



**MATEMATICKO-FYZIKÁLNÍ
FAKULTA**
Univerzita Karlova

BAKALÁŘSKÁ PRÁCE

Martin Homa

RacingCarSim

Katedra softwaru a výuky informatiky

Vedoucí bakalářské práce: Mgr. Jakub Gemrot

Studijní program: Informatika

Studijní obor: Programování a softwarové systémy

Praha 2017

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne 24. 7. 2017

Martin Homa

Názov práce: RacingCarSim

Autor: Martin Homa

Katedra: Katedra softwaru a výuky informatiky

Vedúci bakalárskej práce: Mgr. Jakub Gemrot, Katedra softwaru a výuky informatiky

Abstrakt: Šoférovanie patrí od vzniku automobilu k obľúbeným činnostiam nielen mužského pokolenia. Je s ním spojená dávka zábavy a adrenalínu, ktorú je možné vidieť u automobilových nadšencov každý deň. Práve ich častokrát zaujíma, čo sa skrýva za samotným pohybom auta. Cieľom tejto práce je osloviť túto skupinu ľudí a podať im odpovede na ich otázky zábavnou formou. Za týmto účelom bola vytvorená aplikácia, ktorá na základe pokročilého fyzikálneho modelu poskytuje jej užívateľovi možnosť simulovať jazdenie s automobilom v rôznom prostredí a sledovať, ako sa automobil správa a aké fyzikálne veličiny ho ovplyvňujú. Braný je pri tom ohľad na čo najväčšiu konfigurovateľnosť, preto si môže užívateľ automobil sám nadefinovať a prostredie upravovať. Ak sa tak rozhodne, môže ho zaujímať ešte jedna otázka. Ako rýchlo je možné s týmto autom jazdiť? Odpoveď je taktiež zahrnutá v rámci tejto aplikácie, ktorá implementuje do simulácie umelého hráča. Tento hráč sa snaží naučiť jazdiť po užívateľom definovanej závodnej stope čo najrýchlejšie, pričom proces učenia a jeho súčasti sú taktiež konfigurovateľné. Učenie je avšak veľmi citlivé na nastavenie parametrov a reprezentáciu prostredia, preto dosiahnutie požadovaného výsledku nie je triviálne.

Kľúčové slová: Simulácia, Šoférovanie, Q-learning

Title: RacingCarSim

Author: Martin Homa

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Jakub Gemrot,, Department of Software and Computer Science Education

Abstract: Driving is one of the favourite activities of not just men since the invention of automobiles. A dose of fun and adrenaline is connected to driving, which can be seen among car enthusiasts every day. These people oftentimes want to know what is behind the car movement itself. The aim of this work is to appeal to this type of people, giving them answers to their questions in an entertaining way. An application has been created to fulfil this intention, which, based on an advanced physical model, allows its user to simulate driving an automobile in various environments, observe its behaviour and the physical variables which are affecting it. The main focus is allowing for a great amount of configurability, which is why the user can define the car himself, as well as modify the environment. If he chooses to, he might also be interested in one more question. What is the maximum speed the car can be driven at? The answer is also included in the application, as it implements an artificial player. This player is trying to learn to drive at a user-defined racetrack at the highest speeds possible, while the learning process and its individual components are also configurable. However, learning is highly dependent on its parameters and the representation of the environment, thus achieving the desired result is non-trivial.

Keywords: Simulation, Driving, Q-learning

Veľmi rád by som poďakoval vedúcemu práce Mgr. Jakubovi Gemrotovi za jeho ochotu, ústretovosť a profesionálny prístup, s ktorým mi v rámci pravidelných konzultácií poskytoval cenné rady a dôležité pripomienky.

Obsah

1. Úvod	1
1.1. Ciele	2
1.2. Štruktúra práce	2
2. Užívateľská dokumentácia pre prácu s aplikáciou	4
2.1. Obrazovky	4
2.1.1. Úvodná obrazovka	4
2.1.2. Menu obrazovka	5
2.1.3. Obrazovka konfigurácie simulácie	9
2.1.4. Obrazovka načítania simulácie	11
2.1.5. Obrazovka simulácie	11
2.2. Log aplikácie	12
2.3. Štruktúra aplikácie	14
2.3.1. Adresáre	14
2.3.2. Konfigurácie aplikácie	16
3. Užívateľská dokumentácia simulácie	18
3.1. Parametre simulácie	18
3.1.1. Automobil	18
3.1.2. Mapa	21
3.1.3. Závodná stopa	25
3.2. Režimy simulácie	28
3.2.1. Jazdenie	28
3.2.2. Učenie	28
3.2.3. Prehľad učenia	29
3.3. Nastavenia simulácie	29
3.3.1. Nastavenie rýchlosti	29
3.3.2. Nastavenie priblíženia mapy	30
3.3.3. Ostatné nastavenia grafického prostredia simulácie	30
3.4. Funkcionalita umelého hráča v simulácii	31

3.4.1. Vizualizácia akcii umelého hráča	31
3.4.2. Export tabuľky.....	32
3.4.3. Import tabuľky.....	32
3.4.4. Vizualizácia tabuľky.....	33
3.5. Informačné panely	34
3.5.1. Panel so základnými údajmi o priebehu simulácie.....	34
3.5.2. Panely s informáciami o aktuálnom stave simulácie.....	35
3.5.3. Načítavanie panelov	46
3.6. Konzola	46
3.7. Ovládanie simulácie.....	48
4. Fyzikálny model automobilu	52
4.1. Fyzika pohybu auta po rovnej čiare	52
4.2. Fyzika zatáčania automobilu	64
5. Umelý hráč	71
5.1. Q-learning	71
5.1.1. Základný princíp.....	71
5.1.2. Stav prostredia agenta a akcie agenta	72
5.1.3. Popis iterácii procesu učenia	72
5.1.4. Funkcia reward	73
5.1.5. Kľúčové faktory	74
5.1.6. ϵ -greedy	74
5.2. Adaptovanie umelého hráča do simulácie	75
5.2.1. Definícia stavu prostredia.....	75
5.2.2. Definícia akcie agenta	85
5.2.3. Funkcia reward	86
5.2.4. Ďalšie parametre umelého agenta.....	88
5.3. Konfigurácia umelého hráča	89
5.4. Aplikácia Q-Learning	92
5.5. Analýza dosiahnutých výsledkov	96
6. Vývojárska dokumentácia	98
6.1. Aplikačné rámce a vývojové prostredie	98

6.2. Systémové požiadavky	100
6.3. Základný princíp fungovania aplikácie	100
6.4. Architektúra	102
6.4.1. RacingCarSimulation	102
6.4.2. RacingCarSimulationTests	113
6.4.3. QLearningApplication	114
7. Záver	118
7.1. Možnosti ďalších rozšírení	119
8. Súvisiace práce	120
9. Zoznam použitej literatúry	122
10. Príloha A – Obsah priloženého CD	124

1. Úvod

Aplikácia *RacingCarSim*, alebo v dlhšej forme *Racing Car Simulation* ako samostatná aplikácia je určená predovšetkým pre milovníkov áut, ktorých nezaujíma len samotné jazdenie, ale aj to, ako veci spojené s jazdením a fungovaním automobilu fungujú. Užívateľovi totiž poskytuje možnosť simulovať jazdu s autom v konfigurovateľnom prostredí. Táto simulácia sa riadi fyzikálnym modelom, ktorý reflektuje správanie sa auta v reálnom svete, čo určite ocenia nadšení závodníci. Tento fyzikálny model ako hlavná časť aplikácie na základe ovládania od užívateľa umožňuje autu vykonávať v prostredí pohyb a poskytuje množstvo zaujímavých informácií, ktoré sú užívateľovi sprostredkované skrz údaje zobrazované počas simulácie na obrazovke. Užívateľ tým pádom môže nie len auto ovládať, ale sledovať pri tom vývoj hodnôt rôznych fyzikálnych veličín, ktoré na auto pôsobia. Automobil, ktorý bude v simulácii ovládať si pri tom môže nadefinovať celkom sám. Potrebuje k tomu iba menšiu znalosť toho, čo jednotlivé parametre auta, ktorých hodnotu musí poskytnúť, znamenajú. Aplikácia je pri tom prispôsobená tak, aby mal užívateľ možnosť ju čo najviac konfigurovať a prispôbiť svojim predstavám.

Nemenej významná súčasť tejto aplikácie zasahuje do sveta umelej inteligencie. Simuláciu je totiž možné prepnúť do špeciálneho režimu, v ktorom auto neovláda užívateľ, ale umelý hráč. V tomto režime sa umelý hráč snaží naučiť so zadaným automobilom jazdiť po užívateľom zvolenej závodnej stope. Proces učenia je pri tom možné sledovať v reálnom čase priamo v simulácii, kedy užívateľ môže vidieť, ako umelá inteligencia ovláda auto ako postupuje vo svojom učení. Všetky podstatné informácie, ktoré súvisia s týmto procesom môže užívateľ vidieť skrz údaje zobrazené na obrazovke tak isto, ako v prípade informácií týkajúcich sa fyzikálneho modelu auta.

Racing Car Simulation predstavuje aplikáciu, ktorá funguje samostatne. Má vyvinuté svoje vlastné grafické užívateľské prostredie, ktoré pomocou rôznych obrazoviek komunikuje s užívateľom.

1.1. Ciele

1. Vypracovať na pohľad príjemne grafické prostredie pre komunikáciu aplikácie s užívateľom, ktoré bude dostačujúce pre účely simulácie a pre zobrazovanie potrebných údajov
2. Implementovať netriviálny fyzikálny model auta a prostredia, ktorý bude odrážať správanie sa automobilov v reálnom svete
3. Nájsť vhodnú metódu učenia umelého hráča a túto metódu implementovať
4. Vymyslieť vhodnú reprezentáciu prostredia pre umelého hráča
5. Vypracovať vhodnú reprezentáciu závodnej stopy a implementovať mechanizmus, ktorý poskytuje potrebné údaje pre umelého hráča
6. Vyvíjať aplikáciu s ohľadom na ďalšie rozširovanie
7. Zaistiť čo najväčšiu možnú konfigurovateľnosť jednotlivých komponent aplikácie

1.2. Štruktúra práce

Hlavným účelom tejto práce je poskytnúť užívateľovi dostačujúce informácie na to, aby sa po spustení aplikácie v jej prostredí nestratil, aby bol oboznámený so všetkou dostupnou funkčnosťou a ovládaním a aby mu bolo vysvetlené, ako fungujú dôležité súčasti simulácie (napríklad fyzikálny model) a ako s nimi pracovať. Okrem toho táto práca disponuje vývojárskou dokumentáciou, ktorá vo všeobecnosti opisuje, ako bola táto aplikácia implementovaná, aká je jej štruktúra, v akom prostredí beží a aké sú jej požiadavky. Vývojárovi navyše poskytuje základný prehľad o jednotlivých súčastiach kódu ako sú menné priestory (namespace) alebo použité triedy.

V kapitole 2. a 3. si v rámci užívateľskej dokumentácie predstavíme základné funkcie aplikácie a jej logické celky, ktoré dajú užívateľovi prehľad o používaní aplikácie a simulácie. V kapitole 4. si popíšeme fyzikálny model automobilu do detailov tak, ako je v aplikácii implementovaný. Kapitola 5. nás zatiahne do časti

práce, ktorá nám predstaví metódu Q-learning, ktorá je použitá pre účely učenia umelého hráča. V rámci tejto kapitoly sa taktiež dozvieme, ako je táto umelá inteligencia adaptovaná simulácii a ako ju konfigurovať. V nasledujúcej 6. kapitole je popísaná vývojárska dokumentácia, ktorá je potrebná pre porozumenie kódu a použitým postupom.

2. Uživatelská dokumentácia pre prácu s aplikáciou

Kapitola *Uživatelská dokumentácia pre prácu s aplikáciou* v rámci tejto práce predstavuje sprievodcu užívateľa pri práci s aplikáciou počas jej behu. Poskytuje informácie popisujúce to, čo môže užívateľ v užívateľskom rozhraní vidieť, ako toto rozhranie ovládať a kde hľadať potrebné informácie. Komunikácia medzi aplikáciou a užívateľom

Aplikácia má vypracované vlastné grafické prostredie, ktoré sprostredkovať jej komunikáciu s užívateľom. Toto grafické prostredie je tvorené z niekoľkých obrazoviek, ktoré sú jeho základnými prvkami a ktoré zobrazujú všetky informácie poskytované grafickým prostredím.

Komunikácia užívateľa s aplikáciou je sprostredkovaná skrz klávesnicu, ktorá je využívaná počas celého behu aplikácie ako hlavný zdroj vstupu od užívateľa. V niektorých situáciách je možné aplikáciu ovládať taktiež myšou, avšak tá v rámci tejto komunikácie nie je nepotrebná.

2.1. Obrazovky

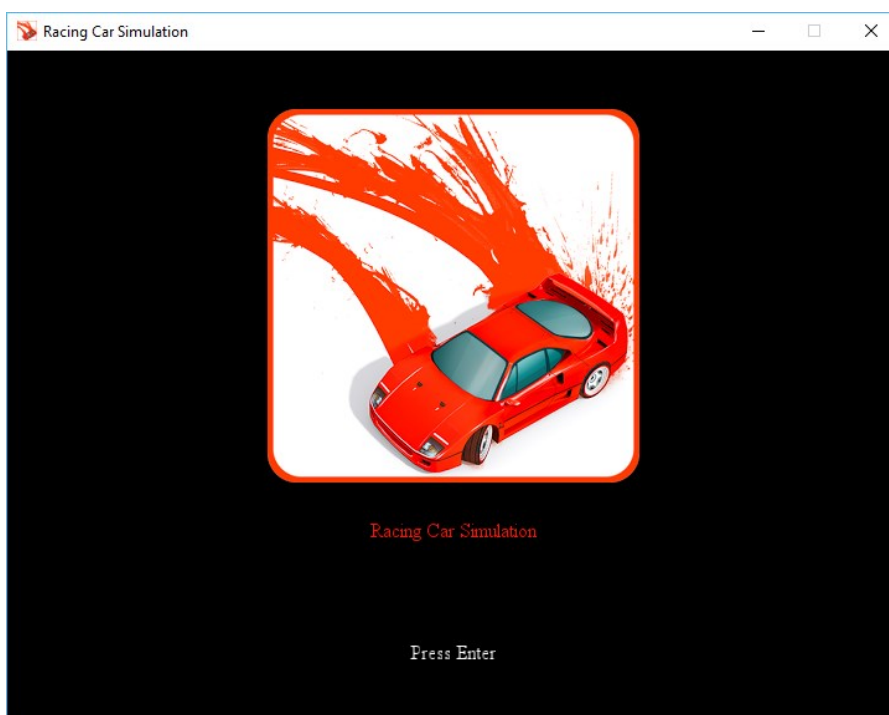
Grafické prostredie sa skladá celkovo z piatich obrazoviek. Tieto obrazovky na seba istým spôsobom naväzujú, pričom každá z nich má svoj špecifický význam. Po prechode na novú obrazovku je vždy možné stlačením klávesy *Esc* vrátiť aplikáciu na predchádzajúcu obrazovku.

2.1.1. Úvodná obrazovka

Úvodná obrazovka (viď obrázok 2.1.) predstavuje prvý bod, kedy sa užívateľ stretne s užívateľským rozhraním. Táto obrazovka je vytvorená za dvomi účelmi:

1. Privítať užívateľa a zoznámiť ho s prostredím aplikácie
2. Inicializovať hru

Keďže táto obrazovka je prvá, ktorá je po spustení aplikácie zobrazovaná, počas jej zobrazovania je väčšinou nutné vykonať základné inicializačné procesy a poprípadne na pozadí spustiť prednahrávanie objektov, ktoré budú neskôr v aplikácii použité. Po vykonaní všetkých potrebných operácií sa pre pokračovanie v aplikácii užívateľovi zobrazí na obrazovke text, ktorý ho vyzýva k stlačeniu klávesy *Enter*, čím spustí prechod na ďalšiu obrazovku.



Obrázok 2.1.: Ukážka úvodnej obrazovky

2.1.2. Menu obrazovka

Menu obrazovka (viď obrázok 2.2.) je hlavným navigačným prvkom v aplikácii. V strede tejto obrazovky je umiestnené vždy jedno menu s položkami, ktoré môžu mať rôznu funkčnosť. V menu môže byť aktívna vždy iba jedna položka, ktorá je graficky vyznačená červenou farbou.

Prepínanie medzi jednotlivými položkami je možné prostredníctvom šípok hore/dole alebo pomocou kurzoru myši. Ak sa užívateľ rozhodne použiť myš, musí jej kurzorom prejsť nad požadovanú položku, ktorá je v zapätí označená ako aktívna. Aktívnu položku je následne možné potvrdiť stlačením klávesy *Enter* alebo kliknutím ľavého tlačidla na myši.

V rámci tejto obrazovky sú k dispozícii 3 typy menu:

1. **Hlavné menu**

Hlavné menu je zobrazené vždy ako prvé menu po prechode na túto obrazovku a poskytuje základnú navigáciu v aplikácii. Skladá sa z piatich položiek:

1. Simulation
2. Options
3. QLearning
4. SplashScreen
5. Exit

Položka *Simulation* predstavuje prvú a zároveň hlavnú položku tohto menu. Ak je táto položka aktívna, po jej potvrdení dochádza k prechodu na ďalšiu obrazovku, konkrétne obrazovku konfigurácie simulácie (viď kapitolu 2.1.3.).

Položka *Options* vedie do hlavných nastavení užívateľského rozhrania. Po potvrdení tejto položky dochádza k zmene menu. Hlavné menu nahradí menu nastavení.

Položka *QLearning* slúži na spustenie aplikácie QLearning, ktorá je súčasťou tejto práce (viď kapitolu 5.4.).

Potvrdení položky *SplashScreen* dochádza k zmene obrazovky späť na úvodnú obrazovku.

Položka *Exit* slúži pre okamžité ukončenie aplikácie. Po jej potvrdení je aplikácia bez varovného vyzvania pre opätovné potvrdenie zatvorená.

2. Menu nastavení

Menu nastavení je zobrazené, ak užívateľ v hlavnom menu potvrdí položku *Options* a slúži pre navigáciu medzi nastaveniami užívateľského prostredia. Obsahuje nasledujúce štyri položky:

1. Graphics
2. Sound
3. Controls
4. Back

Položka *Graphics* vedie do menu nastavení grafiky. Položky *Sound* a *Controls* sú v tejto verzii aplikácie nedostupné. Po ich potvrdení sa zobrazí okno, ktoré informuje užívateľa o ich nedostupnosti. V tomto menu sú prítomné za účelom ďalšieho rozvoja aplikácie, v rámci ktorého budú slúžiť na prechod do nastavení zvuku a nastavení ovládania a priradenia kláves. Položka *Back* slúži pre návrat do hlavného menu.

3. Menu nastavení grafiky

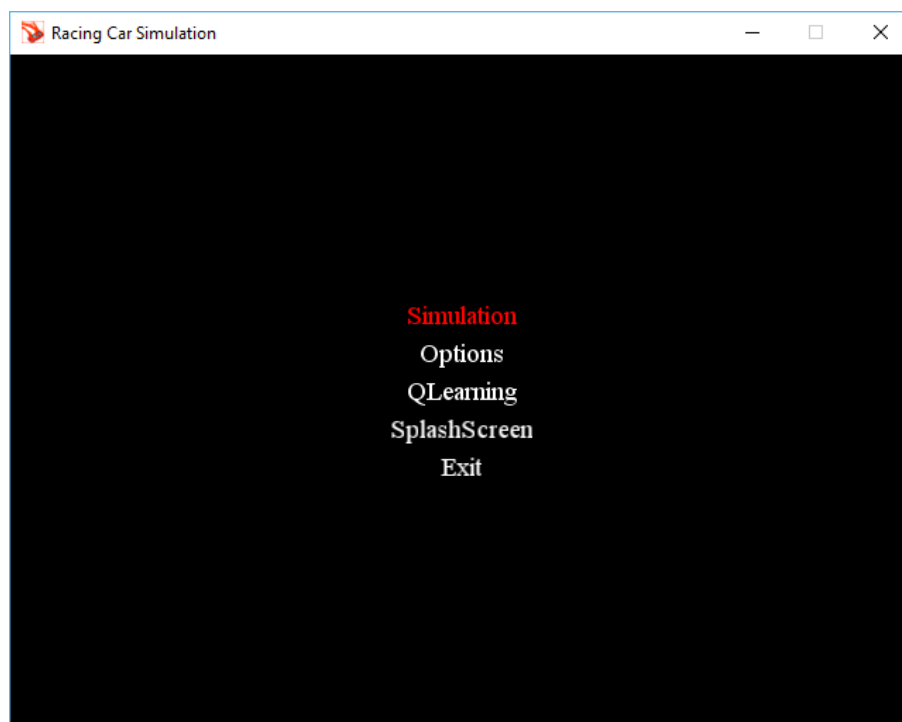
Nastavenie grafického prostredia je potrebnou súčasťou tejto aplikácie. Tieto nastavenia umožňujú užívateľovi prispôbiť vzhľad aplikácie jeho požiadavkám. Jedným z najdôležitejších nastavení grafiky je nastavenie rozlíšenia okna, v ktorom sú všetky obrazovky aplikácie vykresľované. Toto nastavenie je nevyhnutné k tomu, aby užívateľ mohol čo najlepšie prispôbiť zobrazovanie aplikácie svojmu monitoru. Menu nastavení grafiky slúži k týmto účelom a obsahuje dve položky:

1. Screen resolution
2. Fullscreen

Položka *Screen resolution* slúži na nastavenie rozlíšenia obrazovky. Položka je zobrazovaná v tvare „Screen resolution: $W \times H$ “, kde W predstavuje počet bodov v horizontálnom smere a H počet bodov vo vertikálnom smere. Pomocou šípok doľava/doprava môže užívateľ prepínať medzi dostupnými rozlíšeniami a klávesou *Enter* potvrdiť zmenu rozlíšenia. Dostupné rozlíšenia a ich modifikácia sú popísané v podkapitole 2.4.2.

Položka *Fullscreen* sa používa pre zapnutie/vypnutie režimu celej obrazovky. Ak je režim celej obrazovky zapnutý, rozlíšenie obrazovky je interne automaticky nastavené na rozlíšenie monitora, teda nemusí sa zhodovať s rozlíšením, ktoré je uvedené v položke *Screen resolution*. Až po vypnutí tohto režimu dochádza k zmene rozlíšenia podľa položky na aktuálne nastavené rozlíšenie obrazovky.

Prechod medzi uvedenými typmi menu je možný potvrdením príslušnej položky. Pre návrat na predchádzajúce menu slúži klávesa *Esc*.



Obrázok 2.2.: Ukážka obrazovky menu

2.1.3. Obrazovka konfigurácie simulácie

Simulácia pre svoj beh potrebuje niekoľko vstupných parametrov, ktoré musia byť zadané užívateľom. Tieto parametre sú do simulácie nahrávané zo súborov uložených na disku, ktoré ich reprezentujú. Zvolenie týchto parametrov je užívateľovi poskytnuté skrz túto obrazovku, na ktorej sa nachádza špeciálne menu obsahujúce 5 položiek (viď obrázok 2.3.).

Ako prvé, čo je užívateľ povinný si zvoliť je samotný automobil, ktorý si praje v simulácii ovládať a mapa, ktorá bude použitá na reprezentáciu prostredia automobilu a simulácie. Tieto dve vstupné parametre sú nevyhnutne potrebné pre spustenie simulácie. Ďalšími parametrami, ktoré si užívateľ môže zvoliť sú závodná stopa (viď kapitolu 3.1.3.) a konfigurácia umelej inteligencie (viď kapitolu 5.3.).

Pre zadanie parametrov slúžia prvé štyri položky špeciálneho menu, ktoré sa na tejto obrazovke nachádza. Tieto položky sú na obrazovke zobrazené v tvaroch:

- *Car*: názov zvoleného automobilu
- *Map*: názov zvolenej mapy
- *Racing line*: názov zvolenej závodnej stopy
- *QLearning*: názov konfigurácie umelej inteligencie

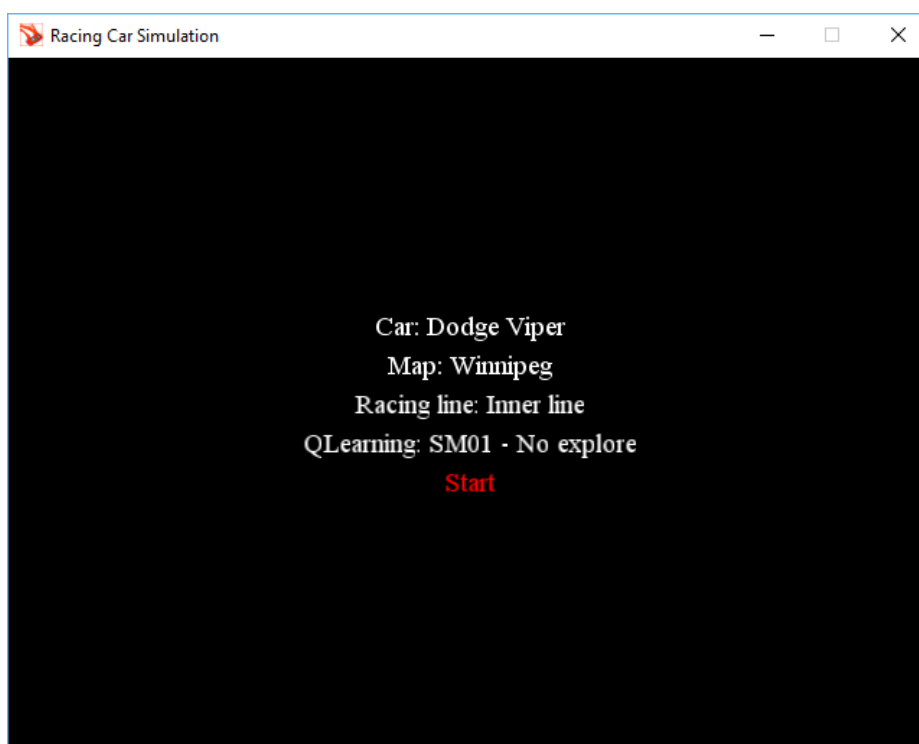
Ak nejaký z týchto parametrov zvolený nie je, namiesto jeho názvu je v položke uvedené „none“. Ak v súbore, ktorý reprezentuje daný parameter nie je uvedené jeho meno, v tejto položke bude namiesto mena parametru zobrazený názov daného súboru.

Užívateľ si pomocou týchto položiek môže vyberať spomedzi dostupných parametrov (viď kapitolu 2.3.1.2.) pomocou šípok doľava/doprava. Okrem dostupných parametrov má užívateľ možnosť zvoliť si parametre, ktoré si sám nadefinoval v ním vytvorených súboroch. Tieto súbory môžu byť umiestnené kdekoľvek na disku a ich výber je umožnený pomocou stlačenia klávesy *Enter*, po ktorom sa v rámci aplikácie zobrazí dialógové okno pre vybratie súboru, ktorý reprezentuje užívateľom požadovaný parameter. Štruktúra týchto súborov je

popísaná v kapitole 3.1. To, ktorý vstupný parameter užívateľ v tomto momente vyberá je určené tým, ktorá položka je práve aktívna.

V rámci výberu parametrov definovaných užívateľom platí ale jedno obmedzenie. Užívateľ si nemôže vybrať vlastnú závodnú stopu. Závodná stopa sa pevne viaže k zvolenej mape, čo znamená, že užívateľ môže listovať medzi dostupnými závodnými stopami, ktoré sú definované aktuálne zvolenou mapou, no zvoliť si inú závodnú stopu nie je možné.

Posledná položka tohto menu má názov *Start* a slúži na spustenie simulácie. Po jej potvrdení klávesou *Enter* dochádza najprv ku kontrole vstupných parametrov. Ak kontrola nie je úspešná, užívateľ sa o tom dozvie pomocou chybovej hlášky. V opačnom prípade dochádza k načítaniu zvolených parametrov a samotnej simulácie, ktoré prebieha na *obrazovke načítania simulácie* (viď nasledujúcu podkapitolu).



Obrázok 2.3.: Ukážka obrazovky konfigurácie simulácie

2.1.4. Obrazovka načítania simulácie

Obrazovka načítania simulácie slúži k jednoduchému účelu, a to k informovaniu užívateľa o stave načítania simulácie. Na tejto obrazovke je možné vidieť percentuálny progres v načítavaní simulácie a údaj informujúci užívateľa o tom, ktorý komponent simulácie je práve načítavaný. Po úspešnom načítaní dochádza k zmene obrazovky na *obrazovku simulácie* (viď nasledujúcu podkapitolu). V opačnom prípade je užívateľ prostredníctvom chybovej hlášky informovaný o chybe, ktorá počas načítavania nastala a dochádza k prechodu späť na obrazovku konfigurácie simulácie.

2.1.5. Obrazovka simulácie

Obrazovka simulácie (viď obrázok 2.4.) predstavuje najdôležitejšiu obrazovku aplikácie, prebieha v nej samotná simulácia. Jej komponenty majú za úlohu hlavne vizualizovať auto a jeho pohyb na mape, no taktiež sprostredkovať užívateľovi aktuálny stav simulácie a všetky informácie, ktoré sú so simuláciou spojené. Sú formované do troch hlavných celkov:

1. Panel so základnými údajmi o priebehu simulácie (viď kapitolu 3.5.1.)
2. Plocha pre vykresľovanie simulácie
3. Panely s informáciami o aktuálnom stave simulácie (viď kapitolu 3.5.2.)

Vizualizácia simulácie je užívateľovi podaná prostredníctvom plochy pre vykresľovanie simulácie. Na túto plochu sú vykresľované všetky vizuálne prvky simulácie. Týmito prvkami sú automobil, mapa, závodná stopa a vizualizácia umelej inteligencie v podobe šípky, ktorá reprezentuje aktuálnu akciu (viď podkapitolu 5.2.2.) umelého hráča. Plocha je centrovaná vzhľadom k automobilu, ktorý je umiestnený v jej strede a za normálnych okolností sa počas simulácie jeho pozícia na ploche nemení, mení sa iba jeho natočenie. Zvyšná časť plochy je vyplnená mapou, ktorá je v pohybe. Pohyb mapy na ploche odpovedá pohybu automobilu na mape. Výnimka nastáva v situácii, kedy automobil svojim pohybom opúšťa mapu. Plocha už nie je centrovaná vzhľadom k autu a pozícia auta na ploche sa začne meniť

odpovedajúco k pohybu auta na mape. Ak automobil opustí mapu, opustí aj túto plochu, na ktorej ostane vykreslená už iba mapa.



Obrázok 2.4.: Ukážka obrazovky simulácie

2.2. Log aplikácie

Logy sú vo veľkom množstve aplikácii a hier použité ako nástroj, ktorý pasívnym spôsobom informuje užívateľa alebo vývojára o záležitostiach, ktoré sú spojené s behom aplikácie a s činnosťou užívateľa. Keď aplikácia implementuje logovací mechanizmus, väčšinou sa pri každom spustení aplikácie vytvorí na disku textový súbor s unikátnym menom, do ktorého sa počas behu aplikácie vypisujú informácie o tom, ako je aplikácia využívaná, ako sa mení jej stav, aké akcie vykonáva užívateľ, aké chyby sa počas behu objavili a tak ďalej. Väčšinou sú to informácie, ktoré súvisia s vývojom aplikácie alebo informácie, o ktorých užívateľ nepotrebuje byť nutne informovaný prostredníctvom užívateľského rozhrania.

Takýto logovací mechanizmus je súčasťou aj tejto aplikácie. Pri každom spustení sa na disku vytvorí súbor, ktorého meno je v tvare „Gamelog_DD-MM-YY_hh-mm-ss“, kde:

- *DD* značí deň, *MM* značí mesiac a *YY* značí posledné dve číslice roku dátumu, kedy bola aplikácia spustená
- *hh* značí hodinu, *mm* počet minút a *ss* počet sekúnd času spustenia aplikácie

Tento súbor slúži predovšetkým k vývojárskym účelom, avšak užívateľ ho môže použiť v prípade, keď sa niektorá časť aplikácie správa neobvyklým spôsob alebo keď nejaká funkcionálnosť aplikácie nie je dostupná, pretože akákoľvek chyba, ktorá v aplikácii nastane je automaticky zalogovaná.

Nasledujúci riadok predstavuje príklad toho, ako môže vyzeráť zalogovaná informácia:

00:00:11.7242072 [Message] (ScreenManager) Screen changing - new screen type: Simulation

Na začiatku riadku je vždy uvedený čas, kedy bola informácia zalogovaná. Tento čas je relatívny k momentu spustenia hry. Za časom nasleduje typ správy uzavretý do hranatých zátvoriek. Užívateľ sa počas práce s aplikáciou môže stretnúť s 3 typmi logovacích správ:

1. Message: predstavuje neutrálnu správu o tom, že sa v rámci aplikácie niečo udialo
2. Warning: predstavuje upozornenie na situáciu, ktorá by za normálnych okolností nemala nastať, avšak neohrozuje beh a správne chovanie aplikácie
3. Error: predstavuje chybu, ktorá môže viesť k obmedzeniu funkcionality aplikácie alebo dokonca k jej ukončeniu.

Typ správy je nasledovaný odosielateľom logovacej správy. Odosielateľ logovacej správy je informácia, ktorá je určená pre vývojárov a zvyčajne predstavuje triedu, z ktorej bola správa zalogovaná.

Za odosielateľom nasleduje samotný text správy.

Ak počas behu aplikácie dôjde k vyhodneniu výnimky, ktorá nebola zachytená alebo ak ošetrenie výnimky zahŕňa zápis do logu, je do logovacieho súboru zapísaná logovacia správa typu Error, ktorej text okrem správy výnimky zahŕňa aj výpis zásobníka volaní (stack trace).

Logovacie súbory je možné nájsť v adresári, ktorý je popísaný v kapitole 2.4.1.2.. Ukážka takéhoto súboru je obsiahnutá v prílohe A.

2.3. Štruktúra aplikácie

2.3.1. Adresáre

Aplikácia Racing Car Simulation pracuje nad dvomi hlavnými adresármi:

1. Koreňový adresár aplikácie
2. Užívateľský adresár

2.3.1.1. Koreňový adresár

Koreňový adresár je adresár, v ktorom je umiestnený spustiteľný (exe) súbor tejto aplikácie. Slúži ako základný adresár pre súbory, ktoré sú potrebné pre samotný beh aplikácie. Obsahuje 2 dôležité podadresáre:

1. Configurations
2. Content

Adresár *Configurations* obsahuje všetky konfigurácie aplikácie (viď kapitolu 2.3.2.). Jeho podadresáre je možné rozdeliť do dvoch skupín.

1. Adresáre obsahujúce konfigurácie nevyhnutné pre chod a spustenie aplikácie. Medzi tieto adresáre patria adresáre „Screen“ a „Menu“.
2. Adresáre obsahujúce konfigurácie, ktoré nie sú nevyhnutne potrebné pre beh aplikácie, avšak bez týchto konfigurácií je obmedzená jej funkcionálnosť. Medzi tie adresáre patrí adresár „Gameplay“.

Pre úspešný beh aplikácie je potrebné, aby adresáre prvej skupiny existovali a obsahovali potrebné súbory. Z tohto dôvodu je pri zobrazení úvodnej obrazovky vykonaná kontrola ich existencie. Ak niektorý z týchto adresárov chýba, hra nemôže pokračovať a po zobrazení príslušnej chybovej hlášky sa automaticky ukončí. Táto kontrola takisto zahŕňa kontrolu existencie adresárov druhej skupiny, avšak v prípade, že tieto adresáre alebo niektoré z ich podadresárov neexistujú, sú chýbajúce adresáre automaticky vytvorené s prázdny obsahom a informácia o týchto operáciách je zapísaná do logu aplikácie.

Adresár *Content* obsahuje všetok grafický obsah hry, t.j. obrázky a fonty.

2.3.1.2. Užívateľský adresár

Užívateľský adresár je uložený v adresári dokumentov aktuálne prihláseného užívateľa a má cestu:

C:\Users\userName\Documents\RacingCarSimulation

Tento adresár obsahuje súbory reprezentujúce dáta, ktoré sú do simulácie nahrávané vo forme vstupných parametrov (viď kapitolu 3.1.). Tieto súbory sú podľa typu rozdelené do podadresárov:

1. Cars
2. Maps
3. RacingLines
4. QLearning
 - 4.1 QLearning/Configurations
 - 4.2 QLearning/Data

Vymenované podadresáre okrem adresáru QLearning/Data predstavujú prednastavené lokácie pre súbory reprezentujúce vstupné parametre simulácie, ktoré sú v rámci *obrazovky konfigurácie simulácie* reprezentované ako dostupné parametre, z ktorých si užívateľ môže vybrať (viď kapitolu 2.1.3.).

Navyše je v tejto lokácii umiestnený adresár Gamelog, v ktorom sa nachádzajú všetky logovacie súbory vytvorené aplikáciou.

Kontrola validity užívateľského adresára a všetkých jeho podadresárov prebieha taktiež pri zobrazení úvodnej obrazovky. Tieto adresáre avšak nie sú pre samotné spustenie a beh aplikácie potrebné, preto ak neexistujú, sú automaticky vytvorené, avšak ich obsah je prázdny a pre spustenie simulácie je užívateľ nútený vybrať si príslušné parametre pomocou vlastne vytvorených súborov. Vytvorenie chýbajúcich adresárov je dokumentované v logu aplikácie.

2.3.2. Konfigurácie aplikácie

Za účelom čo najväčšej konfigurovateľnosti užívateľského prostredia funguje táto aplikácia na základe konfigurácii v podobe xml súborov uložených na disku. V týchto súboroch sa nachádzajú informácie, ktoré definujú ako obsah, vzhľad a reakciu na vstup užívateľa pre všetky obrazovky, tak aj všetok obsah menu v Menu obrazovke a panely a konzolu v Obrazovke simulácie (viď kapitoly 3.5. a 3.6.). Tieto konfigurácie by za žiadnych okolností nemali byť modifikované neznalým užívateľom, pretože môžu vážne narušiť chod a správne fungovanie aplikácie.

Ku konfiguráciám aplikácie patrí aj súbor *GameConfiguration.xml*, ktorý ale na rozdiel od vyššie spomenutých konfigurácií užívateľ môže modifikovať. Tento súbor obsahuje užívateľské nastavenie aplikácie. V štruktúre tohto súboru užívateľ môže nájsť nastavenie grafiky a priradenie kláves k jednotlivým operáciám, ktoré môže v rámci simulácie vykonať.

Nastavenia grafiky sú uzavreté v elemente *ScreenConfiguration*. V rámci tohto elementu môže užívateľ:

- Nastaviť ľubovoľné rozlíšenie obrazovky, ktoré je obsiahnuté v elemente *ScreenResolution*
- Modifikovať zoznam dostupných rozlíšení, ktorým je možné listovať v nastaveniach. Každý prvok zoznamu je uzavretý v elemente *AvailableScreenResolution*. Pokiaľ zoznam neobsahuje rozlíšenie

monitora alebo minimálne rozlíšenie (640x480), budú tieto dve rozlíšenia do zoznamu pridané automaticky

- Nastaviť zapnutie/vypnutie režimu celej obrazovky pomocou priradenia hodnoty *true/false* do elementu *IsFullScreen*
- Zapnúť/vypnúť automatické centrovanie obrazovky pomocou priradenia hodnoty *true/false* do elementu *AutoCenterDisabled*

Priradenie kláves k operáciám simulácie je uzavreté v elemente *KeysAssignmentConfiguration*. Každé priradenie je reprezentované elementom *KeyAssignment*, ktorý obsahuje:

- kód akcie (element *ActionCode*) (viď tabuľka 3.18.)
- kód klávesy (element *KeyCode*), ktorý predstavuje číslo, ktoré jednoznačne definuje jednu klávesu (viď príloha A)
- názov klávesy (element *KeyName*)
- kód klávesy, ktorá je použitá ako modifikátor, napr. klávesa Ctrl alebo Shift (element *ModifierCode*)

Ak súbor *GameConfiguration.xml* neexistuje alebo je poškodený, v hre bude použitá prednastavená konfigurácia (default configuration). V takomto prípade je o tom užívateľ informovaný správou a súbor *GameConfiguration.xml* je vytvorený/prepísaný.

3. Užívateľská dokumentácia simulácie

Simulácia spolu s jej grafickým prevedením je neoddeliteľnou súčasťou tejto aplikácie. Jej základ tvorí fyzikálny model automobilu (viď kapitola 4.), ktorý sprostredkováva všetko pohyby a správanie sa automobilu na mape. Pred jej načítaním užívateľ musí zadať vstupné parametre, ktorými sú:

- Automobil
- Mapa
- Závodná stopa
- Konfigurácia umelej inteligencie

Parametre automobil a mapa sú povinné, bez nich simuláciu nie je možné spustiť. Závodná stopa a konfigurácia umelej inteligencie patria medzi nepovinné parametre, ktoré pre samotné jazdenie s autom po mape nie sú potrebné. Patria medzi parametre umelej inteligencie, ktorá v rámci simulácie môže byť spustená. Umelá inteligencia, je v aplikácii použitá za účelom naučiť umelého hráča jazdiť s automobilom po definovanej závodnej stope (viď kapitola 5.).

3.1. Parametre simulácie

Takisto ako konfigurácie, vstupné parametre simulácie sú uložené v súboroch vo formáte xml. Ich štruktúra je vysvetlená v nasledujúcich podkapitolách. Ukážku týchto súborov je možné vidieť v príloha A.

3.1.1. Automobil

Automobil v simulácii predstavuje prvok, s ktorým je užívateľ neustále v kontakte. Pre jeho vizualizáciu je potrebné, aby mal každý automobil k sebe priradený obrázok, ktorý bude v simulácii použitý a pre správne fungovanie fyzikálneho modelu je nutné, aby bol model automobilu opísaný množinou parametrov.

Reprezentácia automobilu v aplikácii je rozložená do dvoch samostatných súborov:

1. *car.xml*¹
2. *car-model.xml*

Súbor *car.xml* obsahuje základné informácie o automobile. Medzi tieto informácie patrí:

1. Celý názov automobilu
2. Validná cesta k obrázku automobilu
3. Validná cesta k súboru *car-model.xml*

Pred načítaním obrázku a súboru *car-model.xml* prebieha kontrola týchto údajov. Ak cesta k obrázku alebo k súboru *car-model.xml* nie je validná alebo súbor daný cestou neexistuje, užívateľ je o tom informovaný prostredníctvom chybovej hlášky, načítavanie simulácie sa preruší a aplikácie je vrátená späť na obrazovku konfigurácie simulácie.

V súbore *car-model.xml* sa nachádzajú všetky potrebné parametre auta, ktoré definujú jeho rozmery a hmotnosť, aerodynamiku, motor, prevodovku a kolesá. Tieto parametre sú nasledovné (v zátvorke je uvedený názov príslušného xml elementu):

1. Rozmery (*Dimensions*)
 - a. Dĺžka automobilu udávaná v metroch (*Length*)
 - b. Šírka automobilu udávaná v metroch (*Width*)
 - c. Výška automobilu udávaná v metroch (*Height*)
2. Parametre motora (*Engine*)
 - a. Maximálne otáčky motora udávané v jednotkách (*MaxRpm*)
 - b. Krútiace momenty motora v Newton-metroch (*TorqueAtRpm*)
3. Prevodovka (*Transmission*)
 - a. Stály prevod v jednotkách (*DifferentialRatio*)

¹ názov týchto súborov sa môže byť odlišný, názov *car* je uvedený ako príklad

- b. Účinnosť v percentách (*Efficiency*)
 - c. Zoznam prevodových stupňov, každý stupeň definuje:
 - i. Poradové číslo prevodového stupňa (*Number*)
 - ii. Prevodový pomer (*Ratio*)
- 4. Kolesá (*Wheels*)
 - a. Koeficient priľnavosti pneumatiky v jednotkách (*TyreFrictionCoefficient*)
 - b. Polomer kolesa v metroch (*WheelRadius*)
 - c. Hmotnosť kolesa (*WheelMass*)
- 5. Hmotnosť automobilu v kilogramoch (*Mass*)
- 6. Koeficient zotrvačnosti v jednotkách (*Inertia*)
- 7. Brzdná konštanta v jednotkách (*BrakingConstant*)

Poznámky:

- Krútiaci moment motora je rôzny v závislosti na otáčkach motora, preto je v tomto súbore uvedený zoznam krútiacich momentov. Prvá položka zoznamu definuje krútiaci moment v otáčkach 1000rpm, druhá položka definuje krútiaci moment v otáčkach 2000rpm, atď..
- Číslovanie prevodových stupňov:
 - -1: spiatočka
 - 0: neutrál
 - 1 – n: prevodové stupne 1 až n
- Význam netriviálnych parametrov je vysvetlený v rámci kapitoly 4., v ktorej sú tieto parametre zahrnuté v rámci opisu fungovania fyzikálneho modelu automobilu.

Automobil môže byť v simulácii ovládaný užívateľom alebo umelým hráčom.

Medzi základné akcie, ktorými sa auto ovláda patrí:

1. Ovládanie volantu – zatáčanie doľava/doprava
2. Ovládanie plynového pedálu
3. Ovládanie brzdového pedálu
4. Zaradenie vyššieho rýchlostného stupňa

5. Zaradenie nižšieho rýchlostného stupňa
6. Zmena medzi manuálnou a automatickou prevodovkou
7. Zmena miery zatočenia kolies pri plnom zatočení volantu

V prípade, že automobil ovláda užívateľ je ku každej akcii priradená práve jedna klávesa. Priradenie kláves popisuje kapitola 3.6..

3.1.2. Mapa

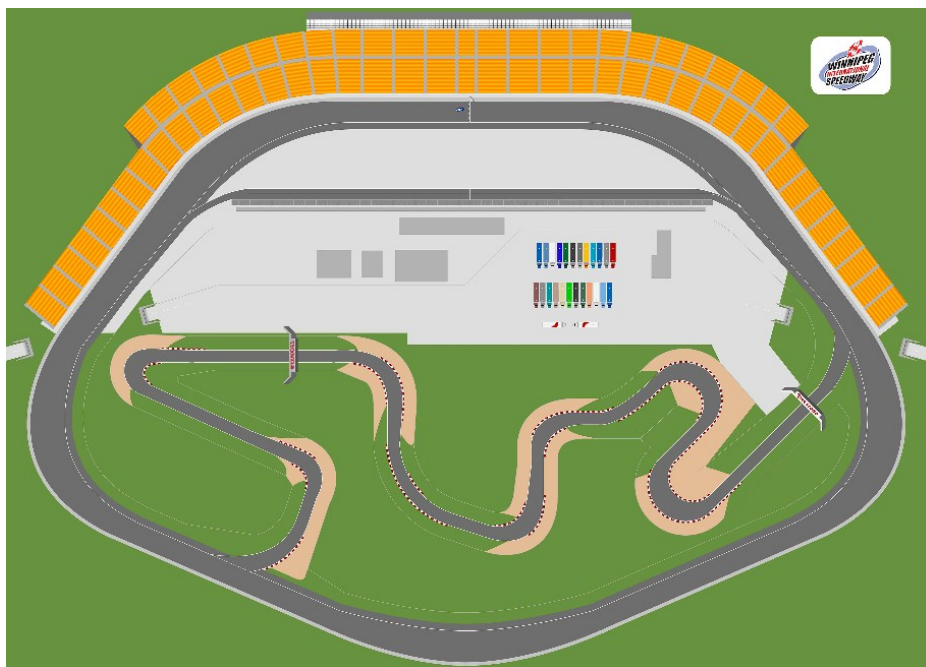
Mapa je neoddeliteľnou súčasťou simulácie, pomocou ktorej je užívateľovi sprostredkované 2D grafické prostredie simulácie. Každá mapa sa skladá z minimálne jedného typu povrchu. Správanie auta a jeho pohyb sa rôzni podľa toho, na akom povrchu sa práve nachádza. Napríklad na asfalte má auto väčšiu priľnavosť ako keď sa pohybuje po trávnom povrchu. Priľnavosť povrchu je definovaná koeficientom priľnavosti, ktorý je používaný v prepočtoch fyzikálneho modelu automobilu. Okrem tohto koeficientu povrch reprezentuje koeficient odporu vozovky, ktorý určuje mieru veľkosti síl, ktoré na auto skrz vozovku pôsobia.

Pre účely simulácie sa v aplikácii nachádzajú 2 typy máp:

1. **Konečná mapa so závodným okruhom (TrackMap)**

Tento typ mapy ponúka užívateľovi možnosť jazdiť s autom po závodnom okruhu. Jej hlavnou časťou je samotná trať, ktorá je určená pre pohyb automobilu. Okrem trate je na mape prítomné okolité prostredie, ktoré môže pozostávať z trávnatých plôch, rôznych budov a iných ciest alebo z výjazdových zón, ktoré sa typicky objavujú v okolí zákrut (viď obrázok 3.1). Tieto jednotlivé časti mapy patria medzi spomínané povrchy mapy a ich vplyv na pohyb automobilu je definovaný koeficientom priľnavosti, ktorý je pre každý povrch uvedený v xml súbore reprezentujúcom danú mapu. Mapa navyše definuje prednastavený koeficient priľnavosti, ktorý je použitý v prípade, že pre povrch, na ktorom sa automobil nachádza z nejakého dôvodu nie je koeficient priľnavosti definovaný.

V simulácii je v tejto verzii aplikácie dostupná jedna takáto mapa, ktorá predstavuje skutočný okruh v kanadskom meste Winnipeg, podľa ktorého je v tejto aplikácii aj nazvaná.

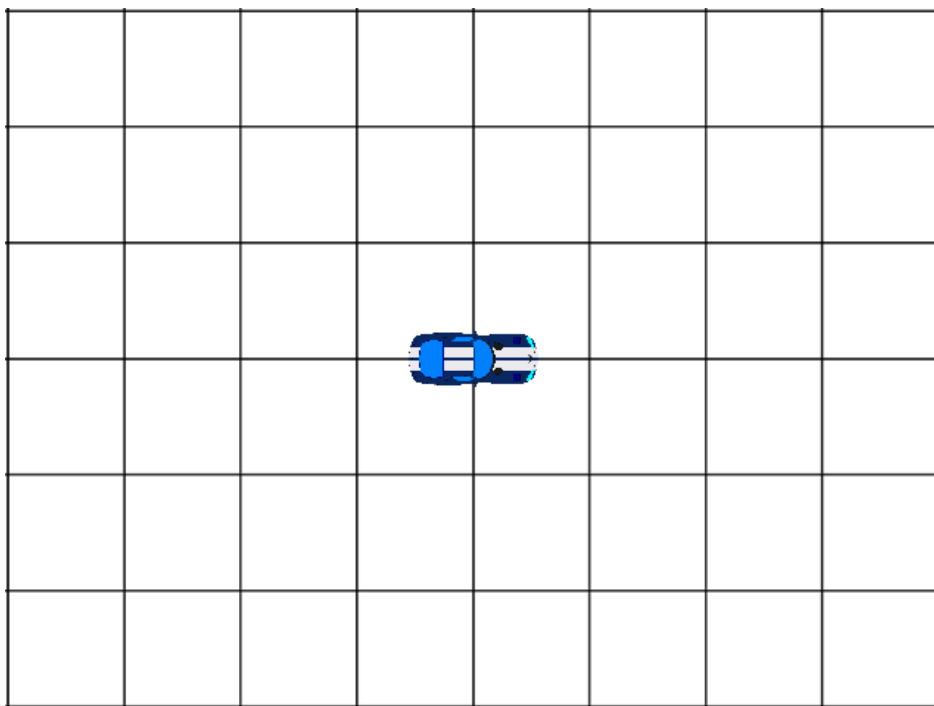


Obrázok² 3.1.: Ukážka mapy so závodným okruhom – Winnipeg

2. Nekonečná mapa (SimulationMap)

Pre testovanie fyziky automobilu je vhodné nebyť obmedzený traťou a ohraňčenou konečnou mapou a namiesto toho mať k dispozícii nekonečný priestor pre ľubovoľný pohyb auta. K tomuto účelu je vytvorená nekonečná mapa, ktorá neobsahuje cestu ani žiadné iné okolné prostredie. Mapa pozostáva z dlaždíc (viď obrázok 3.2.), pomocou ktorých je vizualizovaný pohyb automobilu. Tieto dlaždice tvoria jediný povrch mapy a ich koeficient príľnavosti a koeficient odporu vozovky je konštatne nastavený na 1,0.

² Zdroj obrázku mapy: <http://www.deviantart.com>



Obrázok 3.2.: Ukážka nekonečnej mapy

Mapa je na disku takisto ako automobil reprezentovaná xml súborom. Štruktúra tohto súboru závisí na type mapy, avšak koreňový element nazvaný *MapXmlWrapper* je spoločný. Tento xml element obsahuje práve jeden podelement, pomocou ktorého je definovaný typ mapy, konkrétne to je:

- *TrackMap* pre konečnú mapu so závodným okruhom
- *SimulationMap* pre nekonečnú mapu

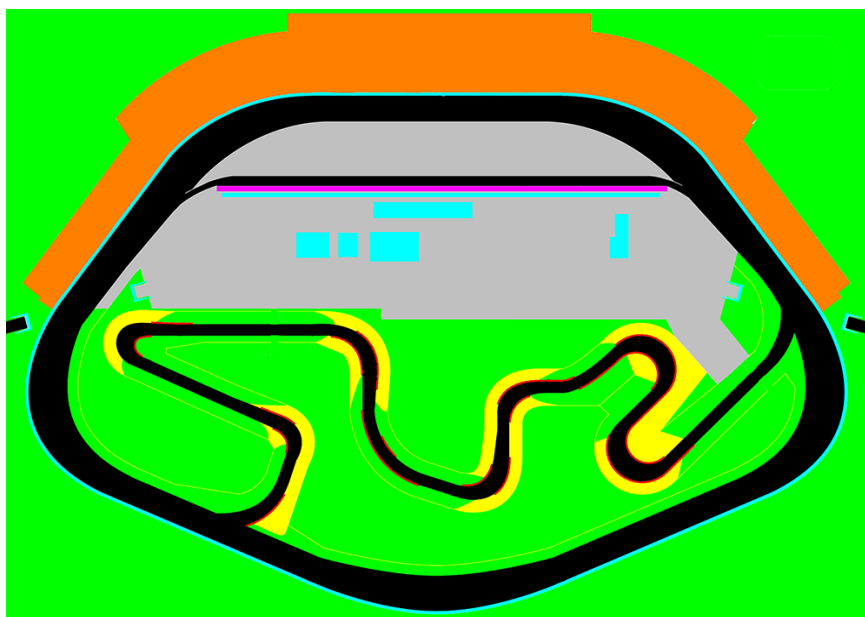
Obsah týchto xml elementov je rôzny, avšak zdieľajú niektoré podelementy, ktoré sú potrebné pre definíciu každej mapy v simulácii. Sú to elementy:

- *Name*: predstavuje meno mapy
- *PixelsPerMeter*: predstavuje počet pixelov, ktoré v prostredí simulácie vymedzujú dĺžku jeden meter
- *RacingLineInfo*: definuje jednu závodnú stopu (viď nasledujúcu podkapitolu)

Okrem týchto spoločných elementov je v reprezentácii mapy podľa jej typu potrebné definovať niekoľko dodatočných údajov. Pre *konečnú mapu so závodným okruhom* sú to jednotlivé povrchy a ich koeficient priľnavosti, pozícia, na ktorej je auto na začiatku simulácie umiestnené a v neposlednej rade samotný obrázok mapy. Pre *nekonečnú mapu* ostáva už iba definícia obrázku jednej dlaždice, keďže mapa obsahuje iba jeden povrch s vopred prednastaveným koeficientom priľnavosti a iníciaľna pozícia auta je definovaná ako bod so súradnicami [0, 0].

Obrázok *mapy so závodným okruhom* je v xml súbore reprezentovaný xml elementom *TrackPath*, ktorý udáva cestu k obrázku mapy. Rovnakým spôsobom je riešený obrázok dlaždice pre *nekonečnú mapu*, ktorého cesta je uložená v xml elemente *TilePath*. Ak sú tieto cesty v zlom formáte alebo odkazujú na neexistujúci obrázok, pri načítaní simulácie sa užívateľovi zobrazí chybová hláška, ktorá ho o tejto skutočnosti informuje a načítavanie sa preruší.

Jednotlivé povrchy *mapy so závodným okruhom* sú reprezentované pomocou dodatočného obrázku, ktorý by mal byť rovnako veľký ako obrázok mapy. Tento obrázok sa skladá z takzvaných zón. Každá zóna je jednoznačne určená unikátnou identifikačnou farbou a v mape ohraničuje oblasť, kde sa daná zóna vyskytuje (viď obrázok 3.3.).



Obrázok 3.3.: znázornenie povrchových zón pre mapu Winnipeg

K tomuto obrázku je ďalej dodaný zoznam informácii o týchto zónach. Každá položka zoznamu definuje jednu zónu, resp. jeden povrch. Na identifikáciu zóny je použitá jej identifikačná farba a k tejto zóne je potom priradený koeficient príľnavosti a názov povrchu.

V xml súbore slúži na definíciu povrchov element *Zones*. Jeho obsah tvorí:

- podelement *ImagePath*, ktorý obsahuje cestu k obrázku, v ktorom sú vymedzené jednotlivé zóny
- podelementy *DefaultFrictionCoefficient* a *DefaultResistanceCoefficient*, ktoré sú použité, ak pri identifikácii zóny dôjde k chybe
- zoznam informácii o týchto zónach. Jednotlivé prvky tohto zoznamu obsahujú elementy, ktoré pre danú zónu definujú:
- Názov povrchu (element *Name*)
- Identifikačnú farbu povrchu (element *Color* s podelementami *R*, *G*, *B*)
- Koeficient príľnavosti (element *FrictionCoefficient*)
- Koeficient odporu vozovky (element *ResistanceCoefficient*)

Iniciálna pozícia automobilu na mape so závodným okruhom je uložená v elemente *StartPosition*. Túto pozíciu definujú súradnice bodu, ktorý udáva miesto na mape a smer, ktorým je auto natočené. Súradnice iniciálneho bodu sú uložené v podelemente *Position* a smer natočenia automobilu v podelemente *Angle*.

3.1.3. Závodná stopa

Závodná stopa je v simulácii zahrnutá predovšetkým za účelom reprezentácie prostredia pre umelého hráča, ktorý sa po tejto závodnej stope snaží naučiť jazdiť čo najrýchlejšie. Závodná stopa je nevyhnutne spojená s mapou, ku ktorej patrí a preto užívateľ v aplikácii nemôže ako parameter simulácie vybrať ľubovoľnú kombináciu mapy a závodnej stopy, závodnú stopu si musí vždy vybrať z množiny závodných stôp, ktorú definuje samotná mapa.

V simulácii existujú dva typy závodných stôp:

- závodná stopa pre mapu so závodným okruhom
- závodná stopa pre *nekonečnú mapu*

Závodná stopa pre *mapu so závodným okruhom* je reprezentovaná kubickou Beziérovou uzavretou krivkou. Takáto reprezentácia sa využíva vo vektorovej grafike, čo je aj hlavný dôvod, prečo bola zvolená za najvhodnejšiu. Užívateľ môže jednoducho vo vhodnom nástroji pre vektorovú grafiku (napríklad freeware Inkscape) kresliť krivku priamo na obrázok mapy a túto krivku uložiť ako súbor vo formáte svg. Túto krivku vo vektorovej grafike reprezentuje množina štvoríc bodov, ktoré dokopy tvoria celú krivku. Každá štvorica bodov jednoznačne definuje jednu časť tejto krivky, ktorá je v rámci tejto aplikácie nazývaná **segment závodnej stopy**. Formát svg je špeciálny formát xml a beziérova krivka je v ňom uložená ako hodnota atribútu *d* v elemente *path* (viď ukážku súboru v prílohe A). Aplikácia tento súbor načíta a krivku prevedie do vlastnej reprezentácie, ktorá je použitá v simulácii a uložená vo vlastnom xml súbore.

Každá mapa vo svojom xml súbore môže definovať ľubovoľný počet závodných stôp. Tieto závodné stopy sú definované v elemente *RacingLineInfo*. Obsah tohto elementu závisí na type mapy. Pre mapu so závodným okruhom platí, že tento element obsahuje tri podelementy:

- *Name*: obsahuje názov závodnej stopy
- *XmlPath*: obsahuje cestu k xml súboru, v ktorom je uložená závodná stopa v reprezentácii simulácie
- *SvgPath*: obsahuje cestu k svg súboru, ktorý obsahuje závodnú stopu ako beziérovu kubickú krivku

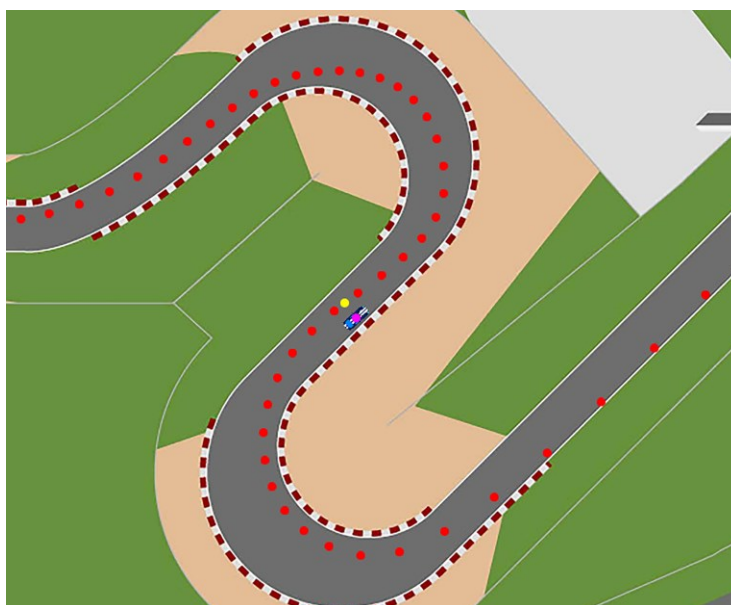
Pri načítaní závodnej stopy je uprednostňovaný xml súbor, v ktorom je uložená závodná stopa v reprezentácii, ktorá môže byť ihneď použitá v simulácii. Ak tento súbor neexistuje, aplikácia načíta svg súbor a beziérovú kubickú krivku, ktorú následne prevedie do vlastnej reprezentácie. Túto reprezentáciu následne uloží do

novovo vytvoreného súboru xml, ktorého cesta je daná v elemente *XmlPath*. Pri opätovnom načítaní simulácie bude krivka načítavaná už z tohto súboru.

To avšak znamená, že ak užívateľ vo svojom nástroji pre vektorovú grafiku modifikuje krivku závodnej stopy, ktorú potom uloží do daného súboru svg a spustí simuláciu, zmeny na danej krivke neuvidí, pretože tieto zmeny sa neuložili do súboru xml. V tomto prípade musí užívateľ súbor xml vymazať a nechať aplikáciu, aby si ho znovu vytvorila s novou, upravenou krivkou.

Závodná stopa pre *nekonečnú mapu* je v tejto verzii aplikácie iba jedna, nekonzfigurovateľná. Ako krivku ju môžeme opísať horizontálnou priamkou, ktorá prechádza bodom, na ktorom je auto umiestnené po načítaní simulácie. Pre účely ďalšieho rozširovania aplikácie sa v xml súbore mapy nachádza element *RacingLineInfo* a obsahuje iba jeden podelement *Name*. Každý takýto element ale odkazuje na tú istú závodnú stopu. V budúcich verziách aplikácie by však bolo možné pridávať do nekonečnej mapy závodné stopy iných tvarov a konfigurovať ich skrz tieto elementy.

Závodná stopa je na mape graficky realizovaná pomocou zobrazovania bodov, ktoré na tejto závodnej stope ležia (viď obrázok 3.4.). Vzdialenosť medzi týmito bodmi je nastaviteľná užívateľom (viď podkapitolu 3.6.)



Obrázok 3.4.: znázornenie závodnej stopy pomocou bodov

3.2. Režimy simulácie

Simulácia môže bežať v celkom troch režimoch. Sú to:

- Jazdenie (Driving)
- Učenie (Learning)
- Prehľad učenia (Learning overview)

3.2.1. Jazdenie

Tento režim je základný režim simulácie. Ak simulácia beží v režime *Jazdenie*, užívateľ ovláda automobil pomocou príslušných kláves na klávesnici. Tento vstup užívateľa je zahrnutý do fyzikálneho modelu automobilu, ktorý na základe vnútorných prepočtov upravuje hodnoty jednotlivých fyzikálnych veličín, ktoré zabezpečujú pohyb auta na trati.

3.2.2. Učenie

V tomto režime kontrolu nad automobilom preberá umelý hráč, ktorý implementuje umelú inteligenciu. V každom prepočte umelej inteligencie je vybraná jedna z množiny akcií, ktoré predstavujú to, čo môže jazdec v danom momente urobiť (pridať plyn, zabrzdíť alebo zatočiť doľava/doprava). Užívateľ v tomto režime sleduje vývoj umelej inteligencie a samotný proces učenia v reálnom čase.

Po prepnutí do tohto režimu je načítaná umelá inteligencia a aplikovaná konfigurácia, ktorú si užívateľ zvolil ako vstupný parameter (viď kapitolu 5.3.). Ak tento parameter nebol zvolený, aplikácia užívateľa vyzve k dodatočnému vybratiu konfigurácie, ktorá je na disku reprezentovaná súborom xml. Ak si užívateľ nevyberie žiaden súbor, aplikácia sa ho opýta, či chce použiť prednastavenú konfiguráciu. Ak užívateľ v takomto prípade zvolí zápornú odpoveď alebo vybraná konfigurácia nie je validná, prepnutie simulácie do režimu učenia neprebehne.

3.2.3. Prehľad učenia

Posledný režim simulácie slúži užívateľovi k tomu, aby si mohol pozrieť, čo sa za uplynulý čas stihla umelá inteligencia naučiť. Tento režim je veľmi podobný režimu učenia s jedným rozdielom – automobil ovláda užívateľ. Vďaka tomu môže auto dostať do rôznych pozícií vzhľadom k závodnej stope a sledovať, čo si umelý hráč myslí, že je v danej situácii najlepšie urobiť, resp. ktorá akcia je pre danú situáciu najvhodnejšia.

3.3. Nastavenia simulácie

Simulácia počas svojho behu umožňuje užívateľovi nastavovať niektoré behové parametre a parametre jej grafického prostredia. Tieto parametre môže užívateľ nastavovať stláčaním príslušných kláves, ktoré sú popísané v podkapitole 3.6..

3.3.1. Nastavenie rýchlosti

Nastavenie rýchlosti patrí medzi najnákladnejšie nastavenia simulácie. Tá za normálnych okolností beží s koeficientom rýchlosti 1.0, čo znamená, že sa všetky prepočty dejú v reálnom čase. Avšak, rýchlosť simulácie môže užívateľ zmeniť príslušnými klávesmi. K dispozícii je niekoľko možností:

1. Zvýšiť rýchlosť simulácie: užívateľ zvýši koeficient rýchlosti o 0,1. Rýchlosť simulácie zhora obmedzená nie je.
2. Znížiť rýchlosť simulácie: užívateľ zníži koeficient rýchlosti simulácie o 0,1. Koeficient rýchlosti je zdola obmedzený na 0,1.
3. Nastaviť koeficient rýchlosti späť na 1,0.

Ďalším spôsobom ako je možné ovládať simuláciu je krokovanie. Keďže sú hodnoty jednotlivých fyzikálnych veličín v reálnom čase prepočítavané niekoľko desiatok krát za minútu, sledovanie ich vývoja môže byť pre užívateľa veľmi obtiažné. Za týmto účelom bolo do hry implementované krokovanie, ktoré môže fungovať v dvoch režimoch:

- Krokovanie prepočtu fyzikálneho modelu
- Krokovanie prepočtu umelej inteligencie

Ak je krokovanie v režime *krokovanie prepočtu fyzikálneho modelu* platí, že ďalší prepočet fyzikálneho modelu prebehne až po stlačení príslušnej klávesy. Ak je krokovanie v režime *krokovanie prepočtu umelej inteligencie* platí, že nasledujúci prepočet umelej inteligencie prebehne až po stlačení príslušnej klávesy.

Zmena režimu krokovania sa udeje po stlačení príslušnej klávesy. Klávesa pre nasledujúci krok je rovnaká pri oboch režimoch krokovania.

Okrem zmeny rýchlosti a krokovania je možné simuláciu zastaviť (Pauza, taktiež stlačením príslušnej klávesy).

3.3.2. Nastavenie priblíženia mapy

V simulácii je možné príslušnými klávesami meniť priblíženie mapy (zoom). Priblíženie nie je zhora obmedzené, avšak zdola áno. Pre mapu so závodným okruhom platí, že znižovať priblíženie je možné iba pokiaľ veľkosť kratšej strany mapy nie je menšia ako veľkosť príslušnej strany plochy, v ktorej je mapa vykresľovaná. Pre *nekonečnú mapu* platí, že priblíženie je zdola ohraničené na 0,1.

Užívateľ má možnosť kedykoľvek stiskom príslušnej klávesy vrátiť priblíženie späť na 1.0, teda predvolené priblíženie.

3.3.3. Ostatné nastavenia grafického prostredia simulácie

Grafické prostredie simulácie je možné upravovať nasledujúcimi spôsobmi vždy stlačením príslušnej klávesy:

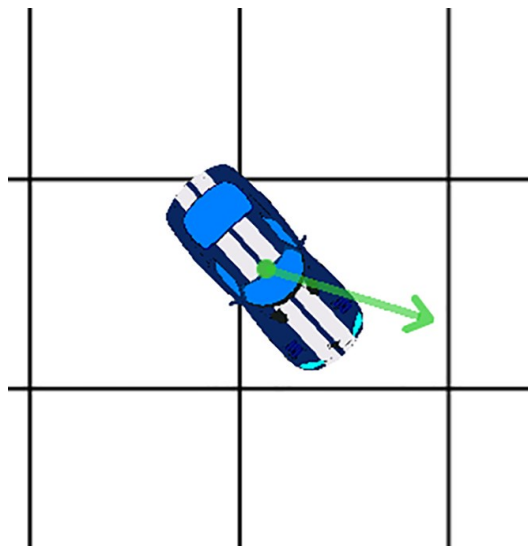
- Zapnúť/vypnúť zobrazenie závodnej stopy na mape
- Zapnúť/vypnúť zobrazenie najbližšieho bodu na závodnej stope vzhľadom k pozícii auta (viď kapitolu 5.2.1) a zobrazenie bodu, ktorý predstavuje pozíciu auta
- Zväčšiť hustotu bodov, ktoré zobrazujú závodnú stopu

- Zmenšiť hustotu bodov, ktoré zobrazujú závodnú stopu
- Zapnúť/vypnúť zobrazenie šípky, ktorá reprezentuje aktuálnu akciu umelej inteligencie (neplatí pre režim „Jazdenie“)

3.4. Funkcionalita umelého hráča v simulácii

3.4.1. Vizualizácia akcii umelého hráča

Proces učenia, ktorý v simulácii prebieha môže užívateľ v reálnom čase sledovať priamo na ploche, kde je simulácia vykresľovaná. Túto možnosť mu sprostredkováva funkcionálita, ktorá má za úlohu vizualizovať na tejto ploche aktuálnu akciu hráča. Táto vizualizácia má podobu šípky, ktorá vychádza zo stredu automobilu. Smerovanie šípky znázorňuje pozíciu volantu, veľkosť šípky znázorňuje pozíciu pedálov. Ak umelý hráč pridáva plyn, šípka je vyfarbená na zeleno a jej veľkosť sa odvíja od polohy plynového pedálu (viď obrázok 3.5.). Ak hráč brzdí, šípka je vyfarbená na červeno a jej veľkosť sa odvíja od polohy brzdového pedálu. Ak agent nepridáva, ani nebrzdí, šípka je vyfarbená sivou farbou. Ak je navyše počiatočný bod šípky zafarbený na oranžovo znamená to, že si umelý hráč vybral náhodnú akciu.



Obrázok 3.5.: znázornenie vizualizácie akcie umelého hráča

Táto vizualizácia je rovnako ako väčšina súčastí umelej inteligencie pre užívateľa modifikovateľná skrz konfiguráciu (viď kapitolu 5.3.).

3.4.2. Export tabuľky

Implementácia umelej inteligencie umožňuje užívateľovi možnosť uložiť priebeh procesu učenia v podobe exportu tabuľky do binárneho súboru na disk. Spolu s tabuľkou sa do tohto súboru ukladá aj aktuálna konfigurácia umelej inteligencie, ktorá popisuje stavy a akcie reprezentované uloženou tabuľkou.

Okrem exportu tabuľky do binárneho súboru má užívateľ možnosť uložiť tabuľku do súboru CSV, ktorý si potom môže otvoriť vhodným tabuľkovým editorom (napríklad Excel z balíku Microsoft Office), v ktorom môže tabuľku a jej hodnoty analyzovať.

3.4.3. Import tabuľky

Spolu s možnosťou exportovať tabuľku je užívateľovi aplikácie poskytnutá možnosť tabuľku z binárneho súboru naimportovať späť do simulácie. Vďaka tomu môže užívateľ proces učenia prerušiť, tabuľku exportovať a učenie spustiť neskôr pomocou importovania danej tabuľky.

Implementácia umelej inteligencie umožňuje importovať tabuľku aj z procesu učenia, ktorý bol konfigurovaný inou konfiguráciou ako je aktuálne použitá konfigurácia. Táto možnosť má však svoje obmedzenia. Hlavné obmedzenie predstavuje nutnosť použiť rovnakú množinu akcií. Okrem tohto obmedzenia ešte musí aktuálna konfigurácia splňovať podmienku, že počet výhľadových bodov musí byť rovnaký ako v konfigurácii importovanej tabuľky.

Import tabuľky s inou konfiguráciou prebieha na základe nasledujúceho princípu, pre ktorého pochopenie je nutné byť zoznámený s fungovaním implementovanej umelej inteligencie (viď kapitola 5.). Z každého stavu S importovanej tabuľky, ktorý je definovaný konfiguráciou uloženou spolu s touto tabuľkou sú vygenerované hodnoty parametrov popisujúce prostredie, ktoré k tomuto

stavu prislúchajú. Z týchto hodnôt je spätne získaný stav na základe aktuálne použitej konfigurácie. V aktuálnej tabuľke tento stav určuje riadok, do ktorého sú prekopírované hodnoty z riadku importovanej tabuľky, ktorý odpovedá stavu S .

3.4.4. Vizualizácia tabuľky

V rámci simulácie je možné stlačením príslušnej klávesy otvoriť nové okno (vid' obrázok 3.6.), v ktorom je zobrazená tabuľka s aktuálnymi Q-hodnotami (vid' kapitola 5.1.3.). Táto tabuľka je naviazaná na proces učenia, čo znamená, že sa v nej hodnoty priebežne upravujú tak, ako sa upravujú v tabuľke Q-learningu. Hodnota, ktorá bola práve zmenená je vyznačená červenou farbou. Navyše, ak je v simulácii zapnutý režim krokovania, pri každej iterácii Q-learningu sa užívateľovi zobrazí bunka, ktorej hodnota bola práve upravená, tým pádom užívateľ nemusí danú úpravu v tabuľke hľadať ručne.

	T: 0 B: 0 S: -1	T: 0 B: 0 S: 0,5	T: 0 B: 0 S: 1	T: 0 B: 1 S: 0	T: 0 B: 1 S: -0,5	T: 0 B: 1 S: -1	T: 0 B: 1 S: 0,5	T: 0 B: 1 S: 1	T: 1 B: 0 S: 0	T: 1 B: 0 S: -0,5	^
[99]S: [0,0, 14,9] D: [109,9, 150,0] AD: [-3,1, -1,9]	-5,32128	-5,203919	-5,232459	-4,630996	-5,315633	-4,74447	-4,730951	-5,333602	-5,258905	-2,778	
[100]S: [0,0, 14,9] D: [109,9, 150,0] AD: [-1,9, -0,6]	-5,36706	-4,813931	-5,546191	-5,537454	-5,527435	-5,281621	-5,516339	-5,527295	-5,52731	-5,317	
[101]S: [0,0, 14,9] D: [109,9, 150,0] AD: [-0,6, -0,4]	-4,752253	-4,716307	-4,947911	-4,898326	-4,99696	-3,352455	-4,655336	-4,883458	-4,036595	-4,944	
[102]S: [0,0, 14,9] D: [109,9, 150,0] AD: [-0,4, -0,2]	-4,5369	-4,661448	-4,633234	-4,819587	-4,192019	-4,678458	-4,837659	-4,756607	-4,864648	-4,464	
[103]S: [0,0, 14,9] D: [109,9, 150,0] AD: [-0,2, 0,0]	-4,778502	-4,963156	-4,689955	-4,943787	-4,850342	-4,646434	-4,866476	-4,715785	-4,827311	-4,700	
[104]S: [0,0, 14,9] D: [109,9, 150,0] AD: [0,0, 0,0]	0	0	0	0	0	0	0	0	0	0	
[105]S: [0,0, 14,9] D: [109,9, 150,0] AD: [0,0, 0,2]	-4,563301	-4,280653	-4,251134	-4,52393	-4,109538	-4,002724	-4,436042	-4,262735	-1,917202	-4,373	
[106]S: [0,0, 14,9] D: [109,9, 150,0] AD: [0,2, 0,4]	-4,001716	-4,470565	-4,060418	-4,327124	-4,471258	-4,253649	-4,390444	-3,397135	-4,095926	-1,213	
											^

Obrázok 3.6.: znázornenie vizualizácie tabuľky

Upozornenie: vizualizácia tabuľky nie je určená pre rozsiahle tabuľky Q-hodnôt, preto je doporučené túto funkcionálnosť využívať iba pri tabuľkách, ktorých počet buniek je menší ako 1000 buniek.

3.5. Informačné panely

Simulácia počas svojho behu poskytuje množstvo informácií o svojom aktuálnom stave. Tieto informácie popisujú hlavne fyzikálny model automobilu a vývoj jednotlivých fyzikálnych veličín, ktoré v tomto modeli pôsobia, no taktiež popisujú prostredie simulácie, stav automobilu a stav umelej inteligencie. Množina týchto informácií by mala byť vhodným spôsobom podaná užívateľovi. Preto boli vytvorené informačné panely, ktoré užívateľovi skrz obrazovku poskytujú ich sledovanie.

Tieto panely sú zobrazované na obrazovke simulácie.

3.5.1. Panel so základnými údajmi o priebehu simulácie

Tento panel je na obrazovke simulácie umiestnený v jej hornej časti a slúži predovšetkým na zobrazenie dvoch základných údajov opisujúcich priebeh a stav simulácie. Konkrétne sú to:

- Čas simulácie

V momente spustenia simulácie sa začína počítať čas jej behu. Vývoj času simulácie sa odvíja od toho, aká rýchlosť je v simulácii nastavená. Pri krokovani alebo pozastavení simulácie sa čas taktiež zastaví.

- Režim simulácie

Okrem týchto údajov panel poskytuje ďalšie dve doplňujúce údaje, ktoré popisujú rýchlosť prepočítavania simulácie a rýchlosť vykresľovania simulácie.

- UPS (Updates per second)

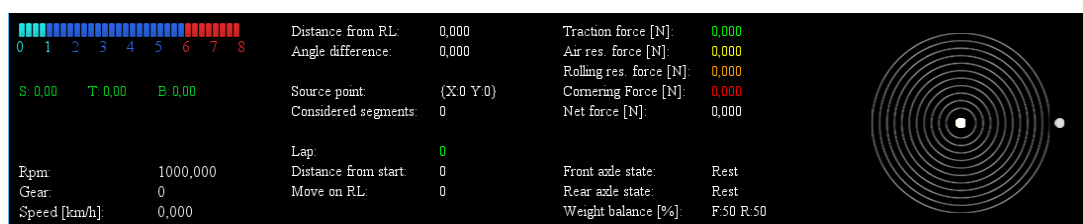
Simulácia a všetky jej komponenty a ich logika sú prepočítavané niekoľko desiatok krát za sekundu. Údaj UPS informuje užívateľa o presnom počte prepočítaní za sekundu.

- FPS (Frames per second)

Tento údaj súvisí s vykresľovaním simulácie a informačných panelov. Tieto komponenty sú na obrazovke prekresľované s istou frekvenciou. Čím je táto frekvencia väčšia, tým plynulejšie a spojitjšie pôsobí chod simulácie na užívateľa. FPS opisuje túto frekvenciu a udáva počet prekreslení za sekundu.

3.5.2. Panely s informáciami o aktuálnom stave simulácie

Panely s informáciami o aktuálnom stave simulácie užívateľ môže nájsť na spodnej časti obrazovky simulácie. Sú zoskupené do jedného celku, v ktorom sú umiestnené vedľa seba (viď obrázok 3.7.)



Obrázok 3.7: zoskupenie informačných panelov

Prednastavený počet panelov, ktoré sú aktuálne v tomto zoskupení zobrazené je 4. Tento počet sa ale môže znížiť v prípade, že sa 4 panely vedľa seba nevojdú na obrazovku kvôli jej malému rozlíšeniu.

Tieto panely zaberajú pomerne veľkú časť obrazovky, čo môže niekedy užívateľovi prekážať. Preto je tieto panely možné skryť stlačením príslušnej klávesy. Pokiaľ sa tak užívateľ rozhodne a panely sa na obrazovku prestanú vykresľovať, nahradí ich jeden malý panel, ktorý sa nachádza na spodnej časti obrazovky a poskytuje iba základne informácie o pohybe auta, ktoré sú dôležité pre užívateľa ako jazdca (viď obrázok 3.8.).



Obrázok 3.8: panel so základnými údajmi o automobile

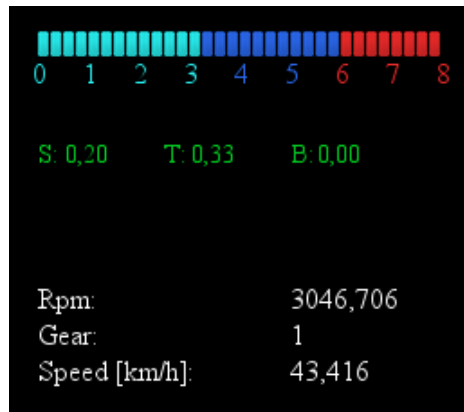
Celkový počet informačných panelov je 9, pričom užívateľ si môže v priebehu simulácie vyberať, ktoré z týchto panelov chce mať zobrazené. Uvažujme, že sú na obrazovke vyhradené 4 miesta (sloty) pre zobrazenie panelov. Do každého z týchto miest je priradený práve jeden panel z množiny dostupných panelov. K týmto štyrom miestam sú navyše priradené 4 klávesy. Stlačením týchto kláves dochádza k zmene panelu na príslušnom mieste. Ako nasledujúci je zvolený panel, ktorý nasleduje v množine dostupných panelov za aktuálne zobrazeným, pre ktorý avšak platí, že v čase zmeny panelu už nie je na obrazovke zobrazený. Ak takýto panel neexistuje, k zmene panelu nedochádza.

V nasledujúcej časti tejto kapitoly sú popísané jednotlivé panely. Informácie, ktoré sú v nich obsiahnuté sú popísané v kapitolách 5. a 6..

3.5.2.1. Panel s údajmi o automobile (Car data panel)

Tento panel obsahuje základné informácie o stave automobilu. Tieto informácie sú dôležité pre užívateľa ako jazdca (viď obrázok 3.9.). Patria k nim:

- Počet otáčok motora (Rpm)
- Počet otáčok motora vizualizovaný skrz jednoduchý otáčkomer
- Aktuálne zaradená rýchlosť (Gear)
- Aktuálna rýchlosť automobilu (Speed)
- Aktuálna pozícia volantu (S)
- Aktuálna pozícia plynového pedálu (T)
- Aktuálna pozícia brzdového pedálu (B)



Obrázok 3.9: Panel s údajmi o automobile

3.5.2.2. Panel s informáciami o pohonnom ústrojenstve automobilu (Drivetrain data panel)

Tento panel užívateľovi poskytuje informácie o pohonnom systéme automobilu (vid' obrázok 3.10). Tento systém opisuje prenos síl medzi motorom a kolesami a ich aplikovanie na povrch vozovky. Prevažne sa jedná o jednotlivé krútiace momenty a prevodové pomery. Konkrétne sú to:

- Krútiaci moment motora (Engine Torque)
- Pohonný krútiaci moment (Drive torque)
- Brzdny krútiaci moment (Brake torque)
- Trakčný krútiaci moment (Traction torque)
- Celkový krútiaci moment vyvinutý na koleso (Net torque)
- Prevodový pomer aktuálne zaradeného prevodového stupňa (Gear ratio)
- Uhlová rýchlosť rýchlosť poháňaných kolies (D.W. Ang. Vel., skratka pre Drive wheels angular velocity)
- Kvocient klzania poháňaných kolies (Slip ratio)

Engine torque [Nm]:	45,749
Drive torque [Nm]:	374,568
Brake torque [Nm]:	0,000
Traction torque [Nm]:	-3774,630
Net torque [Nm]:	-3400,061
Gear ratio [1]:	2,660
D.W. Ang Vel [rad/s]:	35,816
Slip ratio [1]:	0,000

Obrázok 3.10.: Panel s informáciami o pohonnom ústrojenstve automobilu

3.5.2.3. Panel s informáciami o silách pôsobiacich na automobil a o stave náprav (Forces data panel)

Tento panel informuje užívateľa predovšetkým o veľkostiach základných síl, ktoré sú vyvíjané na automobil (viď obrázok 3.11.). K týmto silám patria:

- Trakčná sila (Traction force)
- Sila odporu vzduchu (Air res. force, skratka pre Air resistance force)
- Sila odporu vozovky (Rolling res. force, skratka pre Rolling resistance force)
- Sila zatáčania (Cornering force)
- Celková sila pôsobiaca na automobilu (Net force)

S vývojom týchto síl súvisia ešte niektoré ďalšie faktory, ktoré sú zahrnuté vo fyzikálnom modeli. Z týchto faktorov je v tomto paneli uvedené rozloženie hmotnosti (Weight balance).

Naviac tento panel obsahuje informáciu o stave prednej a zadnej nápravy, ktorá môže mať nasledujúce hodnoty:

- Rest: v tomto stave sú kolesa nápravy v pokoji, resp. neotáčajú sa
- Throttling: opisuje stav, kedy kolesá nápravy vyvíjajú na vozovku sily, ktoré spôsobujú akceleráciu automobilu

- Braking: opisuje opačný stav ako stav *Throttling*, teda stav, kedy sily pôsobiace skrz kolesá vedú k dekcelerácii automobilu
- Rolling: predstavuje stav, kedy kolesá nápravy nevyvíjajú žiadne sily ústiace do akcelerácie/dekcelerácie auta a vykonávajú prirodzený vĺivý pohyb odpovedajúci rýchlosti automobilu

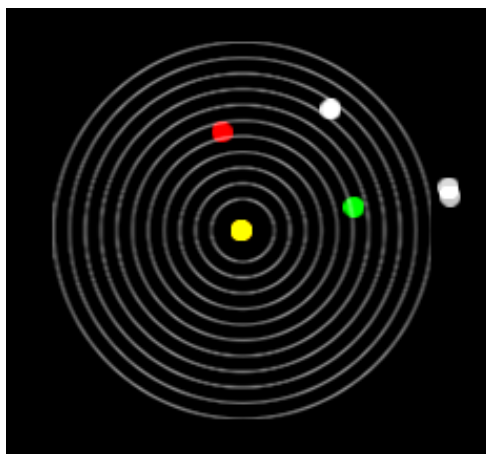
Traction force [N]:	15564,280
Air res. force [N]:	191,388
Rolling res. force [N]:	200,000
Cornering Force [N]:	9369,228
Net force [N]:	18061,620
Front axle state:	Rolling
Rear axle state:	Throttling
Weight balance [%]:	F:19 R:81

Obrázok 3.11.: Panel s informáciami o silách pôsobiacich na automobil

3.5.2.4. Panel vizualizácie síl (Forces visualization panel)

Panel s informáciami o silách pôsobiacich vo fyzikálnom modeli automobilu poskytuje informácie o veľkostiach síl, avšak tieto sily sú určené aj ich smerom, ktorý v tomto paneli zahrnutý nie je.

Za účelom poskytnutia plnej reprezentácie síl (veľkosť a smer) vznikol panel nazvaný *Panel vizualizácie síl*. Tento panel je tvorený sústrednými kružnicami, v ktorých sú vizualizované jednotlivé sily (viď obrázok 3.12.). Ich vizualizácia spočíva v znázornení sily ako vektoru, ktorého počiatok sa nachádza v strede zobrazených sústredných kružníc a ktorého koncový bod vyznačuje v týchto kružniciach smer a zároveň aj veľkosť reprezentovanej sily. Jednotlivé sily sú v tejto vizualizácii odlišené farbami.



Obrázok 3.12.: Panel vizualizácie síl

3.5.2.5. Panel s informáciami o koeficientoch prostredia (Coefficient panel)

Tento panel sprostredkuje užívateľovi hodnoty niektorých koeficientov prostredia (viď obrázok 3.13.). V tejto verzii aplikácie sú k dispozícii nasledujúce koeficienty:

- Hustota vzduchu (Air density)
- Gravitačné zrýchlenie (Gravity acc., skratka pre Gravity acceleration)
- Koeficient priľnavosti vozovky (Surface friction)

Air density:	1,290
Gravity acc.:	9,807
Surface friction:	1,000

Obrázok 3.13.: Panel s informáciami o koeficientoch prostredia

3.5.2.6. Panel s informáciami o závodnej stope (Racing line info panel)

Panel s informáciami o závodnej stope poskytuje údaje, ktoré popisujú polohu a pohyb automobilu voči závodnej stope, základné informácie o závodnej stope a polohe automobilu (viď obrázok 3.14.). Konkrétne to sú:

- Vzdialenosť automobilu od závodnej stopy (Distance from RL, skratka pre Distance from Racing line)
- Uhol medzi priamkou predstavujúcou natočenie (orientáciu) automobilu a dotyčnicou ku krivke závodnej stopy v bode, ktorý je na krivke najbližšie k automobilu (Angle difference)
- Súradnice automobilu na mape, ktoré sú využívané vo výpočtoch závodnej stopy (Source point)
- Index segmentu závodnej stopy. Pri nekonečnej mape je tento segment konštantne nulový.

Závodná stopa na mape so závodným okruhom je uzavretá krivka, na ktorej je vyznačený štart. Každý prechod cez štart začína nové kolo. Na závodnej stope na nekonečnej mape je kolo reprezentované úsečkou, ktorá má vopred definovanú dĺžku. Prejdenie tejto úsečky na závodnej stope predstavuje prejdenie jedného kola. Tieto informácie sú taktiež zahrnuté v tomto paneli. Sú to:

- Aktuálne kolo (Lap)
- Vzdialenosť medzi štartom a bodom na krivke závodnej stopy, ktorý je najbližší vzhľadom k pozícii auta (Distance from start)
- Pohyb na závodnej stope (Move on RL)

Distance from RL:	-9,298
Angle difference:	-0,294
Source point (X):	5457,614
Source point (Y):	1052,484
Considered segments:	0
Lap:	1
Distance from start:	1446,221
Move on RL:	3,181458

Obrázok 3.14.: panel s informáciami o závodnej stope

3.5.2.7. Panel s informácií o umelej inteligencii (Q-learning panel)

Tento panel obsahuje informácie, ktoré popisujú aktuálny stav a aktuálnu akciu umelej inteligencie (viď obrázok 3.15.).

Pre popis aktuálneho stavu sú v paneli uvedené:

- Index aktuálneho stavu (Current state)
- Interval rýchlosti (Speed interval)
- Interval vzdialenosti (Distance interval)
- Interval uhlového rozdielu (Angle diff. Interval, skratka pre Angle difference interval)

Pred hodnotou intervalov je v zátvorke uvedený index príslušného intervalu.

Pre popis aktuálnej akcie sú v paneli uvedené:

- Index aktuálnej akcie (Current action)
- Pozícia plynového pedálu (Throttle)
- Pozícia brzdového pedálu (Brake)
- Pozícia volantu (Steering)

Current state	15737
Speed interval:	(2) [20,067; 40,547]
Distance interval:	(8) [173,460; 219,723]
Angle diff. interval:	(12) [0,2; 0,3]
Current action	0
Throttle:	0
Brake:	0
Steering:	0,000

Obrázok 3.15.: panel s informáciami o umelej inteligencii

3.5.2.8. Panel s informáciami, ktoré analyzujú umelú inteligenciu a jej proces učenia (Q-learning analyzer panel)

Tento panel slúži užívateľovi ako zdroj informácií, ktoré popisujú umelú inteligenciu tak, ako bola nakonfigurovaná a ktoré popisujú samotný proces učenia (vid' obrázok 3.16.).

Údaje obsiahnuté v tomto paneli sú:

- Počet stavov (States count)
- Počet akcií (Actions count)
- Veľkosť tabuľky daná počtom jej buniek (Table cells count)
- Priemerná zmena akcie (Average action change)
- Aktuálna hodnota reward funkcie (Current reward)
- Počet preskúmaní (Explorations)
- Percentuálna miera objavených stavov a objavených buniek tabuľky (Discovered)
- Aktuálny rozdiel v pohybe automobilu na závodnej stope, ktorý je použitý v reward funkcií (Move diff, skratka pre Move difference)

States count:	62073 (27)
Actions count:	231
Table cells count:	14338863
Average action change:	0
Current reward:	0,76752
Explorations:	0
Discovered:	S: 0,008%; C: 0%
Move diff:	150,876

Obrázok 3.16.: Panel s informáciami, ktoré analyzujú umelú inteligenciu a jej proces učenia

3.5.2.9. Panel s informáciami o výhl'adoch umelej inteligencie (Lookouts panel)

Tento panel užívateľovi sprostredkováva informácie o výhl'adoch aktuálneho stavu (viď obrázok 3.17.). Pre každý výhl'ad je v tomto panely možné nájsť nasledujúce údaje:

- Index výhl'adu (Index)
- Interval uhlu výhl'adu (Angle Interval)

Index	0
Angle interval:	(5) [0; 0,2]
Index	1
Angle interval:	(2) [0; 1]

Obrázok 3.17.: Panel s informáciami o výhl'adoch umelej inteligencie

Informačné panely simulácie poskytujú užívateľovi ešte jednu funkciu. Keďže sa dostupné panely dajú rozdeliť do troch skupín podľa ich špecifikácie (panely s informáciami o automobile, panely s informáciami o prostredí, panely s informáciami o umelej inteligencii), v aplikácii je pre každú skupinu definovaná prednastavená štvorica panelov, ktorých zobrazenie je možné nastaviť stlačením príslušnej klávesy. Užívateľ tým pádom, ak chce vidieť informácie napríklad o umelej inteligencii, nemusí vyberať zobrazenie každého jedného panelu samostatne, postačí mu na to stisk jednej klávesy a príslušná štvorica panelov bude automaticky zobrazená.

Prednastavené panely:

- Panely s informáciami o automobile
 - Car data panel
 - Drivetrain data panel
 - Forces data panel
 - Forces visualization panel
- Panely s informáciami o prostredí
 - Car data panel
 - Racing line info panel
 - Coefficient panel
 - Forces visualization panel
- Panel s informáciami o umelej inteligencii
 - Car data panel
 - Racing line info panel
 - Q-learning panel
 - Q-learning analyzer panel

Prednastavené panely sú v simulácii dostupné iba ak boli všetky panely počas načítania simulácie načítané bez chyby a ak je počet zobrazených panelov na obrazovke rovný 4.

3.5.3. Načítavanie panelov

Panely patria k súčasťam aplikácie, ktoré sú modifikovateľné skrz konfigurácie, ktoré sú načítané z disku počas načítania simulácie. Tieto konfigurácie sú uložené pod koreňový adresár v podadresári *Configurations/Gameplay/Panel*. Tieto konfigurácie nie sú nevyhnutne potrebné pre chod aplikácie, preto ak z nejakého dôvodu neexistujú, aplikácia sa spustí normálnym spôsobom a až pri načítaní simulácie sa užívateľ dozvie o ich absencii pomocou informačnej hlášky. Panely, ktorých konfigurácie nie sú načítané v simulácii nebudú použité.

V tomto prípade je dôležité zmieniť, že plocha, v ktorej sú zoskupené informačné panely je tiež panel, ktorý má svoju konfiguráciu. V prípade, že načítanie tejto konfigurácie nebolo úspešne, informačné panely v simulácii nebudú zobrazené vôbec.

3.6. Konzola

Konzoly v aplikáciách a hrách slúžia na komunikáciu medzi užívateľom a aplikáciou. Užívateľ do konzoly zadáva príkazy, ktoré aplikácia spracováva a istým spôsobom na nich reaguje, poprípade do konzoly vypisuje potrebné výstupné informácie.

Konzola je teda vhodným nástrojom pre prijímanie zložitejších požiadaviek od užívateľa a jej prítomnosť v aplikácii umožňuje v rámci ďalšieho vývoja implementovať rôznu funkcionality, ktorá je závislá na vstupe od užívateľa v podobe textu, ktorý do konzoly píše. V tejto verzii aplikácie je konzola použitá predovšetkým na modifikovanie prostredia simulácie.

Fyzikálny model automobilu je závislý na niektorých konštantách a koeficientoch popisujúcich prostredie, na základe ktorých sú prepočítavané

fyzikálne veličiny. K nim patrí napríklad hustota vzduchu, gravitačné zrýchlenie alebo koeficient priľnavosti vozovky. Bolo by vhodné, ak by tieto konštanty a koeficienty bolo možné meniť počas simulácie a sledovať tak fyziku automobilu aj v inom prostredí. Napríklad pri pohybe auta na *nekonečnej mape* sa týmto spôsobom dajú simulovať rôzne povrchy vozovky. V prehnanom prípade je možné simulovať jazdu autom napríklad na Mesiaci, kedy do simulácie stačí zadať správne koeficienty popisujúce povrch vozovky, zmeniť gravitačné zrýchlenie na odpovedajúcu hodnotu a hustotu vzduchu nastaviť na nulu.

K účelu zmeny parametrov simulácie konzola disponuje dvomi typmi príkazov:

1. Príkaz pre zobrazenie hodnoty požadovaných parametrov

Pre zobrazenie hodnoty parametru stačí do konzoly zadať jeho alias. Špeciálny príkazom je príkaz *help*, kedy sa po zadaní tohto príkazu do konzoly vypíše zoznam dostupných parametrov spolu s ich aliasom.

2. Príkaz pre nastavenie hodnoty požadovanej vlastnosti

Pre nastavenie hodnoty parametrov stačí do konzoly napísať jeho alias nasledovaný znakom rovnosti a požadovanou hodnotou. Typ hodnoty zadanej užívateľom sa musí zhodovať s typom nastavovaného parametru. Podporované typy sú:

- Vektor, ktorý musí byť zadán v tvare (x, y)
- Desatinné číslo, ktoré musí byť v tvare 0.99, teda na oddelenie desatinného rozvoja je použitá bodka, nie čiarka
- Celé číslo
- Booleovská hodnota v tvare *true* alebo *false*

Dostupné parametre, ich aliasy a typ:

- Hustota vzduchu.
 - Alias: *AirDensity*

- Typ: desatinné číslo
- Koeficient príľnavosti vozovky.
 - Alias: *SurfaceFriction*
 - Typ: desatinné číslo
- Parameter fyzikálneho modelu, ktorý určuje, či je koeficient príľnavosti vozovky určený mapou alebo užívateľom. Ak má hodnotu *true*, koeficient určuje užívateľ, v opačnom prípade je koeficient vyčítaný z mapy pri každom prepočte.
 - Alias: *SurfaceFrictionOverride*
 - Typ: booleovská hodnota

Konzola a jej vzhľad je taktiež konfigurovateľný. To znamená, že rovnako ako panely je konzola počas načítavania simulácie načítavaná skrz konfiguráciu, ktorá je uložená na disku. Konfigurácia je uložená pod koreňový adresár v podadresári *Configurations/Gameplay/Console* v súbore *Console.xml*. Ak sa pri načítaní konfigurácie z tohto súboru vyskytne chyba, užívateľ je na to upozornený varovnou hláškou a konzola počas simulácie nie je dostupná.

3.7. Ovládanie simulácie

Všetky operácie, ktoré sú užívateľovi v simulácii prístupné cez klávesnicu majú k sebe jednoznačne priradenú klávesu a popríade jej modifikátor. Priradenie kláves k operáciám je definované v súbore *GameConfiguration.xml* (viď kapitolu 2.3.2.).

Pokiaľ sa užívateľ nerozhodne, že toto priradenie bude modifikovať ručne, je použité prednastavené priradenie (default assignment), ktoré je opísané v tabuľke 3.18..

Akcia	Kód akcie	Priradená klávesa
Pozastavenie simulácie (pauza)	SimulationChangePause	P
Zmena režimu simulácie	SimulationChangeMode	M
Resetovanie polohy automobilu	SimulationResetCar	R
Resetovanie simulácie	SimulationReset	LShift + R
Zníženie priblíženia mapy	MapZoomOut	F1
Zvýšenie priblíženia mapy	MapZoomIn	F2
Resetovanie priblíženia mapy	MapResetZoom	F3
Zníženie rýchlosti simulácie	SimulationSpeedDecrease	F5
Zvýšenie rýchlosti simulácie	SimulationSpeedIncrease	F6
Resetovanie rýchlosti simulácie	SimulationSpeedReset	F7
Zapnutie režimu krokovania / nasledujúci krok	SimulationStep	F8
Zmena režimu krokovania	SimulationChangeStepMode	LShift + F8
Pridanie plynu	DriverThrottle	Šípka hore
Brzdenie	DriverBrake	Šípka dole
Zatočenie vľavo	DriverSteerLeft	Šípka vľavo
Zatočenie vpravo	DriverSteerRight	Šípka vpravo
Nastavenie mieri zatočenia 0,1–0,9	-	Num1 – Num9
Nastavenie mieri zatočenia na 1,0	-	Num0
Zaradenie vyššieho prevodového stupňa	DriverShiftUp	D
Zaradenie nižšieho prevodového stupňa	DriverShiftDown	A
Zmena medzi manuálnom a automatickou prevodovkou	DriverChangeTransmission	S

Nahratie konfigurácie umelej inteligencie	QLearningLoadConfiguration	LCtrl + 1
Export tabuľky umelej inteligencie	QLearningExportTable	LCtrl + 2
Export tabuľky umelej inteligencie do CSV	QLearningExportTableToCSV	LCtrl + 3
Import tabuľky umelej inteligencie	QLearningImportTable	LCtrl + 4
Zapnúť/Vypnúť vykresľovanie šípky aktuálnej akcie	QLearningChangeDrawArrow	LCtrl + 5
Zobraziť/skryť vizualizáciu tabuľky	QLearningShowHideTable-Visualizer	LCtrl + 6
Zapnúť/vypnúť zobrazenie najbližšieho bodu na závodnej stope vzhľadom k automobilu	RacingLineEnableDisableProvider	LCtrl + 7
Zapnúť/Vypnúť zobrazenie závodnej stopy	RacingLineChangeVisibility	LCtrl + 8
Zvýšiť počet bodov závodnej stopy	RacingLineIncreaseDensity	LCtrl + 9
Znížiť počet bodov závodnej stopy	RacingLineDecreaseDensity	LCtrl + 0
Zobrazenie pomocného panela pre priradenie kláves	Help	LCtrl
Zobrazenie/skrytie konzoly	ConsoleOpenClose	; (Tilde)
Skrolovanie konzoly smerom hore	ConsoleScrollUp	Šípka hore
Skrolovanie konzoly smerom dole	ConsoleScrollDown	Šípka dole
Zobrazenie/skrytie panelov	SimulationPanelCollapseShow	Tab
Zmena panela na 1. pozícii	SimulationPanelChangeFirst	F9
Zmena panela na 2. pozícii	SimulationPanelChangeSecond	F10
Zmena panela na 3. pozícii	SimulationPanelChangeThird	F11
Zmena panela na 4. pozícii	SimulationPanelChangeFourth	F12

Nastavenie prednastavených		
panelov: Panely s informáciami	-	LShift + F9
o automobile		
Nastavenie prednastavených		
panelov: Panely s informáciami	-	LShift + F10
o umelej inteligencii		
Nastavenie prednastavených		
panelov: Panely s informáciami	-	LShift + F11
o prostredí		

Tabuľka 3.18.: tabuľka jednotlivých akcií, ktoré môže užívateľ v simulácii vykonať, ich kódov a k nim priradených kláves

4. Fyzikálny model automobilu

Fyzikálny model automobilu a jeho prostredia je najdôležitejšia časť simulácie. Vďaka tomuto modelu môže automobil na mape vyvíjať pohyb a simulovať tak správanie sa automobilu v reálnom svete. Fyzikálny model pracuje s množinou fyzikálnych veličín ako sú sily alebo krútiace momenty a ich hodnoty prepočítava v reálnom čase. Navyše je závislý na konštantách a koeficientoch, ktoré popisujú prostredie a ktorých hodnoty vystupujú v rovniciach prepočtu, medzi ktorými sú propagované.

Tento model reaguje na vstup od užívateľa alebo od umelého hráča v podobe hodnôt, ktoré predstavujú aktuálnu pozíciu volantu a plynového a brzdového pedálu. Ďalším vstupom fyzikálneho modelu sú parametre automobilu uvedené v jeho modeli (viď kapitolu 3.1.1.).

Ako výstup fyzikálny model automobilu poskytuje sadu fyzikálnych veličín popisujúcich pohyb automobilu a rotáciu automobilu okolo vlastnej osi.

4.1. Fyzika pohybu auta po rovnej čiare

Pre kompletne pochopenie fungovania tohto modelu je potrebné, aby bolo vysvetlené, ako vôbec automobil vyvíja pohyb vpred/vzad a ako sa krútiaci moment, ktorý generuje motor prenáša skrz kolesá na vozovku.

Za týmto účelom uvažujme, že sa automobil pohybuje po rovnej čiare.

Na začiatok je nutné dodať, že fyzikálny model pracuje so silami, ktoré majú svoju veľkosť udávanú v jednotkách Newton, ale aj svoj smer. Preto sú sily v tomto modeli reprezentované vektormi.

Najvýznamnejšou silou, ktorá pôsobí vo fyzikálnom modeli je sila trakčná, ktorá spôsobuje pohyb automobilu vpred. Poháňané kolesá, ktoré sa roztočia vplyvom motora pôsobia silou na povrch vozovky v smere opačnom vzhľadom k orientácii auta a zároveň podľa tretieho Newtonového zákona, povrch vozovky

pôsobí rovnakou silou v smere opačnom, teda v smere orientácie auta na kolesá, čím auto tlačí vpred. Táto sila, ktorá spôsobuje pohyb auta vpred sa nazýva trakčná sila. Ak by táto sila bola jediná, ktorá na auto pôsobí, auto by bolo schopné zrýchľovať do nekonečných rýchlostí. Preto je dôležité, aby bolo najprv vysvetlené, aké odporové sily na auto pôsobia.

Prvou významnou odporovou silou je sila odporu vzduchu (aerodynamic drag). Táto sila pôsobí v smere opačnom k smeru pohybu auta a je daná predpisom:

$$F_{ad} = 0,5 * c_d * A * \rho * v^2, \text{ kde}$$

- c_d značí koeficient aerodynamického trenia. Hodnota tohto koeficientu závisí na tvare auta a zisťuje sa vo veterných tuneloch
- A značí plochu, ktorú vytvára predná časť automobilu
- ρ predstavuje hustotu vzduchu, približne $1,29 \text{ kg/m}^3$
- v predstavuje rýchlosť automobilu

Druhá a v rámci tohto modelu posledná odporová sila je sila odporu vozovky (rolling resistance force). Táto sila, ktorá pôsobí proti pohybu kolies na vozovke je zapríčinená valivým pohybom pneumatík, kedy vzniká trenie medzi gumeným materiálom pneumatík a povrchom vozovky. Je daná predpisom:

$$F_{rr} = - c_{rr} * c * v, \text{ kde}$$

- c_{rr} predstavuje koeficient odporu vozovky
- c predstavuje konštantu, pomocou ktorej je veľkosť odporovej sily prispôbovaná reálnemu prostrediu
- v značí rýchlosť automobilu

Analýzou tohto predpisu je možné zistiť, že automobil pôsobením tejto sily nikdy úplne nezastaví, pretože jej veľkosť závisí na rýchlosti automobilu. Čím má automobil menšiu rýchlosť, tým menšia odporová sila na neho pôsobí a tým menej dekceleruje. To znamená, že rýchlosť konverguje k nule, avšak nikdy nebude nulová. Preto je v rámci tohto modelu táto sila pri pohybe automobilu zdola ohraničená

vhodnou konštantou, ktorá najreálnejšie simuluje spomalenie vozidla v nízkych rýchlostiach.

Ak sú známe veľkosti trakčnej sily a odporových síl, je z nich možné vypočítať výslednú silu, ktorá pôsobí na automobil a to predpisom:

$$F = F_{traction} + F_{ad} + F_{rr}$$

Je dôležité, že smery týchto síl sa líšia. Trakčná sila pôsobí na auto v smere jeho pohybu, odporové sily pôsobia v smere opačnom. Z výslednej sily, ktorá na auto pôsobí sa vypočíta akcelerácia automobilu podľa druhého Newtonového zákona:

$$a = F/M, \text{ kde}$$

- a značí akceleráciu automobilu, ktorá je taktiež reprezentovaná vektorom a jej veľkosť je udávaná v jednotkách m/s^2
- F značí celkovú silu, ktorá na auto pôsobí
- M predstavuje celkovú hmotnosť automobilu v kilogramoch

V prípade, že poznáme akceleráciu automobilu, je možné integrovaním času získať vývoj pohybu automobilu. Pohyb automobilu je reprezentovaný vektorom V , ktorého smer udáva smer pohybu automobilu a ktorého veľkosť udáva rýchlosť pohybu automobilu v jednotkách m/s . Prepočet vektoru V je daný následovným predpisom:

$$V = V + a * t, \text{ kde}$$

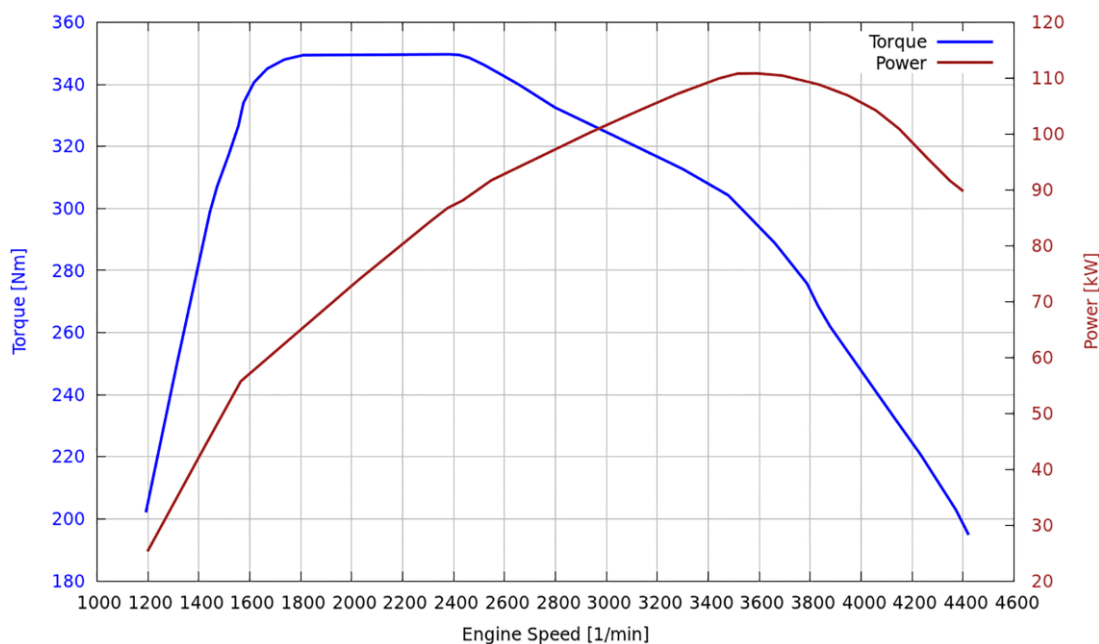
- V predstavuje vektor pohybu automobilu
- a predstavuje vektor akcelerácie automobilu
- t predstavuje čas, ktorý uplynul od posledného prepočtu fyzikálneho modelu

Rovnakým spôsobom je možné získať zmenu polohy auta na mape. Označme P vektor, ktorý predstavuje pozíciu automobilu na mape. Potom následovným predpisom je možné získať novú pozíciu automobilu po prepočte fyzikálneho modelu:

$$P = P + V * t, \text{ kde}$$

- P predstavuje pozíciu auta na mape
- V predstavuje vektor pohybu automobilu
- t predstavuje čas, ktorý uplynul od posledného prepočtu fyzikálneho modelu

K tomu, aby takýto model fungoval je potrebné vedieť vypočítať trakčnú silu, ktorá bola na začiatku tejto podkapitoly spomenutá. Pre výpočet tejto sily je ale nutné vysvetliť, ako sú sily postupne propagované od motora smerom ku kolesám. Motor, ktorý je zdrojom pohybu auta disponuje krútiacim momentom. Krútiaci moment predstavuje rotačný ekvivalent sily a jeho veľkosť sa vyjadruje v jednotkách Nm (Newtonmeter). Motor svojimi spaľovacími procesmi roztáča svoju rotačnú časť nazvanú kľukový hriadeľ, ktorej rýchlosť otáčania je vyjadrená v jednotkách rpm (Rotates per minute). Táto rýchlosť otáčania je všeobecne nazývaná *počet otáčok motora*. Veľkosť krútiaceho momentu, ktorý motor vyvíja je rôzna vzhľadom k aktuálnemu počtu otáčok motora. Túto závislosť je možné vidieť v grafe 4.1..



Graf³ 4.1: závislosť medzi krútiacim momentom (červená krivka) a otáčkami motora

Počet otáčok motora je možné získať na základe otáčania sa poháňaných kolies, ktoré sú spojené s motorom. Najprv však musíme vedieť, ako tieto kolesá roztočiť. Ak je známy počet otáčok motora, pomocou parametrov motora uvedených na vstupe fyzikálny model získa aktuálny krútiaci moment motora. Tento moment pôsobí na kľukovom hriadeli, ktorý je cez spojku, prevodovku a diferenciál spojený s poháňanými kolesami, z čoho aktívnymi prvkami, ktoré menia krútiaci moment sú prevodovka a diferenciál.

Prevodovka sa skladá z ozubených kolies, ktoré prislúchajú jednotlivým prevodovým stupňom a ktoré sú definované prevodovým pomerom (gear ratio) ako parametrom prevodového stupňa. Prevodovka navyše svojou činnosťou generuje teplo, ktoré odvádza istú časť energie, preto jej ďalším parametrom je účinnosť, ktorá sa udáva v percentách.

³ Zdroj: <https://subdiesel.files.wordpress.com/2011/05/powergraphsubaru.png>

Diferenciál podobne ako prevodovka sa tiež skladá z ozubených kolies, ktoré definuje istý prevodový pomer, avšak v tomto prípade je iba jeden a nazýva sa *stály prevod*.

Krútiaci moment, ktorý vychádza z diferenciálu je už priamo aplikovaný na kolesá a jeho veľkosť je daná predpisom:

$$T_{drive} = T_{engine} * r_g * r_d * n, \text{ kde}$$

- T_{drive} predstavuje pohonný krútiaci moment pôsobiaci na kolesá
- T_{engine} predstavuje aktuálny krútiaci moment, ktorý je generovaný motorom.
- r_g značí prevodový pomer aktuálne zaradeného rýchlostného stupňa
- r_d značí prevodový pomer stáleho prevodu v diferenciáli
- n predstavuje účinnosť prevodovky

Prevodové pomery, ktoré sú v týchto rovniciach zvyšujú/znižujú krútiaci moment, no zároveň v opačnom pomere znižujú/zvyšujú počet otáčok za minútu. Vďaka znalosti týchto pomerov je už možné vypočítať aktuálne otáčky motora z uhlovej rýchlosti otáčania poháňaných kolies nasledovne:

$$rpm = \omega * r_g * r_d * \frac{60}{2*\pi}, \text{ kde}$$

- rpm označuje počet otáčok motora
- ω predstavuje uhlovú rýchlosť kolies
- r_g značí prevodový pomer aktuálne zaradeného rýchlostného stupňa
- r_d značí prevodový pomer stáleho prevodu v diferenciáli
- $\frac{60}{2*\pi}$ predstavuje konverzný faktor medzi jednotkami rad/s a rpm

Z aktuálnych otáčok je možné detekovať, aký maximálny krútiaci moment je motor aktuálne schopný vyvinúť. Aktuálny krútiaci moment motora sa získava vynásobením aktuálneho maximálneho krútiaceho momentu motora a číslom z intervalu $[0, 1]$, ktoré udáva aktuálnu pozíciu plynového pedála.

Pohonný krútiaci moment, ktorý pôsobí na zadné kolesá je jeden z dvoch faktorov, ktoré ovplyvňujú roztočenie kolies. Druhým z faktorov je trakčný krútiaci moment, ktorý pôsobí v opačnom smere, ako pohonný krútiaci moment. Pripomeňme si, že trakčná sila je sila, ktorou povrch vozovky pôsobí na povrch kolies, čím auto tlačí vpred. Pôsobením tejto sily vzniká na kolesách trakčný krútiaci moment, ktorý je vypočítaný ako:

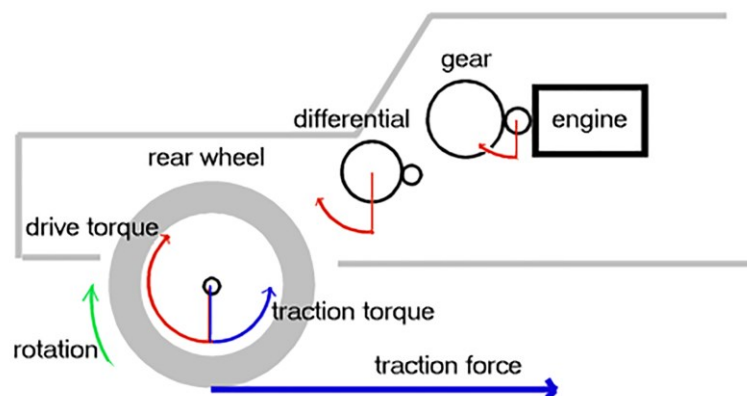
$$T_{traction} = F_{traction} * R, \text{ kde}$$

- $T_{traction}$ predstavuje trakčný krútiaci moment
- $F_{traction}$ predstavuje trakčnú silu pôsobiacu na povrchu kolesa
- R značí polomer kolesa

Výsledný krútiaci moment, ktorý na kolesá pôsobí je znázornený obrázkom 4.3. a daný predpisom:

$$T = T_{drive} - T_{traction}, \text{ kde}$$

- T predstavuje výsledný krútiaci moment
- T_{drive} predstavuje pohonný krútiaci moment pôsobiaci na kolesá
- $T_{traction}$ predstavuje trakčný krútiaci moment



Obrázok⁴ 4.3.: znázornenie jednotlivých krútiacich momentov a trakčnej sily

⁴ Zdroj:

www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html

Výsledný krútiaci moment pôsobiaci na kolesá spôsobuje roztočenie kolies spôsobom, že v nich vyvinie uhlovú akceleráciu, ktorá integráciou času vedie k uhlovej rýchlosti otáčania kolies. Uhlovú akceleráciu, podobne ako akceleráciu automobilu je možno vypočítať druhým Newtonovým zákonom, ktorý ale nepracuje so silami a hmotnosťou, ale s krútiacim momentom a zotrvačnosťou.

Vzorec pre výpočet zotrvačnosti telesa nie je univerzálny, pretože zotrvačnosť závisí na tvare daného telesa. Pre pevný valec je tento vzorec nasledovný:

$$I = M * \frac{r^2}{2}, \text{ kde}$$

- I značí zotrvačnosť telesa (Inertia)
- M značí hmotnosť telesa
- r predstavuje polomer podstavy telesa

Po získaní zotrvačnosti kolies je možno vypočítať uhlovú akceleráciu kolies a následnou integráciou času novú uhlovú rýchlosť otáčania kolies.

$$a_{angular} = T / I, \text{ kde}$$

- $a_{angular}$ predstavuje uhlovú akceleráciu kolies v jednotkách rad/s^2
- T predstavuje celkový krútiaci moment pôsobiaci na kolesá
- I predstavuje zotrvačnosť kolies

$$\omega = \omega + a_{angular} * t, \text{ kde}$$

- ω značí uhlovú rýchlosť otáčania kolies v jednotkách rad/s
- $a_{angular}$ predstavuje uhlovú akceleráciu kolies
- t predstavuje čas, ktorý uplynul od posledného prepočtu fyzikálneho modelu

V tejto fáze prepočtu fyzikálneho modelu automobilu je vhodné vysvetliť, čo zapríčiňuje pohyb automobilu a čo je to vlastne trakčná sila. K zmene pohybu automobilu dochádza, keď je uhlová rýchlosť poháňaných kolies o niečo väčšia ako

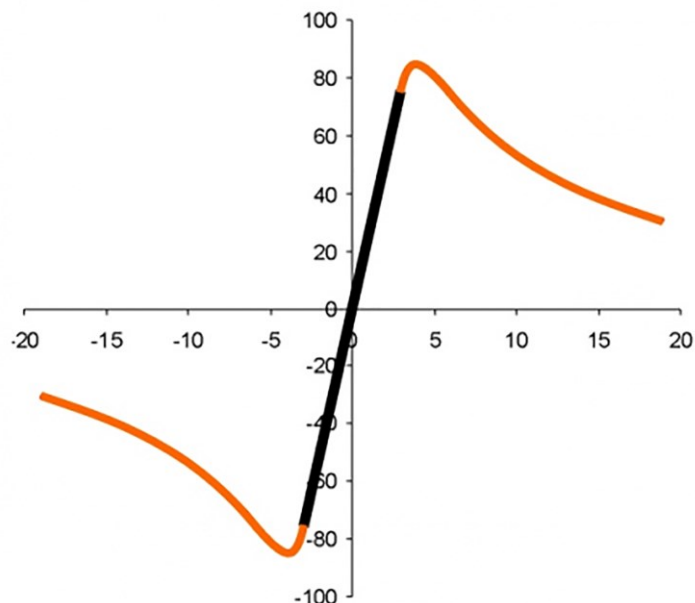
uhlová rýchlosť, ktorá odpovedá pohybu auta. Tento rozdiel v uhlovej rýchlosti zapríčiňuje, že sa povrch kolies po vozovke kĺže, čo medzi vozovkou a kolesami vytvára treciu silu, ktorej smer je zhodný s orientáciou automobilu. V skutočnosti práve táto sila ako reakcia na kĺzanie kolies tlačí auto vpred a práve táto sila je označovaná ako trakčná sila. Veľkosť trakčnej sily pritom závisí na tom, ako veľmi sa kolesá po vozovke kĺžu. K vyjadreniu intenzity kĺzania kolies po vozovke sa štandardne používa veličina zvaná kvocient kĺzania (slip ratio), ktorá je vyjadrená v percentách a daná predpisom:

$$\sigma = (\omega * R - v) / |v|, \text{ kde}$$

- σ značí kvocient kĺzania
- ω predstavuje uhlovú rýchlosť kolies
- v predstavuje rýchlosť automobilu

Kvocient kĺzania je nulový, ak uhlová rýchlosť kolies odpovedá pohybu automobilu, teda kolesa sa po vozovke nekĺžu. Ak automobil brzdí a kolesa sú zablokované (netočia sa), kvocient je rovný -1. Ak auto akceleruje v doprednom smere, kvocient je kladný a môže byť väčší ako 1, ak je intenzita kĺzania veľká.

Kvocient kĺzania neposkytuje reálne veľkosti trakčnej sily, pretože táto sila závisí hlavne na tom, aká tiažová sila na dané koleso pôsobí. Čím je koleso viac zaťažené, resp. čím je hmotnosť auta vyššia a teda aj sila, ktorá pôsobí na koleso vo vertikálnom smere nadol, tým väčšie trenie bude vznikať medzi povrchom kolesa a povrchom vozovky. Kvocient kĺzania udáva iba koeficient trakčnej sily v intervale $[-1, 1]$, ktorým sa tiažová sila vyvinutá na koleso násobí. Závislosti medzi kvocientom kĺzania a koeficientom trakčnej sily, ktorá bude v rámci tejto práce označovaná ako φ je znázornená v grafe 4.4..



Graf⁵ 4.4: závislosti medzi kvocientom klzania na ose X udanom v percentách a koeficientom trakčnej sily na ose Y udanom v percentách

Z grafu je možné vidieť, že ak je rozdiel v uhlovej rýchlosti príliš veľký a kolesa s veľkou intenzitou prešmykujú, veľkosť koeficientu trakčnej sily a teda aj trakčnej sily samotnej je malá, čo odpovedá správaniu sa auta v reálnom svete. Veľkosť trakčnej sily dosahuje svoj vrchol približne v hodnote 4%, kedy uhlová rýchlosť kolies zľahka prekračuje uhlovú rýchlosť odpovedajúcu pohybu automobilu.

Výsledná trakčná sila je teda vypočítaná predpisom:

$$F_{traction} = \varphi(\sigma) * F_g, \text{ kde}$$

- $F_{traction}$ predstavuje trakčnú silu
- φ predstavuje funkciu, ktorá na základe uvedeného grafu predstavuje závislosť medzi kvocientom klzania a koeficientom trakčnej sily
- σ značí kvocient klzania

⁵ Zdroj: <http://the-contact-patch.com/figure/C2015-Selfalign-1498743843.jpg>

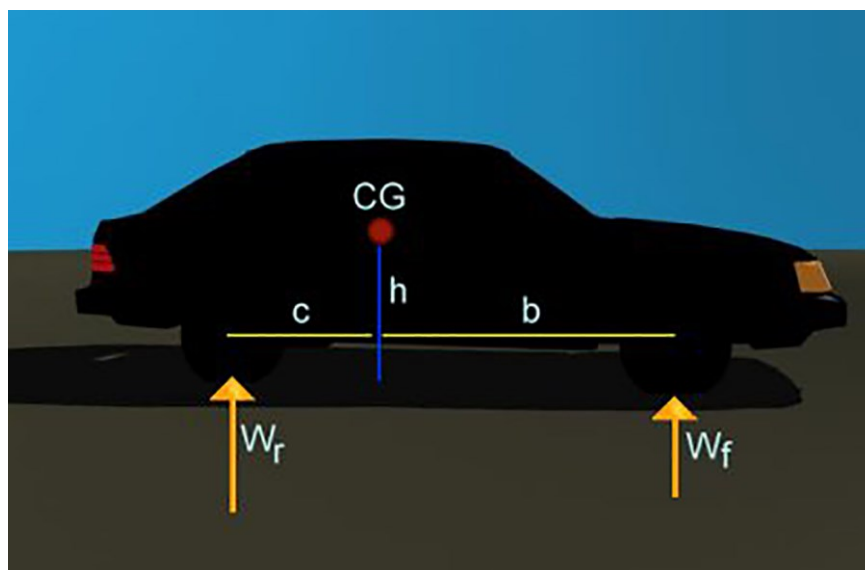
- F_g predstavuje ťiažovú silu vyvinutú na poháňané kolesá

Ku konečnému dopočtu trakčnej sily je ešte potrebné zistiť, aká veľká tiažová sila na dané kolesá pôsobí. V prípade, že je auto v pokoji je tiažová sila rozložená medzi prednú a zadnú nápravu v istom pomere. Tento pomer je definovaný vzdialenosťami medzi nápravami a ťažiskom auta (CG – center of gravity) nasledovne:

$$W_f = \frac{c}{L} * W,$$

$$W_r = \frac{b}{L} * W, \text{ kde}$$

- W_f a W_r značia zaťaženie prednej a zadnej nápravy
- W predstavuje celkovú tiažovú silu auta
- Vzdialenosti c , b a L označujú vzdialenosti podľa obrázku 4.5.



Obrázok⁶ 4.5.: znázornenie síl a vzdialeností spojených s rozložením hmotnosti

⁶ Zdroj:

www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html

Rozloženie záťaže na nápravy sa avšak zmení, ak auto začne akcelerovať alebo dekcelerovať. Pri akcelerácii sa ťažisko a rozloženie váhy auta posúva viac k zadnej náprave, pri dekcelerácii viac k prednej náprave. Toto rozloženie váhy je závislé na akcelerácii a je definované predpismi:

$$W_f = (c/L) * W - (h/L) * M * a,$$

$$W_r = (b/L) * W + (h/L) * M * a, \text{ kde}$$

- W_f a W_r značia zaťaženie prednej a zadnej nápravy
- W predstavuje celkovú ťažovú silu auta
- Vzdialenosti c , b , h a L označujú vzdialenosti podľa obrázku 4.5.
- M označuje hmotnosť automobilu
- a predstavuje akceleráciu (kladná hodnota) alebo dekceleráciu (záporná hodnota)

S rozložením záťaže taktiež súvisí ešte jedna podstatná vec, ktorá sa týka trakčnej sily. Pri akcelerácii sa záťaž na zadnú nápravu zvýši, čo znamená, že sa zvýši aj trakčná sila. Avšak každá pneumatika má svoju hranicu, ktorá určuje maximálnu trakčnú silu, ktorá na pneumatiku môže pôsobiť. Rôzne zmesi gúm ako materiál, z ktorého je vyrobená pneumatika znesú rôzne maximálne množstvo trenia. Toto množstvo je vyjadrené koeficientom trenia pneumatiky (tyre friction coefficient), ktorý patrí medzi parametre automobilu. Maximálna trakčná sila je potom vyjadrená predpisom:

$$F_{max} = c_{friction} * W, \text{ kde}$$

- F_{max} označuje maximálnu trakčnú silu
- $c_{friction}$ predstavuje koeficient trenia pneumatiky
- W predstavuje celkovú ťažovú silu automobilu

Ťažová sila automobilu je vypočítaná ako:

$$W = M * g, \text{ kde}$$

- W označuje tiažovú silu
- M označuje hmotnosť automobilu
- g označuje gravitačné zrýchlenie, ktoré ma vo fyzikálnom modeli hodnotu $9,80665 \text{ m/s}^2$.

Opis prepočtu fyzikálneho modelu sa doteraz sústredil iba na to, ako je možné auto rozháňať v doprednom smere. Dôležité ale tiež je, ako fyzikálny model simuluje brzdenie alebo pohyb vzad.

Brzdenie je v rámci tohto modelu simulované nahradením trakčnej sily silou brzdnou, ktorá pôsobí v opačnom smere. Jej veľkosť je vypočítaná na základe konštanty, ktorá opisuje, akú veľkú silu sú brzdy schopné vyvinúť pri maximálnom brzdnom zaťažení. Táto konštanta patrí medzi parametre auta. Brzdná sila, rovnako ako trakčná sila vyvíja na kolese brzdný krútiaci moment. Táto sila je vypočítaná nasledovne:

$$F_{braking} = -u * c_{braking} * p, \text{ kde}$$

- $F_{braking}$ označuje brzdnú silu
- u označuje jednotkový vektor, ktorý predstavuje orientáciu auta
- $c_{braking}$ označuje brzdnú konštantu
- p označuje pozíciu brzdového pedálu, ktorá nadobúda hodnoty v intervale $[0, 1]$

Pohyb vzad je v tomto modeli simulovaný jednoduchým zaradením spiatočky, ktorej prevodový pomer je záporný, čo zaistí otočenie znamienka pohonného krútiaceho momentu, teda bude pôsobiť v opačnom smere.

4.2. Fyzika zatáčania automobilu

Fyzikálny model automobilu tak ako bol zatiaľ opísaný umožňuje užívateľovi simulovať reálne správanie auta pri jazde po priamke. Druhá časť tohto modelu sa

zameriava na simulovanie zatáčania auta v rôznych rýchlostiach, kedy môže dochádzať k nedotáčavosti alebo na druhej strane k pretáčavosti auta.

Tak ako pri pohybe auta po rovnej krivke, aj pri zatáčaní sa vo fyzikálnom modeli objavujú sily, ktoré modifikujú pohyb auta. Okrem týchto síl sú v ňom prítomné krútiace momenty, ktoré pôsobia na celé auto ako jedno teleso a modifikujú jeho natočenie. Tieto veličiny sa vyvíjajú v reálnom čase.

Základným podnetom pre zatočenie automobilu je zmena pozície volantu, ktorá vyúsťuje k natočeniu kolies na prednej náprave. Týmto nastáva situácia, kedy sú kolesá natočené do iného smeru ako je smer ich pohybu, čo má za následok vývoj tretej sily, ktorá pôsobí na kolesá v smere kolmom na ich orientáciu a ktorá má v konečnom dôsledku dopad na zmenu pohybu automobilu, ktorý sa odkloní od rovnej krivky. Vývoj tejto sily je popísaný v nasledujúcich odsekoch.

Pre simulovanie reálneho správania sa auta na trati nie je možné predpokladať, že jeho kolesá sa hýbu v smere ich otáčania. Kolesá sú pripevnené k autu, ktoré má svoju hmotnosť a jeho reakcia na zatáčanie nie je okamžitá, naopak, postupne sa vyvíja v čase. Automobil ako teleso sa v rámci svojho pohybu môže otáčať okolo vlastnej osi istou uhlovou rýchlosťou. Táto uhlová rýchlosť sa vyvíja v čase na základe uhlovej akcelerácie na rovnakom princípe ako sa skrz akceleráciu vyvíja pohyb automobilu v prípade jazdy po rovnej krivke. Uhlová akcelerácia je závislá na krútiacom momente, ktorý na auto pôsobí a na zotrvačnosti auta, ako celku.

Taktiež sa automobil nie vždy pohybuje v rovnakom smere ako je smer jeho natočenia. Stačí si predstaviť pretekára, ktorý zatáčku prechádza driftom. V takomto prípade okrem pohybu vpred auto vyvíja aj istý pohyb do strany.

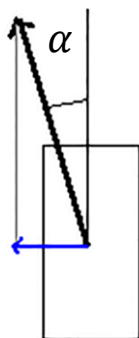
Otáčanie sa automobilu okolo vlastnej osi a pohyb automobilu do strany sú dôležité faktory, ktoré ovplyvňujú vývoj síl spojených so zatáčaním.

Základným princípom zatáčania je, že pneumatiky pri voľnom pohybe v smere ich otáčania vytvárajú len minimálne trenie, avšak, ak by sa mali pohybovať do strany, resp. do smeru kolmého na smer ich otáčania, intenzita trenia sa prudko zvýši. Pneumatiky nie sú prispôbosené na pohyb do strany a akýkoľvek takýto pohyb

znamená, že sa pneumatika po povrchu musí kĺzať, čo vytvára treciu silu, ktorá na pneumatiku v tomto kolmom smere pôsobí. Táto sila spôsobuje zatáčanie automobilu a jej veľkosť je potrebné v rámci fyzikálneho modelu vypočítať.

Pre výpočet tejto sily je potrebné zistiť veľkosť uhla, ktorý vyjadruje mieru odklonenia reálneho pohybu pneumatiky od smeru natočenia automobilu. Tento uhol sa nazýva uhol kĺzania (z anglického pojmu slip angle) a predstavuje ekvivalent kvocientu kĺzania pre zatáčanie. Čím väčší tento uhol je, tým viac sa pneumatika po vozovke kĺže v kolmom smere vzhľadom k orientácii automobilu. Po vypočítaní jeho veľkosti je možné určiť veľkosti síl, ktoré na kolesa v kolmom smere pôsobia.

Uvažujme jednu pneumatiku na prednej náprave a situáciu, že jazdec neotočil volantom. Pneumatika je natočená v smere, ktorý sa zhoduje so smerom natočenia automobilu. Ako už bolo spomenuté, smer pohybu auta nemusí byť vždy rovnaký so smerom jeho natočenia. Vektor pohybu automobilu a aj uvažovaného kolesa sa teda dá rozložiť na dve zložky – zložku súbežnú so smerom orientácie automobilu, ktorá predstavuje pohyb pneumatiky vpred (označme v_{long}) a zložku kolmú na smer orientácie automobilu, ktorá predstavuje pohyb pneumatiky do strany (označme v_{lat}). Dané rozloženie znázorňuje obrázok 4.5..



Obrázok⁷ 4.6.: znázornenie rozkladu vektoru pohybu pneumatiky

⁷ Zdroj:

www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html

Označme veľkosť vektoru pohybu kolesa ako v . Potom veľkosť súbežnej zložky je rovná $\cos \alpha * v$ a veľkosť kolmej zložky je rovná $\sin \alpha * v$. Ak auto nevykonáva otáčavý pohyb okolo vlastnej osi, úhol medzi súbežnou zložkou a vektorom pohybu (úhol označený písmenom α) predstavuje uhol kĺzania.

Komplikovanejšia situácia nastáva, ak sa auto otáča okolo vlastnej osi s uhlovou rýchlosťou ω . V takomto prípade kolesá na prednej náprave svojim otáčavým pohybom opisujú kružnicu so stredom v ťažisku auta a s polomerom b , ktorý je určený vzdialenosťou medzi ťažiskom a prednou nápravou. Táto vzdialenosť patrí medzi parametre automobilu. Uhlová rýchlosť ω *rad/s* znamená, že kolesá na tejto kružnici za sekundu opíšu kružnicový oblúk s dĺžkou ω – násobok polomeru, teda ich rýchlosť je rovná $\omega * b$ *m/s*. Dôležité je, že kolesá pri pohybe po tejto kružnici v mieste dotyku s vozovkou vykonávajú pohyb kolmý na smer ich otáčania, vďaka čomu môžeme tento pohyb jednoducho pripočítať ku kolmej zložke vektoru pohybu pneumatiky v_{lat} .

Týmto spôsobom sa podarilo pre predné pneumatiky skombinovať pohyb auta do strany a otáčanie auta okolo vlastnej osi. Vektor pohybu pneumatiky, ktorý bol rozložený do súbežnej a kolmej zložky teraz predstavuje skutočný pohyb pneumatiky vzhľadom k povrchu vozovky.

Uhol kĺzavosti pre predné kolesá môže byť vypočítaný z pravouhlého trojuholníka nasledujúcim spôsobom:

$$\alpha_{front} = \tan^{-1}((v_{lat} + \omega * b) / |v_{long}|)$$

Teraz je možno uvažovať, že sa užívateľ rozhodol zatočiť volantom. Ale keďže už je známy uhol odklonu pohybu od smeru natočenia auta, stačí od tohto uhla už iba odpočítať uhol, ktorý udáva, ako veľmi sú predné kolesá vytočené. Tým pádom kompletná rovnica pre výpočet uhla kĺzavosti pre predné kolesá je:

$$\alpha_{front} = \tan^{-1}((v_{lat} + \omega * b) / |v_{long}|) - \gamma * \text{sgn}(v_{long}), \text{ kde:}$$

- γ predstavuje uhol vytočenia predných kolies

Uhol vytočenia predných kolies je násobený znamienkom súbežnej zložky vektoru v , ktoré rozlišuje, či sa automobil pohybuje dozadu alebo dopredu.

Uhol kĺzania pre zadné kolesá je veľmi podobný až na dve zmeny.

- Kolesa sa nevytáčajú, preto v rovnici uhol ich vytočenia nie je prítomný
- Keď sa auto točí okolo vlastnej osi, zadné kolesa sa v laterálnom smere (do strany) po vozovke kĺžu v opačnom smere ako predné kolesá, preto je nutné zmeniť znamienko pri rýchlosti

Predpis na výpočet uhlu kĺzania pre zadné kolesa je teda nasledovný:

$$\alpha_{rear} = \tan^{-1} \left((v_{lat} - \omega * b) / |v_{long}| \right)$$

Po získaní veľkostí uhlov kĺzania kolies na prednej a zadnej náprave je možné vypočítať veľkosti laterálnych trecích síl, ktoré na tieto kolesá pôsobia v smere kolmom na natočenie automobilu. Tak isto, ako pri získavaní trakčnej sily, tieto trecie sily závisia hlavne na zaťažení jednotlivých kolies. Čím je koleso viac zaťažené, tým väčšie trenie vzniká v mieste kontaktu pneumatiky s vozovkou. Zaťaženie kolesa v podobe tiažovej sily je násobené koeficientom laterálnej trecej sily v intervale $[-1, 1]$, ktorý závisí iba na uhle kĺzania.

Túto závislosť je možné vidieť v grafe 4.2, ktorý síce znázorňuje závislosť kvocientu kĺzania a koeficientu trakčnej sily, no táto krivka má rovnaký tvar aj pre závislosť medzi uhlom kĺzania a koeficientom laterálnej trecej sily, pretože sa stále jedná o vývin veľkosti trecích síl medzi povrchom pneumatiky a vozovky.

Predpis pre výpočet laterálnej trecej sily je teda nasledovný:

$$F_{lateral} = \varphi(\alpha) * F_g, \text{ kde}$$

- $F_{lateral}$ predstavuje laterálnu treciu silu
- φ predstavuje funkciu, ktorá na základe uvedeného grafu predstavuje závislosť medzi uhlom kĺzania a koeficientom laterálnej trecej sily

- α značí uhol kĺzania
- F_g predstavuje tiažovú silu vyvinutú na koleso

Po získaní veľkostí laterálnych trecích síl pre kolesá prednej a zadnej nápravy nasleduje výpočet hlavnej sily, ktorá spôsobuje zatáčanie automobilu. Jej názov by sa z anglického pojmu cornering force dal preložiť ako sila zatáčania. Táto sila pôsobí na ťažisko auta v smere kolmom na jeho orientáciu a je kombináciou získaných laterálnych síl. Vypočíta sa jednoducho podľa predpisu:

$$F_{cornering} = F_{lateral,front} + F_{lateral,rear}, \text{ kde}$$

- $F_{lateral,front}$ predstavuje laterálnu treciu silu prednej nápravy
- $F_{lateral,rear}$ predstavuje laterálnu treciu silu zadnej nápravy

Sila zatáčania pôsobí na automobil ako na kompletne teleso, teda na každú jednu jeho časť. Zabezpečuje to, že sa automobil nebude pohybovať po priamke, ale napríklad po kružnici. Automobil ale potrebuje vyvinúť aj isté otáčanie sa okolo vlastnej osi, aby mohol meniť svoje natočenie. K tomu je potrebné do tohto modelu zahrnúť krútiaci moment, ktorý pôsobí na celý automobil a v čase mení uhlovú rýchlosť jeho otáčania okolo vlastnej osi.

Tento krútiaci moment sa počíta podľa nasledujúceho predpisu:

$$T = -F_{lateral,front} * b + F_{lateral,rear} * c, \text{ kde}$$

- T značí krútiaci moment
- $F_{lateral,front}$ predstavuje laterálnu treciu silu prednej nápravy
- $F_{lateral,rear}$ predstavuje laterálnu treciu silu zadnej nápravy
- b a c označujú vzdialenosti medzi ťažiskom a prednou nápravou a medzi ťažiskom a zadnou nápravou

Z krútiaceho momentu je podľa druhého Newtonového zákona vypočítaná uhlová akcelerácia:

$$a_{angular} = T/I, \text{ kde}$$

- $a_{angular}$ predstavuje uhlovú akceleráciu
- T značí krútiaci moment
- I predstavuje zotrvačnosť automobilu

Integráciou času je následne z uhlovej akcelerácie získaná nová hodnota pre uhlovú rýchlosť otáčania auta okolo vlastnej osi a následne nové natočenie automobilu.

$$\omega = \omega + a_{angular} * t, \text{ kde}$$

- ω predstavuje uhlovú rýchlosť
- $a_{angular}$ predstavuje uhlovú akceleráciu
- t predstavuje čas uplynulý od posledného prepočtu fyzikálneho modelu

$$\beta = \beta + \omega * t, \text{ kde}$$

- β predstavuje aktuálny uhol natočenia auta vzhľadom k mape
- ω predstavuje uhlovú rýchlosť
- t predstavuje čas uplynulý od posledného prepočtu fyzikálneho modelu

5. Umelý hráč

Vytvorenie umelého hráča patrí taktiež k cieľom tejto aplikácie. Umelý hráč, alebo taktiež nazývaný video-game bot (skrátene bot) predstavuje v tejto aplikácii software, ktorý ovláda auto namiesto užívateľa. S týmto autom sa snaží jazdiť po vopred definovanej závodnej stope a to čo najrýchlejšie. Umelý hráč je implementovaný za použitia umelej inteligencie zvanej Q-learning.

Umelý hráč je v rámci aplikácie zakomponovaný do simulácie, ktorú je pre jeho aktivovanie potrebné zapnúť do režimu *Learning*. Po prepnutí do tohto režimu sa spúšťa proces učenia, ktorý môže užívateľ v reálnom čase sledovať priamo na ploche, kde je simulácia vykresľovaná. K podrobnejšiemu popisu priebehu tohto procesu sú užívateľovi sprostredkované rôzne informácie s ním spojené, ktoré sú zobrazované v príslušných informačných paneloch.

5.1. Q-learning

5.1.1. Základný princíp

Q-learning patrí k jedným z najpoužívanějších typov umelej inteligencie. Táto metóda nachádza uplatnenie od robotiky až po počítačové hry, v ktorých implementuje učiacich sa agentov. Proces učenia je založený na minimalizácii chyby a má iteratívny charakter. Veľmi zjednodušene by sa dal opísať ako metóda pokus-omyl. V každej iterácii sa vykoná jedna operácia a vypočíta sa odmena, ktorá reprezentuje to, či vykonaná operácia viedla k úspechu alebo nie. Táto odmena predstavuje rozdiel medzi skutočným a žiadaným stavom prostredia. Predpokladajme, že daná operácia, ktorú agent vo svojom prostredí vykonal nevedla k úspechu. Ak sa potom počas procesu učenia ocitne agent druhýkrát v tej istej situácii, už vie, že táto operácia nie je vhodná a teda skúsi vykonať operáciu inú, ktorá môže byť lepšia. Týmto spôsobom postupom času agent po veľkom množstve iterácii zistí, čo je v danej situácii vykonať najlepšie.

5.1.2. Stav prostredia agenta a akcie agenta

Pre proces učenia je nutné, aby bolo prostredie agenta nejakým spôsobom reprezentované. Táto reprezentácia je zhrnutá v množine stavov, ktoré môže prostredie nadobúdať a ktoré sú rozlišované Q-learningom. Stav môže zahŕňať ľubovoľnú informáciu z prostredia, no jeho definícia je často kľúčom prvkom k úspešnému procesu učenia. V každej iterácii je z prostredia vyčítaný jeho aktuálny stav, na základe ktorého je určené, čo sa v danom prostredí udialo.

Okrem stavu prostredia je potrebné naviac opísať, akú operáciu môže učiť sa agent v danom prostredí vykonať. Takýto úkon je v terminológii Q-learningu nazvaný akcia.

5.1.3. Popis iterácii procesu učenia

Základnou úlohou metódy Q-learning je nájsť pre každý stav tú najlepšiu akciu. Ak sa to podarí, proces učenia môže byť ukončený, pretože množinou stavov a k nim prislúchajúcich najlepších akcií je opísané žiadané správanie agenta.

Pre nájdenie najlepšej akcie stavu je potrebné, aby pre daný stav boli vykonané všetky možné akcie. Po vykonaní akcii je obdržaná odmena, ktorá určuje, či vykonanie danej akcie v danom stave viedlo k úspechu alebo nie. Pre reprezentáciu toho, aká vhodná je daná akcia pre daný stav existuje v Q-learningu tabuľka. Riadky tejto tabuľky predstavujú stavy prostredia, stĺpce predstavujú akcie. Potom hodnota v tabuľke na riadku x a v stĺpci y hovorí, aká dobrá je akcia y pre stav x a táto hodnota sa nazýva Q-hodnota alebo anglicky Q-value.

Počas procesu učenia sa hodnoty v tejto tabuľke neustále upravujú ako sa Q-learning postupne učí. Pri základnom spôsobe učenia je pri tom postup v jednej iterácii nasledovný:

1. Z prostredia vyčítaj aktuálny stav S
2. Pre stav S nájdí najlepšiu akciu a tú vykonaj
3. Uprav Q-hodnotu najlepšej akcie podľa získanej odmeny

Najlepšia akcia pre stav S je definovaná v tabuľke. Je to akcia, ktorá má v rámci riadku predstavujúcom stav S najvyššiu Q -hodnotu. Po vykonaní akcie a získaní odmeny sa Q -hodnota prepočítava podľa nasledujúcej rovnice:

$$Q(s, a) = (1 - \alpha) * Q(s, a) + \alpha * (r + \gamma * (\max_{a'}(Q(s', a')))), \text{ kde}$$

- s predstavuje stav prostredia pred vykonaním akcie
- a predstavuje vykonanú akciu
- $Q(s, a)$ predstavuje Q -hodnotu akcie a pre stav s
- α označuje parameter „intenzity“ učenia (learning rate parameter)
- r označuje odmenu získanú vykonaním akcie a
- γ značí „zľavový“ parameter (discount parameter)
- s' predstavuje stav prostredia po vykonaní akcie
- $\max_{a'}(Q(s', a'))$ predstavuje Q -hodnotu najlepšej akcie pre stav s'

Tento predpis je parametrizovaný dvomi parametrami. Parameter intenzity učenia (learning rate) určuje, v akom rozsahu nahradia novo získané informácie tie, ktoré boli uložené v poslednom prepočte tejto Q -hodnoty. Ak je nastavený na 0, agent sa nebude schopný nič naučiť. Naopak ak je nastavený na 1 znamená to, že agent bude uvažovať vždy iba novo získané informácie a to, čo sa naučil pred tým nebude brané do úvahy.

Druhý parameter – „zľavový“ parameter (discount parameter) definuje dôležitosť budúcich odmien, ktoré budú agentovi pridelené. Ak je tento parameter nastavený na 0, agent v každej iterácii uvažuje iba tú odmenu, ktorá mu je práve pridelená, zatiaľ čo nastavenie tohto parametru na 1 zabezpečuje, že agent sa bude zameriavať na najvyššiu možnú dlhodobú odmenu.

5.1.4. Funkcia reward

Funkcia reward udáva, či aktuálne vykonaná akcia priblížila stav prostredia k požadovanému stavu alebo nie. Mala by teda vracať kladné hodnoty v prípade, ako sa nový stav viac priblížil požadovanému stavu, resp. vykonaná akcia v danom stave viedla k úspechu. V opačnom prípade to závisí na verzii Q -learningu. Väčšinou

funkcia reward vracia záporné hodnoty ak sa stav zhoršil, avšak existujú aj metódy, ktoré v takomto prípade vracajú nulu a nechávajú tak v tabuľke propagovať iba kladné hodnoty.

5.1.5. Kľúčové faktory

Kľúčovými faktormi pre správne prepočítavanie hodnôt v tabuľke sú:

- dobre definované stavy, ktoré obsahujú údaje reprezentujúce to, čo chceme, aby sa agent naučil
- dobre definovaná funkcia reward poskytujúca odmeny

Pri nevhodnej definícii stavov alebo funkcie reward môže nastať, že jednotlivé Q-hodnoty v tabuľke sa nikdy neustália alebo sa budú približovať jednej a tej istej hodnote. V opačnom prípade dochádza po istom množstve iterácii k ustáleniu hodnôt v tabuľke, kedy sa pre každý stav našla tá správna akcia. Doba, po ktorej dochádza k ustáleniu hodnôt je závislá predovšetkým na počte stavov a akcií, avšak všeobecne sa Q-learning považuje za metódu s dlho trvajúcim procesom učenia.

5.1.6. ϵ -greedy

Metóda Q-learning existuje v rôznych variantoch, ktoré sa od seba odlišujú napríklad algoritmom pre výber akcie pre nový stav. V základnom variante je za akciu pre nový stav vyberaná najlepšia akcia, teda akcia, ktorá ma najvyššiu Q-hodnotu. Avšak niekedy je vhodné nevyberať stále tú najlepšiu akciu, pretože keď sa Q-hodnoty začnú ustáľovať, skrz túto akciu sa prostredie začne dostávať vždy do toho istého nového stavu, ktorý už bol na úkor ostatných stavov prítomný vo veľkom množstve iterácii. Rovnako dochádza k tomu, že Q-hodnoty iných akcií ako najlepšej akcie zrazu prestanú byť prepočítavané, čo môže mať za následok to, že v skutočnosti najlepšiu akciu agent nezistí, pretože sa k nej už v rámci procesu učenia nedostane.

Na predídenie tohto javu sa používa ϵ -greedy implementácia, v ktorej prepočte raz za čas dôjde k tomu, že namiesto najlepšej akcie je vybraná akcia náhodná. To,

ako často k tomu dochádza je definované pomocou parametru ε , ktorý nadobúda hodnoty v intervale $[0, 1]$. V každej iterácii sa vygeneruje náhodné číslo x z toho istého intervalu a ak platí podmienka $x < \varepsilon$, namiesto najlepšej akcie je pre daný stav vybraná akcia náhodná.

5.2. Adaptovanie umelého hráča do simulácie

Pri znalosti princípov metódy Q-learning, ktorá v tejto aplikácii implementuje umelého hráča je potrebné ešte opísať jej naviazanie na prostredie simulácie a automobilu. V súvislosti s týmto naviazaním sa objavuje niekoľko otázok:

- Ako opísať množinu stavov?
- Aké akcie môže umelý hráč v rámci simulácie vykonávať?
- Ako definovať reward funkciu?

5.2.1. Definícia stavu prostredia

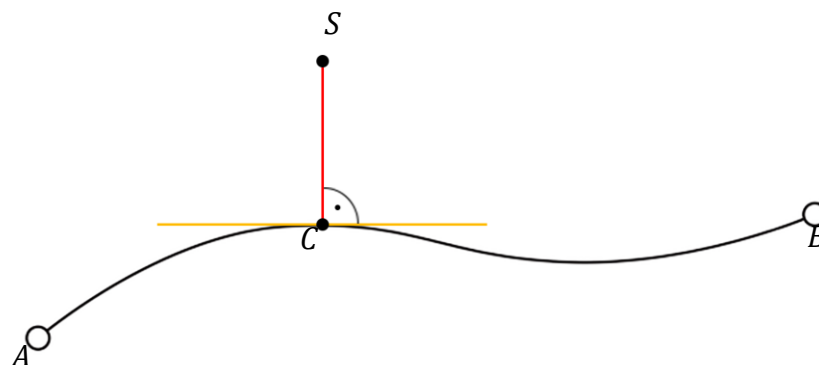
Stav je v rámci implementácie Q-learning jednoznačne určený svojim indexom, ktorý zároveň označuje práve jeden riadok v tabuľke Q-hodnôt. Pri každej iterácii procesu učenia je z prostredia získaný stav a jeho index.

Umelý hráč sa v rámci simulácie snaží naučiť jazdiť s autom po vopred definovanej závodnej stope čo najrýchlejšie. K reprezentácii stavu je teda nevyhnutné použiť parametre, ktoré kompletne popisujú stav automobilu vzhľadom k závodnej stope.

Pre určenie týchto parametrov je ale najprv vhodné definovať najbližší bod závodnej stopy vzhľadom k polohe automobilu. Je to bod, pre ktorý platí:

- Leží na krivke závodnej stopy
- Jeho vzdialenosť od stredu automobilu je najmenšia možná

Takýto bod je za normálnych okolností na krivke práve jeden a je to priesečník krivky závodnej stopy a priamky, ktorá prechádza stredom automobilu a je kolmá na dotyčnicu ku krivke závodnej stopy v tomto bode (viď obrázok 5.1.).



Obrázok 5.1.: znázornenie najbližšieho bodu závodnej stopy vzhľadom k automobilu. Krivka je určená bodmi A a B , bod S predstavuje automobil a bod C najbližší bod

Tejto definícii vyhovuje viac bodov iba v prípade, že závodná stopa opisuje kružnicový oblúk kružnice k a poloha automobilu je zhodná so stredom kružnice k . V takomto prípade je za najbližší bod považovaný ten bod, ktorý vyhovuje tejto definícii a zároveň je najbližšie k predošlému najbližšiemu bodu. Ak sa auto nehýbe, najbližší bod sa nemení. Ak je auto v pohybe, nasledujúci najbližší bod už bude určený jednoznačne, pretože poloha automobilu už nebude zhodná so stredom kružnice k .

Keď už je najbližší bod závodnej stopy vzhľadom k automobilu riadne zavedený, je možné definovať parametre, ktoré opisujú stav automobilu vzhľadom k závodnej stope. Sú to hlavne parametre, ktoré opisujú polohu automobilu vzhľadom ku krivke závodnej stopy. Konkrétne sú to:

- Vzdialenosť od závodnej stopy

Tento parameter predstavuje vzdialenosť medzi stredom automobilu a najbližším bodom závodnej stopy vzhľadom k automobilu. Táto hodnota

môže byť kladná ak sa automobil nachádza na ľavej strane od závodnej stopy alebo záporná, ak sa automobil nachádza na pravej strane závodnej stopy.

- Úhlový rozdiel so závodnou stopou

Tento parameter predstavuje uhol, ktorý zvierajú priamka prechádzajúca stredom automobilu v smere jeho natočenia a dotyčnica ku krivke závodnej stopy v najbližšom bode závodnej stopy vzhľadom k automobilu.

Tieto parametre opisujú polohu automobilu vzhľadom k závodnej stope. Avšak to pre účely učenia nestačí, pretože auto na mieste, kde sa práve nachádza môže vyvíjať istý pohyb, ktorý je podstatný preto, akú akciu je potrebné v danom stave vykonať. Preto je do reprezentácie stavu zakomponovaná taktiež rýchlosť automobilu.

Mohlo by sa zdať, že množina týchto troch parametrov je pre účely učenia dostačujúca. Opak je však pravdou. Dostačujúca je iba v prípade, že sa umelý hráč snaží naučiť jazdiť s autom po rovnej závodnej stope. Čo ale v prípade, ak závodná stopa obsahuje zatáčky? Umelý hráč potrebuje o ich existencii vedieť, aby mohol vhodne zareagovať. Za týmto účelom je do reprezentácie stavov pridaný ešte jeden parameter, a to výhľadové body.

Výhľadové body sú body, ktoré ležia na krivke závodnej stopy v istej konštantnej vzdialenosti pred najbližším bodom závodnej stopy vzhľadom k automobilu. Tým, ako sa auto pohybuje, hýbe sa aj najbližší bod na závodnej stope a to znamená, že sa hýbu aj výhľadové body. Laicky povedané, auto si tieto body tlačí pred sebou.

Výhľadové body slúžia na detekciu zákrut. Tie sú v reprezentácii stavu detekované uhlom α , ktorý zvierajú dotyčnica závodnej stopy v najbližšom bode a priamka prechádzajúca týmto bodom a výhľadovým bodom. Ak je závodná stopa rovná, uhol α je nulový. V momente, keď sa pred autom v istej vzdialenosti objaví zákruta a výhľadový bod do tejto zákruty vstúpi, uhol α nadobudne buď kladnú alebo zápornú hodnotu.

Počet výhľadových bodov a ich vzdialenosť od najbližšieho bodu na závodnej stope je daná v Q-learning konfigurácii (viď podkapitolu 5.3.).

Kombinácia týchto štyroch parametrov reprezentuje aktuálny stav prostredia. Avšak, pri získavaní konkrétneho stavu z tejto kombinácie údajov narážame na problém, pretože všetky tieto údaje majú spojitý charakter a vymedzujú nekonečný počet stavov. Z tohto dôvodu je nutné tieto parametre diskretizovať. Diskretizácia je v simulácii zabezpečená pomocou vymedzenia intervalov, do ktorých hodnoty jednotlivých parametrov spadajú.

Pre každý parameter stavu platí:

Obor hodnôt, ktoré môže parameter nadobúdať je rozdelený do intervalov, pričom počet týchto intervalov je zhora obmedzený. Obor hodnôt parametrov môže byť nekonečný (napríklad hodnota parametru *vzdialenosť od závodnej stopy* má obor hodnôt reálne čísla), preto je pre takéto parametre uvedená hranica, za ktorou všetky hodnoty automaticky spadajú do posledného intervalu a ktorá je v rámci terminológie tejto práce nazývaná *maximálna uvažovaná hodnota*. Učiaci sa agent potrebuje taktiež vedieť zdetekovať situáciu, kedy je hodnota daného parametru rovná nule, pretože takáto hodnota môže reprezentovať žiadaný stav. Z tohto dôvodu je do množiny intervalov zahrnutý jeden špeciálny interval, ktorý obsahuje práve jednu hodnotu, nulu. Každému intervalu je priradený index z množiny $\{0, 1, \dots, n - 1\}$, kde n predstavuje počet intervalov. Pri každej iterácii procesu učenia je z prostredia vyčítaná hodnota daného parametru, ktorá jednoznačne určuje práve jeden interval, do ktorého spadá. Index tohto intervalu je spolu s indexami intervalov ostatných parametrov použitý pre výpočet indexu stavu.

V rámci implementácie umelého hráča existujú tri typy stavov:

- State
- MirroredState
- DynamicMirroredState

To, ktorý typ stavov bude v simulácii použitý je uvedené v konfigurácii Q-learningu.

5.2.1.1. Typ State

Tento typ stavu predstavuje základný typ stavu prostredia. Intervaly jednotlivých parametrov majú konštantnú veľkosť definovanú užívateľom. Počet intervalov závisí na veľkosti intervalov a veľkosti množiny, ktorú dokopy vymedzujú. Pre jednotlivé parametre sú tieto množiny uvedené v tabuľke 5.1..

Názov parametru	Obor hodnôt
Vzdialenosť od závodnej stopy	$[-a, a]$, kde a predstavuje maximálnu uvažovanú hodnotu
Uhlový rozdiel so závodnou stopou	$(-\pi, \pi]$
Rýchlosť	$[0, a]$, kde a predstavuje maximálnu uvažovanú hodnotu
Uhol výhľadového bodu	$[0, a]$, kde a predstavuje maximálnu uvažovanú hodnotu

Tabuľka 5.1: Obory hodnôt pre jednotlivé parametre stavu

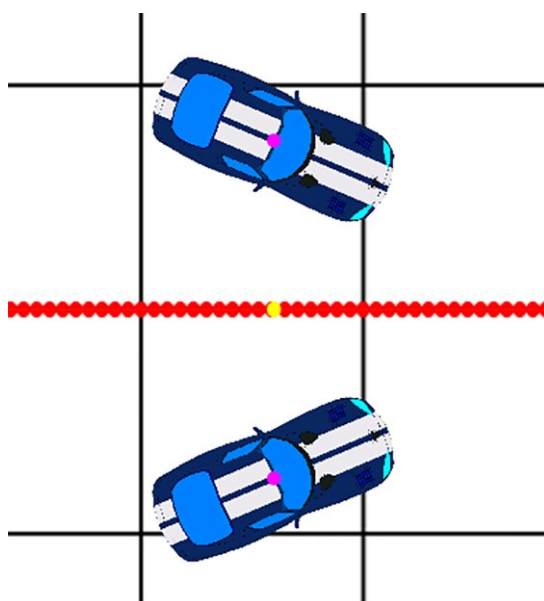
Maximálne uvažované hodnoty jednotlivých parametrov spolu s veľkosťami intervalov a definíciou výhľadov môže užívateľ modifikovať skrz Q-learning konfiguráciu, ktorá je do simulácie nahrávaná. Konfigurácia je opísaná v kapitole 5.3..

5.2.1.2. Typ MirroredState

Typ *MirroredState* bol v implementácii Q-learningu zavedený za účelom zníženia počtu stavov. Odvíja sa od typu *State*, pričom zmena sa nachádza v parametri *vzdialenosť od závodnej stopy*. Analýzou stavov typu *State* bolo zistené, že počet stavov je možné znížiť na polovicu.

Zníženie počtu stavov je možné vďaka tomu, že stav prostredia, v ktorom sa automobil nachádza na pravej strane závodnej stopy vo vzdialenosti d a je vytočený od závodnej stopy o uhol α sa dá reprezentovať pomocou stavu, v ktorom sa automobil nachádza na ľavej (opačnej) strane závodnej stopy v rovnakej vzdialenosti d a je vytočený od závodnej stopy o uhol $-\alpha$.

Tieto dva stavy predstavujú situáciu, kedy je poloha a natočenie automobilu na jednej strane závodnej stopy zrkadlovým odrazom polohy a natočenia automobilu na druhej strane závodnej stopy. Tieto dve polohy spolu s natočením automobilu sú osovo súmerne podľa osi, ktorá je daná dotyčnicou ku závodnej stopke v jej najbližšom bode vzhľadom k automobilu (viď obrázok 5.2.).



Obrázok 5.2.: znázornenie zrkadlových stavov

Vďaka tejto skutočnosti je možné uvažovať iba jednu stranu závodnej stopy a v prípade, že sa automobil nachádza na druhej strane závodnej stopy, je tento stav skrz osovú súmernosť namapovaný na uvažovanú stranu. Takéto namapovanie stavu zahŕňa zmenu znamienka hodnoty parametrov *vzdialenosť od závodnej stopy*, *uhlový rozdiel so závodnou stopou* a zmenu znamienka uhlov všetkých výhľadových bodov.

Stavy tohto typu majú parameter *vzdialenosť od závodnej stopy* na rozdiel od stavov typu *State* vždy nezáporný. Týmto postupom sa znížil počet stavov na polovicu.

5.2.1.3. Typ *DynamicMirroredState*

Tento typ stavov sa odvíja od typu *MirroredState*. To znamená, že počet stavov je oproti typu *State* znížený o polovicu.

Pri analýze procesu učenia bolo zistené, že čím viac sa hodnoty jednotlivých parametrov blížia k nule, tým presnejšie ich je potrebné v stave popisovať. Ako príklad je vhodné uviesť parameter *vzdialenosť od závodnej stopy*. Uvažujme, že užívateľ chce vzdialenosť 500 pixelov od závodnej stopy reprezentovať pomocou desiatich intervalov. To znamená, že veľkosť jedného intervalu je nastavená na 50. Vo väčších vzdialenostiach je táto veľkosť dostačujúca, avšak pri menších vzdialenostiach od závodnej stopy to predstavuje príliš veľkú odchýlku. Uvažujme, že sa automobil nachádza 30 pixelov od závodnej stopy a umelý hráč sa snaží dostať s autom späť na závodnú stopu, aby po nej mohol pokračovať ďalej. V tomto prípade je ale schopný detekovať iba 2 situácie:

- Automobil sa nachádza presne na závodnej stope
- Automobil sa nachádza vo vzdialenosti do 50 pixelov od závodnej stopy

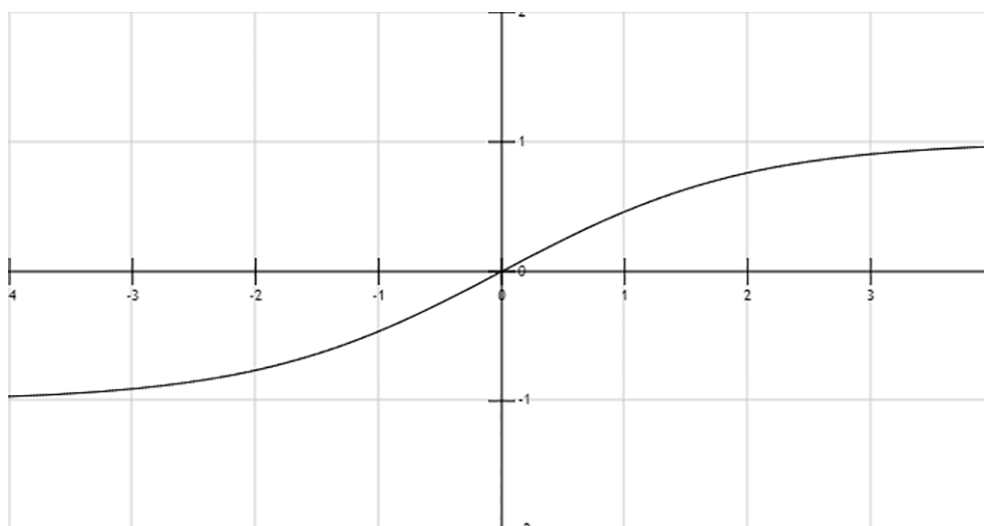
S takto nastavenými stavmi je pre agenta nemožné splniť jeho cieľ, pretože na to potrebuje sa k závodnej stope priblížiť a v správnej vzdialenosti pred závodnou stopou zabočiť tak, aby natočenie automobilu vzhľadom k závodnej stope vyrovnal a mohol po závodnej stope pokračovať. Túto vzdialenosť avšak nie je schopný detekovať. Prvé, čo bude schopný detekovať je, že sa nachádza na závodnej stope, avšak v tom momente už je neskoro zatačať, pretože automobil na zmenu otočenia potrebuje istý čas, takže závodnú stopu prejde a dostane sa na jej druhú stranu, kde nastane rovnaká situácia ako na začiatku.

Na predídenie takýchto situácií bolo zavedený tento typ stavu. Intervaly v týchto stavoch totižto nemajú konštantnú veľkosť. Ich veľkosť je vypočítavaná na základe funkcie známej ako *sigmoid*. Jej predpis je nasledovný:

$$S(x) = \frac{1}{1 + e^{-x}}$$

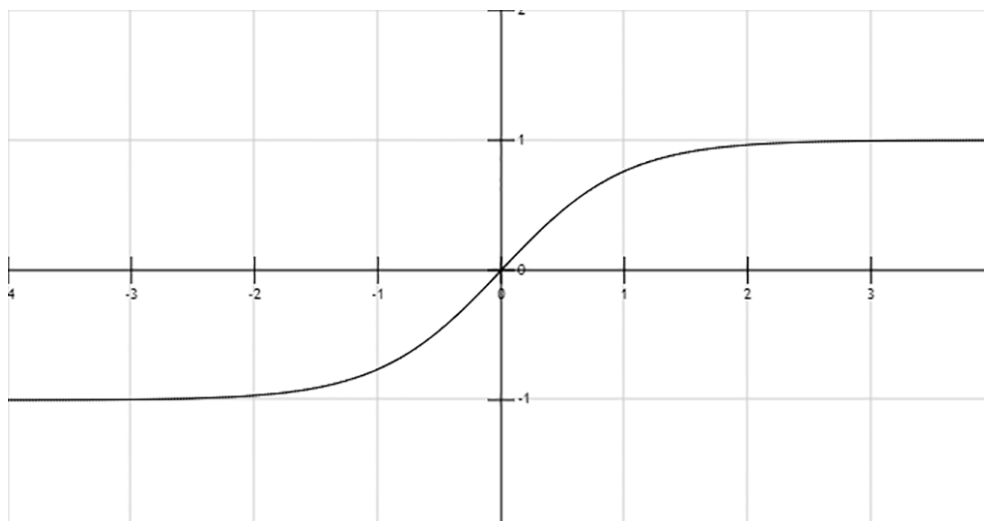
Táto funkcia je všeobecne používaná na mapovanie reálnych čísiel na interval $(0, 1)$. Pre účely tejto práce je však potrebné túto funkciu upraviť tak, aby reálne čísla rovnakým spôsobom mapovala na interval $(-1, 1)$ a miera jej stúpania bola parametrizovateľná (viď obrázok 5.3. a 5.4.). Výsledný predpis funkcie s parametrom b je teda:

$$S(x) = 1 - \frac{2}{1 + e^{b \cdot x}}$$



Graf⁸ 5.3: graf upravenej *sigmoid* funkcie s parametrom b rovným 1

⁸ Zdroj: <http://fooplot.com/>



Graf⁹ 5.4.: graf upravenej *sigmoid* funkcie s parametrom b rovným 2

Táto funkcia je použitá na získanie intervalov pre parametre *vzdialenosť od závodnej stopy* a *rýchlosť*. Užívateľ v konfigurácii namiesto veľkosti intervalu zadáva veľkosť takzvaného *segmentu sigmoidu* (označme s) a parameter sigmoidu b . Maximálna uvažovaná hodnota v konfigurácii ostáva. Jednotlivé intervaly sú potom získané nasledovným spôsobom:

- Interval $[0,1]$ je rozdelný na podintervaly o veľkosti s . Veľkosť posledného intervalu je prispôbená tak, aby neprekročil hodnotu 1.
- Pre každý podinterval v tvare $[x, y)$ definujeme interval parametru ako $[S^{-1}(x), S^{-1}(y))$. Ak hodnota $S^{-1}(y)$ prekročí maximálnu uvažovanú hodnotu, je maximálnou uvažovanou hodnotou nahradená a tento interval je posledný definovaný pre daný parameter.

Pre znázornenie výsledkov tohto postupu viď tabuľku 5.5..

Funkcia *sigmoidu* použitá v tomto postupe je parametrizovaná parametrom b , ktorý je uvedený v konfigurácii.

⁹ Zdroj: <http://fooplot.com/>

Podinterval intervalu [0,1]	Vypočítaný interval
[0, 0.1)	[0, 20.067)
[0.1, 0.2)	[20.067, 40.547)
[0.2, 0.3)	[40.547, 61.904)
[0.3, 0.4)	[61.904, 84.730)
[0.4, 0.5)	[84.730, 109.861)
[0.5, 0.6)	[109.861, 138.629)
[0.6, 0.7)	[138.629, 173.460)
[0.7, 0.8)	[173.460, 219.723)
[0.8, 0.9)	[219.723, 294.444)
[0.9, 1.0)	[294.444, 500)

Tabuľka 5.5.: znázornenie vypočítavania intervalov skrz funkciu *sigmoid*. Hodnota použitého parametru je 0.01, hodnota veľkosti *segmentu sigmoidu* je nastavená na 0,1. Maximálna uvažovaná hodnota je 500.

Stav typu *DynamicMirroredState* sa odlišuje od stavu typu *MirroredState* taktiež intervalmi parametru *uhlový rozdiel so závodnou stopou*, ktoré taktiež nemajú konštantnú veľkosť. Pre ich veľkosť už ale nemôže byť použitá funkcia Sigmoid z dôvodu veľkého nárastu veľkosti posledných intervalov. Preto veľkosti týchto intervalov definuje sám užívateľ v konfigurácii Q-learningu. V rámci tejto konfigurácie užívateľ zadáva zoznam, ktorého prvky definujú aké veľké intervaly majú byť použité na ktorej podmnožine oboru hodnôt tohto parametru. Prvok takéhoto zoznamu je definovaný pomocou elementu *AngleDifferenceIntervals*, ktorý obsahuje dva podelementy:

- *IntervalsSize*, ktorý určuje veľkosti intervalov v danej podmnožine oboru hodnôt
- *MaxAngleDifference*, ktorý podmnožinu definuje prostredníctvom hodnoty, ktorá je v ňom uvedená. Túto hodnotu označme *max*.

Podmnožina oboru hodnôt je potom daná intervalom $[min, max)$, kde hodnota min je rovná hodnote max uvedenej v predchádzajúcej podmnožine. Ak táto podmnožina uvedená nebola, hodnota min je rovná 0. Tento podelement je nepovinný a v prípade, že ho užívateľ nepoužije je hodnota max nastavená na maximum oboru hodnôt π a tým je definovanie intervalov ukončené.

Posledná odlišnosť tohto stavu od stavu typu *MirroredState* spočíva v zadávaní delta-hodnoty pre parametre *vzdialenosť od závodnej stopy*, *uhlový rozdiel so závodnou stopou* a *rýchlosť*. Počas procesu učenia sa často stáva, že automobil zide zo závodnej stopy a pri svojom návrate sa k nej priblíži, no jeho vzdialenosť od závodnej stopy sa už nevráti presne na hodnotu 0, no namiesto toho sa hodnota vzdialenosti pohybuje v blízkom okolí nuly. To spôsobuje to, že stav, ktorý je reprezentovaný touto nulou sa počas simulácie prestane prepočítavať. Delta hodnota, ktorú zadá užívateľ skrz konfiguráciu určuje blízke okolie nuly, do ktorého keď hodnota parametru spadá, agent túto hodnotu reprezentuje akoby bola nulová. Pre parameter *rýchlosť* je však doporučené nastaviť delta-hodnotu na nulu, pretože agent by mal rozoznať situáciu kedy automobil zastal úplne a kedy ešte vykonáva nejaký pohyb.

5.2.2. Definícia akcie agenta

Počas simulácie a procesu učenia by mal viesť agent vykonávať s autom také akcie, aby bol schopný uviesť auto do pohybu a viesť ho po závodnej stope. K týmto akciám patrí:

- Pridanie plynu
- Zabrzdenie
- Zatočenie volantom

Množina akcií pre agenta je oproti množine dostupných akcií pre užívateľa zjednodušená. Užívateľ môže navyše meniť prevodové stupne manuálne. V rámci procesu učenia je však agentovi priradená automatická prevodovka, preto tieto úkony

agent nepotrebuje, čo má pozitívny vplyv na proces učenia, keďže sa tým znižuje počet akcií a teda aj veľkosť tabuľky Q-learningu.

Každá akcia má pri načítaní umelého hráča priradený unikátny index, ktorý sa nemení počas celého procesu učenia a tento index jednoznačne určuje jeden stĺpec v tabuľke Q-hodnôt.

Akcia teda pozostáva z troch parametrov:

- Pozícia plynového pedálu
- Pozícia brzdového pedálu
- Pozícia volantu

Tieto parametre majú avšak spojitý charakter, takisto ako parametre stavov. Pozícia plynového pedálu a brzdového pedálu je reprezentovaná reálnym číslom z intervalu $[0, 1]$, pozícia volantu je reprezentovaná reálnym číslom z intervalu $[-1, 1]$. Tieto intervaly je potrebné diskretizovať, aby mohla byť množina akcií konečná a správne definovaná.

V rámci diskretizácie je z týchto intervalov vybraných niekoľko hodnôt, ktoré sú od seba rovnako vzdialené a ktoré udávajú dostupné hodnoty jednotlivých parametrov, ktoré môže agent pri vykonaní akcie aplikovať na automobil. Vzdialenosti medzi týmito hodnotami sú užívateľom uvedené v konfigurácii Q-learningu.

Ak sú v simulácii použité stavy typu *MirroredState* alebo *DynamicMirroredState* a pri získavaní stavu z prostredia dochádza k namapovaniu aktuálneho stavu na jeho zrkadlový odraz, znamienko hodnoty pozície volantu je v rámci tohto mapovania taktiež zmenené.

5.2.3. Funkcia reward

Základným predpokladom pre funkciu reward je poskytovanie hodnôt, ktoré odrážajú to, čo sa má agent počas procesu učenia naučiť. Správna definícia pre funkciu reward je jedným z hlavných faktorov úspechu procesu učenia.

Požiadavky na agenta sú nasledovné:

- Agent sa má naučiť jazdiť s automobilom po závodnej stope
- Agent má po tejto stope jazdiť čo najrýchlejšie

Funkcia reward by mala reflektovať tieto požiadavky vo svojom predpise. Požiadavka pre pohyb automobilu po závodnej je v implementácii Q-learningu vyjadrená pomocou vzdialenosti od závodnej stopy. V ideálnom prípade, keď sa automobil pohybuje po závodnej stope, je vzdialenosť od závodnej stopy rovná nule. Požiadavka pre čo najrýchlejší pohyb na závodnej stope je vo funkcii reward sprostredkovaná pomocou hodnoty, ktorá určuje, akú vzdialenosť automobil na závodnej stope prešiel. Táto hodnota je vypočítaná skrz najbližší bod závodnej stopy vzhľadom k automobilu. Tento bod sa pohybuje po závodnej stope a vzdialenosť, ktorú na závodnej stope prejde medzi dvomi iteráciami procesu učenia je definovaná ako *pohyb automobilu po závodnej stope*.

Pri znalosti týchto dvoch hodnôt je potrebné ich vhodným spôsobom skombinovať do výslednej hodnoty reprezentujúcej odmenu, ktorú funkcia reward poskytuje. Hodnoty popisujúce vzdialenosť od závodnej stopy a pohyb automobilu po závodnej stope majú rôznu váhu a za účelom zjednotenia ich váh sú tieto hodnoty prepočítané pomocou parametrizovanými funkciami *sigmoid*, ktoré ich namapujú na interval $[-1, 1]$. Parametre týchto funkcií zadáva užívateľ prostredníctvom konfigurácie. Keďže vzdialenosť od závodnej stopy pre účely reward funkcie nezávisí na tom, či je automobil umiestnený na pravej alebo ľavej strane závodnej stopy, vystupuje táto hodnota v reward funkcii v absolútnej hodnote.

Reward funkcia je daná nasledujúcim predpisom:

$$R = S_b(m) - S_c(\text{abs}(d)), \text{ kde:}$$

- R značí hodnotu reward
- S_p predstavuje funkciu sigmoidu parametrizovanú parametrom p
- b predstavuje parameter pre pohyb na závodnej stope
- c predstavuje parameter pre vzdialenosť od závodnej stopy

- m označuje hodnotu pohybu na závodnej stope
- d označuje hodnotu vzdialenosti od závodnej stopy

5.2.4. Ďalšie parametre umelého agenta

V implementácii umelého hráča je zakorenených ešte niekoľko prídavných parametrov, ktoré ovplyvňujú priebeh učenia agenta. Medzi tieto parametre patria:

- Spôsob vyplnenia tabuľky počas inicializácie Q-learningu

Pri načítaní a inicializácii umelej inteligencie dochádza k vytvoreniu tabuľky. Hodnoty, ktoré táto tabuľka po vytvorení obsahuje sa líšia variantami metódy Q-learning.

Užívateľ si skrz konfiguráciu môže vybrať, akým spôsobom má byť tabuľka pri jej vytvorení vyplnená. Dostupné spôsoby sú:

- Tabuľka po jej vytvorení nebude vyplnená
- Tabuľka bude po jej vytvorení vyplnená náhodnými hodnotami v intervale $[0, 1]$
- Tabuľka bude po jej vytvorení vyplnená nulami
- Tabuľka bude po jej vytvorení vyplnená hodnotou zadanou užívateľom prostredníctvom konfigurácie
- Vzdialenosť určená pre reset pozície automobilu

Tento parameter bol zavedený z dôvodu výskytu situácií, kedy sa automobil príliš vzdialil od závodnej stopy a agent na tento stav nevedel vhodne reagovať. Preto ak užívateľ nastaví tento parameter v konfigurácii, bude poloha automobilu resetovaná späť na závodnú stopu zakaždým, keď jeho vzdialenosť od závodnej stopy prekročí zadanú hodnotu.

- Minimálny čas medzi iteráciami prepočtu Q-learningu

Tento parameter zohráva významnú rolu v procese učenia. V simulácii totiž platí, že za normálnych okolností je iterácia procesu učenia vykonávaná s rovnakom frekvenciou ako prepočet fyziky automobilu. To má za následok jednoduchý jav, kedy automobil nestíha zareagovať na akciu agenta a sily vo fyzikálnom modeli sa počas jedného prepočtu nestihnú dostatočne vyvinúť a zapôsobiť na automobil. Preto je zavedený tento parameter, ktorým užívateľ, ak ho nastaví, zadáva dĺžku minimálneho časového úseku, počas ktorého nedochádza k iterácii procesu učenia. Tento časový úsek by mal poskytovať fyzikálnemu modelu dostatok priestoru na to, aby akcia, ktorú agent vykonal viedla k reálnej reakcii automobilu.

5.3. Konfigurácia umelého hráča

Umelá inteligencia patrí ku komponentom tejto aplikácie, ktoré je možné konfigurovať pomocou údajov uvedených v xml súbore. Zadanie konfigurácie je nutným prvkom pri načítaní a spustení procesu učenia, pretože obsahuje dôležité údaje, ktoré sú nevyhnutné pre reprezentáciu prostredia a akcií, ale aj pre samotný prepočet Q-hodnoty počas jednej iterácie.

Táto podkapitola má za úlohu informovať užívateľa o štruktúre tohto súboru. Jednotlivé parametre sú vysvetlené v podkapitolách 5.1. a 5.2.. Štruktúra tohto súboru je vyjadrená pomocou tabuľky 5.6., ktorá priradzuje (pre užívateľa) dôležitým xml elementov ich význam.

Názov xml elementu	Význam xml elementu
QLearningConfiguration	Predstavuje koreňový element, ktorým začína konfigurácia
ManagerConfiguration	Element s týmto názvom zoskupuje konfiguráciu stavov a akcií
StatesConfiguration	Predstavuje konfiguráciu stavov typu <i>State</i>
MirroredStatesConfiguration	Predstavuje konfiguráciu stavov typu <i>MirroredState</i>
DynamicMirroredStates-Configuration	Predstavuje konfiguráciu stavov typu <i>DynamicMirroredState</i>
SpeedIntervalSize	Veľkosť intervalov parametru <i>rýchlosť</i> pre stavy typu <i>State</i> a <i>MirroredState</i>
MaxSpeed	Maximálna uvažovaná hodnota parametru <i>rýchlosť</i>
DistanceIntervalSize	Veľkosť intervalov parametru <i>vzdialenosť od závodnej stopy</i> pre stavy typu <i>State</i> a <i>MirroredState</i>
MaxDistance	Maximálna uvažovaná hodnota parametru <i>vzdialenosť od závodnej stopy</i>
AngleDifferenceIntervalSize	Veľkosť intervalov parametru <i>uhlový rozdiel so závodnou stopou</i> pre stavy typu <i>State</i> a <i>MirroredState</i>
DistanceSegmentsSize	Veľkosť segmentu sigmoidu pre parameter <i>vzdialenosť od závodnej stopy</i>
DistanceSigmoidParameter	Parameter funkcie sigmoidu pre parameter <i>vzdialenosť od závodnej stopy</i>
DistanceDelta	Delta hodnota pre parameter <i>vzdialenosť od závodnej stopy</i>
SpeedSegmentsSize	Veľkosť segmentu sigmoidu pre parameter <i>rýchlosť</i>

SpeedSigmoidParameter	Parameter funkcie sigmoidu pre parameter <i>rýchlosť</i>
SpeedDelta	Delta hodnota pre parameter <i>rýchlosť</i>
AngleDifferenceDelta	Delta hodnota pre parameter <i>uhlový rozdiel so závodnou stopou</i>
StaticLookOutsConfiguration	Predstavuje konfiguráciu výhľadových bodov
LookOut	Predstavuje konfiguráciu jedného výhľadového bodu
AngleIntervalsSize	Veľkosť intervalov uhla výhľadového bodu
MaxAngle	Maximálna uvažovaná hodnota uhla výhľadového bodu
Distance	Vzdialenosť medzi výhľadovým bodom a najbližším bodom závodnej stopy vzhľadom k automobilu
ActionsConfiguration	Konfigurácia akcií agenta
MirroredActionsConfiguration	Konfigurácia akcií agenta, ktoré sa odvíjajú od stavov typu <i>MirroredState</i> a <i>DynamicMirroredState</i>
ThrottleIntervalSize	Vzdialenosť medzi vybranými hodnotami parametru akcie <i>pozícia plynového pedálu</i>
BrakeIntervalSize	Vzdialenosť medzi vybranými hodnotami parametru akcie <i>pozícia brzdového pedálu</i>
SteerIntervalSize	Vzdialenosť medzi vybranými hodnotami parametru akcie <i>pozícia volantu</i>
QLearningParameters	Predstavuje element, ktorý zoskupuje parametre súvisiace s prepočtom Q-hodnôt
LearningRate	Parameter intenzity učenia (learning rate)
DiscountValue	Zľavový parameter (discount value)
ExplorationParameter	Parameter udávajúci hodnotu ϵ pri ϵ -greedy variante Q-learningu

RewardProviderConfiguration	Element, ktorý zlučuje parametre reward funkcie
MoveSigmoidParameter	Parameter funkcie sigmoid pre <i>pohyb na závodnej stope</i>
DistanceSigmoidParameter	Parameter funkcie sigmoid pre <i>vzdialenosť na závodnej stope</i>
TableConfiguration	Element, ktorý zoskupuje konfiguráciu tabuľky Q-hodnôt
FillMode	Spôsob vyplnenia tabuľky po jej vytvorení, prípustné hodnoty: <i>None/ Random/ Zero/ Value</i>
Value	Hodnota, ktorou má byť vyplnená tabuľka, ak je zvolený spôsob <i>Value</i>
GraphicsConfiguration	Element zoskupujúci konfiguráciu vizualizácie Q-learning v simulácii
ActionArrowPath	Cesta ku konfigurácii šípky, ktorá vizualizuje aktuálnu akciu agenta
ResetDistance	Vzdialenosť určená pre reset pozície automobilu
UpdateIntervalMilliseconds	Minimálny čas medzi iteráciami prepočtu Q-learningu

Tabuľka 5.6.: zoznam elementov v konfigurácii Q-learningu a ich význam

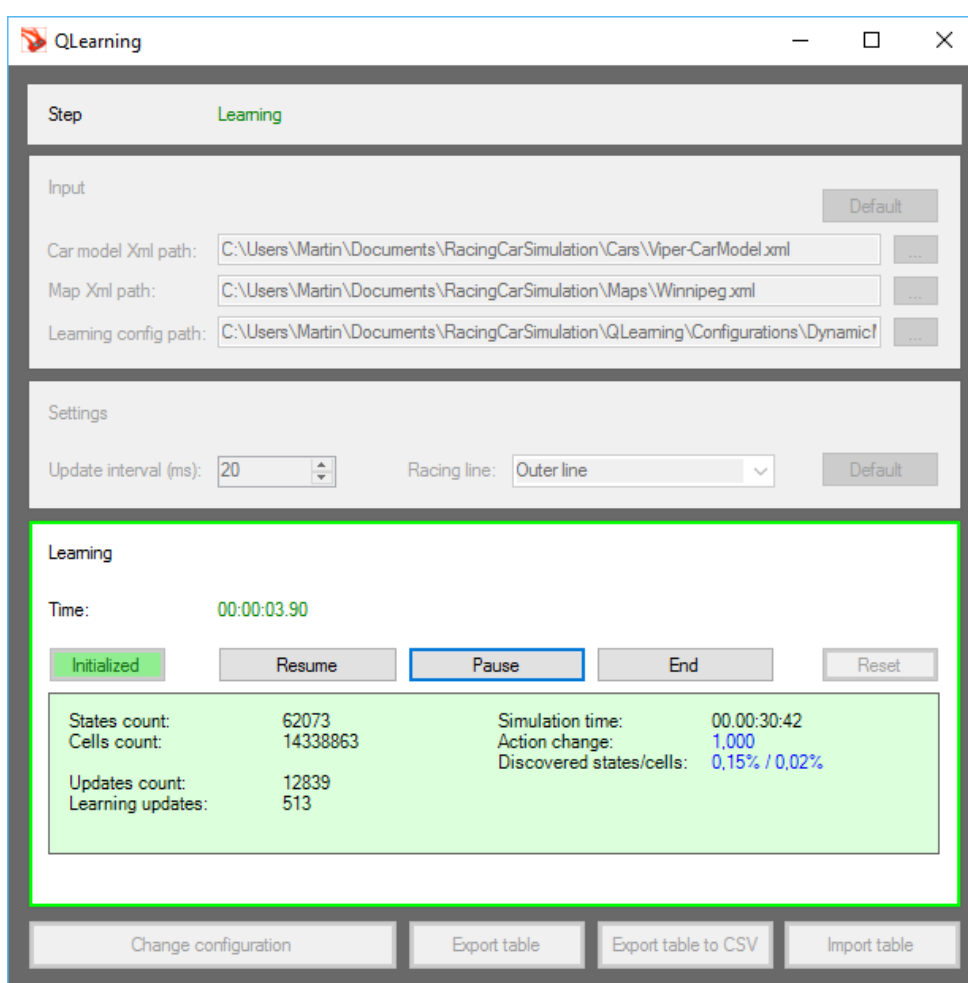
5.4. Aplikácia Q-Learning

Proces učenia metódy Q-learning všeobecne trvá dlhú dobu. Naviac v aplikáciách, ktoré implementujú túto metódu prebehne za sekundu niekoľko tisíc iterácií. V simulácii, pokiaľ je nastavená minimálna doba medzi iteráciami prepočtu Q-learningu, prebehne za sekundu maximálne niekoľko desiatok iterácií. To znamená, že čas, za ktorý dôjde k ustáleniu Q-hodnôt v tabuľke počas simulácie sa môže blížiť týždňom, možno aj mesiacom (záleží od veľkosti tabuľky).

Z tohto dôvodu bola v rámci tejto práce vytvorená aplikácia nazvaná *Q-Learning*. Jedná sa o okennú aplikáciu, ktorá využíva tú istú implementáciu umelej

inteligencie, simulácie a fyzikálneho modelu automobilu ako aplikácia *Racing Car Simulation*, avšak iterácie tejto implementácie sa nedejú za prítomnosti grafického rozhrania, ale na pozadí v jednej slučke, pomocou ktorej je kladený dôraz na čo najväčšiu rýchlosť procesu učenia, kedy sa počet iterácii za sekundu pohybuje v rádoch tisícoch. Užívateľ tak nemá možnosť sledovať, čo agent aktuálne vykonáva, avšak výsledky sa namiesto mesiacov môžu dostaviť za pár hodín.

Užívateľské rozhranie aplikácie Q-Learning je možno vidieť na obrázku 5.7..



Obrázok 5.7.: ukážka aplikácie *QLearning*

Ovládanie aplikácie je jednoduché. Užívateľ musí najprv zadať vstupné parametre simulácie tak, ako v aplikácii *Racing Car Simulation*. Vstupné parametre sú reprezentované xml súbormi. Týmito parametrami sú:

- Model automobilu (súbor car-model.xml)
- Mapa
- Konfigurácia Q-learningu

Po zadaní týchto parametrov a ich potvrdení stlačením tlačidla *Confirm input* prebehne kontrola ich validity. Ak súbory reprezentujúce vstupné parametre simulácie sú poškodené, alebo k nim užívateľ nemá prístup, alebo dáta uložené v nich majú zlý formát, užívateľ je informovaný o chybe a musí nevalidné parametre zadať znova.

Ak kontrola prebehne bez chyby, aplikácia sa posunie do ďalšieho kroku, v ktorom si užívateľ vyberá nastavenia simulácie. Medzi tieto nastavenia patrí:

- Závodná stopa
- Čas, ktorý je propagovaný do prepočtov simulácie

Závodné stopy sú definované načítanou mapou, užívateľovi stačí si len vybrať zo zoznamu dostupných závodných stôp. Čas, ktorý je propagovaný do prepočtov simulácie predstavuje čas, ktorý v simulácii prebehol od posledného prepočtu. Po potvrdení nastavení stlačením tlačidla *Confirm settings* sa aplikácia posúva do ďalšieho kroku, v ktorom už bude prebiehať samotné učenie.

Na začiatku musí užívateľ simuláciu inicializovať stlačením tlačidla *Initialize*. Počas inicializácie sa načítava Q-learning, vytvára sa tabuľka, simulácia sa prepína do režimu *Learning* a pripravuje sa na spustenie. Po inicializácii simulácie má užívateľ možnosť do simulácie naimportovať tabuľku uloženú na disku.

Stlačením tlačidla *Begin* sa spúšťa proces učenia. Počas učenia je možné tento proces kedykoľvek pozastaviť stlačením tlačidla *Pause*. V tomto stave je možné vyexportovať tabuľku ako medzivýsledok stlačením tlačidla *Export table* alebo *Export table to CSV*, znova spustiť proces učenia stlačením tlačidla *Resume* alebo tento proces úplne ukončiť stlačením tlačidla *End*.

Po ukončení procesu učenia aplikácia poskytuje užívateľovi možnosť tabuľku exportovať alebo proces učenia vyresetovať pomocou tlačidla *Reset*, kedy sa simulácia dostane do stavu tesne po inicializácii.

Ak proces učenia nebeží alebo nie je pozastavený, užívateľ môže kedykoľvek zmeniť zadané vstupné parametre stlačením tlačidla *Change configuration* a celý proces opakovať znovu.

Aplikácia počas procesu učenia poskytuje užívateľovi niektoré informácie týkajúce sa umelej inteligencie. Tieto informácie sú:

- States count: počet stavov Q-learningu
- Cells count: počet buniek v tabuľke Q-hodnôt
- Updates count: počet prepočtov simulácie, ktoré prebehli za poslednú sekundu
- Learning updates: počet prepočtov umelej inteligencie, ktoré prebehli za poslednú sekundu
- Simulation time: nasimulovaný čas v simulácii
- Action change

Qlearning si udržuje informáciu o tom, ktorá akcia bola pre daný stav vybratá naposledy. Ak sa novo vybratá akcia s touto akciou nezhoduje, navýši sa interné počítadlo zmien stavov. *Action change* vyjadruje percentuálny počet týchto zmien ako podiel počtu iterácií, kedy došlo k zmene akcie a počtu všetkých iterácií. Tieto počty sú každú sekundu vynulované, preto sa hodnota *Action change* viaže ku konkrétnej sekunde

- Discovered states/cells: vyjadruje podiel medzi počtom stavov/buniek tabuľky, ktoré boli v priebehu učenia objavené a počtom všetkých stavov/buniek tabuľky

5.5. Analýza dosiahnutých výsledkov

Ako už bolo v predošlých kapitolách spomenuté, kľúčom k úspechu metódy Q-learning je vhodné definovanie stavov a funkcie reward. Tieto dve faktory boli aj v skutočnosti tým, čo si v rámci učenia vyžadovalo najviac času na analýzu procesu učenia, rozboru výsledku a ďalšieho výskumu.

Týmito postupmi bolo zistené, že najväčším problémom sa stal spojený charakter prostredia, ktorý naviac nezlepšoval ani pokročilý fyzikálny model automobilu. Diskretizácia prostredia popísaná v kapitole 5.2.1. síce je konfigurovateľná pomocou rôzneho určovania veľkosti intervalov, avšak častokrát sa stávalo, že ani to nebolo dostačujúce na to, aby si učiaci sa agent mohol vyvinúť správne a žiaduce chovanie. Riešením by sa mohlo stať zjemnenie prostredia na väčšie množstvo stavov za účelom toho, aby umelý hráč bol schopný detekovať viac rozličných situácií, pre ktoré ak by sa naučil, čo je správna akcia, tieto vedomosti by mohli dokopy ústiť k požadovanému správaniu. Avšak zväčšenie množiny stavov, o ktorom je reč predstavuje zmenšenie intervalov jednotlivých parametrov prostredia minimálne na polovicu. Parametre sú 4, to znamená zväčšenie tabuľky na 16násobok jej pôvodnej veľkosti, čo by malo tragické následky na dĺžku trvania procesu učenia. Ideálne by bolo nájsť kompromis medzi veľkosťou stavovej množiny a časom, ktorý sme ochotní investovať na proces učenia. Avšak tento kompromis nezaručí dokonalý výsledok.

Rozborom sa zistilo, že umelý hráč je schopný v istých situáciách zistiť, ako sa priblížiť k závodnej stope, no to je tá ľahšia časť. Komplikovanejšie je zariadiť to, aby sa vedel na tejto stope aj udržať. V drvivej väčšine prípadov sa stalo to, že sa umelý hráč k závodnej stope približoval, za čo samozrejme vzhľadom k ostatným akciám dostával najlepšie odmeny, avšak túto závodnú stopu stále prešiel a dostal sa na jej druhú stranu, na ktorej začal rovnaký postup aplikovať znova. Ustáliť sa na závodnej stope sa mu nepodarilo ani raz, pretože mu k tomu chýbala dostatočná jemnosť reprezentácie prostredia.

Ďalším problémom, s ktorým sa umelý agent stretol bol už vyššie spomenutý fyzikálny model automobilu. Tento model poskytuje reálne správanie automobilu v prostredí, čo avšak nie stále pomáha procesu učenia, hlavne keď agent vykoná akciu, ale reakcia zo strany automobilu sa dostaví až o nejaký moment neskôr, pretože sa všetky fyzikálne veličiny zapojené do tohto modelu prepočítavajú v reálnom čase. Ďalšie faktory, ktoré spomaľovali a v istej miere aj znemožňovali proces učenia sú spojené s otáčaním automobilu okolo vlastnej osi.

Automobil si počas svojho pohybu vyvíja isté uhlové rýchlosti otáčania sa okolo vlastnej osi, ktoré neboli zahrnuté do reprezentácie stavu prostredia. Vďaka tomu sa stávalo, že sa agent ocitol dva krát v tom istom stave, avšak prvýkrát sa auto otáčalo okolo vlastnej osi a druhýkrát nie. On to ale nevedel zdetekovať a tak tieto stavy bral ako rovnaké a reagoval na nich rovnakým spôsobom, čo v prvej situácii viedlo k inému výsledku ako v druhej situácii. Tieto javy zabraňujú ustáleniu Q-hodnôt v tabuľke, pretože odmena, na základe ktorej sa prepočítavajú môže byť ako dôsledok týchto dvoch rôznych situácií odlišná vo veľkej miere.

S procesom učenia bolo spojených ešte niekoľko komplikácií. Keďže je Q-learning určený prevažne pre diskrétny priestor, po vykonaní akcie vždy dochádza ku nejakej zmene. Väčšinou sa po vykonaní akcie zmení stav alebo sa stav nezmení, ale táto skutočnosť definuje túto akciu. V spojitom prostredí sa stávalo, že pokiaľ sa zmenil stav (napríklad pokiaľ sa automobil rozbehol alebo zatočil) prešlo niekoľko iterácií procesu, počas ktorých ostal stav rovnaký nezávisle na vybranej akcii a teda nebolo možné agentovi presne určiť, či práve vykonáva dobrú alebo zlú akciu.

Ako dôsledok týchto komplikácií bolo častokrát pozorované to, že Q-hodnoty v tabuľkách nemali ani po väčšom množstve iterácií požadované hodnoty a že ich ustálenie je veľmi nejednoznačné. V rámci vývoja tejto práce bolo veľa krát skúšané nastaviť parametre a poopraviť implementáciu procesu učenia tak, aby bolo toto chovanie eliminované v čo najväčšej miere. Implementácia bola napríklad poopravená tak, aby k iterácii prepočtu dochádzalo až vtedy, keď dôjde v prostredí k zmene stavu a nie medzi tým. Avšak ani to neviedlo k celkom úspešnému výsledku. Preto je k uzatvoreniu umelého hráča potrebný ešte ďalší výskum.

6. Vývojárska dokumentácia

V tejto kapitole je vo všeobecnosti popísané, aká je štruktúra a architektúra tejto práce z pohľadu jej vývoja, aké techniky a praktiky v nej boli použité, no taktiež v akom vývojovom prostredí je implementovaná, aké aplikačné rámce (framework) a knižnice využíva a aké sú jej požiadavky a podporované platformy. Vývojárovi poskytuje základný prehľad o jednotlivých súčiastiach kódu, či už sa jedná o väčšie štrukturálne celky v podobne menných priestorov (namespace) alebo o ich podčasti predstavujúce rôzne triedy, ktoré implementujú jednotlivé objekty v aplikácii použité.

6.1. Aplikačné rámce a vývojové prostredie

Jednými z prvých otázok, ktorými sa musí programátor pred vývojom práce zaoberať sú otázky spojené s cieľovou platformou aplikácie, s výberom toho správneho programovacieho jazyka a aplikačného rámca (framework) a s výberom tých správnych technológií, ktoré budú v tejto implementácii použité.

Pre implementáciu tejto aplikácie bol jednoznačne zvolený framework **.NET**, ktorý disponuje veľkým množstvom knižníc poskytujúcich rôznu podpornú funkcionality a vlastným behovým prostredím. S týmto použitím je spojená jedna veľká výhoda, keďže tento framework je obsiahnutý v každom operačnom systéme **Windows**, ktorý zároveň predstavuje cieľovú platformu tejto aplikácie. Vývoj tejto práce začal vo frameworku **.NET** vo verzii 4.5, ktorá sa počas vývoja nezmenila.

Použitie tohto aplikačného rámca jednoznačne určuje výber vývojového prostredia, ktorým sa stal software **Microsoft Visual Studio 2015**.

K vývoju aplikácie *Racing Car Simulation* bolo potrebné použiť ešte niekoľko ďalších nástrojov a aplikačných rámcov. Charakter aplikácie totižto vyžadoval, aby implementácia umožňovala vytvoriť grafické užívateľské prostredie, ktoré sa musí prekresľovať minimálne niekoľko desiatok krát za sekundu, aby bolo užívateľovi možné podať spojitý obraz. Taktiež bol požadovaný vhodný mechanizmus, ktorý by

zabezpečoval všetku logiku aplikácie a implementáciu jej prepočtov. V rámci týchto požiadaviek bol zvolený framework **Microsoft XNA**, ktorý predstavuje voľne dostupnú sadu nástrojov a manažovaného behového prostredia určeného predovšetkým pre vývoj hier, ktorých charakter sa zlučuje s charakterom tejto práce. Microsoft XNA je založený na frameworku .NET a vyskytuje sa vo verziách, ktoré bežia na platformách *Windows*, *Windows Phone* a *Xbox 360*.

Pri vývoji bol taktiež s veľkou intenzitou braný ohľad na budúce rozširovanie aplikácie. Jednou z otázok, ktoré boli kladené bola otázka, či by nebolo vhodné pripraviť implementáciu na budúce rozšírenie aj na iné platformy ako je platforma *Windows* a platformy s ňou spojené. Táto otázka bola v rámci cieľov práce zodpovedaná kladne, čo malo za následok hľadanie vhodného spôsobu, ako aplikácie vyvinuté vo frameworku Microsoft XNA naportovať aj na iné platformy. Toto hľadanie vyústilo do rozhodnutia o použití frameworku **Monogame**¹⁰. Monogame predstavuje open-source implementáciu frameworku Microsoft XNA 4, ktorej cieľom je umožnenie vývojárom na platformách *Xbox 360*, *Windows* a *Windows Phone* naportovať ich hry aj na iné platformy, ktoré hranie hier podporujú. Medzi tieto platformy patria:

- Platformy desktopových počítačov
 - Windows Store Apps (8.1 and 10)
 - Windows Win32 (OpenGL & DirectX)
 - Linux (OpenGL)
 - Mac OS X (OpenGL)
- Platformy mobilných zariadení
 - Android (OpenGL)
 - iPhone/iPad (OpenGL)
 - Windows Phone (8.1 and 10)
- Platformy konzol (dostupné iba pre registrovaných vývojárov)
 - PlayStation 4

¹⁰ www.monogame.net

- PlayStation Vita
- Xbox One (both UWP and XDK)
- Nintendo Switch
- Iné platformy
 - tvOS

Implementácia je teda pripravená na rozšírenie na ďalšie platformy, avšak za účelom tejto práce bola odladená verzia iba pre jej cieľovú platformu Windows.

Pri zoznámení sa so všetkými použitými aplikačnými rámcami je ešte nevyhnutne vybrať programovacie jazyky, ktoré budú pre implementáciu použité. Tieto jazyky musia byť podporované všetkými použitými aplikačnými rámcami. K účelu implementácie aplikácie a všetkých jej komponentov bolo rozhodnuté, že v rámci celej aplikácie bude použitý jednotný jazyk a ako najvhodnejší programovací jazyk spĺňujúci všetky požiadavky bol vybraný jazyk **C#** v jeho aktuálne najvyššej verzii 6.0, pomocou ktorého je možné písať úhl'adný a manažovaný kód.

6.2. Systémové požiadavky

Minimálne systémové požiadavky sú určené predovšetkým frameworkom Monogame. Tieto požiadavky sa rôznia podľa platformy, na ktorej aplikácia beží. Pre platformu Windows existuje iba jedna požiadavka.

- DirectX vo verzii minimálne 9.0c, ktorý je podporovaný grafickou kartou na danom zariadení

6.3. Základný princíp fungovania aplikácie

V rámci vývojárskej dokumentácie je veľmi žiadúce vysvetliť, ako je vlastne zabezpečené fungovanie tejto aplikácie. Tieto poznatky sú pre vývojára potrebné, pretože bez nich len ťažko zistí, ako jednotlivé časti implementácie fungujú.

Ako už bolo spomenuté v kapitole 6.1., táto aplikácia je implementovaná vo frameworku Monogame, ktorý je vybudovaný nad frameworkom Microsoft XNA 4.

Framework Monogame umožňuje rozšíriteľnosť aplikácie na iné platformy, no pre fungovanie aplikácie je dôležitý práve framework Microsoft XNA. Tento framework bol vybratý na základe požiadaviek, ktoré umožňujú v aplikácii:

- vybudovať logiku, ktorá je neustále prepočítavaná
- vytvoriť GUI (grafické užívateľské prostredie), ktoré je neustále prekresľované

Tieto požiadavky musia byť nejakým spôsobom zahrnuté v implementácii aplikácie. K tomuto účelu slúži trieda `Microsoft.Xna.Framework.Game`, ktorá predstavuje samotnú aplikáciu a ktorá poskytuje nasledujúcich 6 dôležitých virtuálnych metód:

- `Run` – zavolaním tejto triedy dôjde k spusteniu aplikácie
- `Initialize` – metóda, ktorá slúži na inicializovanie aplikácie
- `Update` – každé volanie tejto metódy predstavuje jeden prepočet logiky
- `Draw` – každé volanie tejto metódy slúži na jedno prekreslenie GUI
- `LoadContent` – metóda, ktorá slúži na nahratie všetkého aplikačného obsahu
- `UnloadContent` – metóda, ktorá slúži na uvoľnenie aplikačného obsahu z pamäte

Princíp fungovania je nasledovný. Na inštancií triedy `Microsoft.Xna.Framework.Game` je zavolaná metóda `Run`, čím dochádza k spusteniu aplikácie. Po tomto spustení najprv dochádza k volaniu metódy `Initialize` a metódy `LoadContent`. Po vykonaní týchto dvoch metód sa v rámci behového prostredia frameworku Microsoft XNA spúšťa špeciálny cyklus, ktorý zabezpečuje volanie metód `Update` a `Draw`, čo zabezpečuje chod hry.

Pre vytvorenie vlastnej aplikácie vo frameworku Microsoft XNA je teda potrebné vytvoriť triedu, ktorá je potomkom triedy `Microsoft.Xna.Framework.Game` a ktorá konkretizuje (override) jej virtuálne metódy vlastnými metódami, v ktorých začína implementácia behu aplikácie.

6.4. Architektúra

Ako bolo spomenuté v podkapitole 6.1., ako vývojové prostredie pre implementáciu práce bol použitý software Microsoft Visual Studio 2015. Od tohto vývojového prostredia sa odvíja aj štruktúra práce, ktorá pozostáva z nasledujúcich troch projektov:

1. RacingCarSimulation
2. RacingCarSimulationTests
3. QLearningApplication

Architektúra týchto projektov je popísaná v nasledujúcich podkapitolách. Architektúra sa popisuje od jej najväčších celkov, ktorými sú menné priestory.

6.4.1. RacingCarSimulation

Projekt RacingCarSimulation je najväčšou a najdôležitejšou časťou práce, predstavuje totiž aplikáciu *Racing Car Simulation*. Jeho architektúra je zložená z viacerých menných priestorov, ktoré budú v tejto kapitole postupne opísané v poradí podľa ich dôležitosti. Tento opis má vývojárovi poskytnúť všeobecný prehľad o tom, kde v kóde hľadať implementáciu jednotlivých funkcionalít a ktoré triedy ich predstavujú. Vzhľadom na veľký rozsah tejto architektúry je pre každý menný priestor popísaná množina tried alebo menných podpriestorov, ktoré v rámci implementácie aplikácie zohrávajú významnú úlohu.

1. RacingCarSimulation.Main

Menný priestor RacingCarSimulation.Main je síce z pohľadu veľkosti jeho obsahu najmenší, no zároveň pre vývojára najdôležitejší, pretože obsahuje vstupný bod (entry point) aplikácie. Obsahuje jednu statickú triedu **Program**, ktorá disponuje jednou statickou metódou, a to metódou **Main**.

V rámci tejto metódy sa vytvorí inštancia triedy *RacingCarSimulation* (vid' menný priestor *RacingCarSimulation.Game*), ktorá predstavuje celú aplikáciu. Na tejto inštancii je raz zavolaná metóda *Run*, ktorou sa spustí beh aplikácie.

2. **RacingCarSimulation.Game**

Tento menný priestor obsahuje triedy, ktoré predstavujú samotnú aplikáciu a jej konfiguráciu.

Najvýznamnejšou triedou je trieda **RacingCarSimulation**, ktorá je potomkom triedy *Microsoft.Xna.Framework.Game*, čo znamená, že skrz túto triedu sú do hry sprostredkované základné metódy poskytované frameworkom Microsoft XNA (vid' podkapitolu 6.3), ktorými je zabezpečený beh aplikácie. V rámci tejto triedy dochádza predovšetkým k inicializácii aplikácie, k nahratiu grafického obsahu nutného pre zobrazenie prvej obrazovky a k volaniu metód *Update* a *Draw* behovým prostredím, ktoré je propagované skrz celú architektúru k objektom, ktoré sa práve prepočítavajú alebo vykresľujú.

Ďalšou významnou triedou v tomto mennom priestore je trieda **GameManager**, ktorá sa v implementácii vyskytuje ako statický singleton. To znamená, že počas behu aplikácie existuje iba jedna inštancia tejto triedy, ktorá je statická a verejná, teda prístupná z každého objektu. Slúži pre poskytovanie informácii a parametrov týkajúcich sa celej aplikácie a na nastavovanie niektorých týchto parametrov. Je zložená z komponent a v rámci tohto menného priestoru k týmto komponentom patria:

- *ScreenComponent*: predstavuje nastavenia okna aplikácie
- *DirectoryManager*: slúži pre prácu s adresármi (vid' kapitolu 2.3.1.)
- *KeyAssignmentComponent*: slúži pre poskytovanie informácii o priradení kláves k jednotlivým operáciám, ktoré môže užívateľ v aplikácii vykonávať (jednotlivé triedy spojené s touto komponentou sú zoskupené v mennom podpriestore *RacingCarSimulation.Game.KeyAssignments*)

Jednou z ďalších tried, ktoré sa v tomto mennom priestore nachádzajú je **GameConfiguration**, ktorá predstavuje model konfigurácie aplikácie tak, ako je popísaná v podkapitole 2.3.2..

3. **RacingCarSimulation.Screens**

Užívateľské grafické rozhranie, ktoré je v aplikácii použité sa skladá z rôznych obrazoviek (viď kapitolu 2.1.). Z vývojárskeho hľadiska je okrem implementácie samotných obrazoviek taktiež potrebné zaručiť, aby sa tieto obrazovky mohli v rôznych situáciách správne meniť. Menný priestor **RacingCarSimulation.Screens** zahŕňa triedy, ktoré k týmto účelom slúžia.

Najdôležitejšou triedou v rámci tohto menného priestoru je trieda **ScreenManager**. Táto trieda je v aplikácii prítomná ako statický singleton podobne ako v prípade triedy *GameManager*. To zaručuje, že inštancia tejto triedy je prístupná z ktoréhokoľvek miesta v kóde tohto projektu. Táto trieda poskytuje metódy spojené so zmenou obrazoviek. Jej dôležitou položkou je položka *CurrentScreen*, ktorá predstavuje aktuálne zobrazenú obrazovku. Vďaka tejto položke sa je možné dostať ku všetkému verejnému obsahu, ktorý sa na danej obrazovke nachádza.

Každá obrazovka je v kóde reprezentovaná jednou triedou, pričom tieto triedy sú: *SplashScreen*, *MenuScreen*, *SimulationConfigurationScreen*, *LoadingScreen* a *SimulationScreen*. Tieto triedy zlučuje spoločný predok - trieda **GameScreen**.

Zaujímavou triedou v tomto mennom priestore je trieda **ScreensCache**, ktorej inštancia je prístupná skrz inštanciu triedy *ScreenManager*. Táto trieda slúži na prednahrávanie objektov po zobrazení úvodnej obrazovky.

4. **RacingCarSimulation.Graphics**

Menný priestor *Graphics* vznikol za účelom oddelenia implementácie tried, ktoré sa podieľajú na tvorbe vizuálneho obsahu aplikácie. Tento obsah sa počas behu aplikácie môže vyskytovať v dvoch podobách. Buď je na obrazovku vykresľovaná textúra alebo text. K vykresleniu tohto obsahu je okrem samotného obsahu potrebné

do metódy Draw zadať aj množstvo parametrov ako je pozícia, otočenie, škálovanie, počiatok (origin), farba, priesvitnosť a tak ďalej. Hodnoty týchto argumentov musia byť niekde uložené a z tohto dôvodu vznikla v implementácii najdôležitejšia trieda tohto menného priestoru, **Image**. Táto trieda neslúži len na uchovanie hodnoty daných argumentov ale vývojárovi poskytuje nad vykresľovaním grafického obsahu abstrakciu, ktorá sa stala základom vytvorenia vlastného grafického API aplikácie. Vývojár ma v rámci tejto abstrakcie k dispozícii množstvo vlastností a metód, ktoré mu vo veľkej miere uľahčujú prácu s grafickým obsahom.

Trieda **Image** je abstraktná. Od nej sú derivované nasledujúce triedy:

- **Picture**: slúži na vykresľovanie textúr
- **TextImageSlim**: slúži na vykresľovanie textu
- **ColorImage**: slúži na vykresľovanie jednofarebnej plochy (vhodné napríklad pre pozadia obrazoviek)

Do tohto menného priestoru patria ešte ďalšie triedy, ktoré sú taktiež spojené s grafickým API aplikácie a triedou *Image*. Sú to triedy, ktoré poskytujú triede *Image* dve doplnujúce funkcionality:

- Podpora efektov

K týmto triedam patrí predovšetkým trieda **ImageEffectManager**, ktorá slúži pre manažment efektov v rámci obrázku a potom triedy, ktoré predstavujú jednotlivé efekty. Tieto triedy majú spoločného abstraktného predka – triedu **ImageEffect**. Všetky triedy spojené s podporou efektov sú separované v mennom podpriestore `RacingCarSimulation.Graphics.ImageEffects`

- Podpora pre prácu s kurzorom myši

Túto funkcionality dopĺňa iba jedna trieda nazvaná **ImageEventManager** a jej úlohou je poskytovať vývojárovi pre daný obrázok sadu udalostí, ktoré sú naviazané na vstup myši. K tejto triede

je vyhradený menný podpriestor

RacingCarSimulation.Graphics.ImageEvents

5. **RacingCarSimulation.Gameplay**

Časť implementácie vymedzená týmto menným priestorom sa viaže k obrazovke simulácie a všetkým jej komponentom okrem umelého hráča, ktorého implementácia je oddelená v samostatnom mennom priestore RacingCarSimulation.QLearning.

Významnými prvkami, s ktorými sa užívateľ v obrazovke simulácie stretáva sú automobil, mapa a závodná stopa. Implementácia automobilu je komplexnejšia, keďže zahŕňa automobil ako ho je možné vidieť na mape, model reprezentujúci jeho jednotlivé parametre a fyzikálny model. Tieto tri komponenty automobilu sú implementované v triedach **Car**, **CarModel** a **CarPhysics**. S automobilom je spojená ešte jedna podstatná trieda nazvaná **ElectronicControlUnit**, ktorá predstavuje hlavnú elektronickú riadiacu jednotku automobilu, v rámci ktorej je implementovaný napríklad ovládač automatickej prevodovky alebo obmedzovač otáčok motora. Týmto triedam sú priradené menné podpriestory RacingCarSimulation.GamePlay.Car a RacingCarSimulation.GamePlay.CarPhysics.

S automobilom je v súvislosti so simuláciou spojený taktiež jazdec, ktorý je v rámci tejto aplikácie implementovaný v samostatnej triede **Driver**. Počas simulácie môžu existovať dve inštancie tejto triedy, a to jazdec, ktorého predstavuje samotný užívateľ (potomok **PlayerDriver**) alebo jazdec, ktorý je spojený s umelým hráčom. Tieto triedy majú vyhradený vlastný menný podpriestor RacingCarSimulation.Gameplay.Driver.

Reprezentáciu mapy v implementácii aplikácie predstavuje trieda **Map**, ktorá je abstraktná a má dvoch potomkov: **TrackMap** predstavujúca mapu so závodným okruhom a **SimulationMap** predstavujúca nekonečnú mapu (viď kapitolu 3.1.2.), ktoré je možné nájsť v mennom podpriestore RacingCarSimulation.GamePlay.Map.

Podobne je riešená implementácia závodnej stopy, ktorá obsahuje abstraktné triedy **RacingLine** a **RacingLineDataProvider**. Trieda *RacingLine* implementuje

závodnú stopu a jej reprezentáciu a vizualizáciu v simulácii.

RacingLineDataProvider implementuje mechanizmy, ktoré počítajú všetky dáta spojené so závodnou stopou a ktoré sú ako parametre poskytované umelému hráčovi. Každá z týchto dvoch tried má dvoch potomkov, ktorí sa viažu k jednému typu mapy. Títo potomkovia sú *TrackRacingLine*, *TrackRacingLineDataProvider*, *SimulationRacingLine* a *SimulationRacingLineDataProvider* a sú zoskupené do menného podpriestoru *RacingCarSimulation.GamePlay.RacingLine*.

Najpodstatnejšou časťou tohto menného priestoru je simulácia a jej grafické prevedenie. Implementácia týchto dvoch prvkov bola zámerne oddelená, aby bolo možné simuláciu spúšťať aj bez jej grafického prostredia. Po zoznámení sa zo základnou architektúrou jednotlivých prvkov simulácie je možné opísať jej reprezentáciu v kóde. Implementácia simulácie obsahuje nevizuálne prvky a je zakotvená v triede **Simulation** v mennom podpriestore *RacingCarSimulation.Gameplay.Simulation*. Jej grafické prevedenie je reprezentované triedou **Board**, ktorá obsahuje všetky vizuálne zložky simulácie a je taktiež ako simulácia umiestnená do samostatného menného podpriestoru *RacingCarSimulation.Gameplay.Board*.

V tomto momente sa avšak môže niekto pozastaviť nad otázkou, ako sú rozdelené prvky simulácie na vizuálne a nevizuálne. Napríklad mapa je určite vizuálny prvok avšak musí byť prítomná aj v triede simulácie. Riešenie je nasledovné. Mapa ako vizuálny prvok je umiestnená v triede *Board* avšak dáta, ktoré pre simuláciu poskytuje sú zoskupené rozhraním *IMapDataProvider*, ktoré implementuje trieda *Map*. Inštancia tohto typu je v triede *Simulation* uložená ako dátová položka. Rovnako to platí pre závodnú stopu, ktorej trieda implementuje rozhranie *IRacingLineDataProvider*. Týmto konceptom boli nakoniec opísané všetky prvky simulácie, čo navyše zahŕňa auto a jeho fyzikálny model. Týmto do implementácie pribudli rozhrania *ICarDataProvider* a *ICarPhysicsDataProvider*. Ich zavedenie malo jednoduchý dôvod, a to potrebu v implementácii reprezentovať parametre prostredia, ktoré slúžia pre umelého hráča na získavanie aktuálnych stavov. Zoskupením všetkých týchto rozhraní vznikla trieda **EnvironmentDataProvider**, ktorá implementuje rozhranie

IEnvironmentDataProvider a ktorá predstavuje množinu všetkých parametrov prostredia simulácie. Táto trieda spolu so spomenutými rozhraniami je oddelená v samostatnom mennom podpriestore *RacingCarSimulation.Gameplay.DataProvider*.

Na obrazovke simulácie ešte navyše vystupujú komponenty informačného charakteru v podobe panelov. Každý panel, ktorý je v obrazovke simulácie použitý má v implementácii vyhradenú jednu triedu. Tieto triedy sú zoskupené pomocou abstraktnej triedy **Panel**, ktorá je ich spoločným predkom. Tieto triedy sú spolu obsiahnuté v mennom podpriestore *RacingCarSimulation.Gameplay.Panel*.

Posledným komponentom obrazovky simulácie je konzola. Konzola ako vizuálny prvok je reprezentovaná triedou **Console**. Navyše má konzola svoj ovládač, ktorý slúži na spracovávanie príkazov užívateľa. Tento ovládač je v implementácii zakomponovaný do triedy **ConsoleController**. Triedy spojené s konzolou sú zoskupené v mennom podpriestore *RacingCarSimulation.Gameplay.Console*.

6. *RacingCarSimulation.QLearning*

Ako názov napovedá, tento menný priestor zahŕňa kompletnú implementáciu umelého hráča. Skladá sa z množstva tried, ktoré túto implementáciu rozdeľujú na jednotlivé komponenty, ktoré na seba hierarchicky nadväzujú.

Základnou triedou predstavujúcou celú umelú inteligenciu je trieda **QLearning**, ktorá je parciálna a je rozdelená do súborov *QLearning.cs* a *QLearning.Controller.cs* za účelom oddeliť implementáciu samotného procesu učenia (obsiahnutú v súbore *QLearning.cs*) od vlastností a metód, ktoré by mala trieda reprezentujúca Q-learning v kóde poskytovať.

Jej pomocnou triedou je trieda nazvaná **QLearningManager**, ktorej implementácia pre proces učenia sprostredkováva získanie indexu aktuálneho stavu a vykonanie akcie. Táto trieda je zložená z dvoch komponent. Stavový komponent, ktorý v implementácii reprezentuje trieda **StateComponent** slúži na vyčítanie aktuálneho stavu z prostredia a poskytnutie jeho indexu triede *QLearningManager*. Komponent akcií je reprezentovaný triedou **ActionComponent** a slúži na vykonanie aktuálnej akcie danej triedou *QLearningManager*.

Jednotlivé typy stavov prostredia umelého hráča (viď kapitolu 5.2.1.) sú v implementácii zakomponované do tried **State**, **MirroredState** a **DynamicMirroredState**. Vzhľadom k nadväznosti týchto stavov je trieda *DynamicMirroredState* potomkom triedy *MirroredState*, ktorá je potomkom triedy *State*. Vývojár ich môže nájsť v mennom podpriestore `RacingCarSimulation.QLearning.States`.

So stavmi sú taktiež spojené výhľadové body (viď kapitolu 5.2.1), ktorých implementácia je od implementácie stavov oddelená do triedy **LookOuts**, ktorá interne obsahuje zoznam všetkých výhľadových bodov pre daný stav. Jednotlivé výhľadové body sú implementované v triede *LookOut*. Tieto dve triedy sú abstraktné a derivujú ich triedy predstavujúce dva rôzne typy výhľadových bodov. Avšak po analyzovaní procesu učenia bol nakoniec vyvíjaný iba jeden typ, ktorý implementuje výhľadový bod tak ako je popísaný v kapitole 5.2.1.. Tento typ výhľadových bodov je zakorenený v triedach *StaticLookOuts* a *StaticLookOut*. Všetky triedy spojené s výhľadovými bodmi sú sústredené v mennom podpriestore `RacingCarSimulation.QLearning.States.LookOuts`.

Akcia umelého hráča je zahrnutá v triede **Action**. Táto trieda ma jedného potomka, konkrétne triedu **MirroredAction**, ktorá by mala byť použitá v prípade, že je prostredie reprezentované stavmi *MirroredState* alebo *DynamicMirroredState*. Tieto dve triedy je možné nájsť v mennom podpriestore `RacingCarSimulation.QLearning.Actions`.

Tabuľka Q-hodnôt je implementovaná pomocou dvojrozmerného poľa. Nad týmto poľom je však vybudovaná abstrakcia, ktorú predstavuje trieda **QLearningTable**, ktorá poskytuje pomocné metódy pre prácu s touto tabuľkou.

Umelý hráč je v rámci tejto práce založený na konfigurácii, ktorá obsahuje hodnoty jednotlivých parametrov procesu učenia a definície stavov a akcií. Táto konfigurácia je zhrnutá v triede **QLearningConfiguration**. Jej architektúra odráža hierarchickú architektúru triedy *QLearning* a jej položiek, čo znamená, že každá z vyššie spomenutých tried má svoju konfiguráciu. V implementácii je reprezentovaná jednou triedou a tieto triedy na seba podľa rovnakej hierarchie nadväzujú. Navyše

každá konfigurácia poskytuje jednu statickú položku, ktorá predstavuje danú konfiguráciu s prednastavenými (default) hodnotami. Všetky triedy spojené s konfiguráciou sú uvedené v mennom podpriestore `RacingCarSimulation.QLearning.Configurations`.

Vizualizácia akcie pomocou šípky je zahrnutá v mennom podpriestore `RacingCarSimulation.QLearning.Graphics`, kde ju implementuje trieda **ActionArrow**.

7. **RacingCarSimulation.Common**

Tento menný priestor bol vytvorený za účelom zoskupenia tried, ktorých funkcionality má všeobecný charakter a v rámci implementácie by mohla byť alebo je použitá na viacerých miestach. Skladá sa z nasledujúcich podpriestorov:

7.1. `RacingCarSimulation.Common.Csv`

Podpriestor `Csv` obsahuje jednu triedu a to **CsvManager**, ktorá poskytuje sadu metód pre vytvorenie ľubovoľného CSV súboru.

7.2. `RacingCarSimulation.Common.Data`

Podpriestor `Data` obsahuje niekoľko významných tried, ktoré väčšinou predstavujú abstrakciu nad menným priestorom `System.Reflection`.

- `DataActivator`: poskytuje metódy pre vytvorenie inštancie objektu, ktorého typ je zadaný ako parameter metódy.
- `PropertyManager`: slúži pre získanie alebo nastavenie hodnôt zadaných v konzole
- `DataManager.IDManager`: slúži pre manažovanie unikátnych identifikačných čísel v rámci jedného behu aplikácie

7.3. `RacingCarSimulation.Common.Exceptions`

V tomto mennom podpriestore sa nachádzajú triedy, ktoré dedia od triedy `System.Exception` a definujú typy výnimiek, ktoré môžu v aplikácii nastať.

Okrem týchto tried je tu prítomná trieda **ExceptionHandler**, ktorá sa v rámci celej implementácii aplikácie používa na ošetrovanie vyhodnených výnimiek.

7.4. RacingCarSimulation.Common.Extensions

Tento podpriestor obsahuje 3 typy tried:

- Statické triedy obsahujúce extension metódy pre rôzne typy objektov
- Statické triedy, ktorých metódy sa používajú pre neštandardné konverzie medzi rôznymi typmi objektov
- Statické triedy definujúce komparátory

7.5. RacingCarSimulation.Common.FilesManagement

Podpriestor *FilesManagement* je v implementácii zavedený kvôli triede **FileManager**, ktorá pomocou dialógových okien sprostredkováva súbory vybrané užívateľom.

7.6. RacingCarSimulation.Common.Gamelog

V tomto podpriestore sa nachádza trieda **Gamelog**, ktorá je zodpovedná za vytvorenie logovacích súborov a následný zápis logovacích správ

7.7. RacingCarSimulation.Common.IO

Ako už názov naznačuje, tento podpriestor disponuje triedami, pomocou ktorých sú zabezpečené vstupno-výstupne operácie v implementácii. Najvýznamnejšou triedou, ktorá sa používa skoro v každej časti implementácie aplikácie je **XmlManager**, ktorý slúži na serializáciu/deserializáciu objektov do/z xml súborov uložených na disku.

7.8. RacingCarSimulation.Common.Svg

V tomto mennom priestore sa nachádzajú triedy, ktoré sa v rámci simulácie používajú na vyčítavanie rôznych kriviek zo súborov svg, ktoré sú v rámci týchto súborov reprezentované postupnosťou špeciálnych príkazov.

7.9. RacingCarSimulation.Common.UserInput

Menný podpriestor *UserInput* zoskupuje triedy, ktoré sa v aplikácii starajú o vstup užívateľa. Najdôležitejšou triedou v rámci tohto priestoru je trieda **InputManager**, ktorá reprezentuje stav klávesnice a myši a poskytuje množinu metód, ktoré vývojár môže použiť na naviazanie žiadanej funkcionality k zvolenej klávese.

8. RacingCarSimulation.Menu

Tento menný priestor poskytuje implementácie prvkov, ktoré užívateľ môže nájsť na menu obrazovke. Konkrétne sa jedna o reprezentáciu jednotlivých menu, ktoré sú na obrazovke zobrazené a o mechanizmus, pomocou ktorého je sprostredkovaný prechod medzi týmito menu.

Najdôležitejšou triedou tohto menného priestoru je trieda **MenuManager**, ktorej inštancia je obsiahnutá v triede *MenuScreen*. Táto trieda sa stará o to, aby na obrazovke bolo vždy zobrazené to správne menu. Menu je implementované pomocou triedy *Menu* a jeho položky pomocou triedy *MenuItem*, ktorá je abstraktná. Keďže každá položka môže mať inú funkcionality (zmena obrazovky, prechod na iné menu, ukončenie aplikácie, zmena rozlíšenia), jednotlivé typy týchto položiek sú podľa funkcionality rozdelené do tried, ktoré sú derivované od triedy *MenuItem* a ktorých implementácia sa na základe požadovanej funkcionality líši.

6.4.2. RacingCarSimulationTests

Projekt `RacingCarSimulationTests` je v rámci tejto práce zahrnutý za účelom poskytnutia nástroja, pomocou ktorého by bolo možné otestovať správne fungovanie takých komponent aplikácie *Racing Car Simulation*, ktorých zložitosť si to vyžaduje. Takýto nástroj poskytuje **Unit Test** framework, ktorý je plne podporovaný samotným Visual Studiom a slúži k testovaniu .NET kódu.

Tento projekt obsahuje sadu Unit testov pre testovanie funkčnosti implementácie umelej inteligencie. V rámci tohto projektu existujú dva menné priestory:

1. `RacingCarSimulationTests.QLearning`

Tento menný priestor obsahuje jedinú triedu **`QLearningTests`**. Táto trieda je parciálna a rozložená do dvoch súborov:

- `QLearningImportExportTests.cs` – obsahuje Unit testy pre testovanie exportovania a importovania tabuľky Q-hodnôt
- `QLearningStatesTests.cs` – obsahuje Unit testy pre správne získavanie stavov z prostredia a vypočítavanie ich indexov, pre všetky typy stavov (viď kapitolu 5.2.1.)

2. `RacingCarSimulationTests.QLearning.TestDataStructures`

Tento menný priestor disponuje triedami, ktoré v Unit testoch reprezentujú prostredie simulácie. Keďže počas Unit testu simulácia nebeží a pre získanie stavov je potrebné poznať parametre prostredia, tieto parametre musia byť vygenerované.

Hlavnou triedou je trieda **`TestEnvironmentDataGenerator`**, ktorá implementuje rozhranie *`IEnvironmentDataProvider`*. Táto trieda obsahuje inštancie ďalších tried obsiahnutých v tomto mennom priestore, ktoré dokopy implementujú rozhrania *`ICarDataProvider`*, *`ICarDataPhysicsProvider`*, *`IRacingLineDataProvider`* a *`IMapDataProvider`*. Každá z týchto tried navyše

implementuje rozhranie *IEnumerable* a špecializuje sa na jednu zložku prostredia simulácie. Tieto triedy sú určené na generovanie tých parametrov prostredia, ktoré sú spojené s ich špecializáciou a toto generovanie je implementované pomocou enumerátorov za pomoci príkazu **yield return**.

V triede *TestEnvironmentDataGenerator* sú hodnoty získané z týchto enumerátorov kombinované pomocou triedy *TestEnvironmentDataCombiner*, ktorá taktiež implementuje rozhranie *IEnumerable*. Toto rozhranie je parametrizované jej vnorenou (nested) triedou *Data*. Trieda *Data* obsahuje všetky parametre prostredia, ktoré sú v rámci *TestEnvironmentDataCombiner* kombinované tak, aby spolu vygenerovali všetky možné kombinácie, ktoré môžu v prostredí nastať.

6.4.3. QLearningApplication

Tento projekt predstavuje aplikáciu *QLearning* (viď kapitolu 5.4.). Využíva GUI (grafické užívateľské prostredie) **Windows Forms**, ktoré je poskytované frameworkom .NET a ktoré sa používa pre tvorbu okenných aplikácií prostredníctvom vytvárania takzvaných formulárov.

Architektúra tejto aplikácie je založená na vzore **Model-View-Controller**, od ktorého sa odvíjajú menne priestory. Tie sú nasledovné:

1. QLearningApplication.Control

Tento menný priestor obsahuje triedy, ktoré sú spojené s ovládaním aplikácie. Tieto triedy sú:

1.1 Controller

Trieda *Controller* sprostredkováva komunikáciu medzi triedou *SimulationController* (viď bod 1.2) a grafickým užívateľským prostredím. Poskytuje sadu metód, ktoré sú z GUI volané, keď užívateľ vykoná nejakú akciu. Tieto metódy na základe akcie upravujú stav grafického užívateľského prostredia pomocou volania príslušných metód v triedach menného priestoru

`QLearningApplication.View` a zároveň tieto akcie prekladajú do podoby, na ktoré je simulácia schopná reagovať. Akcie v tejto podobe potom propaguje do triedy *SimulationController*, ktorej inštanciu obsahuje ako svoju dátovú položku. Taktiež táto trieda implementuje načítavanie vstupných parametrov simulácie zo súborov, ktorých cestu užívateľ zadáva v užívateľskom prostredí.

Metódy, ktoré poskytujú výstupné dáta pre užívateľské prostredie alebo obsahujú procesy, ktorých vykonávanie trvá istú dobu a na ktoré užívateľské prostredie čaka, sú **asynchrónne**.

1.2 SimulationController

Trieda *SimulationController* predstavuje medzikrok medzi triedou *Controller* a implementáciou simulácie. Jej najdôležitejšou dátovou položkou je inštancia samotnej simulácie (triedy *Simulation*) a vlákno, v ktorom je umiestnená slučka predstavujúca prepočty simulácie. Táto trieda je zodpovedná za všetku komunikáciu zo simuláciou. To zahŕňa vyvolanie jej inicializácie, dodanie vstupných parametrov, ktoré sú sprostredkované pomocou triedy *Controller* a volanie metód simulácie, ktoré súvisia s jej behom.

2. QLearningApplication.View

Triedy tohto menného priestoru slúžia pre komunikáciu s užívateľom. Táto komunikácia môže prebiehať rôznymi spôsobmi, na ktorých sú ostatné menné priestory nezávislé. Jej hlavnou triedou je trieda **View**, ktorá je abstraktná a poskytuje množinu abstraktných metód, ktoré dokopy zahŕňajú všetky akcie, ktoré môže užívateľ v aplikácii vykonávať a všetky akcie, pomocou ktorých trieda *Controller* upravuje stav užívateľského rozhrania.

Túto triedu implementujú dvaja potomkovia:

2.1 ConsoleView

Trieda *ConsoleView* je v rámci implementácie prázdna a poskytuje prípravu na budúce rozšírenie aplikácie QLearning, v ktorom táto aplikácia komunikuje s užívateľom cez konzolu.

2.2 FormView

Trieda *FormView* slúži na sprostredkovanie akcií, ktoré popisujú metódy triedy *View* formuláru, ktorý v aplikácii QLearning predstavuje okenné grafické užívateľské prostredie. Tento formulár je implementovaný pomocou triedy **MainForm**, ktorá je potomkom triedy *System.Windows.Form*. Trieda *FormView* obsahuje inštanciu typu *MainForm* a vo svojich metódach komunikuje s týmto formulárom.

V mennom priestore *QLearningApplication.View* je ešte dôležitou položkou enumerácia nazvaná **Step**, ktorá obsahuje výčet hodnôt reprezentujúcich aktuálny krok aplikácie. Pri spustení aplikácie má aktuálny krok hodnotu *Step.Input*. Tento krok znamená, že v aplikácii sa práve zadávajú vstupné parametre. Keď užívateľ tieto parametre zadá a klikne na tlačidlo *Confirm Input*, nastáva kontrola vstupných údajov, ktorú predstavuje krok *Step.CheckingInput*. Po úspešnej kontrole užívateľ zadáva nastavenia simulácie, čo reprezentuje krok *Step.Settings*, a tak ďalej. Dokopy má enumerácia *Step* 14 hodnôt a slúži ako nástroj pre komunikáciu medzi triedami *Controller* a *View*, v rámci ktorej má *Controller* prístup k aktuálnemu kroku a môže ho na základe rôznych situácií meniť.

Pri zmene tohto kroku trieda *FormView* túto informáciu propaguje triede *MainForm*, ktorá podľa hodnoty kroku mení vzhľad formuláru.

3. QLearningApplication.Models

Menný priestor QLearningApplication.Models disponuje triedami, ktoré definujú objekty použité v rámci tejto aplikácie. Týmito objektami sú:

- **InputPaths:** štruktúra, ktorá obsahuje cesty k súborom reprezentujúcim vstupné parametre simulácie, ktoré užívateľ do aplikácie zadal
- **Input:** štruktúra, ktorá obsahuje vstupné parametre simulácie nahrané triedou *Controller* zo súborov, ktorých cesty sú zadané v *InputPaths*
- **Settings:** štruktúra obsahujúca nastavenia simulácie, ktoré užívateľ do aplikácie zadal
- **Statistics:** štruktúra obsahujúca informácie o aktuálnom stave simulácie, ktoré sú zobrazované užívateľovi

7. Záver

V rámci tejto práce bola vytvorená aplikácia, ktorá umožňuje jej užívateľovi simulovať jazdu automobilu v rôznych prostrediach. Pri vypracovávaní tejto práce bol kladený dôraz na splnenie všetkých cieľov, ktoré boli určené v úvode. V súvislosti s týmito cieľmi boli dosiahnuté nasledovné výsledky.

Aplikácia má vypracované grafické prostredie, ktoré je nastaviteľné a konfigurovateľné. Splňa všetky predpoklady, ktoré boli dané simuláciou. Zároveň aplikácia počas simulácie poskytuje užívateľovi množstvo informácií, ktoré sú spojené ako s fyzikálnym modelom a fyzikálnymi veličinami, ktoré sú v ňom zahrnuté, tak aj s prostredím a procesom učenia

V práci bol implementovaný pokročilý fyzikálny model, ktorý reflektuje správanie sa automobilu v reálnom svete.

Do simulácie bol implementovaný umelý hráč, ktorý sa s automobilom snaží naučiť jazdiť po vopred definovanej závodnej stope. V súvislosti s týmto bola zvolená metóda učenia Q-learning, ktorá bola naviazaná na prostredie simulácie.

Pri vývoji aplikácie bol kladený dôraz na čo najväčšiu konfigurovateľnosť, vďaka čomu je užívateľovi poskytnuté nadefinovať vlastný automobil pomocou zadania požadovaných hodnôt jeho parametrov a upravovať prostredie simulácie skrz konzolu, ktorá je súčasťou aplikácie. Užívateľ môže taktiež do aplikácie zadávať hodnoty parametrov procesu učenia a konfigurovať jeho stavy a akcie.

V rámci tejto práce bola vypracovaná reprezentácia závodnej stopy pomocou kubickéj Bézierovej krivky, ktorú môže užívateľ kresliť vo vhodných programoch pre vektorovú grafiku. Závodná stopa bola skrz prostredie simulácie zahrnutá do procesu učenia pomocou vypracovania množiny parametrov, ktoré ju popisujú.

Aplikácia bola vyvíjaná s ohľadom na ďalšie rozšírenia. Návrhy na niektoré z nich sú uvedené v nasledujúcej podkapitole.

7.1. Možnosti ďalších rozšírení

Aplikácia je pripravená na rozširovanie rôznymi smermi. V tejto kapitole sú opísané ich návrhy.

- Najväčším možným rozšírením, na ktoré je aplikácia pripravená je rozšírenie na inú platformu. Vďaka použitiu aplikačných rámcov uvedených v kapitole 6.1. je toto rozšírenie možné na viacero platforiem.
- Rozšírenie aplikácie *Qlearning* aj do konzolovej podoby. Architektúra tejto aplikácie je na to pripravená, stačilo by už iba doplniť implementáciu zopár metód.
- Rozšírenie množiny nastavení dostupných v *menu nastavení* (viď kapitolu 2.1.2) o možnosť meniť priradenie kláves k akciám simulácie priamo v aplikácii.
- Pridanie zvuku do simulácie. Framework Microsoft XNA podporuje do aplikácie pridávať zvuky a prehrávať ich. Vďaka tomu by bolo možné do aplikácie pridať hudbu a minimálne zvuk motora automobilu.
- Zväčšiť množinu parametrov a koeficientov prostredia, ktoré je možné nastavovať skrz konzolu.
- Rozšírenie fyzikálneho modelu automobilu. V súčasnej verzii aplikácie *Racing Car Simulation* sú jednotlivé fyzikálne veličiny prepočítavané v rámci prednej a zadnej nápravy. Táto implementácia by sa dala rozšíriť na prepočítavanie fyzikálnych veličín na jednom kolese namiesto nápravy, čo by viedlo k ďalšiemu priblíženiu správania sa automobilu reálnemu svetu. Implementácia je už na to sčasti pripravená.
- Doplnenie databázy dostupných automobilov a máp, ktoré by boli v aplikácii užívateľovi okamžite dostupné.
- Doplnenie množiny panelov. V súčasnej verzii existuje v rámci simulácie 9 informačných panelov, ktoré avšak nepokrývajú celú množinu dostupných informácií. Zároveň by bolo možné do týchto panelov pridať ďalšie vizualizácie v podobe grafov vývoja veľkostí jednotlivých síl v reálnom čase

8. Súvisiace práce

V rámci tejto kapitoly je v krátkosti popísaná jedna hra, ktorá má podobný charakter ako *Racing Car Simulation*. Z aplikácii, ktoré implementujú fyzikálny model automobilu a prostredia na úrovni približnej tejto práci a zároveň sa venujú umelej inteligencii je najznámejšia hra *TORCS*.

TORCS alebo *The Open Racing Car Simulator* začínal približne rovnako ako aplikácia *Racing Car Simulation*. Jej vývoj začal v roku 1997 kedy sa volala ešte len *Racing Car Simulator* (pozor na podobnosť mien). V tom čase to bola ešte 2D hra, ktorá implementovala jednoduchú fyziku automobilu. Keď si odmyslíme odstup času od jej vývoja a s tým spojený rozdiel v použitých technológiách a fakt, že *TORCS* v tom čase neimplementoval žiadnu umelú inteligenciu, tak sa táto hra podobala aplikácii obsiahnutej v tejto práci. Obsahovala jednoduché grafické prostredie, ktoré sa avšak viac sústreďovalo na závodenie ako na simuláciu ľubovoľného jazdenia za účelom sledovania fyzikálneho modelu.

Postupom času sa *TORCS* vyvíjal, grafické prostredie sa zmenilo v 3D, do aplikácie boli postupne pridané zvuky, fyzikálny model bol vylepšovaný, prostredie taktiež. Jeho vývoj istým spôsobom reflektuje kapitolu 7.1., v ktorej bolo zmienených zopár návrhov na ďalšie rozšírenie aplikácie *Racing Car Simulation*.

V dnešnej podobe *TORCS* predstavuje open-source 3D simuláciu automobilového závodenia (viď obrázok 8.1.), ktorá je dostupná na mnohých platformách, napríklad Linux, Mac OS X, Windows, FreeBSD. Všetci ju ale poznajú z iného dôvodu. Prostredie, s ktorým sú vďaka open-source dostupnosti simulácie užívatelia-programátori oboznámení sa stalo útočiskom pre závodenie medzi umelými hráčmi. Pravidelne sú konané akcie, v rámci ktorých programátori nechávajú namiesto seba závodieť agenta, ktorého naimplementovali a potom ich výsledok podrobujú rôznym rozborom.



Obrázok¹¹ 8.1.: Ukážka simulácie hry *TORCS*

¹¹ Zdroj <http://www.linuxlinks.com/article/20080510070725976/TORCS.html>

9. Zoznam použitej literatúry

Julius Mackerle: *Automobil s lepší účinností*. 1. Vydanie, Praha SNTL, 1985

Chuck Edmondson: *Fast Car Physics*. JHU Press, 2011, ISBN: 9780801898235

Brian Beckman: *The Physics of Racing* [online]
http://phors.locost7.info/files/Beckman_-_The_Physics_of_Racing.pdf

An introduction to car physics modelling for games [online]
<http://www.asawicki.info/Mirror/Car%20Physics%20for%20Games/Car%20Physics%20for%20Games.html>

Child, C.H.T a Trusler, B.P.: *Implementing Racing AI using Q-Learning and Steering Behaviours*. UK, 2014

S. J. Russel a P. Norvig: *Artificial Intelligence: A Modern Approach*, 3. vydanie. Prentice Hall, Upper Saddle River, USA ©2009, ISBN:0136042597 9780136042594

I. Millington a J. D. Funge: *Artificial intelligence for games*. 2. vydanie. Morgan Kaufmann Publishers, Burlington, 2009.

Cardamone, Loiacono a Lanzi: *Proceedings of the 11th Annual conference on Genetic and evolutionary computation, Evolving competitive car controllers for racing games with neuroevolution*. USA ©2009, ISBN: 9781605583259

O. Stanley & Miikkulainen: *Evolutionary Computation, Evolving Neural Networks through Augmenting Topologies*. 2014, ISSN: 10636560

Onieva, Pelta, Alonso, Milanés a Perez: *2009 IEEE Symposium on Computational Intelligence and Games, A modular parametric architecture for the TORCS racing engine*. 2009, ISBN: 9781424448142

Chad Carter: *Microsoft XNA Unleashed: Graphics and Game Programming for Xbox 360 and Windows*. 1. vydanie. Sams Publishing, 2007, ISBN: 0672329646 9780672329647

Chad Carter: *Microsoft XNA Game Studio 3.0 Unleashed*. 1. vydanie. Sams Publishing, 2009, ISBN: 0672330229 9780672330223

Luke Drumm: *Microsoft XNA 4.0 Game Development Cookbook*. Packt Publishing, 2012, ISBN: 1849691983 9781849691987

A.T. Chamillard: *Beginning C# Programming with MonoGame*. Burning Teddy, 2015

10. Príloha A – Obsah priloženého CD

- Adresár *Aplikacia/Binaries* obsahujúci spustiteľný súbor aplikácie *Racing Car Simulation* a aplikácie *QLearning*
- Adresár *Aplikacia /Solution* obsahujúci celú prácu otvorenú vo Visual Studiu skrz súbor *RacingCarSimulation.sln*
- Adresár *Aplikacia /RacingCarSimulation*, ktorý predstavuje užívateľský adresár (viď kapitolu 2.3.1.2.) a ktorý je pre načítanie dostupných parametrov (viď kapitolu 2.1.3.) nutné skopírovať do dokumentov aktuálne prihláseného užívateľa v lokácii: `C:\Users\userName\Documents\`
- Adresár *Dokumentacia/Priklady suborov* obsahujúci:
 - Ukážky xml súborov reprezentujúcich všetky vstupné parametre simulácie popísané v kapitole 3.1., konkrétne sú to:
 - *Car.xml* a *Car-model.xml* pre reprezentáciu automobilu
 - *TrackMap.xml* pre reprezentáciu mapy so závodným okruhom
 - *SimulationMap.xml* pre reprezentáciu nekonečnej mapy
 - *QLearningConfiguration.xml* pre reprezentáciu konfigurácie umelej inteligencie
 - Súbor *RacingLine.svg* obsahujúci kubickú Bézierovú krivku reprezentujúcu závodnú stopu
 - Ukážku súboru *GameConfiguration.xml*
 - Ukážku logovacieho súboru *Gamelog.txt*
- Súbor *Sprievodca ovladanim.pdf* umiestnený v adresári *Dokumentacia*, ktorý obsahuje tabuľku dostupných operácií, pomocou ktorých je možné ovládať aplikáciu. Tejto tabuľke je k dostupným operáciám uvedené prednastavené priradenie kláves. Keďže je pre väčšinu operácií možné zmeniť priradenie klávesy (viď kapitolu 2.3.2.), v tejto tabuľke sú uvedené

kódy reprezentujúce tieto akcie. V tomto súbore je obsiahnutá ešte druhá tabuľka, ktorá pre jednotlivé klávesy uvádza ich číselné kódy.

- Táto prácu vo formáte *pdf*, umiestnená v adresáry *Dokumentacia*
- Súbor ReadMe.txt, ktorý opisuje základnú štruktúru prílohy