Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

Simulated Car Racing Competition 2011

Dokumentácia k projektu

Tím číslo: 20
Vedúci projektu: Ing. Peter Vilhan
Predmet: Tvorba softvérového systému v tíme
Študijný program: Informačné systémy/Softvérové inžinierstvo
Akademický rok: 2010/2011

Bc. Maroš Bednár
Bc. Adam Brček
Bc. Marek Briš
Bc. Marián Florek
Bc. Vojtech Juhász
Bc. Ivan Valenčík
Bc. Juraj Kosmeľ

Obsah

0. Úvod

0.1 Účel a rozsah dokumentu

Predkladaný dokument obsahuje dokumentáciu k tímovému projektu *Simulation racing car*, ktorý riešime v rámci predmetu Tímový projekt. Forma dokumentu zodpovedá pravidlám a požiadavkám spomenutého predmetu. Obsahuje analýzu, špecifikáciu a návrh riešenia, neskôr pribudne aj opis implementácie, testovania a používateľská príručka. Finálna verzia dokumentu bude mať predpokladaný rozsah niekoľko desiatok strán.

0.2 Prehľad dokumentu

Dokumentácia je rozdelená do viacerých, už vyššie spomenutých, kapitol. Prvá kapitola obsahuje analýzu problémovej oblasti, opis súťaže, podmienok účasti ako aj jazdných vlastností a možností auta.

V časti špecifikácia definujeme ciele, ktoré sa budeme snažiť splniť a tiež sa už venujeme konkrétnym požiadavkám kladeným na vytváraného autopilota.

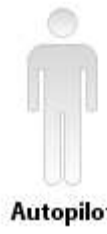
Návrh obsahuje konkrétne navrhované riešenie a identifikáciu jednotlivých častí a modulov autopilota spolu s popisom.

Spomenuté časti sú prvým výsledkom našej práce na projekte a budú postupne doplnené, najmä čo sa návrhu týka o detailné popisy a riešenia. Taktiež pribudne popis samotnej implementácie, testovania, ako aj používateľská príručka k vytvorenému prototypu.

0.3 Použité skratky a pojmy

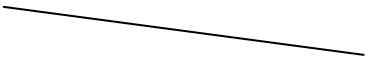
TORCS – The open racing car simulator
autopilot (bot) – inteligentný agent, ktorý v prostredí TORCS riadi vozidlo
tick – doba čakania servera na odozvu

0.4 Použitá notácia



hráč

prípad použitia



asociácia

cieľ

stav

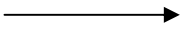
začiatočný stav

koncový stav

aktivita

rozhodovací blok

zlučovací blok



prechod medzi stavmi a akciami

1. Analýza

Kapitola obsahuje analýzu problémovej oblasti, v našom prípade svet TORCS a analyzuje existujúce riešenia autopilotov fungujúcich v tomto prostredí.

1.1 TORCS – The Open Racing Car Simulator

TORCS [4] je open source 3D simulátor automobilových pretekov dostupný na platformách Linux, FreeBSD, Mac OS X a Windows. Pôvodne ho vytvorila dvojica autorov Eric Espié a Christophe Guionneau, v súčasnosti projekt vyvíja Bernhard Wymann. TORCS je napísaný v programovacom jazyku C++ a umožňuje závodiť predprogramovaným agentom, *botom*, ako aj hráčom za pomoci klávesnice či myši.

TORCS je považovaný za jednu z najlepších open source alternatív ku komerčným závodným hrám z viacerých dôvodov [3]:

- Je založený na prepracovanom fyzikálnom modeli, ktorý berie do úvahy množstvo aspektov, ako napríklad kolízie, trakciu, aerodynamiku, spotrebu paliva a ďalšie.
- Vizualizácia je založená na sofistikovanom 3D modeli.
- Hráči majú k dispozícii vyše 40 tratí rozdelených do troch kategórií – klasické cestné preteky (road), ovály (oval) a hrboľaté nespevnené trate na štýl plochej dráhy (dirt).

Hlavným využitím je však vývoj vlastných botov, prípadne úprava existujúcich vyše 50 protivníkov kvôli dosiahnutiu čo najlepších výsledkov. Títo agenti sa zostavujú ako samostatné C++ moduly, ktoré je možné ľahko skompilovať a pridať do hry. V každom kontrolnom kroku (*tick*) má bot prístup k aktuálnemu stavu hry, ktorý obsahuje informácie o aute a trati, ako aj o protivníkoch. Bot kontroluje auto pomocou plynového pedálu a brzdy, radiacej páky a volantu.

Platforma TORCS sa využíva na usporadúvanie viacerých medzinárodných súťaží, z ktorých najvýznamnejšie sú Majstrovstvá jazdcov a podujatia organizované platformou The TORCS Racing Board. Novšími súťažami sú The Simulated Car Racing Championship a Demolition Derby – tieto využívajú odlišný prístup k informáciám o trati než je tomu v pôvodnom simulátori.

1.1.1 The Simulated Car Racing Championship

Súťaž je založená na platforme TORCS, avšak je potrebné doinštalovať softvér, ktorý pôvodnú architektúru rozširuje v troch bodoch [3]:

- Architektúra sa mení na klient-server aplikáciu: boti sú spúšťaní ako externé procesy prepojené s pretekovým serverom pomocou UDP spojení.
- Do hry je zapracovaný reálny čas: v každom hernom ticku (približne 20 ms simulovaného času) server odošle senzorové vstupy každému botovi a čaká 10 ms reálneho času na prijatie akcie od agentov. Ak nie je prijatá žiadna akcia, simulácia pokračuje ďalej a použije sa naposledy prijatá akcia.
- Softvér vytvára fyzické oddelenie medzi kódom jazdca a pretekovým serverom pomocou vytvorenia abstraktnej vrstvy senzorov a efektorov – vďaka nej sa

obmedzuje prístup k informáciám o trati dostupných v pôvodnej hre. Dostupné senzory a efektory (použité v ročníku 2010) sú uvedené v tabuľkách 01 a 02.

Tab. 01 *Popis dostupných efektorov 2010.*

Názov	**Rozsah**	**Popis**
accel	[0; 1]	Virtuálny plynový pedál
brake	[0; 1]	Virtuálny brzdový pedál
clutch	[0; 1]	Virtuálny spojkový pedál
gear	-1,0,1,...,7	Rýchlostný stupeň
steering	[-1; 1]	Hodnota zatočenia: -1 a 1 znamenajú naplno doprava resp. doľava, čo zodpovedá uhlu 0,785398 rad.
focus	[-90; 90]	Smer *focus* diaľkomerov v stupňoch.
meta	0; 1	Príkaz meta-riadenia: 0 predstavuje žiadnu zmenu; 1 žiadosť pre súťažný server o reštart preteku.

Tab. 02 *Popis dostupných senzorov.*

Názov	**Rozsah (jednotky)**	**Popis**
angle	$[-\pi; \pi]$ (rad)	Uhol medzi orientáciou auta a orientáciou osi trate.
curLapTime	$[0; \infty]$ (s)	Čas odjazdený počas aktuálneho kola.
damage	$[0; \infty]$ (body)	Aktuálne poškodenie auta (čím vyššia hodnota, tým väčšie poškodenie).
distFromStart	$[0; \infty]$ (m)	Vzdialenosť auta od štartovej čiary.
distRaced	$[0; \infty]$ (m)	Vzdialenosť prejdená od začiatku preteku.
focus	[0; 200] (m)	Vektor 5 diaľkomerov; každý vracia vzdialenosť medzi okrajom trate a autom v rozsahu 200 metrov. Senzory vzorkujú 5stupňový priestor pozdĺž smeru poskytnutého od klienta (akciou *focus*). Senzory môžu byť použité len raz za sekundu simulovaného času, v opačnom prípade poskytnú hodnotu mimo rozsahu (platí aj v prípade, že auto je mimo trate).

fuel	[0; ∞] (l)	Aktuálne množstvo paliva.
gear	{-1,0,1,…,7}	Aktuálny prevod (-1 je spiatočka, 0 neutrál).
lastLapTime	[0; ∞] (s)	Čas prejdenia posledného kola.
opponents	[0; 200] (m)	Vektor 36 senzorov zameraných na súperov: každý pokrýva úsek 10 stupňov v rozsahu 200 metrov a vracia vzdialenosť najbližšieho oponenta v pokrytej oblasti. Senzory pokrývajú celý priestor okolo auta, v smere hodinových ručičiek od π po -π vzhľadom k osi auta.
racePos	{1,2,…,N}	Pozícia v preteku vzhľadom na protivníkov.
rpm	[2000; 7000] (ot/min)	Počet otáčok za minútu motora.
speedX	[-∞; ∞] (km/h)	Rýchlosť auta pozdĺž pozdĺžnej osi auta.
speedY	[-∞; ∞] (km/h)	Rýchlosť auta pozdĺž priečnej osi auta.
speedZ	[-∞; ∞] (km/h)	Rýchlosť auta pozdĺž osi *z* auta.
track	[0; 200] (m)	Vektor 19 diaľkomerov; každý vracia vzdialenosť medzi okrajom trate a autom v rozsahu 200 metrov. Senzory pokrývajú priestor pred autom, v smere hodinových ručičiek od π/2 po –π/2 vzhľadom k osi auta. Ak sa auto nachádza mimo trate, návratová hodnota je mimo rozsahu.
trackPos	(-∞; ∞)	Vzdialenosť medzi autom a osou trate. Hodnota je normalizovaná vzhľadom na šírku trate: 0 predstavuje auto na osi trate, -1 v prípade, že auto je na pravom okraji a 1 v prípade, že auto je na ľavom okraji trate.
wheelSpinVel	[0; ∞] (rad/h)	Vektor 4 senzorov reprezentujúci rýchlosť otáčok kolies.
z	[-∞; ∞] (m)	Vzdialenosť ťažiska auta od povrchu trate pozdĺž osi *z*.

Cieľom majstrovstiev je zostrojiť bota pre závodné auto, ktorý bude súťažiť na niekoľkých neznámych tratiach najprv sám (na čas), a potom proti ostatným súperom. Boti vnímajú závodné prostredie radom senzorov a uskutočňujú typické jazdné akcie pomocou efektorov.

Majstrovstvá sa skladajú z deviatich pretekov uskutočňovaných počas troch súťaží (na každej sa konajú tri preteky) a v tomto formáte fungujú od roku 2009 (prvý pretek založený na rovnakom softvéri bol o rok skôr). Každý rok dochádza k jemným úpravám týkajúcich sa efektorov (pridanie spojky) či senzorov, napríklad zmena ich rozsahu. Významnou črtou je tiež zavedenie šumových senzorov – jedná sa o senzory focus, opponents a track z tabuľky 02. Podstatou je úprava návratovej hodnoty senzorov o takzvaný normálny šum, čiže o štandardnú odchýlku rovnajúcu sa istému percentu z nameraných hodnôt, pričom sa toto percento medzi jednotlivými ročníkmi majstrovstiev mení.
Každý pretek sa skladá z troch častí [2]:

- *Tréning (Warm-up)* – každý pilot jazdí sám; táto fáza je určená na zber dôležitých informácií o trati a na vylepšenie správania pilota.
- *Kvalifikácia (Qualifying)* – každý pilot jazdí sám na všetkých troch pretekoch súťaže; osem pilotov, ktorí prejdú najväčšiu vzdialenosť, sa kvalifikuje do závodu.
- *Samotný závod (Race)* – osem najlepších pilotov jazdí spolu. Preteky sa skladajú z ôsmich behov na každej z troch tratí – v každom behu štartuje pilot z iného miesta. Na konci preteku sú pilotom pridelené body systémom 10-8-6-5-4-3-2-1. Dodatočné dva body získa pilot, ktorý dosiahol najrýchlejšie kolo ako aj pilot, ktorý absolvoval preteky s najmenším poškodením auta.

1.1.2 Demolition Derby

Súťaž [1] je svojou architektúrou podobná The Simulated Car Racing Championship, rovnako je nutné doinštalovať softvér ovplyvňujúci pôvodný TORCS. Prvý ročník sa mal uskutočniť v roku 2010, z dôvodu nedostatku účastníkov však bol zrušený.

Cieľom je zostrojiť bota pre závodné auto, ktorý dokáže efektívne narážať do protivníkov, pričom sa bude snažiť minimalizovať vlastné poškodenie, podstatou je teda zničenie všetkých súperov, víťazom je posledné stojace auto. Demolition Derby sa odohráva na veľkej kruhovej trati (dĺžka 640m, šírka 90m, povrchom je asfalt). Senzory a efektory sú podobné ako pri súťaži The Simulated Car Racing Championship (tabuľky 01 a 02), odstránené sú efektory *speedZ* a *z*.

Model poškodenia je navrhnutý tak, že cieľom je naraziť do ostatných áut z boku alebo zozadu. Čelné nárazy, rovnako ako nárazy do steny nevedú k poškodeniu.

Všetci prihlásení piloti sa musia kvalifikovať do záverečného boja súťažením s každým protivníkom v kvalifikačných 1vs1 zápasoch. Po každom zápase, ktorý trvá maximálne 5 minút simulovaného času, auto s menším poškodením získava bod. Osem najlepších pilotov potom súťaží proti sebe v rovnakom čase v desiatich finálových zápasoch. V týchto súbojoch sa všetkým autám vynuluje poškodenie zakaždým, keď je nejaký účastník vyradený z boja. Najčastejší víťaz sa stane šampiónom Demolition Derby.

1.2 Existujúce riešenia

V nasledujúcej časti sú opísané 3 konkrétne riešenia iných tímov, ktoré dosiahli prvé miesta v rokoch 2008, 2009 a 2010. V každom je opísaný algoritmus, klady a zápory.

1.2.1 LUIGI CARDAMONE

Víťaz z roku 2008. Hlavnou ideou tohto riešenia bolo vyvinúť konkurencieschopného pilota, ktorý by využíval čo najmenej vstupov. Architektúra pozostáva z vyvíjajúcich sa neurónových sieti implementujúcich základné jazdné správanie, spojené s hotovým správaním pre štart, obnovu po kolízii, preradzovaním rýchlostí a predbiehaním protivníkov.
Vstupom do neurónovej siete boli informácie priamo súvisiace s riadením, čiže rýchlosť a senzory zachytávajúce okraje trate. Ako výstup bola zvolená dvojica natočenie kolies a pridávanie/brzdenie. Ak sa pilot nachádzal na rovnej dráhe, výstupy pre brzdenie a pridávanie boli komplete ignorované a auto akcelerovalo na plný plyn. Tento návrh nútil pilota jazdiť čo najrýchlejšia a zabraňoval plytvaniu času na bezpečné ale pomalé jazdenie.

Rozhodujúcou schopnosťou pilota je obiehanie protivníkov. V prípade tohto riešenia bolo explicitne implementované tak, že upravovalo výstupy neurónovej siete dvoma spôsobmi. Ak sa pilot nachádzal na rovnej dráhe, vždy sa pokúsil o predbiehací manéver. Ak sa však blížila zákruta, brzdil tak, aby nedošlo ku kolízii.

Pre radenie rýchlostí a obnovovanie sa z kolízií využili predprogramované techniky z príkladu pilota voľne dostupného spolu s TORCS. Štart preteku označili ako základnú a veľmi podstatnú časť preteku z dvoch dôvodov:

1. dochádza tam často ku zrážkam a auto sa môže značne poškodiť
2. je možné obehnúť väčšie množstvo áut naraz čo prispeje k lepšiemu konečnému výsledku

Vo svojom riešení predstavili veľmi jednoduchú štartovaciu techniku, založenú na rýchlom presunutí sa na okraj trate a predbehnutí čo najviac protivníkov hneď na začiatku.

Na vývoj neurónovej siete využili techniku NEAT, založenú na veľmi podobnom princípe ako evolučné algoritmy. Každú neurónovú sieť testovali jedno kolo na jednej konkrétnej trati z menom Wheel 1, ktorá obsahuje väčšinu zaujímavých vlastností z tratí, ktoré TORCS obsahuje. Na výpočet vhodnosti pre jednotlivé neurónové siete využívali tri hodnoty a to prejdená dráha, priemerná rýchlosť a počet tikov servera kedy sa pilot nachádzal mimo trate.

Medzi hlavné nedostatky riešenia patrí neschopnosť pilota ísť aspoň trochu podľa ideálnej stopy, vzhľadom k tomu, že väčšinu trate jazdí stredom. Nie je implementovaný žiadny druh učenia sa počas pretekov vzhľadom k tomu, že sa využíva vyvinutá neurónová sieť. Medzi kladné stránky patrí množina vedomostí o trati, ktorá je minimálna a vysoký stupeň zovšeobecnenia, čo znamená vytvorenie jednej neurónovej siete pre všetky trate a to iba na základe jedného kola na jednej trati.

1.2.2 AUTOPIA

Tím Autopia bol víťazom posledného ročníka súťaže typu championship v roku 2010. Líder tohto tímu, Enrique Onieva, sa zúčastnil viacerých predchádzajúcich ročníkov a zaoberá sa modulárnou parametrizovateľnou architektúrou. V tomto riešení rozdelili pilota na viac modulov, ktoré riešia jednoduchšie oblasti riadenia/problémy. Presnejšie jedná sa o šesť nasledovných modulov:

- kontrola radenia
- maximálna rýchlosť - modul určujúci maximálnu rýchlosť na segmente

- kontrola rýchlosti - modul kontrolujúci brzdu a plyn; slúži na upravenie rýchlosti, aby vyhovovala modulu maximálnej rýchlosti; okrem toho implementuje ABS a TCS
- kontrola riadenia - ovláda natočenie kolies
- modul učenia – deteguje segmenty trate, kde by sa mala rýchlosť upravovať, vo väčšine prípadov ide o rovinky alebo o miesta pred ostrými zákrutami
- modul protivníkov - aplikuje zmeny v riadení, aby vyhovovali situáciám, keď sú oponenti blízko

Na určenie rýchlosti využívajú fuzzy logiku, kde implementovali sedem jednoduchých pravidiel. Takýto prístup im umožňuje rýchlu odozvu a možnosť venovať sa ostatným zložitejším úlohám riadenia. Pravidlá vo fuzzy logike vylepšovali genetickými algoritmami na viacerých tratiach v rámci TORCS.

Modul protivníkov označili ako kritický, pretože stratégia protivníkov je neznáma a je ťažké sa rozhodnúť či predbiehať alebo brzdiť. Ako výsledok hĺbkovej analýzy implementovali jednoduché pravidlá ako je náhle zatočenie alebo znižovanie rýchlosti tesne pred zrážkou.

Výhody sú v jednoduchosti a parametrizovateľnosti riešenia, čo umožňuje ľahké rozšírenie. Modul protivníkov zabezpečuje obiehanie viacerých protivníkov s utŕžením čo najnižšieho poškodenia auta a umožňuje rozšíriteľnosť, čo sa týka zabraňovaniu v obiehaní inými protivníkmi.

1.2.3 COBOSTAR

Ako prvý skočil v roku 2009. Pilot je kombinácia techniky, ktorá vstup v podobe senzorov premieňa na akciu, a princípu úpravy správania sa, ktorý vylepšuje aktuálnu akciu za pomoci nejakého odhadu alebo vedomosti o budúcnosti alebo minulosti. Základné riadenie zahrňuje iba vzdialenosť a uhol najdlhšieho senzora k okraju trate. Pre riadenie mimo trate sa uvažuje uhol k osi trate a vzdialenosť k trati. Táto technika bola optimalizovaná a rozšírená o ABS, ASR, recovery modul, kontrolu skoku, detektor nárazov a vyhýbanie sa protivníkom.

Optimalizácia sa vykonávala nad zoskupením 16 parametrov, kde jednotlivé skupiny testovali na všetkých tratiach v rámci TORCS. Vhodnosť skupiny parametrov určovali na základe vzdialenosti, ktorú prešiel počas 10000 tikov servera. Nakoniec bola zvolená skupina ktorá prešla najväčšiu vzdialenosť vzhľadom ku každej trati, týmto spôsobom vylúčili lokálne najlepšiu skupinu a zvolili si všeobecne najvýhodnejšiu skupinu parametrov.

Učenie počas jazdenia bolo založené na závažnosti nehody a krokov pred nehodou, čo mohlo viesť k zníženiu rýchlosti alebo upraveniu správania v ďalšom kole. Implementovali monitorovania protivníkov takým spôsobom, že do senzorov merajúcich vzdialenosť ku krajom trate premietli miesto a čas za aký sa autá zrazia a následne sa riadili podľa základnej techniky. Takto vytvorili efektívne a robustné správanie zohľadňujúce správanie protivníkov.

Pozitíva tohto riešenia sú v jednoduchom riadení a prepojení riadenia s vyhýbaním sa kolíziám s protivníkmi. Avšak toto jednoduché riadenie, ktoré berie do úvahy iba senzory merajúce vzdialenosť od okraju trate, nie je optimálne a nepočíta zo žiadnymi inými údajmi o trati.

2. Špecifikácia

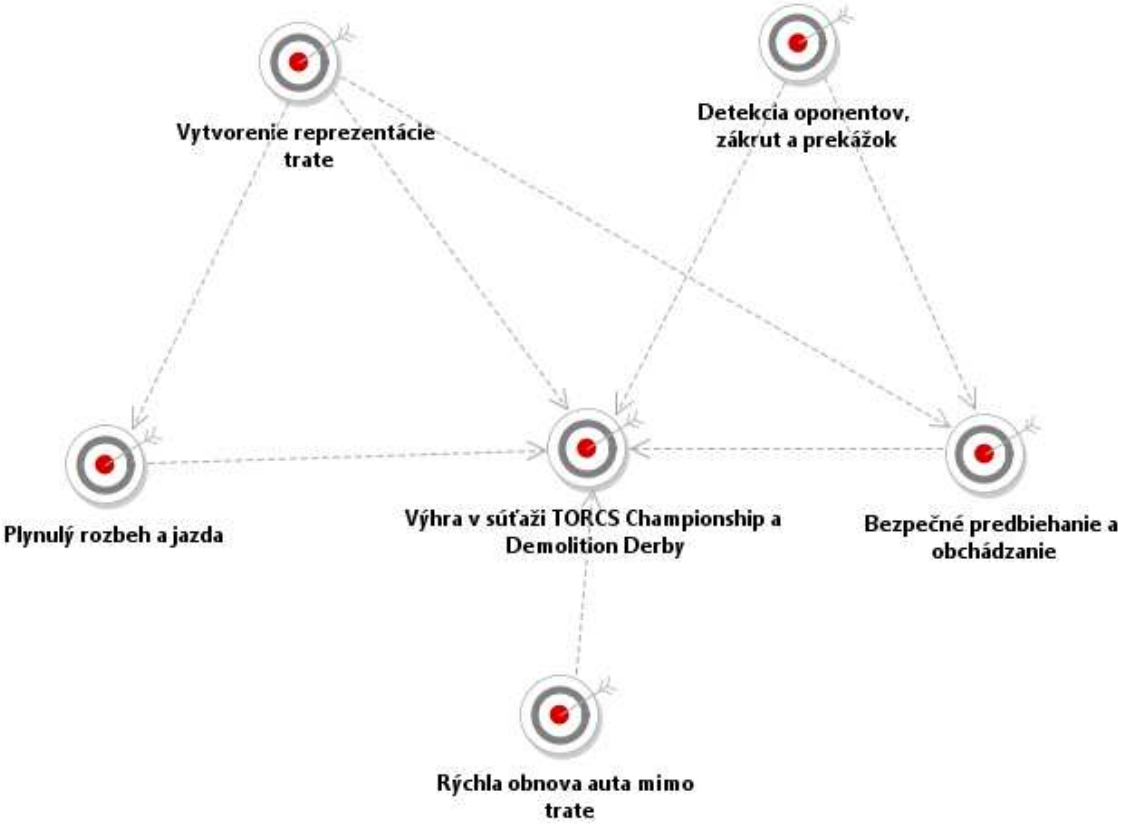
V tejto kapitole sa bližšie pozrieme na požiadavky kladené na nami vytváraného autopilota ako aj ciele, ktoré náš projekt sleduje, a ktoré sa snažíme naplniť. Taktiež priblížime správanie autopilota, ktoré bude ilustrované diagramom prípadov použitia.

2.1 Ciele projektu

V nasledujúcich riadkoch uvádzame sumarizáciu cieľov, ktoré sme si stanovili. Identifikované ciele sú ilustrované diagramom cieľov na Obr. 01.

Identifikované ciele:

- Výhra v súťaži TORCS Championship a Demolition Derby
  - Vytvorenie reprezentácie trate - je potrebné vytvoriť matematickú reprezentáciu trate, aby sme boli schopní určiť čo najlepšiu stopu, ktorou náš autopilot pôjde. Zároveň táto reprezentácia poslúži aj na konfiguráciu rôznych nastavení jazdy, ako rýchlosť, miesta brzdenia, či častých a možných kolízií so súpermi.
  - Detekcia oponentov, zákrut a prekážok - je potrebné detegovať počas jazdy priestor, v ktorom sa auto nachádza, z dôvodu predvídania a prispôsobenia jazdy resp. prípravy na obiehanie.
  - Bezpečné predbiehanie a obchádzanie - splnenie tohto cieľa významne pomôže naplneniu celkového cieľa, ktorým je cieľ vyhrať. Prípadná kolízia pri predbiehaní alebo obchádzaní spôsobí stratu času, ktorá môže mať za následok celkový neúspech v súťaži.
  - Rýchla obnova auta mimo trate - taktiež dôležitý cieľ, keďže v súťaži ide o čas a čím rýchlejšie sa dokážeme dostať späť na trať po vybočení z nej, tým sú naše šance na úspech väčšie.
  - Plynulý rozbeh a jazda - plynulosť jazdy je určite dôležitým krokom k víťazstvu a k naplneniu hlavného cieľa.



Obr. 01 *Identifikované ciele*

2.2 Špecifikácia požiadaviek

Systém bude založený a testovaný na platforme TORCS, ktorá je bližšie špecifikovaná v časti *Analýza*. Našou úlohou je implementovať autopilota, ktorý bude schopný jazdiť na tratiach tejto platformy, a to v dvoch súťažiach:

- Championship
- Demolition Derby,

ktoré sú taktiež detailne popísané v analýze. Našou hlavnou úlohou bude navrhnúť autopilota, ktorý prejde určenú trať čo najrýchlejšie, respektíve s čo najmenším poškodením. Musíme sa zamerať na viaceré oblasti, čo sa jazdy a jazdných vlastností vozidla týka. Dôležité je zvoliť správnu metodiku jazdy, pamätať si trať alebo reagovať na aktuálny stav auta na trati. Trate sú vytvorené na rôznych povrchoch, pričom na každom sú jazdné vlastnosti auta iné a preto je potrebné prispôsobiť spôsob jazdy vzhľadom na aktuálny povrch. Na naučenie trate a zistenie všetkých jej vlastností máme k dispozícii testovacie kolá, tzv. warm up, počas ktorých prechádzame traťou a môžeme zbierať informácie potrebné pre vytvorenie ideálnej stopy a konfiguráciu ostatných parametrov tak, aby sme dosiahli čo najlepší čas na danej trati.

Dôležitou oblasťou je obnova auta po zídení z trate. Tu je potrebné definovať všetky možnosti, ktoré môžu nastať a taktiež spôsoby riešenia daných stavov. Ďalšou oblasťou, na

ktorú sa musíme zamerať, je predbiehanie oponentov, kde je potrebné monitorovať aktuálnu polohu oponentov, našu rýchlosť ako aj možnosť predbiehania vzhľadom na trať.

V súťaži Demolition Derby je potrebné zamerať sa najmä na spôsob ako čo najdlhšie udržať auto nepoškodené a zároveň zničiť súperov. Musíme riešiť možné spôsoby vyhýbania a narážania. V súťaži Demolition Derby je potrebné zamerať sa najmä na spôsob ako čo najdlhšie udržať auto nepoškodené a zároveň zničiť súperov čelnými nárazmi. Za týmto účelom treba vymyslieť techniky jazdy, ktoré autu umožnia vyhýbať sa útokom súperov a dajú mu schopnosť rozoznať vhodné príležitosti na vykonanie narazenia do súpera. Auto pri tom musí byť schopné zvládať rôzne počty súperov, keďže kvalifikácia sa odohráva systémom jeden proti jednému a vo finále proti sebe bojuje 8 áut naraz, pričom sú postupne vyraďované.

2.3 Identifikácia prípadov použitia a hráčov

V súvislosti s naším systémom nevystupujú žiadni hráči, ak pravdaže nerátame používateľa, ktorý spustí zdrojové kódy autopilota na serveri TORCS. Jediným zmysluplným hráčom je samotný autopilot, ktorý vykonáva rôzne aktivity v súvislosti s prechádzaním trate. Pre vyššie spomenutého hráča sme identifikovali tieto prípady použitia (Obr. 02):

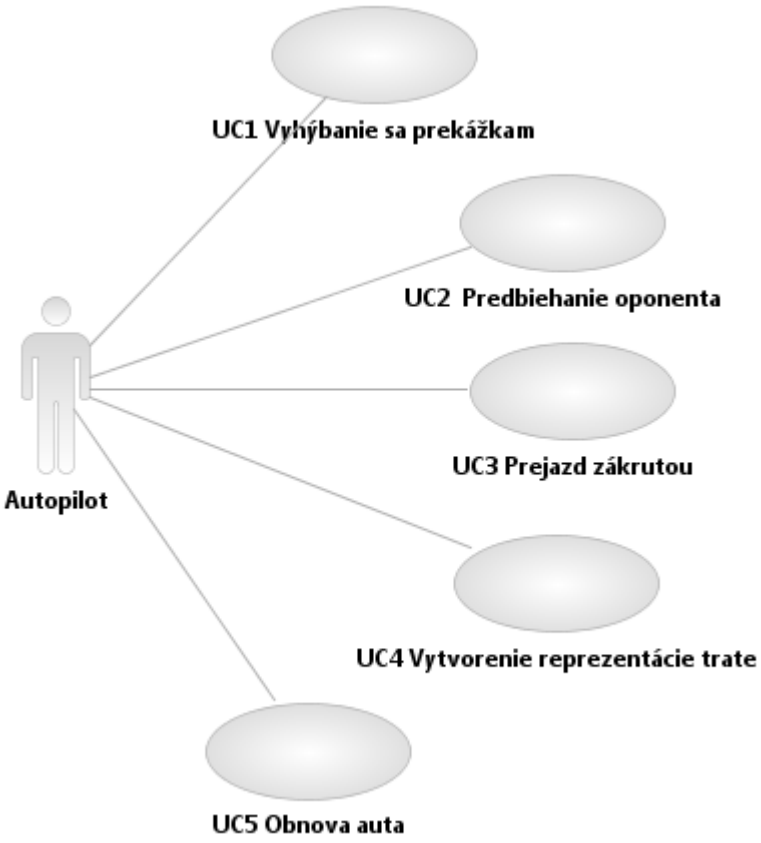
- *UC1 Vyhýbanie sa prekážkam* - autopilot po identifikácii prekážky na trati vyhodnotí situáciu a zvolí vhodný spôsob obídenia danej prekážky v prípade potreby aj zastavenie vozidla

- UC2 *Predbiehanie oponenta* - autopilot deteguje oponenta, vyhodnotí možnosť predbiehania vzhľadom na možnosti trate a rýchlosť, určí ideálnu stopu predbiehania a predbehne daného oponenta

- UC3 *Prejazd zákrutou* - autopilot identifikuje zákrutu, vzhľadom na zapamätanú trať zvolí vhodný spôsob prejdenia zákruty - miesto vjazdu a výjazdu, rýchlosť, natočenie kolies a pod.

- UC4 *Vytvorenie reprezentácie trate* - autopilot prechádza trať viackrát za sebou a zisťuje všetky potrebné parametre - vytvára reprezentáciu trate pre potreby následnej súťaže a identifikáciu čo najideálnejšej stopy

- *UC5 Obnova auta* - autopilot po vyjdení z trate určí vzhľadom na svoju polohu a smer vhodný spôsob obnovy a vracia sa späť na trať



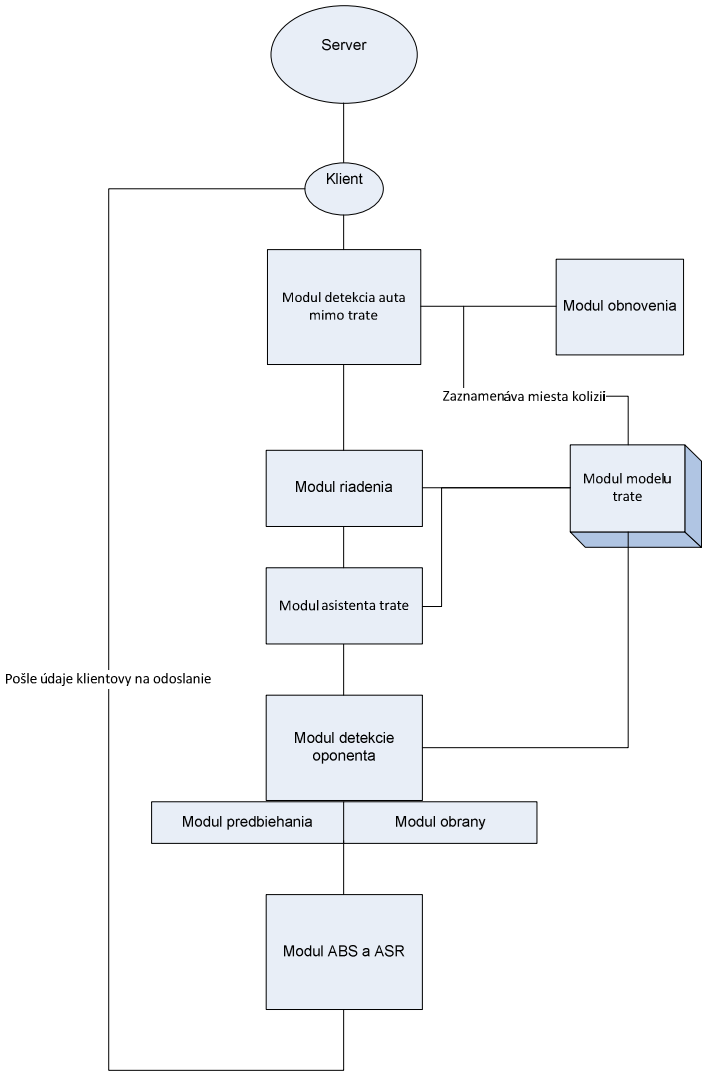
Obr. 02 *Prípady použitia*

3. Návrh

Naším cieľom je vytvoriť autopilota, ktorý bude schopný dosahovať najlepšie možné výsledky pri nasadení aj do vopred neznámej trate.

3.1 Základná koncepcia architektúry

Keďže našou úlohou je zúčastniť sa v oboch súťažiach, je potrebné navrhnúť architektúru autopilota v čo najväčšej miere prenosnosti. Takto navrhnutý autopilot bude ľahko zostrojiteľný pre akúkoľvek súťaž, pretože každá jedna potrebuje základné moduly (napríklad modul pre ovládanie plynu a brzdy, modul riadenia, modul radenia a pod.). Po viacerých návrhoch architektúr sme sa rozhodli práve pre architektúru zobrazenú na Obr. 03.



Obr.03 Základný návrh architektúry autopilota

3.2 Návrh modulov

Podkapitola obsahuje opis jednotlivých modulov zo základnej koncepcie architektúry autopilota. Nevyhnutné moduly pre chod autopilota:

- modul klienta
- modul detekcie auta mimo trate
- modul obnovenie
- modul riadenie

Modul klienta
Klient slúži na komunikáciu so serverom. Prijíma údaje zo senzorov, s ktorými jednotlivé moduly pracujú a menia ich. Na základe týchto údajov klient spätne posiela na server akcie, ktoré je potrebné vykonať. Komunikácia prebieha v tzv. „Tick-och“ čo je približne 20 milisekúnd.

Tento modul nie je potrebné implementovať, keďže kód, a tým aj celá funkcionalita komunikácie klienta so serverom je poskytovaná v rámci súťaže Torcs.

Modul riadenia (Driving)
Modul pre riadenie má za úlohu prispôsobovať rýchlosť auta a smer jazdy. Rýchlosť jazdy je ovplyvnená tromi efektormi a to plyn, brzda a prevodový stupeň. Pri zmene prevodového stupňa je za určitých okolností vhodné aktivovať aj efektor pre spojku. Smer jazdy je možné meniť podľa nastavenia efektora pre natočenie volantu.

Úlohou tohto modulu je taktiež detekcia aktuálneho stavu pilota, t.j.:

- detekcia pilota mimo trate – na základe detekovania pozície mimo trate zvolí modul obnovenia
- detekcia oponentov - detekuje dobiehajúceho alebo približujúceho sa oponenta. Na základe informácií v modeli trate rozhodne, či má autopilot zvoliť modul predbiehania alebo obrany.

Modul obnovenia (Recovery)
Tento modul slúži na návrat autopilota z mimo jazdnej dráhy v čo najkratšom čase. V prvom kroku modul zistí, v ktorej časti mimo trate sa nachádza. Následne zistí, v akom smere sa auto nachádza vzhľadom k trati. Na základe týchto informácií prispôsobí nastavenia hodnôt efektorov.

Modul modelu trate (Lap model)
Na dosiahnutie čo možno najlepších výsledkov je potrebné vedieť ako trať vyzerá. Na to slúži práve tento modul, ktorý zaznamenáva prejdenú trasu cesty spoločne s časom. Každým kolom sa porovnáva čas, za ktorý bola trať prejdená. Ak bol čas kratší ako čas nachádzajúci sa v modeli, tak sa model aktualizuje. Model bude obsahovať aj miesta, v ktorých prišlo ku kolíziám.

Modul asistenta trate (Line Assistent)

Na dosiahnutie maximálnej možnej rýchlosti na trati dohliada modul asistenta trate. Úlohou tohto modulu je určenie ideálnej stopy na trati a následná korekcia na ideálnu stopu ak sa v nej nenachádza.

Modul detekcie oponenta

Úlohou modulu je detekcia dobiehajúceho alebo približujúceho sa oponenta. Na základe informácií v modeli trate rozhodne, či má autopilot zvoliť modul predbiehania alebo obrany.

Modul predbiehania (Overtake)

Tento modul zaisťuje predbiehanie oponenta v prípade, že pri predbiehaní alebo počas predbiehania autopilot nevyletí mimo trať.

Modul obrany (Defense)

Podobnú úlohu ako modul predbiehania má modul obrany. Rozdiel je v tom, že autopilot sa snaží ubrániť si pozíciu od oponenta ale len za cenu nevyletenia z dráhy alebo osobného kontaktu.

Modul ABS a ASR

Modul slúži na reguláciu stlačenia brzdy v prípade ABS, aby nevznikali súvislé brzdné dráhy, na ktorých je auto neovládateľné. Pri ASR sa reguluje stlačenie plynu na úroveň, aby nedochádzalo k prekĺzavaniu kolies, čo taktiež vedie k neovládateľnému autu.

4. Prototyp

Kapitola opisuje moduly navrhnuté a implementované do prototypu vypracovaného počas zimného semestra.

4.1 Modul riadenia

Základný modul vykonáva esenciálne funkcie ako točenie, pridávanie, brzdenie, radenie. Je schopný bezpečne prejsť akúkoľvek trať bez jej modelu. Zároveň v prípade detekcie výnimočného stavu, ako je približovanie oponenta, či pozícia mimo trate, zavolá príslušný modul, ktorý týmto preberá riadenie pilota, až kým sa nedostane naspäť do pôvodného stavu.

4.1.1 Prejazd zákrut

Návrh

Plynulé točenie v zákrutách

Rýchly prejazd prvého kola je potrebný na kategorizovanie zákrut vo fáze warm up. Aby sme dosiahli presnejšie údaje v modeli trate je potrebné dosiahnuť postupné plynulé točenie volantom v zákrutách. Takýto prístup je oveľa presnejší napríklad oproti riadeniu podľa najdlhšieho senzora využívaného tímom COBOSTAR (kapitola 1.2.3), kde je využitých iba niekoľko stavov natočenia volantu ako 10°, 20° alebo plné vytočenie. Tieto tri stavy viedli k neprimeranému mykaniu auta v zákrutách, čo spôsobovalo pokles otáčok a rýchlosti.

Prejazd prvým kolom

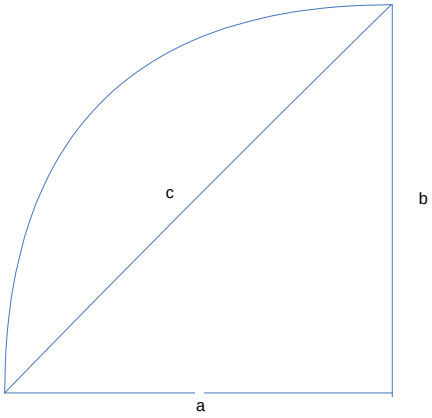
Pilot pozná dva úseky trate a to rovina a zákruta. Na akom úseku sa momentálne nachádza determinuje zo senzorov merajúcich vzdialenosť k okrajom trate. Na základe týchto senzorov determinuje aj smer zákruty. Natočenie volantu a rýchlosť sa určí z hodnoty vrátenej senzorom predlžujúcim pozdĺžnu os auta. Vzhľadom na to, že v prvom kole je kritické to, aby pilot ostal celý čas na trati, zajazdil celé kolo bez kolízie, znížili sme maximálnu rýchlosť počítanú zo vzorca (formula a), kde v je zisťovaná rýchlosť, g je gravitačné zrýchlenie, μ frikcia trate a s dráha k okraju trate zo senzora predlžujúceho pozdĺžnu os auta skrátená o polovicu šírky trate, o 60 km/h. Takto počítaná rýchlosť sa využíva aj na rovine aj v zákrute.

(a) $$v = \sqrt{2 \cdot g \cdot \mu \cdot s}$$

Na rovine pilot v prvom kole ide rovnobežne s osou a po strede trate. V zákrute vychádza z funkcií vytvorených logistickou regresiou, kde ako vstup ide dĺžka senzora predlžujúceho pozdĺžnu os auta a výstupom je natočenie volantu.

Odvodenie funkcií zatáčania

Meranie spočívalo v zaznamenávaní šírky prejdeného výseku kruhu, čiže premennej *c*, pri množine 20 natočení volantu (Obr. 04). Meranie prebiehalo na rovine. Dĺžka *c* sa určovala pomocou Pytagorovej vety z údajov *a* a *b*, ktoré sa determinovali zo senzorov pre pozíciu na trati a šírku trate (*a*) a zo senzoru prejdená vzdialenosť (*b*). Takto vznikli usporiadané dvojice *[c,natočenie]*. Pre usporiadané dvojice sme cez logistickú regresiu odhadli funkcie. Pre väčšiu presnosť sme interval *c* rozdelili na tri podmnožiny určené jednotlivými natočeniami. Trojicu určených vzťahov je vidieť vo formule b.



Obr.04 *Výpočet natočenia z prepony c*

(b)

ak c > 138 tak

$$natočenie = 0.0075$$

inak ak c > 43 tak

$$natočenie = \frac{0{,}8409}{c - 26{,}9920}$$

inak ak c > 15 tak

$$natočenie = \frac{2{,}2216}{c - 9{,}921}$$

inak ak c > 6,3545 tak

$$natočenie = \frac{3{,}8193}{c - 6{,}3542}$$

inak

$$natočenie = 1$$

Implementácia

Implementácia prebehla podľa návrhu. Na určovanie smeru zákruty sú využívané senzory s indexmi 8,9 a 10. Pri krátkych a prudkých zákrutách vznikali problémy, že pilot točil príliš skoro a príliš prudko, čo viedlo k zmene smeru zákruty. Tento problém bol riešený cez zásobník posledných 5 hodnôt smeru zákruty, čím sa vlastne vyhladia anomálie. Ďalej prebehla optimalizácia cez úpravu dĺžky nameranej senzorom predlžujúceho pozdĺžnu os auta a to tak, že sa z nej odpočítava šírka trate. Následne ak je zákruta s malým polomerom, čiže hodnota senzora je kratšia ako 43 a dlhšia ako 15, zvýši sa dĺžka o 5%. Ak je hodnota senzora kratšia ako 15 metrov, tak sa táto hodnota zvýši o 10%.

Testovanie

Testovanie prebiehalo sledovaním správania pilota na rôznych tratiach dodávaných s frameworkom Torcs počas implementácie. Z testovania vyplynuli zmeny uvedené v implementácii. Vzhľadom k tomu, že riadenie využíva senzory, ktoré ovplyvňuje šum, bude potrebné prekryť senzor predlžujúci pozdĺžnu os auta viacerými senzormi. Následne urobiť priemer hodnôt a ten využívať na výpočet zatočenia.

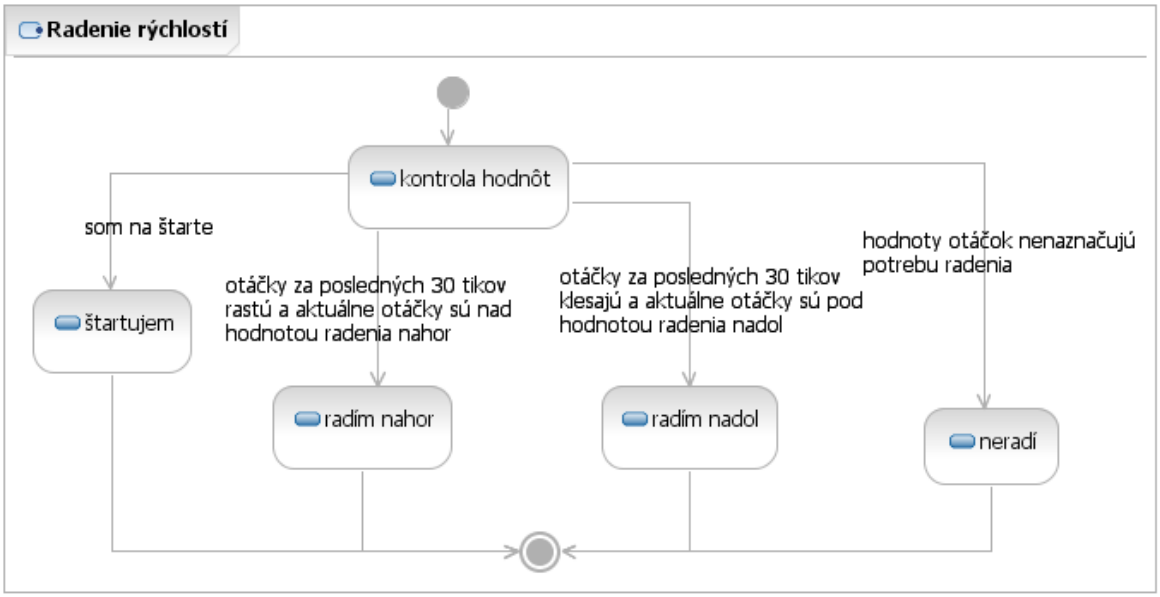
4.1.2 Radenie prevodových stupňov

Návrh

Návrh spočíva v tom, že pri radení neberieme do úvahy iba intervaly otáčok pre jednotlivé rýchlosti, ale aj správanie sa otáčok za posledných niekoľko tikov, ako aj vo využití spojky. Zavedením dvoch stavov

- otáčky klesajú
- otáčky stúpajú

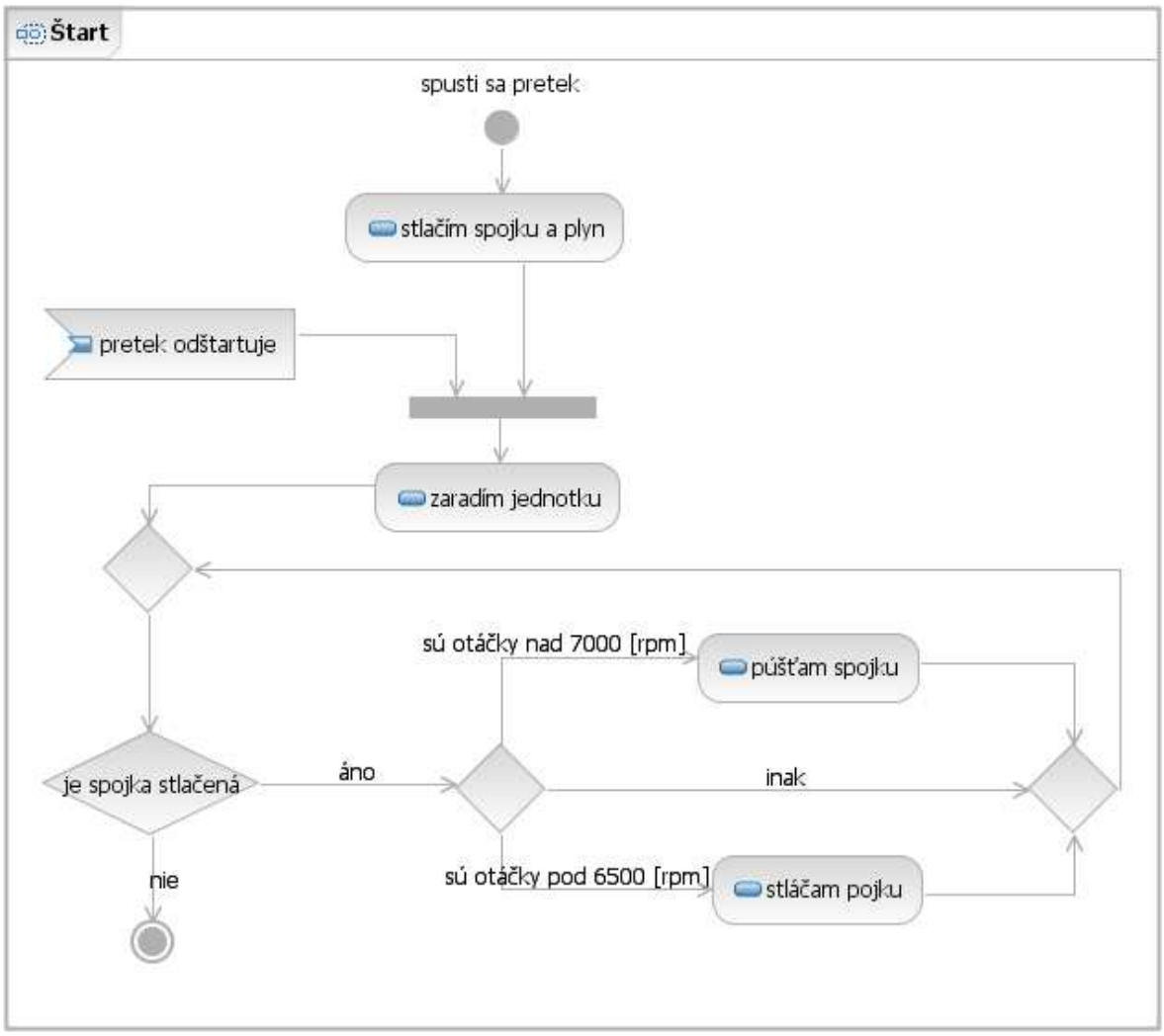
sme dosiahli zúženie intervalov pre jednotlivé prevodové stupne. Na obrázku Obr. 05 je vidieť stavy, v akých sa radenie môže nachádzať. Od ročníka 2010 je v ponuke efektorov aj spojka. Avšak nie je nutné ju používať pri radení, vzhľadom k tomu že serveru posielame údaj o prevodovom stupni a on na základe nami poslanej hodnoty a predchádzajúcej hodnoty prehodí/neprehodí stupne automaticky. Rozhodli sme sa spojku využívať na štarte.



Obr. 05 *Stavový diagram pre radenie*

Používanie spojky (asistencia pri štarte)

Z analýzy vyplynulo, že štart v pretekoch je veľmi dôležitý a s pridaním spojky sa umožnilo pilotovi vytvoriť si náskok už na začiatku pretekov. Asistencia pri štarte je založená na udržaní otáčok vozidla na hodnotách, ktoré mu umožňujú najväčšie zrýchlenie. Toto sa dá zabezpečiť stlačením spojky. Spojku aktivujeme pri prevodových stupňoch jedna a dva. Pri jednotke sa spojka využíva iba pri štarte. Spojka pri dvojke sa využíva počas celého preteku a to iba pri radení z jednotky na dvojku. Auto má najväčšie zrýchlenie od 7000 otáčok vyššie. Na obrázku Obr. 06 je proces asistencie pri štarte pomocou spojky vyobrazený diagramom aktivít.



Obr. 06 *Diagram aktivít pre využitie spojky pri štarte*

Implementácia

Pre jednotlivé prevodové stupne sme identifikovali otáčkové intervaly uvedené v Tab 03.

Tab. 03 *Otáčkové intervaly*

Stupeň	1	2	3	4	5	6
Radenie hore	9500	9500	9500	9500	9500	0
Radenie dole	0	4300	6200	7000	7300	7700

To, či otáčky rastú alebo klesajú, sa určuje z množiny po sebe nasledujúcich otáčok za posledných 30 tikov. To, či množina rastie alebo klesá, určujeme na základe prevládania lokálnych rastov alebo klesaní. Množina otáčok prestavuje rad. Čiže v praxi sa porovnajú susediace dvojice otáčok z radu a podľa toho, či väčšina porovnaní rastie alebo klesá, aj

množina otáčok rastie alebo klesá. Takéto určovanie rastu ošetruje situácie, pri ktorých napríklad v dôsledku prudkého zatočenia tri hodnoty rapídne klesnú ale celkové otáčky stále rastú.

Po radení nahor alebo nadol sa nasledujúcich osem tikov nesmie radiť a ani množina otáčok sa neaktualizuje. Týchto osem je doba, pokiaľ vozidlo preradí a údaj o otáčkach nabehne, čiže motor sa dostal do záberu s novým prevodovým stupňom.

Testovanie

Testovanie prebiehalo sledovaním správania pilota na rôznych tratiach dodávaných s frameworkom Torcs počas implementácie. Pri procese radenia sa využívajú senzory bez šumu. Zúženie intervalov otáčok a zavedenie stavu otáčok umožňuje včasnejšie podradenie, čím si pilot udrží vyššie otáčky a akceleráciu. Z testovania vyplynulo, že pri použití asistencie pri štarte z využitím spojky sa zrýchlenie z 0 na 100 km/h zrýchlilo v priemere o 1 sekundu. Framework Torcs neuvažuje opotrebenie spojky. V prípade, že by toto bolo v budúcnosti doimplementované, mohlo by to spôsobiť problémy, pretože nami zvolený prístup spojku príliš zaťažuje.

4.1.3 Meranie odporu povrchu

Návrh

Konštanta trenia medzi pneumatikou a povrchom je dôležitá pri väčšine výpočtov určujúcich krajné hodnoty pre správanie vozidla na danej trati pri riadení. Meranie frikcie, čiže odporu alebo trenia povrchu, je založené na rovnosti kinetickej energie a práce spravenej frikciou pri zablokovaných kolesách v šmyku [5]. Čiže odpor povrchu zistíme z rýchlosti, kedy sa kolesá zablokujú a brzdnej dráhy od tohto miesta.

Implementácia

Z formuly *a* sme si vyjadrili μ, viď formulu *c*. Po nadobudnutí rýchlosti 125 km/h začne pilot naplno brzdiť. Brzdnú dráhu *s* meria od miesta kde sa mu zablokujú všetky štyri kolesá. Rýchlosť *v* je zaznamenaná tiež v mieste, kde sa zablokujú kolesá. Výpočet sa prevedie až keď auto úplne zastaví.

(c) $$\frac{v^2}{2 \cdot g \cdot s} = \mu$$

Všetky údaje potrebné na výpočet frikcie vieme zistiť z hodnôt zo senzorov. Využívajú sa senzory, ktoré neovplyvňuje šum.

Testovanie

Zvolená metóda na meranie frikcie na rôznych povrchoch pri rôznych rýchlostiach dáva výsledky s malou odchýlkou, preto je možné zmeniť rýchlosť merania podľa dĺžky identifikovanej roviny na ktorej sa bude merať.

4.2 Modul Recovery

Pilot sa počas svojej jazdy môže dostať do rôznych situácií, ktoré nie sú typické pre jazdu po trati. Jedná sa najmä o také situácie, kedy sa pilot buď prešvihnutím najvyššej povolenej rýchlosti, ktorou je možné zákrutu bezpečne prejsť, alebo po kolízii s iným pilotom dostane mimo trate. Vzhľadom na zvolenú modulárnu architektúru v týchto situáciách preberá úplnú kontrolu nad pilotom *modul Recovery*, ktorého úlohou je navrátiť pilota späť na trať, a to v čo najmenšom možnom čase. V každom *tiku* sa zisťuje, či sa pilot nachádza v stave, kedy je potrebné tento modul použiť.

4.2.1 Návrh

Východiskom pre riešenie prípadov, kedy je potrebné na riadenie aplikovať modul Recovery, je identifikovanie stavov, v ktorých sa môže pilot nachádzať a sú pre riadenie atypické, pilot sa s nimi nevie sám vysporiadať.

Pilot mimo trate

Najčastejšie vyskytujúcim sa stavom bude stav, kedy sa pilot bude nachádzať mimo trate. Ako prvé je potrebné uvedomiť si, že existujú dva hlavné stavy, do ktorých sa môže pilot dostať vybehnutím mimo trate. A to:

- Pilot mimo trate NAĽAVO
- Pilot mimo trate NAPRAVO

Tieto stavy identifikujeme pomocou senzora, ktorý vracia pozíciu na trati a to konkrétne číslo väčšie ako 1, v prípade, že sa pilot nachádza naľavo od trate a číslo menšie ako -1, keď sa pilot nachádza napravo od trate. Toto je prvý predpoklad, na základe ktorého sa identifikuje vhodnosť použitia modulu Recovery. Z informácie o pozícii pilota mimo trate, naľavo alebo napravo, nie je možné určiť ďalšie správanie sa pilota, a preto sa ešte stále nevie, aké hodnoty treba nastaviť efektorom. Takže bude potrebná ďalšia identifikácia stavov (podstavov) v rámci stavu naľavo alebo napravo od trate. Identifikovali sme nasledujúce stavy:

Pilot je na ĽAVEJ strane mimo trate

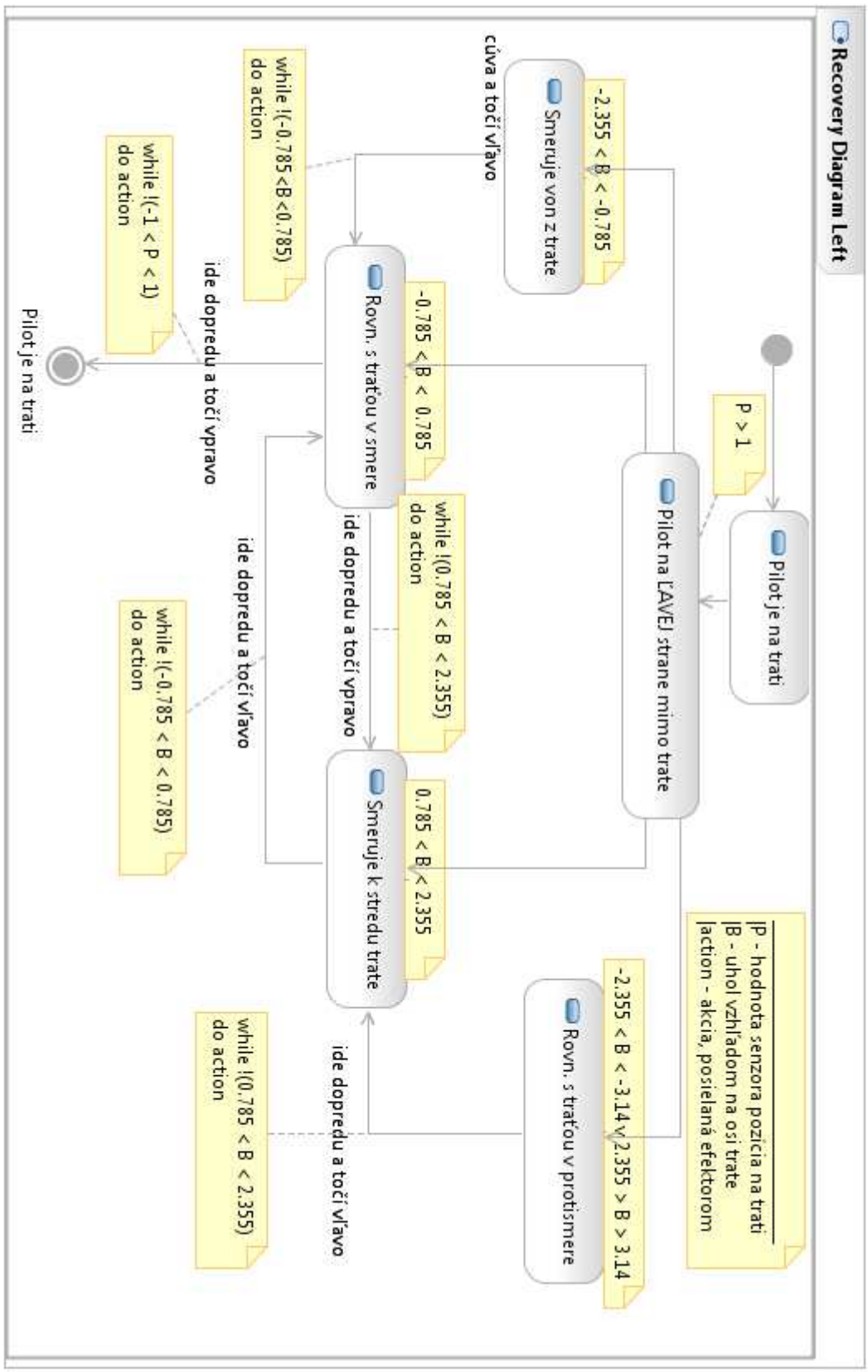
- Rovnobežne s traťou v protismere
- Smeruje von z trate
- Rovnobežne s traťou v smere trate
- Smeruje k stredu trate

Pilot je na PRAVEJ strane mimo trate

- Rovnobežne s traťou v protismere
- Smeruje von z trate

- Rovnobežne s traťou v smere trate
- Smeruje k stredu trate

Každý z týchto stavov bude špecifikovaný podmienkou, ktorá ho ohraničuje a tá bude vymedzená uhlom vzhľadom na os trate. Čo je podstatné je to, že každý stav bude určovať hodnoty efektorov, čiže to, ako sa má pilot správať, čo sa týka radenia a riadenia. Podrobnosti o prechodoch medzi jednotlivými stavmi od počiatku, kedy je pilot na trati, až po stav, kedy sa dostane opäť na trať, sú popísané v stavovom diagrame na obrázku Obr. 07. Na diagrame je znázornený hlavný stav, kedy je pilot na ľavej strane mimo trate, pretože diagram stavu, kedy je pilot na pravej strane mimo cesty je podobný. Líši sa len zrkadlovo preklopenými ohraničeniami vstupnej podmienky každého stavu.



Obr. 07 *Stavový diagram: Pilot na Ľavej strane mimo trate*

Špecifické stavy

Okrem uvedených stavov mimo trate naľavo a napravo sú identifikované aj ďalšie špecifické stavy, do ktorých sa môže pilot počas jazdy dostať. A to konkrétne tieto:

- Jazda v protismere
- Pilot je na trati, zablokovaný

Jazda v protismere

Môže sa stať, že sa pilot dostane do stavu, kedy bude pokračovať v jazde aj keď by bol v protismere, v domnienke, že ide správne. Príčinou je to, že na základe senzorov použitých pre riadenie nie sme schopní zistiť, či sa pilot pohybuje v smere trate alebo v protismere. Preto identifikujeme stav „v protismere“ a to tak, že sa zo senzora zistí natočenie pilota vzhľadom na os trate a od ďalšieho senzora aj to, že pilot nie je mimo trate, ale stále sa nachádza na trati. Podľa informácie o pozícii na trati, získanej zo senzora, a natočenia pilota vzhľadom na trať sa riadenie odovzdá modulu Recovery, kde sa s pilotom bude narábať tak, ako by bol mimo trate a tým pádom sa dostane do konkrétnych stavov, ktoré budú nastavovať aj hodnoty efektorov.

Pilot je na trati, zablokovaný

Ďalší zo špecifických stavov, ktorý by mohol nastať je ten, že pilot bude na trati, kolmo na jej os a bude zablokovaný pri okraji. Tento prípad môže nastať na typoch trate, ktoré nemajú miesto „mimo“ trate, ale sú ohraničené bariérami. Identifikovanie tohto stavu nie je jednoduché, a bude si vyžadovať informáciu o tom, či je pilot v pohybe, alebo stojí. Bude treba postupovať nasledovne:

- Zistiť či pilot stojí na mieste (rýchlosť za posledných 20 tikov je menšia ako určitá hodnota)
- Odovzdať riadenie modulu Recovery (s pilotom sa bude narábať tak, ako keby sa nachádzal mimo trate)

Je možné, že špecifických stavov je viac, tie však budú odhalené až testovaním naimplementovaného návrhu modulu Recovery.

4.2.2 Implementácia

Keďže vychádzame z modulárnej architektúry, modul Recovery bol implementovaný ako samostatná triedu, ktorej objekt bol zakomponovaný do triedy pilot, konkrétne do metódy *control*. S modulom Recovery taktiež súvisí aj ESP, pretože pri prevzatí riadenia treba eliminovať šmyky použitím ABS a ASR metód, ktorými táto trieda disponuje. Čiže, implementácia modulu Recovery sa dotýka troch tried a to:

- *Prototype* (controller, prototyp pilota)

- *RecoveryModule* (hlavná trieda modulu Recovery)
- *ESP* (trieda s ABS a ASR)

Rozhodnutie, kedy a ako sa použije modul Recovery, sa vykonáva v metóde *control*, a to takmer na jej začiatku, pred celým riadením. Na základe údajov zo senzorov sa vyhodnocuje niekoľko po sebe nasledujúcich podmienok. Tieto podmienky overujú, či sa pilot nachádza v stave, kedy je potrebné na riadenie pilota aplikovať modul Recovery. Podmienky overujú všetky možné stavy, tak pilota mimo trate ako aj stavy špecifické. Ak sa pilot v niektorom z týchto stavov nachádza, riadenie preberá modul Recovery až dovtedy, pokiaľ pilot nie je opäť vrátený na trať (na trati, v smere trate) alebo riadne nepokračuje v jazde. Takto identifikovaný stav je postúpený ďalej a to triede *RecoveryModule* a jej metódam.

Trieda RecoveryModule

Okrem iných, obsahuje nasledujúce metódy, ktoré je potrebné spomenúť:

- **public void checkRecovery**(SensorModel sm){...}
- **public** Action **recoverLeft**(SensorModel sm){...}
- **public** Action **recoverRight**(SensorModel sm){...}

Metóda *checkRecovery* sa volá v každom tiku v metóde *control*, vypočítava priemernú rýchlosť za posledných 20 tikov, ktorá naznačuje, či je pilot v pohybe, alebo stojí. Ak sa identifikuje, že pilot má rýchlosť blízku nule, riadenie preberá *modul Recovery*.

Najpodstatnejšie metódy sú však *recoverLeft* a *recoverRight*, ktoré sú volané z metódy *control* po tom, ako je identifikovaná situácia, kedy je pilot mimo trate alebo v iných špecifických situáciách. Každá z týchto metód obsahuje sériu podmienok, ktoré bližšie špecifikujú stav vyžadujúci obnovenie. Na základe toho, v akom konkrétnom stave sa pilot nachádza, sa nastavia hodnoty efektorom (akcia), ktorá je následne vrátená metóde *control* a ňou sa pilot v danom tiku riadi.

Z triedy ESP sa využíva len metóda ASR na eliminovanie šmyku pri akcelerácii, nakoľko modul Recovery používa brzdu len minimálne.

4.2.3 Testovanie

Po implementácii prototypu modulu Recovery bol tento modul testovaný. Testovanie bolo rozdelené na testovanie samostatných stavov identifikovaných v návrhu. Na testovanie nebol použitý žiadny externý nástroj, úspešnosť či neúspešnosť implementácie posúdil človek na základe toho, že videl, ako sa pilot v stavoch recovery správa.

Pilot mimo trate

Pilotovi, ktorý mal určenú najvyššiu povolenú rýchlosť na bezpečný prechod zákruty, bola táto rýchlosť upravená tak, aby bolo zabezpečené vybehnutie z trate. Rýchlosť bola zvýšená o 50 km/hod, čo bolo dostatočné na sledovanie obnovenia pilota z miesta mimo trate znovu na trať. Boli odhalené drobné chyby, ktoré boli spätne v implementácii upravené.

Špecifické stavy

Testovanie špecifických situácií prebiehalo poloautomaticky. Poloautomaticky preto, lebo pilot bol manuálne nastavený do špecifického stavu a to hrubou úpravou kódu počas niekoľkých tikov. Po uplynutí tikov na umiestnenie pilota do špecifického stavu sa opäť spustilo hlavné riadenie pilota, bežné správanie sa. Avšak hneď sa identifikoval jeden zo špecifických stavov (keďže sa do neho pilot externe dostal) a sledovalo sa plynulé vrátenie sa do normálneho stavu a pokračovanie v jazde. Aj toto testovanie odhalilo niektoré chyby, ktoré boli spätne predmetom implementácie.

4.2.4 Časti modulu Recovery identifikované ako neúspešné

Pilot, nad ktorým preberie riadenie modul Recovery, sa pohybuje so zaradeným rýchlostný stupňom výlučne 1 alebo R.

Pokus 1: Otázkou teda bolo, či je potrebné v recovery móde (stavoch) preraďovať rýchlostné stupne alebo nie. Keďže túto úlohu nebolo možné riešiť samotnou analýzou, pretože povrch mimo trate je takmer nepredvídateľný a správanie sa pilota na ňom je atypické oproti normálnemu správaniu sa na trati. Taktiež bola otázka, či zachovať rýchlostný stupeň po vybehnutí z trate.

Výsledok 1: Radenie v module Recovery sa implementáciou a následným testovaním ukázalo ako úplne nevyhovujúce, pretože pilot preukázal vyššiu stabilitu na povrchu mimo trate so zaradeným prevodovým stupňom 1. To znamená, že aj ponechaný rovnaký rýchlostný stupeň po vybehnutí z trate, aký bol tesne pred vybehnutím, vykazoval nestabilitu a spôsoboval horšie ovládanie pilota.

Pokus 2: Keďže sa predchádzajúca implementácia ukázala ako nepresvedčivá, predmetom na ďalšiu implementáciu bolo preradenie rýchlostného stupňa 1 na 2 a to v prípade, ak sa pilot dostáva z miesta mimo trate priamym pohybom, ktorý smeruje k trati.

Výsledok 2: Aj táto možnosť bola testovaním zamietnutá a spätne vyhodená z implementácie. Je pravdou, že pilot po preradení rýchlostného stupňa na 2 dosiahol vyššiu rýchlosť a tým pádom sa dostal skôr na trať. Avšak vo viacerých prípadoch sa stalo, že vďaka tomu, že rýchlosť bola vyššia, riadenie nedokázalo pilota zosúladiť so smerom trate a dôsledkom toho pilot prešiel cez trať a dostal sa opäť mimo trate. Obnovenie pilota z miesta mimo trate nie je časovo zanedbateľné, preto sme radenie rýchlostných stupňov (okrem 1 a R) v akomkoľvek recovery móde (stave) vylúčili.

4.3 Modul modelu trate

4.3.1 Požiadavky na model

Vytvorenie modulu trate vychádza z potreby autopilota učiť sa informácie o trati z dôvodu ich neskoršieho zužitkovania pre lepší prejazd okruhu. Od modulu na vytváranie modelu trate (rozoznávanie zákrut) je očakávaná schopnosť rozoznať začiatok zákruty a koniec zákruty. Záznam o zákrute musí okrem údajov o časti trate, kde sa zákruta nachádza obsahovať aj údaje, na základe ktorých je možné určiť ako je zákruta prudká, resp. akou rýchlosťou je možné cez zákrutu prejsť.

4.3.2 Reprezentácia trate

Najrealistickejšou reprezentáciou je uloženie trate do určitej dátovej štruktúry vo forme 2D mapy, čiže v pamäti bude uložená postupnosť bodov v dvojrozmernom priestore. Tieto body, určené x-ovou a y-ovou súradnicou predstavujú okraje trate. V prípade úspešnej reprezentácie trate v takomto formáte môžeme uvažovať o využití matematických odhadov ideálnej trajektórie nášho auta. Za optimálnu jazdnú dráhu sa dá považovať určitý pomer medzi najkratšou a najmenej zakrivenou dráhou. V prvom prípade auto prejde v rámci jedného okruhu najkratšiu vzdialenosť, v druhom prípade dosiahneme najväčšiu možnú rýchlosť v danom úseku trate.

Alternatívou je pamätanie si presnej povahy zákrut v jednotlivých úsekoch trate, čiže jej začiatok, koniec a rádius. Tieto informácie predstavujú solídny základ pre učenie autopilota, čiže optimalizáciu prejazdu týmito zákrutami. Povahu zákruty nemusí určovať len jej rádius, ale aj iná hodnota, ktorá konzistentne zachytáva natočenie zákruty.

K vytváraniu modelu trate sa dá pristupovať dvoma základnými spôsobmi alebo ich kombináciou, a to:

- vytváraním modelu trate na základe dát získavaných senzormi, ktoré sú pri preteku ovládaču k dispozícii.
- vytváraním modelu trate na základe akcií, ktoré sú vykonané ovládačom.

4.3.3 Komunikácia rozoznávacieho modulu s ovládačom

Modul modelu trate realizuje výpočty v inom vlákne ako autopilot. Z tohto dôvodu musí byť vyriešená komunikácia medzi týmito vláknami.

Návrh

Od ovládača je vyžadované, aby dával odpovede serveru v čo najkratšom čase, keďže neskoré odpovede majú nepriaznivý vplyv na ovládané vozidlo. Za týmto účelom je vytváranie modelu trate spustené v inom vlákne ako výpočty slúžiace na riadenie. Toto vlákno má nižšiu prioritu vykonávania.

Komunikácia smerom od ovládača k rozoznávaciemu modulu prebieha prostredníctvom jednosmernej fronty. Komunikácie opačným smerom prebieha prostredníctvom iných spôsobov synchronizácie.

Implementácia

Vlákno s vytváraným modelom je označené ako démon a jeho priorita je nastavená na minimálnu. Komunikácia smerom od ovládača k rozoznávaciemu modulu je zabezpečená prostredníctvom kontajnera java.util.concurrent.BlockingQueue<TickInfo>. V triede TickInfo sú prenášané všetky potrebné informácie – hlavne údaje senzorov a odosielanej akcie. Kapacita tohto kontajnera je 1000 prvkov. Zaplnenie kontajnera tejto veľkosti by bez pochýb už predstavovalo príliš veľké oneskorenie a pravdepodobne by nebolo napraviteľné, lebo oneskorenie tohto rozsahu musí mať vážnejšiu príčinu ako dočasné nestíhanie spracovávania. Po naplnení kontajnera sa stáva z posielania ďalších prvkov blokujúce volanie, čo pri neprebiehajúcom spracovávaní informácií o tikoch má za následok zablokovanie ďalšieho vykonávania kódu riadenia.

Ďalšia synchronizácia je realizovaná prostredníctvom statických metód a synchronizačných polí. Hodnotu nastaviteľných senzorov je možné získať aj z vlákna, v ktorom je vytváraný model. Za týmto účelom musia byť vykonané nasledujúce kroky:

1. Vo vlákne s vytváraným modelom je zavolaná metóda, ktorá umiestni požiadavku na zisk hodnoty nastaviteľného senzoru mieriaceho určeným smerom.

2. Vo vlákne vytvárajúcom riadiacu akciu je na základe vytvorenej požiadavky daný príkaz na namierenie senzoru. Toto je vykonané iba za podmienky, že senzor nebol v priebehu poslednej sekundy herného času už použitý (nastaviteľné senzory nie je možné používať častejšie). Pokiaľ bol senzor už použitý, tak je vrátená odpoveď o jeho zaneprázdnenosti.

3. V nasledujúcom tiku je prijatá nová hodnota nastaviteľného senzora, a tá je poslaná žiadajúcemu vláknu. To až teraz pokračuje vo svojom behu, keďže akcia je blokujúca.

4.3.4 Vytváranie modelu trate na základe senzorov ovládača

2D mapa

Prvým návrhom tohto modulu bolo ukladanie aktuálnej pozície auta ako bodu v 2D priestore. Body predstavujúce okraje trate sa určia ako dosah dvoch senzorov kolmých na pozdĺžnu os auta. Tento postup sa ukázal byť nerealizovateľný z dôvodu neexistencie pevného bodu, od ktorého by bolo možné odvodiť aktuálnu pozíciu a smer auta. Všetky dostupné senzory totiž poskytujú len údaje súvisiace s aktuálnou polohou, napríklad uhol k osi trate, k dispozícii však nie je žiaden referenčný bod, vzhľadom ku ktorému by bolo možné určiť relatívnu pozíciu.

Model zákrut z diaľkomerov

Návrh

Po ustúpení od modelu pomocou 2D mapy sme vytvorili návrh modulu, ktorý by na ukladanie jednotlivých zákrut využíval diaľkomery. Vybrali sme po tri senzory na obidvoch stranách auta, ktoré na tento účel využijeme. Na základe údajov z týchto senzorov dostaneme tri body, ktoré potrebujeme na to, aby sme nimi preložili kružnicu. Jej polomer predstavuje rádius zákruty.

Implementácia

Implementácia prebehla podľa návrhu, za použité senzory sme vybrali diaľkomery zvierajúce s priečnou osou auta uhly 30°, 40° a 50°. X-ové súradnice bodov sme vypočítali ako kosínus príslušného uhla vynásobený vzdialenosťou získanou z diaľkomeru, y-ové súradnice ako sínus daného uhla krát získaná vzdialenosť. Z týchto troch bodov sme rádius zákruty získali vzťahom:

$$r = (a * b * c) / (4 * \sqrt{(s * (s - a) * (s - b) * (s - c))}), \text{ kde}$$

$$s = (a + b + c) / 2$$

$$a = \sqrt{((x1 - x2)^2 + (y1 - y2)^2)}$$

$$b = \sqrt{((x2 - x3)^2 + (y2 - y3)^2)}$$

$$c = \sqrt{((x3 - x1)^2 + (y3 - y1)^2)}$$

Testovanie

Implementovaný postup sme testovali pomocou výpisov polomerov v jednotlivých úsekoch trate a ich porovnaním s reálnou hodnotou získanou zo zdrojového súboru trate. Hodnoty sú takmer identické pri základnej simulácii pretekov.

Problém nastal pri aplikovaní umelého šumu využívaného v rámci súťaže Championhsip, ktorý sa okrem ďalších senzorov vzťahuje aj na diaľkomery. Pri testovaní v týchto podmienkach sa dostavili neželané výsledky predstavujúce nepoužiteľné dáta. Dôvodom je vysoká citlivosť použitej metódy na chybné vstupy. Pre určenie polomeru kružnice sú potrebné minimálne jej tri body a chybné súradnice čo i len jedného z nich vedú k nepoužiteľným výsledkom, keďže získaná hodnota je v drvivej väčšine prípadov veľmi vzdialená od tej skutočnej.

Na potlačenie šumu sme použili dva spôsoby. Prvým bola zmena rozloženia diaľkomerov pri inicializácii autopilota, čiže nastavenie viacerých senzorov na ten istý smer. Snahou bolo získanie presnejšej informácie pomocou priemeru viacerých hodnôt. Druhým postupom bolo navzorkovanie väčšieho množstva dát za pomalej jazdy a následné určenie polomeru zákruty ako priemeru určitého intervalu hodnôt.

Ani jeden z týchto prístupov však nepotlačil šum do takej miery, aby bolo možné získať informácie o trati, ktoré by boli ďalej použiteľné.

Model zákrut z osi trate

Návrh

Pri tomto návrhu sme kvôli predchádzajúcej skúsenosti zobrali do úvahy len senzory, ktoré nie sú ovplyvnené šumom. Pre tento účel sme navrhli použiť informácie o vzdialenosti od začiatku trate a polohy v rámci šírky trate. Jedná sa o senzory *getDistanceFromStartLine* a *getTrackPosition*. Myšlienka zostala rovnaká ako pri diaľkomeroch, čiže vypočítanie polomeru zákruty na základe troch bodov. V tomto prípade sa však jedná o body ležiace na osi trate. Postup je použiteľný na takých miestach, kde je natočenie volantu nulové.

Implementácia

Implementácia vychádza z návrhu, body slúžiace na výpočet sú potom definované nasledovne. X-ovú súradnicu predstavuje vzdialenosť od začiatku trate, t.j. údaj získaný zo senzora *getDistanceFromStartLine*. Y-ovú súradnicu predstavuje vzdialenosť od stredu(osi) trate, túto hodnotu dostaneme vzťahom:

getTrackPosition * trackWidth / 2 , kde

trackWidth predstavuje šírku trate nameranú zo senzorov kolmých na pozdĺžnu os auta pri štarte tréningu.

Testovanie

Testovanie prebiehalo rovnakým spôsobom ako pri modeli zákrut z diaľkomerov. Narazili sme však na dva problémy. Prvý priamo súvisí s použitou metódou. Vzdialenosť od začiatku trate síce nepodlieha šumu, avšak táto hodnota nie je dostatočne presná na určenie y-ovej súradnice hľadaného bodu. Vyjadruje totiž vzdialenosť pozdĺž osi trate, čo nekorešponduje presne s polohou auta, ktorá môže byť ľubovoľná v rámci šírky trate.

Navyše použitie tejto metódy je vysoko závislé na spôsobe riadenia auta, keďže vyžaduje úseky trate prejdené s nulovým zatočením volantu. Z týchto dôvodov sme upustili aj od tohto spôsobu reprezentácie zákrut.

4.3.5 Vytváranie modelu trate na základe vykonaných akcií

Model trate vytvorený na základe vykonaných akcií je do veľkej miery závislý na spôsobe jazdy ovládača. Nevýhodou tohto prístupu je, že pokiaľ ovládač nejde presne po strede trate (túto podmienku žiadny z použitých ovládačov nespĺňa), tak nie je možné vytvoriť presný model. Od ovládača je požadované, aby trať prešiel absolútne bezpečne, t.j. aby sa vyhol akémukoľvek vybehnutiu z trate alebo kolízii. Tieto totiž ovplyvnia pohyb vozidla, čo vnesie do modelu chybné údaje.

Ďalšou prekážkou sú rôzne spôsoby prechodu zákrutami. Nájsť všeobecne použiteľný spôsob rozoznávania zákrut pre všetky spôsoby ovládania sa ukázalo byť príliš náročné na implementáciu. Prakticky schodnejšou cestou je vytvorenie algoritmu rozoznávajúceho zákruty pri použití konkrétneho algoritmu riadenia. Za týmto účelom boli vytvorené tri hlavné verzie rozoznávania zákrut, ktoré sú nižšie popísané.

1. verzia – ovládač Prototype idúci podľa najdlhšieho senzora

Návrh

Špecifikom ovládača Prototype idúceho výlučne podľa najdlhšieho senzora je, že zákruty neprechádza plynule, ale prudšie zatáča iba na malých úsekoch, a potom zase prejde malý úsek rovno, čo v jednej zákrute zopakuje niekoľko krát. Zaznamenávané sú aj rovné úseky trate, a to ako rovinky. Model trate je vytváraný pre tento typ riadenia nasledovne:

Zatočenia volantom väčšie ako určená konštanta sú spojené do väčších celkov, pri ktorých sú sledované viaceré hodnoty. Jedná sa hlavne o:

- priemernú prudkosť zatočenia
- maximálnu prudkosť zatočenia
- začiatok a koniec zákruty
- smer zákruty

Na základe pozorovaní sú po sebe nasledujúce celky spojené do ešte väčších celkov, ktoré sú považované za zákruty. Spájané môžu byť iba celky smerujúce tým istým smerom a rovinky. Kedy spojiť celky zatáčajúce rovnakým smerom prerušené rovinkou záleží od pomerov ich dĺžok k rovinke. Konkrétne hodnoty sú určené experimentálne.

Implementácia

Rozoznávanie zákrut je implementované podľa návrhu. Náročnosť algoritmu je lineárna vzhľadom na dĺžku zákrut, ktoré sú modulom rozoznané.

Testovanie

Modul bol vyskúšaný na viacerých tratiach, hlavne CG Speedway a Aalborg. Modul dokáže správne rozoznávať zákruty, avšak ich kategorizácia nie je konzistentná. Ďalej spôsob ovládania na ktorom je založený nie je vhodný na vytváranie modelu trate, keďže viacero zákrut je ignorovaných alebo prejdených špecifickým spôsobom, čo musí nevyhnutne vyústiť do nesprávneho modelu. Tento model by mohol byť využiteľný pravdepodobne len pre ovládač na ktorom je založený.

Pre otestovanie bola implementovaná verzia autopilota (ovládača) Prototype (riadiaceho sa podľa najdlhšieho senzora). Využitie údajov z modelu napomohlo k zlepšeniu celkového času ovládača na trati.

2. verzia – ovládač SimpleDriver

Návrh

Model je vytváraný na základe otáčania volantu. Na potlačenie ojedinelých prudkých otočení volantom sú viaceré po sebe nasledujúce záznamy tikov spriemerované do jednej hodnoty. Tieto hodnoty – sektory – sú ďalej spájané do záznamov o zákrutách. Sledované sú tie isté položky ako v predchádzajúcom spôsobe rozoznávania trate. Na vylepšenie kvality dosiahnutých výsledkov môžu byť použité viaceré techniky:

- Minimálny počet sektorov potrebných na určenie zákruty – zabraňuje rozoznaniu náhlych trhnutí volantom alebo korekciám smeru na rovine, aby boli rozoznané ako zákruty.
- Nastavenie dĺžky sektora do ktorého sú spájané jednotlivé natočenia. V malých dĺžkach sa prejavia aj malé výkyvy, ktoré nie sú súčasťou dlhodobejšieho trendu, takže jedna zákruta môže byť rozdelená do viacerých.
- Nastavenie hodnoty, ktorá je považovaná za zatáčanie. Nižšie hodnoty majú za následok zaznamenanie zákruty aj pri malých korekciách smeru, vyššie spôsobujú ignorovanie zákrut s malým zatočením.

V implementácií je potrebné vyvážiť nastavenie jednotlivých parametrov, a to tak, aby bola metóda použiteľná na všetkých typoch tratí a povrchoch.

Jedným z alternatívnych prístupov je použitie laterálneho zrýchlenia namiesto zatočenia volantom. Toto je taktiež možné skombinovať s rýchlosťou vozidla, čo umožňuje vytvárať model na základe vektoru pohybu.

Implementácia

Rozoznávanie zákrut je implementované podľa návrhu. Náročnosť algoritmu je lineárna vzhľadom na dĺžku zákrut, ktoré sú modulom rozoznané.

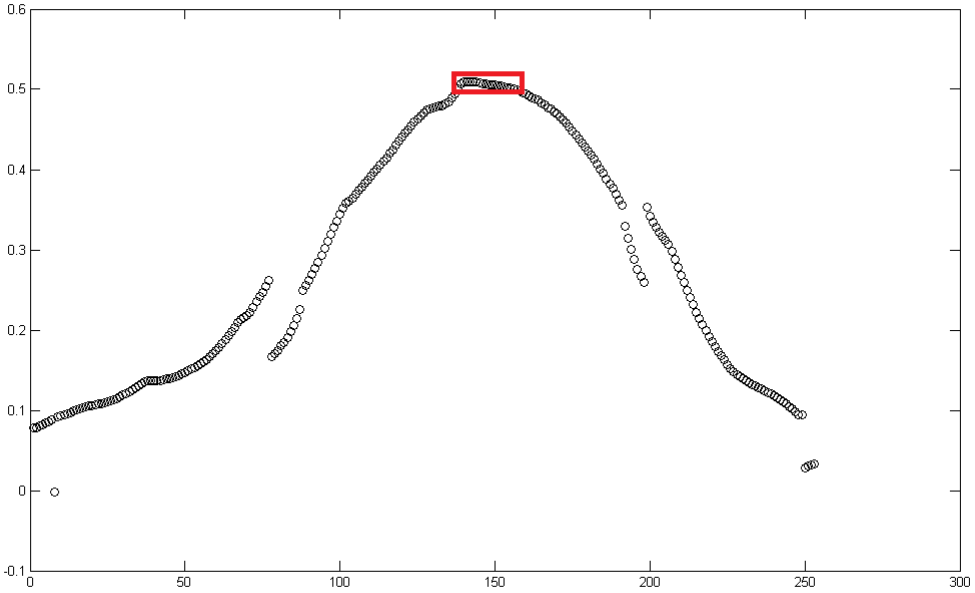
Testovanie

Modul bol vyskúšaný na viacerých tratiach a povrchoch. Modul dokáže správne rozoznávať zákruty, ale niektoré – hlavne dlhé zákruty, ktoré majú veľký rádius, modul buď nerozozná alebo rozozná iba časť z nich. Vytváraná kategorizácia nie je konzistentná. Toto bolo potvrdené aj pokusom o vytvorenie funkcie na kategorizáciu zákrut prostredníctvom nástrojov Matlabu.

3. verzia – ovládač PrototypeFirstTime

Návrh

Modul určuje začiatok zákruty na základe informácie poslanej od riadenia. V prvej verzií túto informáciu využíva aj na určenie konca zákruty. V ďalších verziách je koniec zákruty určený rozoznaním náhleho poklesu otáčania volantu pozorovaného u ovládania PrototypeFirstTime.



Obr. 08 *Vrchol zákruty*

Modul rozoznáva tzv. vrchol zákruty, čo je postupnosť málo sa meniacich hodnôt točenia volantom, ktoré nasledujú priamo po sebe a predstavujú čo najprudšie otočenie volantom Obr. 08. Súčasťou záznamu o zákrute je aj začiatok zákruty a jej koniec.

Implementácia

Rozoznávanie konca zákruty podľa prudkého poklesu otáčania volantom nie zatiaľ implementované. Zvyšok rozoznávania zákrut je implementované podľa návrhu. Náročnosť algoritmu je lineárna vzhľadom na dĺžku zákrut, ktoré sú modulom rozoznané.

Pre určovanie vrcholu zákruty je určená jeho minimálna dĺžka a maximálny rozsah prvkov, ktoré sa v tomto rozsahu nachádzajú.

Testovanie

Model bol otestovaný na obmedzenom počte tratí. Zákruty boli vo väčšine vykonaných testoch rozoznané spoľahlivo. Na rovine s miernym zatočením model spravil viacero záznamov o malých zákrutách, avšak toto správanie môže byť napravené. Konzistencia

kategorizácie zákrut nebola pri vykonaných pozorovaniach ideálna, ale poskytuje dobrý základ pre ďalšiu prácu na tomto modely.

4.4 Modul podpory riadenia pred vjazdom do zákrut

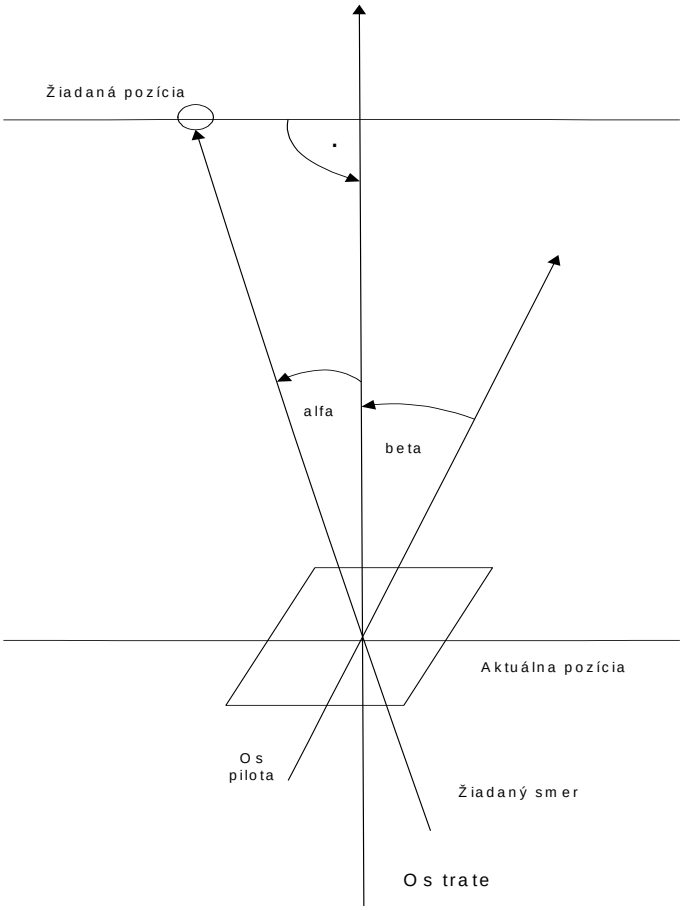
Úlohou tohto modulu je upraviť smer jazdy pilota pred zákrutou na základe predchádzajúcej detekcie tejto zákruty. Podľa informácií o zákrute, t. j. začiatočnej, koncovej pozícii a veľkosti natočenia zákruty je potrebné určiť pozíciu, na ktorej sa pilot pri vjazde do danej zákruty musí nachádzať a podľa toho korigovať natočenie kolies.

4.4.1 Návrh

Pri návrhu musíme zohľadniť viacero faktorov, ktoré sú dôležité pri rozhodovaní o použití podpory riadenia pred zákrutou. Použitie podpory riadenia pred zákrutou je opodstatnené, keď je pred zákrutou primerane dlhý relatívne rovný úsek, na ktorom je možné korigovanie jazdy pilota na určenú pozíciu bez náhlych a veľkých zmien natočenia kolies, čo by mohlo spôsobiť šmyk a pretočenie pilota. Preto sú pred použitím podpory riadenia dôležité najmä dva faktory, a to:

- tvar aktuálneho úseku trate (rovina, zákruta)
- vzdialenosť do začiatku najbližšej zákruty

V samotnej podpore riadenia berieme do úvahy viacero faktorov, od ktorých sa odvíja možnosť ovplyvniť natočenie pilota na žiadanú pozíciu (náhľad možnej situácie na Obr. 09).

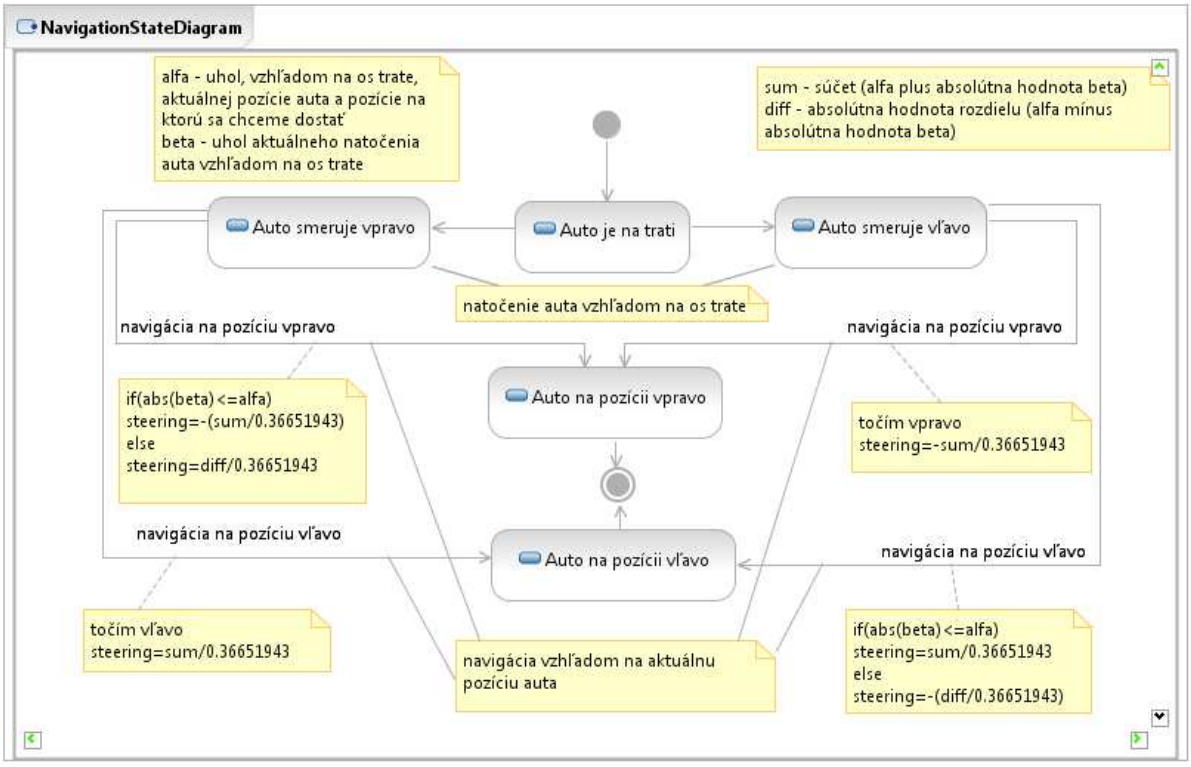


Obr. 09 *Ukážka situácie*

Spomenuté faktory sú nasledovné:

- uhol pilota s osou trate
- aktuálna pozícia vzhľadom na pozíciu, na ktorú sa chceme dostať
- uhol medzi aktuálnou a žiadanou pozíciou vzhľadom na os trate

Na základe týchto faktorov sa pilot môže dostať do viacerých stavov (Obr. 10), pričom korekcia natočenia je špecifická v každom stave.



Obr. 10 *Stavový diagram podpory riadenia*

Ako je ilustrované v stavovom diagrame na Obr. 10, ak je pilot na trati, môže sa nachádzať v dvoch hlavných stavoch, pričom z každého z nich prechádza do ďalších dvoch stavov, v závislosti od polohy aktuálnej a žiadanej pozície. Dôležité sú v tomto prípade uhly *alfa* a *beta* (Obr. 09), t. j. uhol spojnice aktuálnej pozície pilota a žiadanej pozície s osou trate resp. uhol osi pilota s osou trate. Výsledné natočenie pilota sa teda vypočíta buď sumou týchto uhlov alebo ich rozdielom, pričom počítame s ich absolútnymi hodnotami, keďže podľa špecifikácie TORCS môžu nadobúdať aj záporné hodnoty. Výsledný uhol je potrebné predeliť konštantou *0.36651943,* čím získame konečné natočenie kolies pilota, ktoré podľa pozície, t. j. žiadaného smerovania použijeme v zápornej resp. v kladnej hodnote v závislosti od toho, či má pilot smerovať vpravo resp. vľavo.

4.4.2 Implementácia

Na základe návrhu a identifikovaných stavov bola podpora riadenia implementovaná nasledovne:

- pilot natočený vpravo
    - pozícia pilota vľavo vzhľadom na žiadanú pozíciu – v tomto prípade výsledné natočenie kolies vypočítame vzťahom: $\frac{alfa+|beta|}{0.36651943}$

- pozícia pilota vpravo vzhľadom na žiadanú pozíciu – tu záleží od veľkosti uhlov *alfa* a *beta*, t. j. od toho, či pilot smeruje vľavo alebo vpravo resp. či je uhol *alfa* väčší alebo menší ako uhol *beta*
  - *alfa>=beta* - $-\frac{alfa+|beta|}{0.36651943}$
  - *alfa<beta* - $\frac{alfa-|beta|}{0.36651943}$

- pilot natočený vľavo
  - pozícia pilota vľavo vzhľadom na žiadanú pozíciu - tu taktiež záleží od veľkosti uhlov *alfa* a *beta*, t. j. od toho, či pilot smeruje vľavo alebo vpravo resp. či je uhol *alfa* väčší alebo menší ako uhol *beta*
    - *alfa>=beta* - $\frac{alfa+|beta|}{0.36651943}$
    - *alfa<beta* - $-\frac{alfa-|beta|}{0.36651943}$
  - pozícia pilota vpravo vzhľadom na žiadanú pozíciu - v tomto prípade výsledné natočenie kolies vypočítame vzťahom: $-\frac{alfa+|beta|}{0.36651943}$

Riešenie bolo implementované v dvoch metódach, a to *computeAngle(SensorModel sm, double positionDistance, double position)*, čo je metóda, ktorá vypočíta uhol medzi aktuálnou pozíciou auta a bodom na trati pred autom, a metóda *navigateToPosition (Action action, SensorModel sm, **double** positionDistance, **double** position)*, ktorá na základe uhla predchádzajúcej metódy nastaví natočenie kolies.

4.4.3 Integrácia a testovanie

Otestovanie implementovaných metód bolo vykonávané postupne, počas implementácie. Testovacie scenáre boli pre rôzne štartovacie a cieľové pozície, pričom pilot úspešne korigoval smer k žiadanej pozícii.

V prvej fáze prebiehalo testovanie správneho výpočtu uhla medzi aktuálnou a cieľovou pozíciou. Toto testovanie bolo realizované použitím pomocných výpisov, pričom boli dosahované požadované výsledky zodpovedajúce zadaným pozíciám. Uhol sa s približujúcou pozíciou podľa predpokladov zväčšoval.

Po úspešnom otestovaní výpočtu uhlov prebiehalo testovanie korekcie presunu na žiadanú pozíciu na základe vypočítaných uhlov. Toto bolo realizované na rovnom úseku trate tak, že pilot bol nastavený na konkrétnu pozíciu resp. stranu trate, z ktorej sa mal dostať na požadovanú pozíciu, toto sa úspešne podarilo, čo bolo overené pomocnými výpismi a aj

vizuálnou kontrolou. Ako problémový sa v rámci testovania javí finálny vypočítaný steering, ktorý má veľkosť podľa vypočítaného uhla. V prípade keď je uhol veľmi veľký, je veľký aj výsledný steering, čo spôsobuje prudké vytočenie pilota na trati. Vzhľadom na matematickú presnosť výpočtu uhlov a možné ďalšie použitie je optimalizácia steeringu vykonávaná až pri použití metódy podpory riadenia v riadení pilota.

Integrácia podpory riadenia závisí od informácií o modeli trate naučených počas prejazdu prvého kola. Prostredníctvom tohto modelu sú k dispozícii všetky identifikované zákruty spolu s informáciami o začiatkoch, koncoch ako aj ostrosti jednotlivých zákrut. Pri integrácii a rozhodovaní o nasadení podpory riadenia boli rozhodujúce nasledujúce faktory:

- ukončené prvé kolo – naučené zákruty
- dostatočná vzdialenosť pilota pred zákrutou – zamedzenie prudkým zmenám pohybu
- celkové zakrivenie zákruty – rozhodujúce pri výpočte cieľovej vstupnej pozície pred zákrutou
- orientácia zákruty – vpravo alebo vľavo

Testovanie integrácie odhalilo predpokladaný problém, a to nedostatočne plynulý prejazd na požadovanú pozíciu, čo bolo spôsobené výpočtom absolútneho steeringu vzhľadom na uhol medzi aktuálnou a cieľovou pozíciou. Toto bolo úspešne odstránené optimalizáciou veľmi nízkych hodnôt steeringu na 0 a naopak veľmi vysokých hodnôt na 0.2, čo odstránilo spomínaný nedostatok

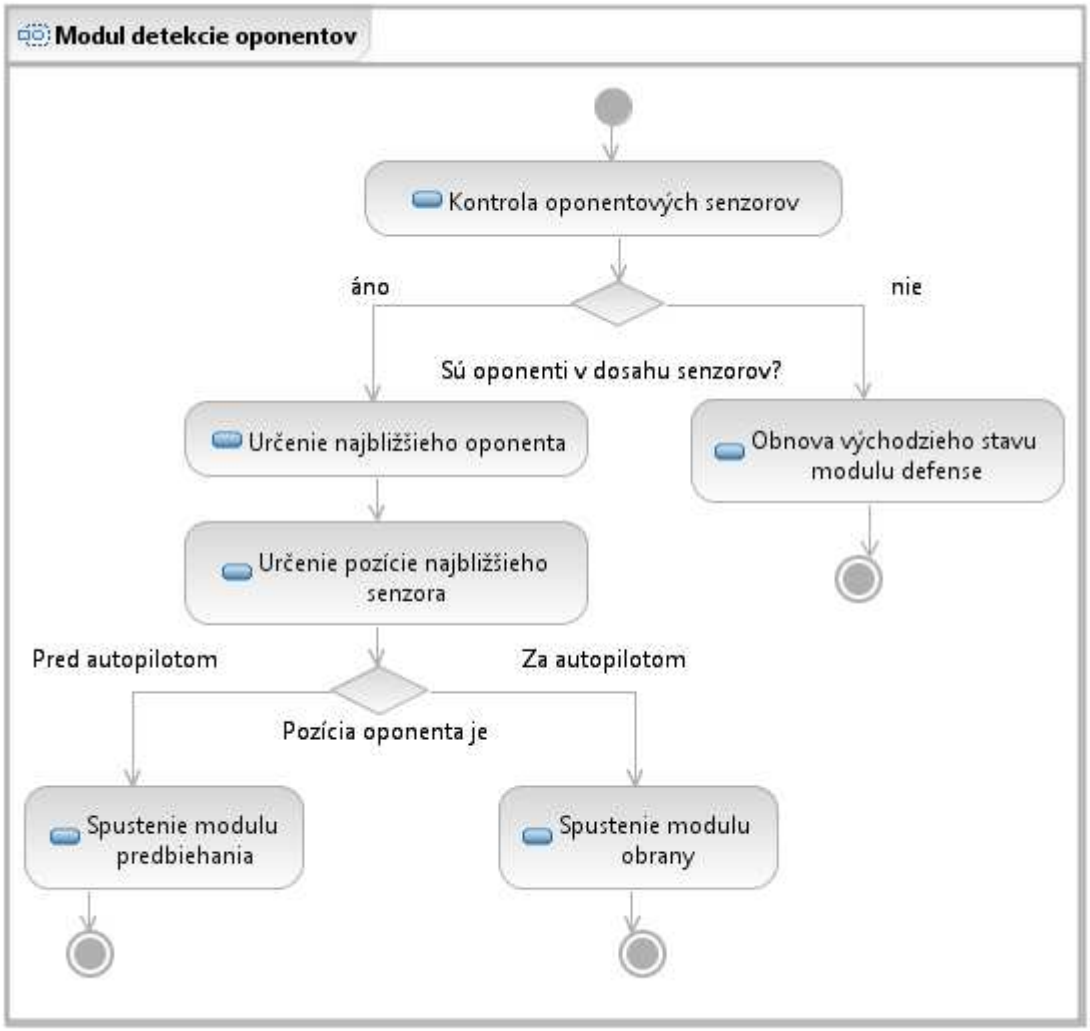
Ďalším problémom bol prejazd zákrut nasledujúcich tesne za sebou, kedy bola podpora riadenia nasadená na minimálny čas, pričom účinok bol viac negatívny ako pozitívny, kvôli prudkému vychýleniu pilota z jazdnej dráhy. Tento problém bol odstránený obmedzením podpory riadenia na veľmi krátkych vzdialenostiach.

Posledný problém vyplýval z nepresného určenia začiatku zákruty v modeli, čo bolo optimalizované a testované posunom bodu začiatku zákruty dopredu resp. dozadu. Nakoniec bol však problém vyriešený optimalizáciou vytvárania modelu trate, čím sa odstránila potreba posunu začiatku zákruty.

4.5 Modul detekcie oponentov

Modul detekcie oponentov (MDO) slúži na detekciu oponentov v blízkosti vozidla. Stavový diagram možno vidieť na Obr. 11. Takáto detekcia je použiteľná dvojako:

1. ako obranný mechanizmus
2. ako mechanizmus na predbiehanie.



Obr. 11 *Modul detekcie oponentov*

Podstata MDO spočíva v tom, že pracuje na najnižšej úrovni. Na základe informácií získaných zo senzorov modifikuje akciu nastavenú ostatnými modulmi (t.j. line assist) podľa svojich potrieb.

Hlavnou úlohou modulu detekcie oponentov je detekcia oponentov a určenie ich pozícií voči autopilotovi. Na základe pozície oponenta tento modul zavolá modul predbiehania alebo modul obrany.

Kvôli zníženiu zaťažovania procesora, sa prechádza pole senzorov detekujúcich oponentov iba každých 20 tikov, t.j. približne 5 krát za sekundu v dobe, kedy autopilot nedeteguje žiadneho oponenta. Pri rozdielnych rýchlostiach nášho auta a oponenta to môže znamenať, že oponenta nemusíme detegovať na hranici dosahu senzorov, ale môže sa stať, že oponent bude detekovaný približne 10 m za týmito hranicami. To znamená, že oponent je detekovaný najmenej v 190 m vzdialenosti od nášho pilota. Táto vzdialenosť je postačujúca na bezpečné vykonanie blokovacích manévrov.

Vstupy:

- Action – akcia určená vyššími modulmi
- SensorModel – stav auta na trate

Výstupy:

- Action – modifikovaná akcia, ktorá je následne posielaná pre servera

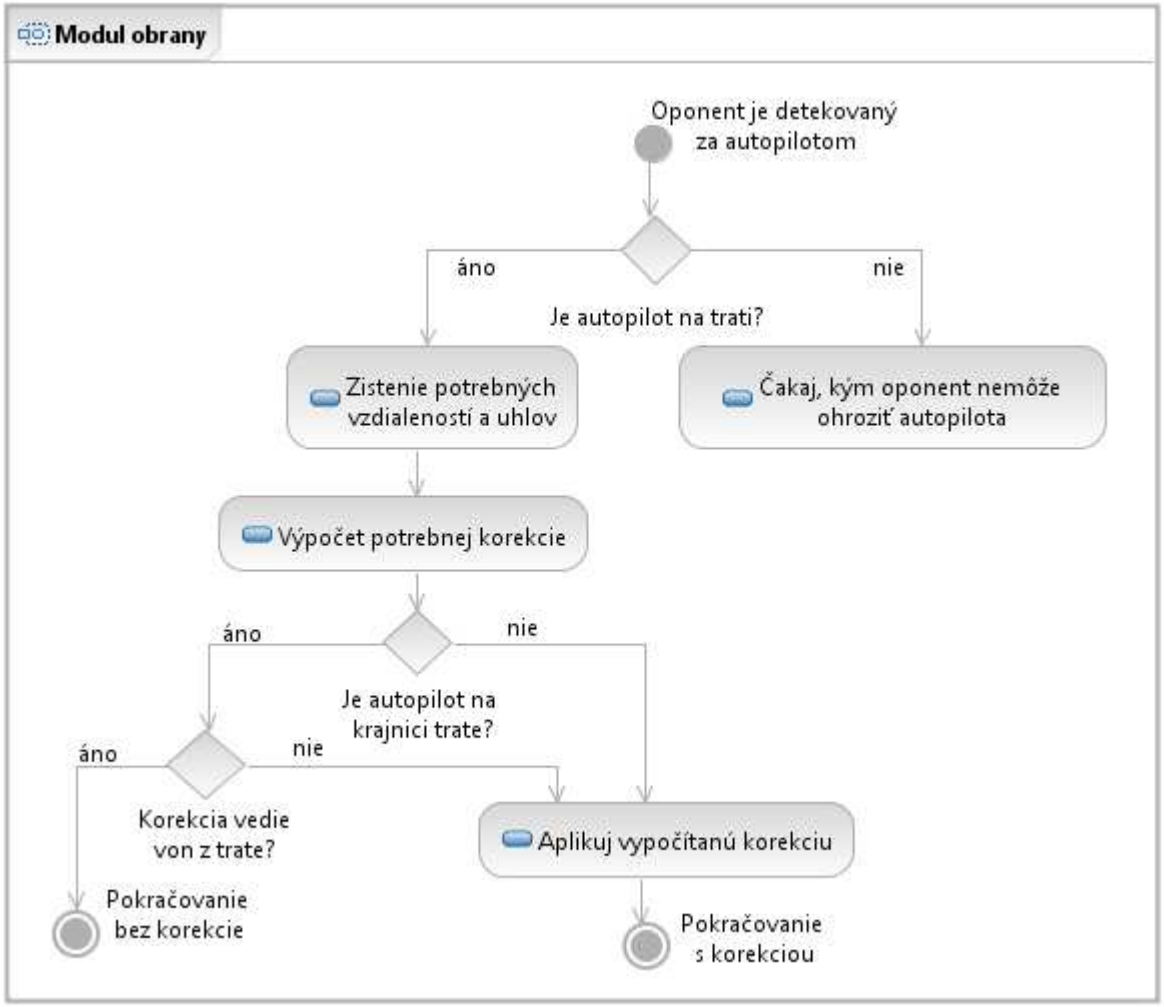
Možné vylepšenia:

MDO pozíciu oponenta určí na základe toho, ktorým senzorom ho vnímal, ale nezohľadňuje zatočenie, resp. pozíciu samotného vozidla. Možným vylepšením je zohľadnenie stavu a pozície vozidla pri určení pozície oponentov.

4.5.1 Modul obrany

Modul obrany (MObr) slúži na obranu voči oponentom, je zavolaný modulom detekcie oponentov vtedy, keď oponent je detekovaný za vozidlom. Činnosť modulu je znázornená na Obr. 12. Typy obrán môžu byť rôzne:

1. obrana voči poškodeniu
2. obrana voči predbiehaniu



***Obr. 12** Modul obrany*

Vstupy:

- Action – akcia určená vyššími modulmi
- SensorModel – stav auta na trate

Výstupy:

- Action – modifikovaná akcia, ktorá je následne posielaná pre servera

Obrana voči poškodeniu

Táto obrana je aktivovaná vtedy, keď modul obnovenia je aktívny a modul detekcie oponentov deteguje oponentov za vozidlom. V prípade, že je detekovaný oponent, je určená jeho vzdialenosť. Ak vzdialenosť vozidla je menšia než určitá hraničná vzdialenosť, modul obrany pozastaví obnovenie. Obnovenie bude znovu spustené, ak žiadne prichádzajúce vozidla neohrozujú autopilota.

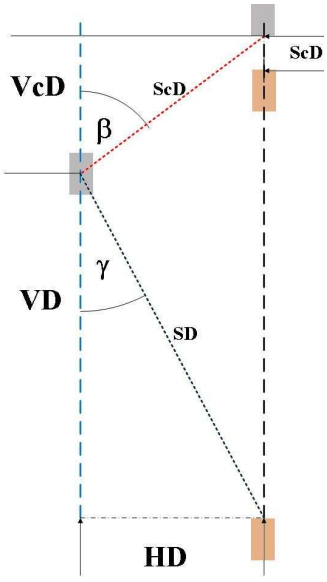
Možné vylepšenia:

Ten typ obrany je možné vylepšiť:

- zohľadnením informácií o trati
- odhadnutím trajektórie oponenta i autopilota a ich následné porovnanie
- odhadnutím času potrebného na obnovenie a jeho porovnanie s časom potrebným na dobehnutie oponenta

Obrana voči predbiehaniu

Táto obrana je aktivovaná vtedy, keď autopilot je na trati a za ním jazdí oponent. Základná koncepcia spočíva v určení miery modifikovania zatočenia volantu vstupnej akcie.



Obr. 13 *Schéma pre výpočet miery zatočenia volantu*

Na Obr. 13 je znázornená schéma, na základe ktorej je vypočítané požadované zatočenie vozidla(β). Sivé obdĺžniky znázorňujú počiatočnú a požadovanú pozíciu autopilota, kým oranžové obdĺžniky znázorňujú aktuálnu a očakávanú pozíciu oponenta.

1. uhol γ a vzdialenosť SD sú známe:
    a. SD je vzdialenosť oponenta, t.j. dĺžka senzora
    b. γ je uhol senzoru, je možné získať nasledovne: γ = IndexSenzora * 10°.
2. HD horizontálna vzdialenosť oponenta: HD = SD*cos γ.
3. VD vertikálna vzdialenosť oponenta: VD = SD*sin γ.
4. je určená aktuálna rýchlosť oponenta (SP) – relatívna rýchlosť. Je určená na základe informácií zhromaždených počas predchádzajúcich Tickov nasledovne: VD_{Δ} = VD – VD_{predch}.
    a. VD_{predch} – je vertikálna vzdialenosť oponenta v predchádzajúcom Tiku.
    b. časový rozdiel medzi aktuálnym a predchádzajúcim Tikom: $T_{\Delta}=T_{cur}-T_{predch}$.
    c. $SP=VD_{\Delta}/T_{\Delta}$.
5. je odhadnutý čas dobehnutia (T_C) nasledovne:
    a. T_C=VD/SP
6. z aktuálnej rýchlosti je vypočítané V_cD = SP_{act} * T_C, kde SP_{act} je aktuálna rýchlosť auto-pilota.
7. je určený uhol β: tgβ=HD/V_CD. Z toho vieme určiť β: β = arctg (tgβ).

Smer zatočenia volantu (znamienko pred ST) závisí od toho, či zatáčame doľava (-) alebo doprava (+). Na rýchlosť zatáčania, tj. na veľkosť ST vplýva pozícia vozidla na trati, t.j. vzdialenosť vozidla od okrajov trate. Čím bližšie je vozidlo k okraju tým viac sa mení zatočenie. Zatočenie sa mení vtedy keď je uhol β nenulový.

Korekciu zatočenia nevykonávame v prípade, že:

1. vozidlo je na pravom okraji a modifikácia by smerovala napravo, tj. von z trate
2. vozidlo je na ľavom okraji a modifikácia by smerovala naľavo, tj. von z trate

Možné vylepšenia:

Modifikácie vykonané týmto modulom sú vykonané reakčne, t.j. zohľadnením iba aktuálneho stavu vozidla. Možným vylepšením je zohľadnenie trate.

4.6 Modul anti-blokovacieho systému (ABS)

Úlohou tohto modulu je kontrola činnosti brzdovej sústavy s cieľom zabrániť úplnému zablokovaniu kolies v prípade brzdenia na klzkom povrchu, čo spôsobí šmyk a neovládateľnosť vozidla. ABS systém sleduje počas brzdenia pohyb kolies a v prípade, že zistí, že niektoré z kolies sa na vozovke začína šmýkať, zmenší alebo úplne preň vypne brzdnú silu. Tým sa snaží medzi vozovkou a pneumatikou udržať valivé trenie a zabrániť klznému treniu, čo dáva predpoklad lepšej ovládateľnosti vozidla. Nevýhodou ABS asistencie je, že v niektorých prípadoch vzniká dlhšia brzdná dráha.

4.6.1 Návrh

Základný princíp systému spočíva postupným pridávaním veľkosti hodnoty stlačenia brzdového pedálu, pokým sa neprekročí hodnota prešmyknutia λ. Štandardne sa uvádza interval 0 až 20 percent prešmyknutia za bezpečné. V prípade prekročenia intervalu dochádza k strate kontroly ovládania vozidla. V prípade prekročenia hodnoty λ sa veľkosť stlačenia brzdového pedálu vráti do predchádzajúceho stavu, kedy nedošlo k prešmyknutiu. Regulovanie intenzity stlačenia brzdového pedálu je po hodnotne 0.1.

4.6.2 Implementácia

Na základe návrhu je potrebné určiť hodnotu prešmyknutia λ.

Pre výpočet prešmyknutia slúži formula d, v ktorej vystupujú veličiny ako aktuálna rýchlosť auta (formula e) a uhlová rýchlosť z kolies auta (formula f) .

(d) $$\lambda = 1 - \frac{vU}{vF} * 100\,[\%]$$

(e) $$vF = \frac{hodnotazosenzora}{3.6}\,[m/s]$$

(f) $$vU = \frac{\sum_{i=0}^{3} otáčkyKolesa_i * rádiusKolesa_i}{4}\,[m/s]$$

V prípade, že hodnota λ je menšia ako 20%, tak sa zvýši hodnota brzdy o 0.1 a táto hodnota sa nastaví aj do pomocnej premennej slúžiacej pre možný návrat v ďalšom tiku. Ak hodnota λ je väčšia ako 20%, tak hodnota brzdy sa nastaví na hodnotu z predchádzajúceho tiku. Následne sa hodnota pomocnej premennej zmenší o 0.1. Ak hodnota brzdy presiahne hodnotu 1, nastaví sa na 1.0. Podobným spôsobom, ak hodnota brzdy klesne pod hodnotu 0, nastaví sa na 0.0.

Riešenie bolo implementované do metódy filterABS, kde vstupom je sensors model a hodnota brzdy z predchádzajúceho tiku. Táto metóda sa nachádza v triede ESP.

4.6.3 Testovanie

Počas testovania navrhnutého systému sme dospeli k záveru, že pri uvádzanej štandardnej hodnote vznikajú pri type povrchu „Dirt“ súvislé brzdné dráhy. Z tohto dôvodu sa bezpečná hodnota prešmyknutia posunula na hodnotu 0 až 15 percent.

4.7 Modul regulácie preklzávania (ASR)

Úlohou tohto modulu je zabránenie pretáčania pohonných kolies pri zrýchľovaní alebo rozbiehaní sa vozidla. Za pomoci tohto systému nenastanú neovládateľné situácie pri akomkoľvek zrýchľovaní.

4.7.1 Návrh

Základný princíp systému vychádza z podobného spôsobu ako pri systéme ABS, len rozdiel je v tom, že sa nepridáva veľkosť hodnoty stlačenia brzdového pedálu ale veľkosť plynového

pedálu, pokým sa neprekročí hodnota prešmyknutia λ. Taktiež ako pri ABS, tak aj pri systéme ASR sa štandardne uvádza interval 0 až 20 percent prešmyknutia za bezpečné. V prípade prekročenia intervalu dochádza k strate kontroly ovládania vozidla. V prípade prekročenia hodnoty λ sa veľkosť stlačenia plynového pedálu vráti do predchádzajúceho stavu, kedy nedošlo k prešmyknutiu. Regulovanie intenzity stlačenia plynového pedálu je po hodnotne 0.1.

4.7.2 Implementácia

Na základe návrhu je potrebné určiť hodnotu prešmyknutia λ.

Pre výpočet prešmyknutia slúži formula g, v ktorej vystupujú veličiny ako aktuálna rýchlosť auta (formula h) a uhlová rýchlosť z kolies auta (formula i) .

(g) $$\lambda = \left(\frac{vU}{vF} - 10\right) * 100\ [\%]$$

(h) $$vF = \frac{hodnota\ zo\ senzora}{3.6}\ [m/s]$$

(i) $$vU = \frac{\sum_{i=2}^{5} otáčkyKolesa_i * rádiusKolesa_i}{2}\ [m/s]$$

Spôsob regulovania plynového pedálu je takmer rovnaký ako pri systéme ABS. Jediný rozdiel je v tom, že sa nezvyšuje alebo neznižuje hodnota brzdového pedálu ale plynového pedálu.

Riešenie bolo implementované do metódy filterASR, kde vstupom je sensors model a hodnota plynu z predchádzajúceho tiku. Tak ako metóda filterABS, tak aj táto metóda sa nachádza v triede ESP.

4.7.3 Testovanie

Pri testovaní vznikali podobné situácie ako pri systéme ABS. Taktiež pri type povrchu „Dirt“ sa ukázalo, že 20 percentná hranica prešmykovania nie je bezpečná. Pri nastavení hranice prešmykovania na hodnotu 15 percent vznikalo pretáčanie kolies už len pri výjazdoch z trávnatých miest. Z tohto dôvodu bola hranica prešmykovania určená na 10 percent.

4.8 Modul Telemetria

4.8.1 Návrh

Motiváciou pre vytvorenie modulu telemetrie je snaha o vytvorenie aplikácie, ktorá umožní spätnú analýzu dát získaných zo senzorov a vyslaných pomocou efektorov. Analýza je založená na zobrazení dát vo forme grafov a následného odčítania požadovaných údajov z tejto obrazovky. Týmto spôsobom je tiež možné porovnať spôsob jazdy viacerých autopilotov, a tak vyhľadať optimálne parametre pre prejazd určitých úsekov trate.

Aplikácia by mala umožniť výber veličín, z ktorých sa grafy zostavia. Vo finálnej fáze bude možné modul spustiť aj ako samostatný program načítavajúci údaje zo súborov. Modul musí spĺňať podmienku prenositeľnosti, čiže musí byť použiteľný pre ľubovoľne zostrojeného

autopilota, ktorý využíva prednastavené klientské rozhranie v použitom programovacom jazyku Java. Aplikácia tak môže byť v budúcnosti použitá ďalšími tímami, ktoré sa rozhodnú zúčastniť súťaže *Championship*.

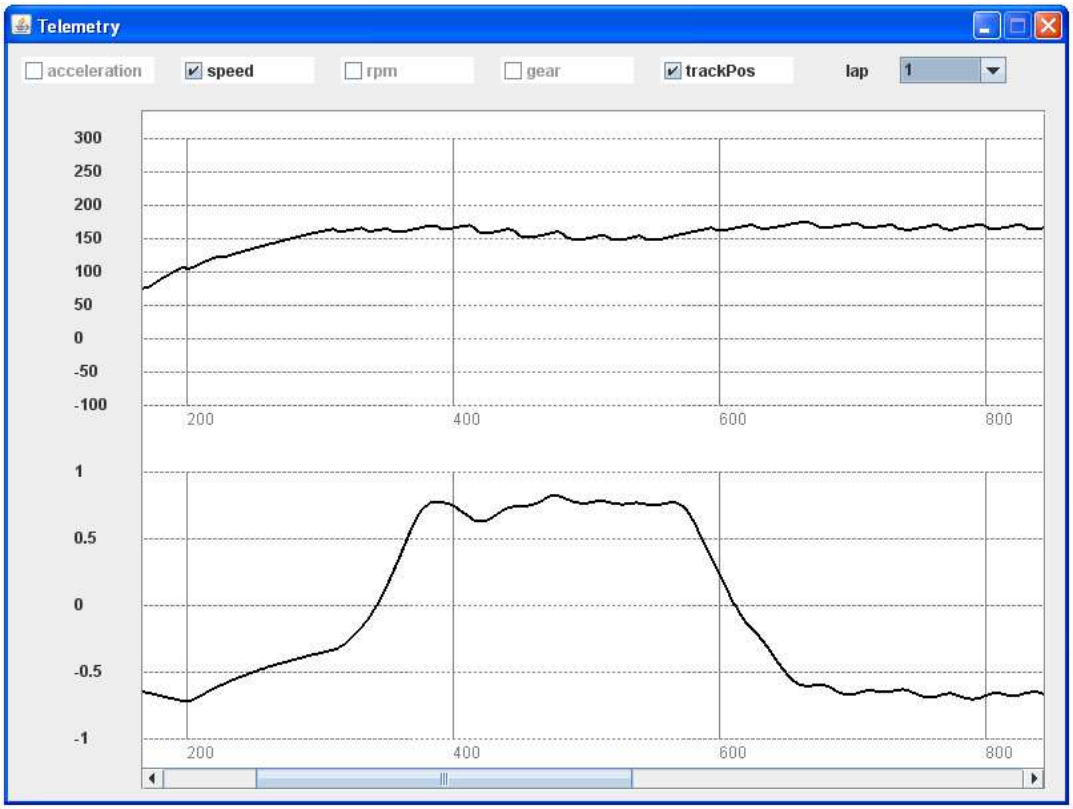
4.8.2 Implementácia

Prototyp modulu je implementovaný pomocou knižnice Swing. Grafická časť je zobrazená pomocou okna JFrame, v ktorom sa nachádzajú komponenty potrebné pre manipuláciu s dátami (checkboxy, combobox) a komponenty slúžiace na samotné zobrazenie dát: štítky (label) a panely, do ktorých sa pomocou technológie Java2D vykresľujú grafy.

Prototyp telemetrie využíva ako zdroj dát údaje získané v jednotlivých tickoch, uložené do vytvorenej triedy DataUnit. Táto štruktúra sa ukladá do poľa (ArrayList), ktoré sa po ukončení preteku, čiže vypnutí servera odošle na vizualizáciu. Ukážku grafického rozhrania predstavuje obrázok č.14.

4.8.3 Testovanie

Testovanie je založené na porovnaní zobrazených údajov na grafe s reálne získanými dátami zo senzorov a dátami odoslanými pomocou efektorov. Touto komparáciou sme overili správnosť zobrazených údajov.



Obr. 14 *Grafické rozhranie modulu Telemetria*

5. Implementácia

5.1 Modul riadenia

Oproti riešeniu, ktoré bolo použité v prototype (kap. 4.1) sa nič nezmenilo.

5.2 Modul Recovery

V návrhu a samotnom riešení modulu Recovery z kapitoly 4.2 nastala len jedna zmena. V riešení, ktoré bolo implementované podľa kapitoly 4.2 nebola uvažované flexibilná zmena akcelerácie. Hodnota akcelerácie bola pomerne vysoká, 0.8 z maximálnej – 1. To často spôsobovalo (hlavne pri rozjazdoch z miesta mimo trate), že pilot pri nízkej rýchlosti a s veľkým natočením kolies začal rýchlo akcelerovať. To malo za dôsledok to, že dochádzalo k pretočeniu automobilu okolo svojej osi aj o 360°, k „hodinám“. Preto bolo navrhnutá a implementovaná modifikácia tohto modulu.

Zmena oproti pôvodnému návrhu a implementácií spočíva v tom, že hodnota akcelerácie závisí od rýchlosti, ktorou sa pilot pohybuje. Hodnoty rýchlosti boli špecifikované do štyroch kategórií a každej z nich bola pridelená jedinečná akcelerácia. Pilot pohybujúci sa menšou rýchlosťou akceleruje pomalšie ako pilot pohybujúci sa rýchlejšie.

Kategórie rýchlosti a im prislúchajúce hodnoty akcelerácie sú implementované nasledovne:

```
if (sm.getSpeed() < 30)
      a.accelerate = ESP.filterASR(sm, 0.5);
else if ((sm.getSpeed() >= 30) && (sm.getSpeed() < 50))
      a.accelerate = ESP.filterASR(sm, 0.7);
else if ((sm.getSpeed() >= 50) && (sm.getSpeed() < 70))
      a.accelerate = ESP.filterASR(sm, 0.8);
else if (sm.getSpeed() >= 70)
      a.accelerate = ESP.filterASR(sm, 1);
```

Testovaním tejto zmeny sme dospeli k lepším výsledkom, preto považujeme túto zmenu za prospešnú. Zlepšenie vidíme ešte v rozlíšení toho, či sa pilot nachádza na trati alebo mimo trate a s tým súvisiacou hodnotou akcelerácie. Pretože pilot sa správa inak mimo trate a inak na trati hoci hodnota akcelerácie je rovnaká.

5.3 Modul modelu trate

5.3.1 Práca s modelom trate

Všetky údaje o trati, t.j. zákruty, frikcia, šírka trate a zaznamenané nehodové udalosti počas prejazdu traťou sú ukladané do triedy TrackData. Údaje tejto triedy sú vytvárané počas prvých kôl, kedy sa pilot učí trať, resp. sú načítané zo súboru, v prípade, že tento je k dispozícii. Hneď po prvých metroch a odmeraní je do modelu uložená frikcia a šírka trate, neskôr, po prejdení prvého kola, aj zoznam zákrut.

Správa zoznamu zákrut

Zoznam zákrut je načítaný a inicializovaný po prejazde prvého kola. Pri prvotnej inicializácii sú zavolané metódy na prepočet vstupných pozícií do zákrut a brzdnej vzdialenosti, na základe ktorých sa následne vypočítava rýchlosť. Vstupné pozície sú vypočítané na základe **kategorizácie zákrut** , ktorá je nasledovná:

- šírka trate >15
    - index zákruty <0.35
    - index zákruty >=0.35
- šírka trate >12
    - index zákruty <0.25
    - index zákruty >=0.25
- šírka trate <=12
    - index zákruty <0.15
    - index zákruty >=0.15

Zákruty sú rozdelené na mierne a prudké podľa šírky trate a následne je pre každý typ prepočítaná vstupná pozícia aj vzdialenosť zahŕňajúc šírku trate a index prudkosti zákruty.

Najdôležitejšou funkciou modelu trate je vzhľadom na zákruty udržiavanie aktuálnej informácie o zákrutách vzhľadom na pozíciu pilota, aby mohli byť podľa blížiacich sa zákrut vypočítavané optimálne parametre jazdy. Túto funkcionalitu zabezpečuje metóda iterate(), ktorá zabezpečuje iterovanie zoznamom zákrut a udržiavanie informácie o nasledujúcej zákrute, predchádzajúcej zákrute a zákrute za nasledujúcou zákrutou. Po každom kole taktiež obnovuje iterátor a zoznam zákrut, aby mohla iterácia prebiehať vždy od začiatku. Menšie problémy nastávali pri inicializácii zoznamu zákrut, keďže do tohto zoznamu boli uložené aj nesprávne zákruty zaznamenané pri vytočení pilota na štarte. Toto bolo odstránené identifikovaním takýchto zákrut a ich odstránením hneď pri počiatočnej inicializácii.

5.3.2 Optimalizácia a zvyšovanie rýchlosti

Keďže rýchlosť, ktorou pilot prechádza trať a je počítaná na základe modelu a najdlhšieho senzora, nie je optimálna, je potrebné a nutné túto rýchlosť v úsekoch kde je to možné optimalizovať smerom nahor. Keďže výpočet rýchlosti je rozdelený na úsek pred zákrutou (na základe brzdnej vzdialenosti z modelu) a úsek v zákrute (na základe najdlhšieho senzora), je potrebné aj optimalizovanie rozdeliť na dve časti. Pred zákrutou bolo navrhnuté a implementované riešenie založené na zvyšovaní rýchlosti v nasledujúcej zákrute, pričom po každom kole sa táto zvýšená rýchlosť prepočíta do brzdnej vzdialenosti. V zákrute bola navrhnutá a implementovaná optimalizácia použitím indexu rýchlosti, ktorý je uložený v každej zákrute modelu a jeho hodnota je na začiatku 1, pričom sa postupne zvyšuje. Indexom rýchlosti sa počas prejazdu zákruty prenásobuje rýchlosť vypočítaná na základe najdlhšieho senzora. Oba riešenia sú založené na laterálnej rýchlosti v zákrute, kedy sa pri prejazde zákrutou vyhodnocuje, či laterálna rýchlosť presiahla povolenú hodnotu, po ktorej už nasleduje šmyk. Hranica laterálnej rýchlosti bola stanovená testovaním vzhľadom na frikciu povrchu trate:

- frikcia < 29 – maximálna laterálna rýchlosť 5
- frikcia > 29 – maximálna laterálna rýchlosť 10

Teda pokiaľ laterálna rýchlosť počas prejazdu zákrutou nepresiahne stanovené hodnoty, zvyšujeme brzdnú vzdialenosť aj index rýchlosti pre danú zákrutu, čo sa následne prejaví pri výpočte rýchlosti počas jazdy. Každá zákruta obsahuje parameter, ktorý nadobúda boolovské hodnoty a vyjadruje, či zákruta už dosiahla maximálnu laterálnu rýchlosť. V prípade, že laterálna rýchlosť presiahne povolenú hranicu, parameter sa nastaví na true, čo signalizuje, že rýchlosť pre danú zákrutu je už maximálna a nebude sa ďalej zvyšovať. Zvyšovanie prebieha nasledovne:

- zvýšenie rýchlosti – rychlost*(1+ (1-index_prudkosti_zákruty)*0.15)
- zvýšenie indexu rýchlosti – index_rychlosti + (1-index_prudkosti_zákruty)*0.15

Uložené a optimalizované brzdné vzdialenosti ako aj indexy rýchlosti sú počas jazdy využívané na výpočet resp. zvýšenie rýchlosti. Z brzdnej vzdialenosti sa vypočíta rýchlosť pred zákrutou a naopak indexom sa prenásobí rýchlosť vypočítaná na základe najdlhšieho senzora.

5.3.3 Skoky

Ďalšia oblasť, ktorú je treba riešiť sú nerovnosti trate. Malé nerovnosti trate nemajú veľký vplyv na jazdu pilota. Závažnejšie sú však také nerovnosti a prevýšenia na trati, po ktorých sa pilot dostáva do skoku. Identifikované skoky sme rozdelili podľa toho, aký dôsledok spôsobujú.

Rozdelenie skokov:

- *Skoky bez dopadu na pilota* – sú to také skoky, po ktorých nedochádza k náhlej zmene smeru jazdy pilota a ani k vyleteniu mimo trate. Tento druh skokov sa vyskytuje na rovných úsekoch trate a má len minimálny dopad na štýl jazdy pilota. Z tohto pohľadu ich považujeme za nepodstatné a nebudeme sa nimi viac zaoberať.

- *Skoky s dopadom na pilota* – ide o skoky, po ktorých dochádza k vyleteniu pilota mimo trate. Na trati sú zväčša situované v zákrutách, tesne pred nimi alebo za nimi. Na tie treba reagovať.

Návrh

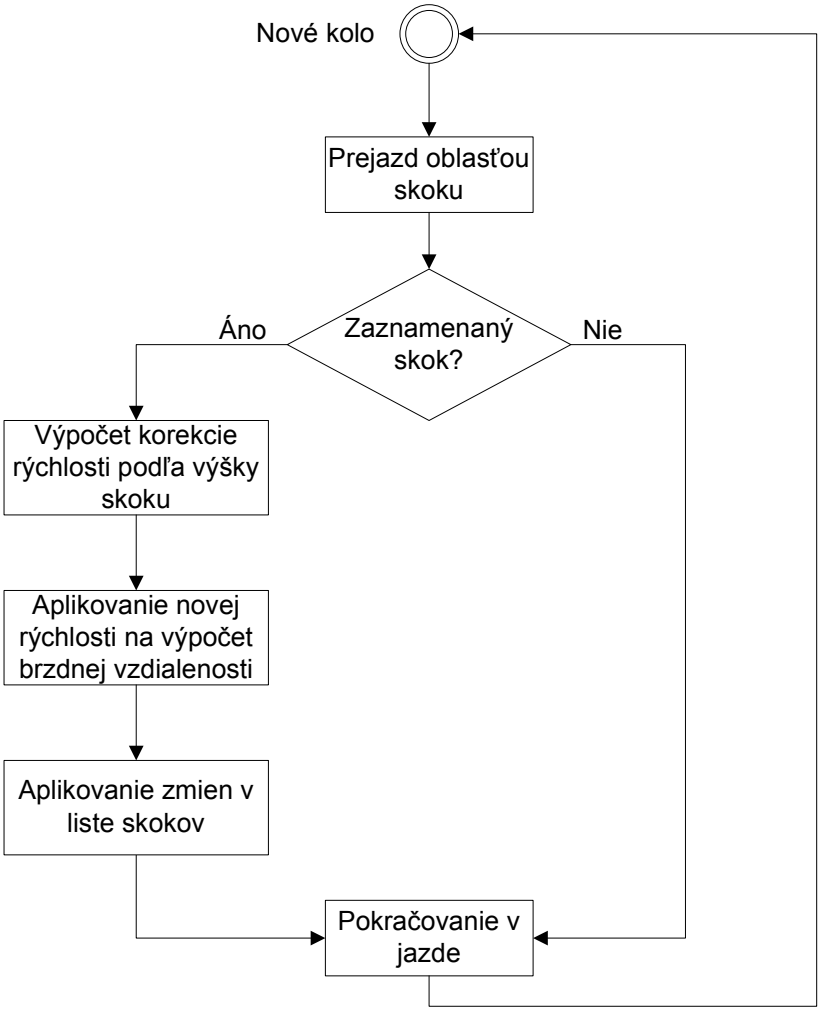
Z hľadiska poznania 3D modelu je vhodné zaznamenať všetky skoky, čiže aj tie kritické a aj tie bez dopadu na riadenie pilota. Ďalej je potrebné o skokoch udržiavať informácie, ktoré nám daný skok identifikujú, ako sú:

- Začiatok skoku (vzdialenosť od začiatku trate)
- Koniec skoku (vzdialenosť od začiatku trate)
- Maximálna výška skoku
- Rýchlosť prejazdu skokom

Tieto informácie je vhodné integrovať do modulu modelu trate, ktorý disponuje informáciami o trati.

Podrobnejšie sa treba zaoberať skokmi, ktoré sú pre nás kritické, čiže majú na pilota dopad a prerušia jeho plynulosť jazdy. Možnosť, ako to riešiť, je zníženie rýchlosti pred skokom tak, aby nedošlo k vyleteniu pilota. Avšak rýchlosť prejazdu oblasťou skoku je potrebné bližšie určiť. Vychádzať sa bude z výšky skoku.

Určenie rýchlosti potrebnej pre bezpečný prejazd oblasťou skoku sa uskutoční počas kôl etapy Warm-Up, kedy je možné na trati učiť sa. Algoritmus eliminácie kritických skokov úpravou rýchlosti môžeme vidieť na obrázku 15.



Obr. 15 *Algoritmus eliminácie skokov*

V každom kole sa upraví rýchlosť a tým aj brzdná dráha k miestu vyletenia podľa výšky skoku. Modifikované údaje sa aktualizujú v liste skokov a podľa nich sa bude pilot riadiť v nasledujúcom kole.

Implementácia

Algoritmus popísaný v návrhu bol kompletne implementovaný. V rámci neho bola implementovaná trieda Event, ktorá obsahuje, okrem iných, metódu na pridávanie nových skokov do zoznamu a aj metódu na korekciu rýchlosti. Skoky sú zaznamenávané a uchovávané v liste LinkedList<EventInfo>. Všetky potrebné informácie o skoku sú držané v triede EventInfo.

Po ukončení etapy Warm-Up sú dáta o skokoch spolu s dátami o modeli trati zapísané do súboru, z ktorého sú v ďalších etapách čítané.

Ďalej, môžu nastať prípady, kedy je skok priamo v zákrute a dôjde k vyleteniu pilota mimo trate. V tomto prípade nastáva korekcia samotnej zákruty a tým pádom nedochádza ku

korekcii rýchlosti odvíjajúcej sa od skoku, pretože by mohlo dôjsť k dvojnásobnej korekcii rýchlosti počas jedného kola, čiže spomalenie by bolo citeľnejšie ako je treba.

5.3.4 Vytváranie modelu trate na základe vykonaných akcií

Algoritmus navrhnutý a implementovaný v rámci prototypu bol spoľahlivý a detekoval všetky zákruty na trati. Jeho jediný problém boli sústavy zákrut v jednom smere bezprostredne za sebou(nebola medzi nimi žiadna rovina). Takéto reálne dve zákruty s rôznymi polomermi identifikoval ako jednu a priradil jej koeficient a rýchlosť z prudšej z nich. Tu dochádzalo k zdržaniu, pretože ak je zákruta prudká a za/pred ňom veľmi jemná , v modeli je to jedna zákruta a rýchlosť v jemnej zákrute je rovnaká ako v prudkej. Preto bola do algoritmu doplnená kontrola takejto situácie.

Doplnenie algoritmu

Pre jednotlivé akcie a zatočenia počas prejazdu zákrutou sa vytvoria intervaly v rozsahu <min zatočenie, max zatočenie> s hodnotami +/- 0.01, čo je maximálny rozdiel. Potom sa pre jednotlivé intervaly vytvoria početnosti. V prípade, že dva intervaly za sebou majú veľké početnosti, tak sa všetky intervaly posunú o maximálny rozdiel(0.01) a znova sa prepočítajú početnosti. Tento prípad prepočítavanie početností sa opakuje maximálne 2 krát, vzhľadom k tomu že pokryjeme všetky intervaly.

Ak už máme početnosti tak pre jednotlivé intervaly vieme podľa nich určiť, či zaznamenaná zákruta je zložená. Ak sa v intervaloch nachádzajú dve a viac početnosti nad 18%, tak sme zákrutu rozdelili podľa toho koľko tam bolo týchto početností. Hodnotu 18% sme dostali testovaním. Medzi rozdelenými zákrutami je vzdialenosť 1 m.

5.4 Modul podpory riadenia pred vjazdom do zákrut

5.4.1 Riadenie v zákrutách

Riedenie pilota počas prejazdu zákrutou je veľmi špecifická činnosť, ktorá v značnej miere závisí od pozície, z ktorej pilot začína manéver prejazdu zákrutou. Informácie, ktoré máme k dispozícii počas prejazdu zákrutou, jedná sa hlavne o uhol vzhľadom na os trate a pozíciu pilota na trati, sú veľmi premenlivé a ťažko sa podľa nich koriguje jazda pilota v zákrute.

Návrh

Pri návrhu prejazdu zákrutami sa bolo potrebné zamerať na dve najdôležitejšie časti tohto úkonu, a to

- **vjazd do zákruty**
- **výjazd zo zákruty**

Do úvahy pripadali dve možnosti realizácie vjazdu aj výjazdu, a to buď korekciou natočenia volantu vzhľadom na najdlhší senzor, alebo implementáciou matematických funkcií a korekciou natočenia vzhľadom na rozdiel aktuálnej a želanej ideálnej pozície v rámci zákruty. Pri druhom spôsobe prejazdu je k dispozícii vzťah, ktorý vychádza z obdĺžnika, ktorého dĺžku (x) vyjadruje vzdialenosť medzi začiatkom zákruty a pozíciou vrcholu zákruty v prípade vjazdu resp. koncom zákruty a vrcholom zákruty v prípade výjazdu. Šírku (y) tohto obdĺžnika vyjadruje rozdiel aktuálnej pozície pilota v rámci šírky trate a želanej pozície v cieli manévru. Posledným koeficientom vzťahu (pos) je aktuálne prejdená vzdialenosť od začiatku zákruty v prípade vjazdu resp. vzdialenosť od vrcholu zákruty v prípade výjazdu. Vzťah, pomocou ktorého vypočítame ideálnu pozíciu, na ktorej by sme mali v danom okamihu vzhľadom na cieľovú pozíciu byť, je veľmi jednoduchý a vyzerá nasledovne:

$$\frac{pos}{x} * y$$

Keďže vypočítaná pozícia je v absolútnych hodnotách, je potrebné ešte jej prepočítanie vzhľadom na smerovanie zákruty a nastavenie správneho znamienka výsledného natočenia volantu podľa špecifikácie TORCS. Výsledné natočenie vypočítame dosadením vypočítanej pozície do vzťahu:

$$\frac{uhol\_k\_osi\_trate - (aktua\ln a\_pozicia - vypoci\tan a\_pozicia)}{0.366519}$$

Pomocou týchto dvoch vzťahov môžeme riešiť vjazd aj výjazd zo zákruty. Celkový prejazd zákrutou by mal teda prebiehať po ideálnej stope, čo je veľmi dobrý predpoklad pre optimalizáciu rýchlosti a tým zaisteniu čo najefektívnejšieho a rýchleho prejazdu.

Implementácia a testovanie

Implementácia prebehla podľa návrhu. Najprv bol implementovaný vjazd zákruty pomocou navrhnutých vzťahov aj pomocou najdlhšieho senzora a následne po otestovaní aj výjazd taktiež použitím spomenutých dvoch prístupov. Použitie vzťahov sa spočiatku javilo ako použiteľné riešenie, aj keď v niektorých zákrutách spôsobovalo problémy. Bolo testované na rôznych tratiach a rôznych zákrutách, no nakoniec bol uprednostnený spôsob prejazdu využitím najdlhšieho senzora, keďže v špecifických zákrutách bolo riadenie neudržateľné a nevyspytateľné, čo sme si samozrejme nemohli dovoliť, keďže nesprávny prejazd, či až nehoda v zákrute by v pretekoch znamenali obrovskú časovú stratu. Implementovaná bola spolu s jazdou podľa najdlhšieho senzora aj korekcia natočenia pri výjazde s cieľom lepšie využiť zákrutu a nechať auto vyniesť na okraj namiesto násilného točenia smerom k stredu trate. Táto možnosť bola aj otestovaná a priniesla uspokojivé výsledky.

5.4.2 Výpočet rýchlosti

Návrh

Na výpočet rýchlosti vyplynul z analýzy vzťah, použitím ktorého môžeme vypočítať aktuálnu možnú rýchlosť dosadením určitej dĺžky maximálnej brzdnej vzdialenosti, t. j. vzdialenosti na ktorej pilot dokáže ubrzdiť. Túto vzdialenosť je možné získať na základe dĺžky najdlhšieho senzora, resp. použitím rovnakého vzťahu avšak odvodením vzťahu na výpočet brzdnej vzdialenosti na základe rýchlosti. Vzťah pre výpočet brzdnej vzdialenosti z rýchlosti je:

$$\frac{rýchlost^2}{2*9.81*\ frikcia}$$

Vzťah pre výpočet rýchlosti z brzdnej vzdialenosti je:

$$\sqrt{2*9.81*\ frikcia}*brzdna\_vzdialenost$$

Na základe týchto dvoch vzťahov je možné vypočítať brzdnú vzdialenosť podľa rýchlosti prejazdu v prvom kole a uložiť ju do modelu trate a jeho jednotlivých zákrut a následne prepočítavať z tejto brzdnej vzdialenosti aktuálnu rýchlosť pred vjazdom a počas prejazdu zákrutou.

Implementácia a testovanie

Implementovaný bol prepočet a uloženie brzdnej vzdialenosti do jednotlivých zákrut modelu trate. Následne vznikli viaceré možnosti ako brzdnú vzdialenosť využiť na výpočet rýchlosti. Prvá implementovaná možnosť bola postupne prepočítavať rýchlosť na základe brzdnej vzdialenosti najbližšej zákruty, ku ktorej sa pripočíta aktuálna vzdialenosť pilota k danej zákrute. Následne bola zákruta prejdená rýchlosťou brzdnej vzdialenosti tejto zákruty, pričom pri výjazde bola táto rýchlosť postupne zvyšovaná. Toto riešenie bolo vzhľadom na nepresnosti modelu, ako aj veľmi pomalou konštantnou rýchlosťou v zákrute nepoužiteľné a bolo ho treba optimalizovať. Hlavný problém výpočtu rýchlosti v zákrute bol, že sme nedokázali uspokojivo simulovať dynamické uberanie a pridávanie rýchlosti v zákrute, čo pri vjazde aj výjazde spôsobovalo šmyk a tým výjazd mimo trate. Optimalizácia bola vykonaná využitím údajov získaných z najdlhšieho senzora a následný výpočet rýchlosti na základe týchto údajov počas jazdy zákrutou. Mimo zákruty ostalo pôvodné vypočítavanie rýchlosti z uložených maximálnych brzdných vzdialeností najbližšej zákruty. Takto sme zabezpečili maximalizovanie rýchlosti pred zákrutou do poslednej možnej pozície, kde by už v prípade najdlhšieho senzora došlo k brzdeniu, a naopak dynamickému prepočtu rýchlosti v zákrute, čo minimalizovalo šmyk a výjazd mimo zákruty.

5.5 Modul detekcie oponentov

Navrhnuté riešenie v kapitole 4.5 sa po dlhšom testovaní ukázalo byť kvôli šumu nepoužiteľné. Z tohto dôvodu sme zapracovali výpočtovo jednoduchšie riešenie menej náchylné na nepresné hodnoty.

Algoritmicky sa jedná o pravidlový systém, ktorý na základe detekcie oponenta určitým senzorom upraví jazdnú dráhu tak, aby nedošlo ku kolízii so súperom. Pre tento účel boli zvolené štyri najprednejšie senzory ako najhlavnejšie senzory, na základe ktorých dochádza k predbiehaniu/vyhýbaniu sa oponentovi. Na základe počtu a miesta detekovaných oponentov v týchto senzoroch, naplánovanej jazdnej dráhy (t.j. natočenia volantu) a polohe auta na trati (zákruta alebo rovina) sa upraví natočenie volantu. Iba v prípade nemožnosti predbiehať (t.j. blokované všetky štyri použité senzory) dôjde k postupnému spomaľovaniu auta.

Počas testovania boli objavené chyby týkajúce sa ukončovania predbiehacieho manévru, kedy autopilot pri vracaní sa do ideálnej stopy často kúskom auta kolidoval s oponentom. Z tohto dôvodu boli použité aj ďalšie oponentské senzory, pomocou ktorých dochádza k jemným korekciám natočenia práve v takýchto situáciách, kedy hrozí riziko bočnej zrážky (súbežná jazda a ukončovacia fáza predbiehacích manévrov.

Štartovacia sekvencia s oponentmi je špecifická situácia, preto sme pristúpili k oddeleniu tejto časti od štandardného obiehania. Obehanie protivníkov je na základe tohto návrhu trvá iba po prvú zákrutu.

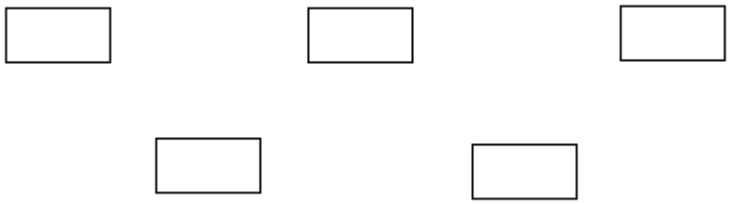
Oponent asistent pri štarte

Štart preteku je kritická časť, kde sa dá veľa získať alebo stratiť. Využitie spojky nám zabezpečuje rýchlejší štart a taktiež polepšenie si v rámci pozícií v preteku hneď na začiatku. Úlohou tejto časti pilota je upraviť riadenie tak, aby sme sa úspešne vyhli oponentom a bezpečne ich obišli. Vzhľadom k tomu, že štartovanie je špecifická oblasť preteku, kde sa pilot nachádza v blízkosti viacerých protivníkov, úplne preberá kontrolu nad riadením táto časť oponent modulu.

Princíp je založený na smerovanie vozidla rovnobežne z osou trate na určenej pozícii podľa formuly (a). Premenné uholKTrati a poziciaNaTrati sú údaje zo senzorov. Konštanta 0.8 je na zjemnenie zatočenia a hodnota v menovateli je uhol 21 stupňov v radiánoch (maximálne natočenie kolies). Celým problémom je určenie pozície(chcenaPozicia), tak aby nedošlo k stretu s oponentmi.

(a)
$$steer = \frac{uholKTrati - (poziciaNaTrati - chcenaPozicia) * 0.8}{0.366519}$$

V obidvoch riešeniach uvažujeme 4 senzory na ľavo a na pravo od osi auta a na oboch stranách senzory na 60 a 90 stupňoch. 60 a 90 stupňové senzory sa kontrolujú, či sa do nás niekto nesnaží naraziť. Minimálna vzdialenosť medzi vozidlami* bola zvolená na 30 centimetrov. Ďalším predpokladom je, že vozidlá štartujú so šachovnicového postavenia (obr.16).



Obr. 16 rozostavenie vozidiel na štarte

Vždy sa pokúšame obchádzať z vonkajšej strany. Ak to však nejde, čiže naša nová pozícia je mimo trať obchádzame stredom trate. V oboch metódach treba do uhla započítať aj uhol osi auta k osi trate ,aby sme mohli predpokladať, že ide o pravouhlý trojuholník

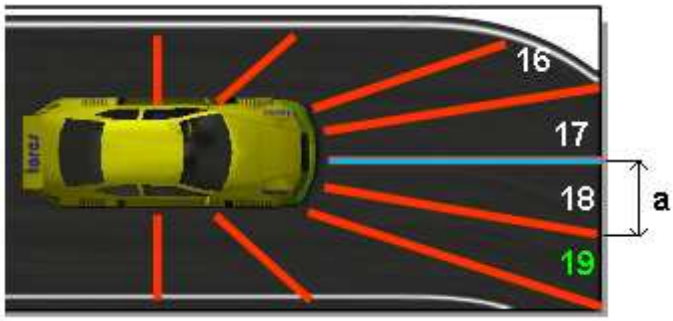
1. Riešenie

Určenie oponentovej pozície na trati a z toho odvodenie našej novej pozície a následné výpočet natočenia kolies (steer). Na určenie oponentovej pozície na trati použijeme pravouhlý trojuholník a goniometrickú funkciu sínus. Počítame protiľahlú odvesnu, určíme pozíciu oponenta na trati odpočítaním/pripočítaním hodnoty odvesny a našej pozície. Ako preponu beriem hodnotu vracanú senzorom. Pri určovaní uhla delíme senzory na ľavé a pravé. Uhol s ktorého počítame sínus, určujeme z indexu senzora(aktSenz), ktorý oponenta zachytil a senzora(lastSenz), ktorý ho snímal predtým. Ak sme v ľavých senzoroch a aktSenz je väčší ako lastSenz, tak uhol je určený ako uhol senzora mínus 10 stupňov. Ak aktSenz je menší ako lastSenz tak uhol je uhol senzora. Na pravej strane sa 10 stupňov odpočítava ak je aktSenz menší ako lastSenz, inak sa berie uhol podľa senzora. Berieme worst case scenario uhly. Našu novú pozíciu vypočítame posunutím od oponentovej pozície o dĺžku v predom definovanej vzdialenosti vozidiel. Vždy sa pokúšame obchádzať z vonkajšej strany. Ak to však nejde, čiže naša nová pozícia je mimo trať obchádzame z stredom trate.

Na obr.2 je modelová situácia. Oponenta sme zachytili senzorom s indexom 19. Predtým bol snímaný senzorom 18. Sú to pravé senzory 18 je menej ako 19, preto uhol senzora 19, čo je 20 stupňov, zmenšíme o 10 stupňov. Následne počítame odvesnu a vzorcom sin(10)*oponentSenzor[19]. Určíme pozíciu oponenta odpočítaním a od našej vzdialenosti od

* aj keď je v texte uvedené 30 centimetrov ako minimálna vzdialenosť sa počíta 30 cm + polovica šírky nášho auta + polovica šírky súperovho auta (v podstate 2*šírak nášho vozidla), lebo senzory merajú od stredu vozidla po stred vozidla.

stredu. K pozícii oponenta pripočítame hodnotu vzdialenosti vozidiel. Overíme či hodnota neukazuje mimo trať, ak nie tak máme našu novú pozíciu.



Obr. 17 výpočet pozície oponenta

2. Riešenie

Spočíva vo výpočte našej novej pozície s dĺžky senzora a indexu senzora, ktorým oponenta vnímame. Uhol v tejto metóde určujem ako uhol pre senzor mínus 10 stupňov(hraničné situácie). Určíme odvesnu a s pravouhlého trojuholníka. Následne porovnávame a s hraničnou hodnotou pre senzor. Hraničná hodnota je určená minimálnou vzdialenosťou vozidiel zmenšenou o hodnotu podielu tangensu uhla a polovice dĺžky auta[**]. Ak prepona a je menšia ako táto hraničná hodnota tak pozíciu auta upravíme tak, aby prepona a bola väčšia nanajvýš rovná hraničnej hodnote. Taktiež sa počíta s tým, či pri zmene našej pozície nedôjde k tomu že pilot zamieri mimo trať.

Implementácia

Riešenie 1 má nedostatok v tom, že pri zmene senzoru sa zmení uhol o 10 stupňov pričom vzdialenosť senzora sa zmení minimálne. Čím sa nová pozícia posunie smerom k predchádzajúcim pozíciám a auto sa na chvíľu približuje a nejde plynule do jednej strany. Keďže je minimálna vzdialenosť je nízka v 80% prípadov dochádza ku kolízii.

Riešenie 2 sa ukázalo ako funkčné. Na zjemnenie točenia sa na začiatku vozidlo posunie so svojej štartovacej pozície na trati na jednu alebo na druhú stranu podľa toho, ktorým senzorom(17/18) vníma oponenta pred sebou. Toto posunutie sa deje pokiaľ sa oponent nedostane do closeArea[***] vzdialenosti voči nášmu pilotovi. Keď je oponent detekovaný v closeArea používa sa už algoritmus z návrhu.

[**] polovica dĺžky auta, preto lebo senzory merajú hodnoty od stredu vozidla

[***] closeArea vzdialenosť na vybraných senzoroch je menšia ako hraničné hodnoty (bude tabulka, su to vypočítané hodnoty sinusov, aby minimálna vzdialenosť sedela).

Ďalšou zmenou je minimálna vzdialenosť medzi vozidlami*, ktorá sa určuje so šírky trate. A to ak je šírka menšia ako 10m je to 10 cm, ak je šírka menšia ako 12m je to 20 cm, ak je šírka menšia ako 15m je to 30 cm inak 60cm. (určené testovaním)

5.6 Modul anti-blokovacieho systému (ABS)

Oproti riešeniu, ktoré bolo použité v prototype (kap. 4.6) sa nič nezmenilo.

5.7 Modul regulácie preklzávania (ASR)

Oproti riešeniu, ktoré bolo použité v prototype (kap. 4.7) sa nič nezmenilo.

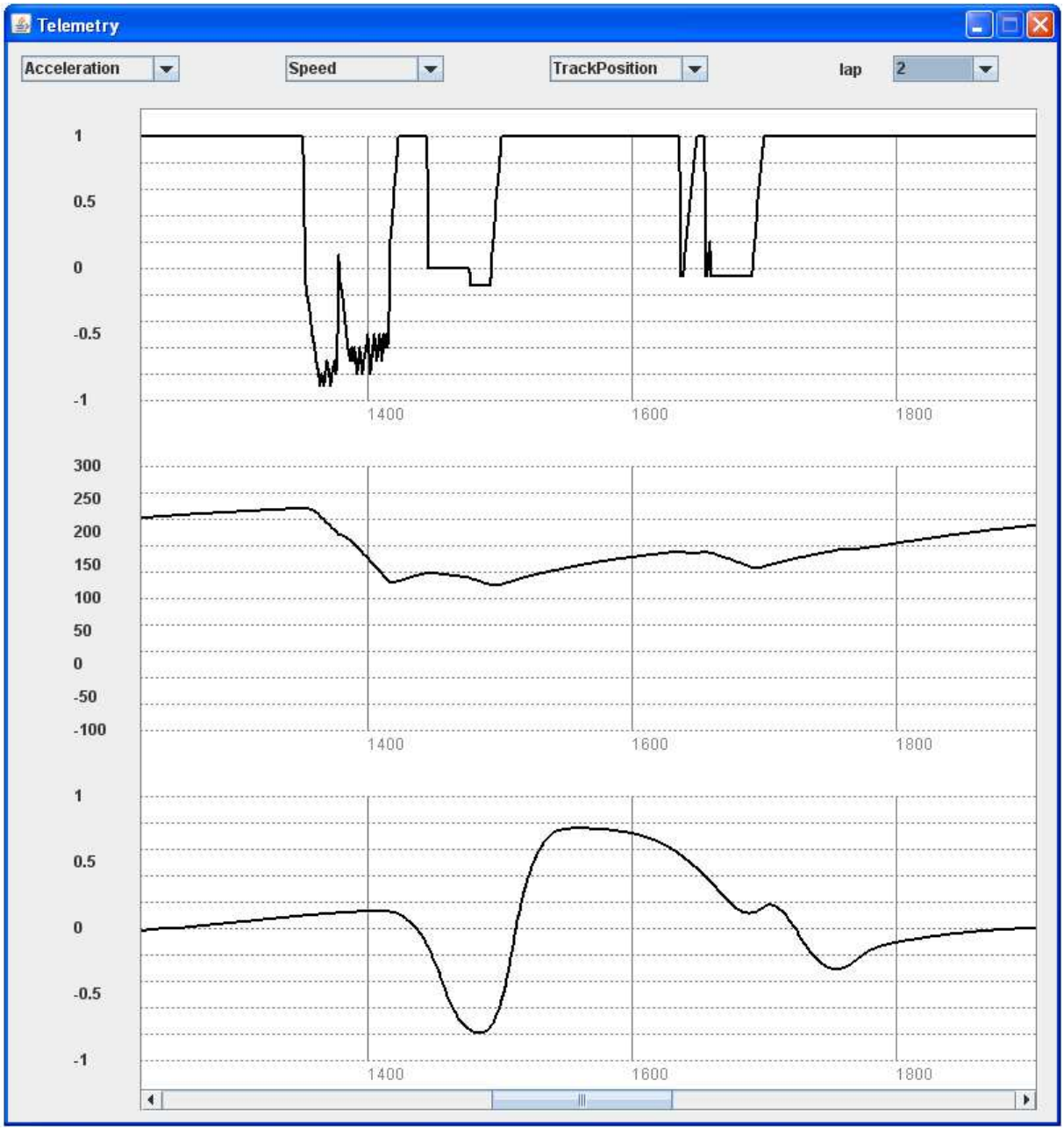
5.8 Modul Telemetria

Oproti riešeniu, ktoré bolo použité v prototype (kap. 4.8) došlo k niekoľkým drobným úpravám. Základná funkcionalita zostala rovnaká avšak tento modul bol vyčlenený do samostatne spustiteľnej aplikácie, keďže je zbytočné, aby sa tento modul aktivoval po každom spustení pilota, analýza sa obvykle robí na väčšom množstve dát.

Z tohto sa po ukončení autopilota (metóda *shutdown()*) vytvorí súbor (XYZ.tad – telemetry analyzer data) s dátami, ktoré je neskôr možné vizualizovať. Identifikátorom súboru je čas vytvorenia, súbor sa ukladá do podadresára, ktorý nesie meno názvu trate v adresári telemetry. Spustiteľná aplikácia (modul) ďalej pracuje s týmito dátami.

Po vizuálnej stránke sme prešli k rozšíreniu simultánne zobrazovaných dát na tri množiny, obsluha modulu sa stala prehľadnejšou a intuitívnejšou. Grafické používateľské rozhranie prezentuje obrázok číslo 17.

Kvôli možnosti použitia tohto modulu aj v nasledovných ročníkoch či dokonca iných súťažiach (zmena počtu prevodových stupňov, väčší maximálne dosiahnuteľný výkon a podobne), aplikácia je navrhnutá tak, že pracuje s dátami naškálovanými v rozmedzí < 1; 1>. Na tento účel slúži funkcia *rescaleTORCS2011()*, ktorá prevedie dáta (napr. rýchlosť) do tohto intervalu. Modul je preto rozšíriteľný, pre iné vstupné hodnoty je potrebné jedine doimplementovať potrebnú *rescaleXXX()* metódu.



Obr. 17 *Graficképoužívateľské rozhranie modulu Telemetria*

6. Zhrnutie

Cieľom tohto zhrnutia je analyzovať a zhodnotiť prácu na predmete Tímový projekt najmä z pohľadu na realizácie projektu v tíme. Keďže sme sa viacerí podobného projektu zúčastnili prvý krát, neboli všetky postupy a činnosti vykonávané dokonale. Je veľa miest, kde by sa dalo pridať, či zlepšiť veci, ktoré počas pôsobenia v tímovom projekte neprebehli správne. Aj napriek uvedeným nedostatkom však hodnotíme pozitívne naše účinkovanie v tíme, ako aj prácu celého tímu. Podarilo sa nám vyskúšať a overiť na vlastných skúsenostiach plánovanie, realizáciu naplánovaných činností, ale aj riešenie neočakávaných problémov, ktoré sa počas semestra vyskytli. Úspešne sme dosiahli stanovené ciele a vytvorili funkčný prototyp, pričom sme sa naučili veľa užitočných, pre viacerých z nás ešte nepoznaných, vecí z oblasti tímového manažmentu. Predmet Tímový projekt nám veľa dal a z chýb, ktoré sme identifikovali, sa môžeme poučiť a v budúcom zamestnaní sa im úspešne vyhýbať.

Nie všetky stanovené ciele sme samozrejme stihli realizovať, čo však bolo rizikom vytvorenia tímu, v ktorom takmer všetci majú žiadne resp. minimálne skúsenosti s realizáciou väčších softvérových projektov. Plánovali sme realizovať aj súťaž *Demolition derby,* čo však už po prvom semestri bol ťažko splniteľný cieľ, keďže prvá súťaž vyžadovala oveľa viac úsilia ako sme plánovali a predpokladali. Pri ďalšom projekte by sme určite takúto chybu v plánovaní nespravili.

Pri finalizácii tejto dokumentácie sme mali za sebou účasť na prvej konferencii, ktorá pre nás skončila dvojakým výsledkom. Na jednej strane bola vcelku úspešná kvalifikácia, kde sa nami vytvorený pilot ukázal ako konkurencieschopné riešenie a skončil na treťom mieste. Na strane druhej bol samotný pretek, kde sa nepremietli priaznivé výsledky kvalifikácie a pilot skončil na poslednom mieste. Výsledky plánujeme analyzovať a pred ďalšími konferenciami napraviť prípadné zistené nedostatky, aby boli ďalšie umiestnenia čo najlepšie.

Počas dvoch semestrov tímového projektu sme odviedli veľký kus práce a vytvorili konkurencieschopného pilota. Veríme, že v prípade ďalšieho pokračovania tejto témy na predmete Tímový projekt, budú môcť ďalšie tímy použiť naše riešenie a stavať na ňom. Je veľa možností a ciest ako vylepšiť nášho pilota a dosiahnuť už nami stanovený cieľ – vyhrať preteky na niektorej z konferencií.

7. Použitá literatúra

[1] Butz, M.V., Linhardt, M.J.: Demolition Derby
http://www.coboslab.psychologie.uniwuerzburg.de/competitions/

[2] Loiacono, D., Cardamone, L.: Simulated Car Racing Championship
http://cig.ws.dei.polimi.it/

[3] Loiacono, D. et al.: The 2009 Simulated Car Racing Championship. In: *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, 2010, 131-147.

[4] Wymann, B., Espié, E.: TORCS
http://torcs.sourceforge.net

[5] Haavasoja, T., Pilli-Sihvola, Y.: Friction as a Measure of Slippery Road Surfaces. In: *Standing International Road Weather Commission*, topic 2, 2010.