http://en.wikipedia.org/wiki/Automated_Guided_Vehicle`

[5] Yoonho Seo, Pius K. Egbelu: *Integrated manufacturing planning for an AGV-based FMS*, Eslevier, International journal of production economics, 1999

[6] Murata, T.:, *Petri Nets: Properties, Analysis and Applications* an invited survey paper, Proceedings of the IEEE, Vol.77, No.4 pp.541-580, April, 1989.

[7] Bieliková, M.: *Ako úspešne vyriešiť projekt*, 1. vyd. Bratislava 2000. 158 s. ISBN 80-227-1329-5