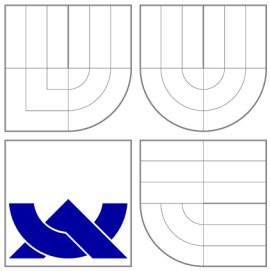
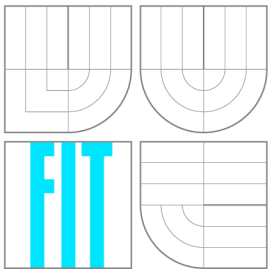


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

ŘÍZENÍ FORMACE VOZIDEL

CONTROLLING OF VEHICLES FORMATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETER REVICKÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN, Ph.D.

BRNO 2016

Abstrakt

Cieľom tejto práce je vytvoriť systém na riadenie kolesových vozidiel s pohybovými obmedzeniami. V práci sa rieši riadenie formácie a jej dynamické prispôsobovanie v závislosti od prekážok v okolí a zároveň aj riadenie vozidiel. Algoritmy riadiace vozidlo sú založené na báze potenciálových polí. Systém je implementovaný v 2D prostredí v hernom engine Unity. V závere je systém otestovaný na rôznych scenároch ako prechod formácie úzkym miestom, prekážka čiastočne blokujúca formáciu, scenár s dynamickou prekážkou apod.

Abstract

The goal of this thesis is to create system for formation management of wheeled vehicles with kinematic constraints. The work presents the way to control vehicle and how to manage formation in presence of obstacles. Algorithms used for vehicle control are based on potential fields. Whole system is implemented in Unity game engine in 2D enviroment. The system is then tested on various scenarios such as passing through narrow passage, obstacle partially blocking formation, dynamic obstacle avoidance etc.

Kľúčové slová

riadenie formácie vozidiel, potenciálové polia, unity, unity3d, vyhýbanie sa prekážkam, dynamická formácia, adaptívna formácia, koordinovaný pohyb, neholonomické obmedzenia

Keywords

vehicles formation control, potential fields, unity, unity3d, obstacle avoidance, dynamic formation, adaptive formation, coordinated movement, non-holonomic constraints

Citácia

REVICKÝ, Peter. *Řízení formace vozidel.* Brno, 2016. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Rozman Jaroslav.

Řízení formace vozidel

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána Ing. Jaroslava Rozmana, Ph.D. Uviedol som všetky literárne pramene a publikácie, z ktorých som čerpal.

.....
Peter Revický
16. mája 2016

Poďakovanie

Týmto sa chcem poďakovať vedúcemu tejto práce Ing. Jaroslavovi Rozmanovi Ph.D. a odbornému konzultantovi Ing. Davidovi Hermanovi za ich cenné rady a tipy pri vypracovávaní tejto práce.

Obsah

Kapitola 1

Úvod

V poslednej dobe sa dostáva do pozornosti potreba riadiť formáciu. Na túto tému je zameraný výskum hlavne v oblasti robotiky. Riadenie formácie tam má význam pri rôznych úlohách ako napr. odhŕňanie snehu na letisku, preskúmavanie prostredia, hliadkovanie a monitorovanie nejakej oblasti pomocou drónov apod. Kooperatívnym spôsobom roboti, podobné úkoly, vykonávajú efektívnejšie a zároveň sa tak znižuje záťaž na jedného robota.

Je trendom, že človek robí menej a menej vecí a na mnoho činností sú nasadené stroje. Podobne je to aj s vozidlami. Dnes sú schopné vozidlá sa riadiť sami. Autonómne vozidla boli originálne vyvinuté pre armádu. Skupina týchto autonómnych vozidiel musí byť tiež riadená organizovaným spôsobom. Ozbrojené sily sú najčastejšou oblasťou kde možno aplikovať pohyb vo formácií. Koordinovaný pohyb im umožňuje dosiahnuť taktický prístup v boji.

Mimo toho sa formácie dajú využiť aj vo vojenských simulátoroch alebo počítačových hrách k dosiahnutiu realistického pohybu skupiny jednotiek.

Táto práca sa bude sústrediť riadenie formácií pozemných vozidiel s pohybovými obmedzeniami čo robí riadenie o čosi komplexnejším. Zameriavať sa bude predovšetkým na vozidlá s podvozkom typu automobil. Systém bude implementovaný v prostredí Unity3D[1] a algoritmy budú prispôsobené pre použitie v počítačovej hre alebo simulátore.

V kapitole dva bude popísané plánovanie trasy, ktoré je prerekvizitou k riadeniu formácie. Spomenuté v nej bude ako v plánovači reprezentovať prostredie, algoritmy na vyhľadávanie cesty a plánovanie pohybu a vysvetlenie pojmov použitých v rámci plánovania. Tretia kapitola priblíži koordinovaný pohyb viacerých vozidiel. Rozoberať sa tam bude ako definovať formáciu a aké metódy sa používajú na jej riadenie. Kapitola štyri sa bude zaoberať návrhom systému. V piatej kapitole budú popísané implementačné detaily. Na záver v kapitole päť bude celý systém otestovaný na rôznych scenároch.

[1] http://www.unity3d.com

Kapitola 2

Plánovanie trasy

Problém hľadania trasy patrí medzi jeden z ústredných problémov a tvorí jednu z najzákladnejších častí umelej inteligencie. Používa sa takmer v každom systéme, ktorý zahŕňa prvky umelej inteligencie ako napr. počítačové hry kde je potreba pohybovať s hernými objektami v priestore, v robotike pri presúvaní robota z bodu A do bodu B alebo v iných systémoch kde je potrebné dostať objekt do cieľovej pozície. Rovnako aj tu aby sa vozidlo respektíve skupina vozidiel vedela dopraviť do destinácie je dôležité aby systém vedel nájsť trasu, ktorá napovie ako sa tam dostať.

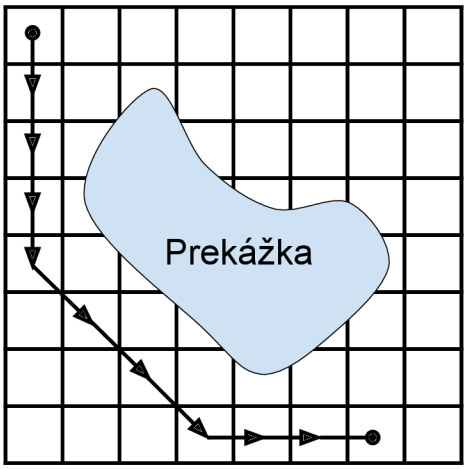
Pre tento účel slúži plánovač. Ten typicky dostane počiatočnú pozíciu spolu s cieľovou a jeho úlohou je nájsť nekolíznu trasu. Nájdená trasa k cieľu predstavuje postupnosť bodov. Pri plánovaní môže zohľadňovať viacero parametrov ako napr. bezpečnosť prepravy, typ podvozku vozidla, predikcia pohybu iných objektov apod. čo ovplyvňuje tvar trasy. Keďže sa v prostredí môžu nachádzať aj prekážky, takáto trasa nemusí vždy existovať. Dobrý plánovač by preto mal zahlásiť chybu v takýchto prípadoch. Táto kapitola je venovaná stavbe takého plánovača. Spomenuté tu budú algoritmy, ktoré sa používajú pri plánovaní trasy a pohybu ako aj metódy na reprezentáciu prostredia.

2.1 Reprezentácia prostredia

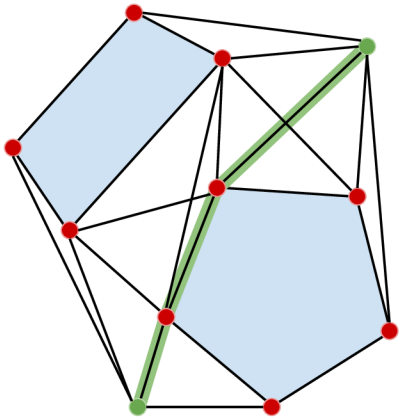
Aby mohol plánovač vyhľadávať trasu musí mať prostredie, v ktorom vyhľadáva nejakým spôsobom reprezentované. Plánovač tak vie, ktorá časť prostredia tvorí prekážku, ktorá časť je prejazdná alebo vie o časti, ktorá je síce prejazdná ale nežiadúca (napr. vtedy ak sa objekt nezmestí cez úzke miesto). Prostredie môže byť reprezentované dvoma spôsobmi a to buď spojito alebo diskrétne. V reálnom svete je prostredie spojité. To ale problém hľadania trasy komplikuje keďže spojitý priestor si možno predstaviť ako množinu s nekonečne mnoho možnými stavmi. Spojitá reprezentácia nájde skôr uplatnenie pri vyhýbaní sa pohyblivým prekážkam než plánovaní. Za účelom zjednodušenia sa obyčajne spojité prostredie rozdelí na konečnú množinu stavov, v ktorých sa môže vozidlo nachádzať. Tento proces sa nazýva diskretizácia prostredia. Nájdenie trasy potom znamená vyhľadať postupnosť stavov vedúcich do cieľového stavu vo vzniklej konečnej množine stavov [4].

Stav typicky predstavuje bod v priestore. Pri diskretizácií teda vznikne množina takýchto bodov, ktorú nazývame uzly. Tie sú určitým spôsobom následne prepojené. Tieto prepojenia označujeme ako hrany. Usporiadaná množina hrán a uzlov tvorí dohromady graf. Možností ako takto rozdeliť prostredie je veľa. Avšak, sú štandardné metódy, ktoré sa používajú najčastejšie. U niektorých je treba rozdeľovať ručne iné to dokážu automaticky [6].

Mriežka Mriežková reprezentácia rozdeľuje 2D-3D prostredie do pravidelných regiónov. Zvyčajne majú tvar štvorca, ale môžu to byť aj trojuholníky či šesťhrany, avšak sú používané zriedkavo. Mriežková forma je jednoduchá a ľahko zrozumiteľná, a kvôli týmto vlastnostiam je to často používaná metóda.



Obr. 2.1: Príklad mriežkovej reprezentácie.



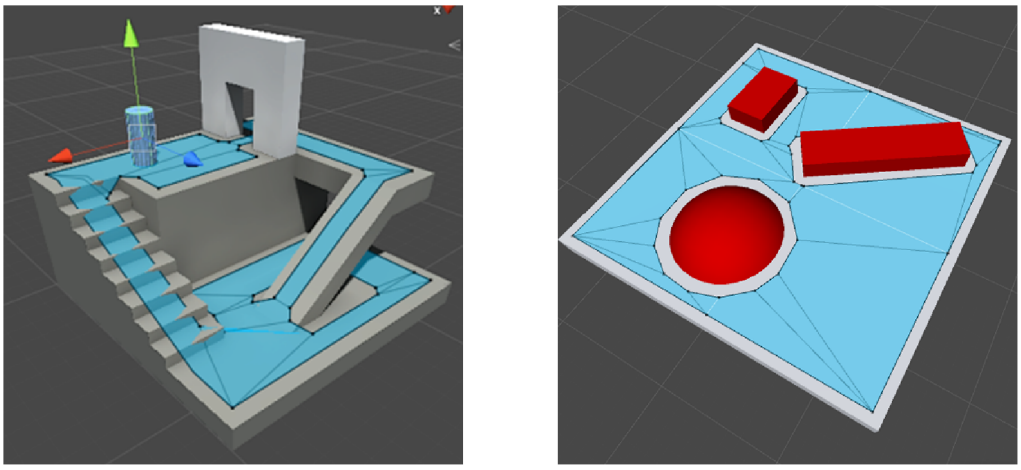
Obr. 2.2: Príklad grafu viditeľnosti.

V rámci mriežky sú tri možnosti ako sa v nej pohybovať. Môže to byť pohyb po dlaždiciach, po hranách dlaždice, alebo po jej vrcholoch. Pri pohybe po dlaždiciach tvorí stred každej dlaždice jeden uzol. Dlaždicový pohyb demonštruje aj obrázok 2.1. Pohyb po hranách a vrcholoch sa používa v prípade ak by dlaždice boli veľké [9].

Grafy viditeľnosti Častá alternatíva k mriežkovej forme je reprezentácia mnohouholníkmi. Rozdiel oproti mriežke je, že uzly sú rozmiestnené nepravidelne. Konkrétne ich tvoria okraje prekážok. Ich vzájomné prepojenia sú tiež iné. Každý uzol je prepojený s iným uzlom len vtedy ak lúč ich spájajúci nekoliduje so žiadnou prekážkou. Z toho plynie aj názov graf viditeľnosti pretože uzly sú spojené len vtedy ak na seba „vidia". Do grafu sa musí pridať ešte počiatočný a cieľový uzol pretože ani jeden z týchto bodov nemusí byť vždy okraj prekážky. Na obrázku 2.2 je vidieť príklad takého grafu. Červené body predstavujú uzly okrajov prekážok a zelené body sú novo pridané uzly s počiatočným a koncovým stavom. Zelená čiara predstavuje nájdenú trasu v grafe.

Navigačná sieť V poslednej dobe veľmi vyskytovaná metóda v počítačových hrách. U grafov viditeľnosti polygóny reprezentujú prekážky. Tu je to presne naopak. Polygóny predstavujú oblasť, po ktorej sa môže objekt pohybovať a platí to, že sa neprekrývajú navzájom. Z hľadiska počítačovej grafiky sú všetky 3D objekty zložené zo siete polygónov (v angl. nazývané mesh). Najčastejšie sú to trojuholníky. Túto grafickú štruktúru možno tiež použiť ako reprezentáciu. V mnohých hrách sú niektoré 3D objekty slúžiace ako „podlaha". Každý polygón na týchto objektoch predstavuje uzol. Uzly sú spojené vtedy, ak polygóny zdieľajú hranu. Väčšinou do procesu tvorby navigačného grafu je zahrnutý dizajnér, ktorý v editore označí podlahové objekty a ten graf automaticky vytvorí [6].

Hlavným motívom prečo sa navigačná sieť používa je ten, že presne vymedzuje pohybovú plochu objektu. Tým neobmedzuje pohyb objektu po presne stanovených bodoch ako pri mriežke ale umožňuje im sa voľne po nej pohybovať a zaručuje, že nájdená trasa bude



Obr. 2.3: Príklad navigačnej siete.

najkratšia. Pretože aj keď objekt potrebuje obísť prekážku, najkratšou cestou je to po obvode prekážky. Ďalší dôvod prečo sa intenzívne používajú v hrách je ten, že eliminujú množstvo nepotrebných uzlov. Vyhľadávanie v grafe je tak rýchlejšie.

2.2 Prehľadávacie algoritmy

Vo chvíli keď sme rozhodnutí akým spôsobom budeme prostredie reprezentovať, potrebujeme algoritmus, ktorý bude v tomto prostredí vyhľadávať cestu. V tejto sekcii sú popísané algoritmy pre taký účel.

2.2.1 A*

Najrozšírenejší algoritmus na vyhľadávanie cesty v grafe. A* je úplný čo znamená, že vždy nájde riešenie ak existuje a zároveň je optimálny čo značí, že nájde najlepšie riešenie, avšak len vtedy ak je heuristika tiež optimálna. Naviac, je jednoduchý na implementáciu. Možno aj to sú dôvody prečo ako primárny vyhľadávací algoritmus.

A* je rozšírený Dijkstrov algoritmus [8]. Kľúčová vec spočíva v zredukovaní počtu prehľadávaných uzlov použitím heuristiky. Má dva zoznamy open a closed. V zozname open si udržuje navštívené uzly, ktoré ešte neboli spracované. V zozname closed sa nachádzajú už spracované uzly. A* je iteratívny algoritmus. V každej iterácií uvažuje jeden uzol a sleduje ohodnotenie uzlov s nim spojených. Označme si taký uzol ako aktuálny uzol a uzly spojené s týmto uzlom ako susedné uzly. Pre každý susedný uzol nájde koncový uzol a zistí celkové ohodnotenie cesty do cieľa pri prechode susedným uzlom. Cenové ohodnotenie uzla *f(n)* je nasledovné:

$$f(n) = g(n) + h(n) \tag{2.1}$$

kde *g(n)* je cena cesty od počiatočného uzlu až po aktuálny uzol a *h(n)* heuristicky predpokladaná cena cesty pri prechode susedným uzlom. Heuristiky používané na kontrolu chovania algoritmu sú napr. Manhattanová vzdialenosť, Euklidová vzdialenosť alebo iné [10]. Každá má svoje výhody v iných situáciách. Heuristika určuje, ktorý uzol „najpravdepodobnejšie" vedie k cieľovému uzlu. Potom čo A* zistí ohodnotenie všetkých susedných uzlov, vloží ich do zoznamu open. Tam sú následne zoradené podľa ceny. V ďalšej iterácií je zo zoznamu open vybraný uzol sa najnižším ohodnotením ako aktuálny uzol.

Algoritmus končí ak uzol s najnižším ohodnotením v zozname open je cieľový uzol alebo ak trasa vedúca k cieľovému uzlu nebola nájdená.

2.2.2 Iné metódy

Algoritmy ako A* patria medzi informované metódy. Tieto metódy, využívajú informáciu o cieľovom stave pri prehľadávaní. Okrem týchto metód, existujú aj metódy neinformované. Tie naopak slepo prehľadávajú graf pretože nemajú žiadnu informáciu o cieľovom stave ani žiadne prostriedky ako aktuálne stavy vyhodnotiť.

Neinformované metódy sú nasledovné:

- **BFS - Breadth First Search** - metóda prehľadávania do šírky.
- **DFS - Depth First Search** - metóda prehľadávania do do hĺbky.
- **IDS - Iterative deepening DFS** - metóda postupného zanorovania.
- **UCS - Uniform cost search** - je metóda rovnakých cien.
- **BS - Bidirectional BFS search** - je metóda obojsmerného prehľadávania.

2.3 Plánovanie pohybu s obmedzeniami

Plánovacie techniky popísané v predošlých sekciách sú založené na predpoklade, že pohybujúci objekt môže instantne meniť svoj smer jazdy. Z hľadiska tejto práce sú pohybujúcimi objektami vozidlá, ktorých podvozky sú zhodné s podvozkom bežného automobilu. Vozidlá tak možno zaradiť medzi kolesové systémy kde kolesá rolujú v smere, v ktorom sú natočené. Existujú podvozky, ktorých kolesá sú dizajnované tak aby sa vozidlo vedelo pohybovať a natočiť do ľubovoľného smeru. V tomto prípade sú zadné kolesá fixné a predné ovládateľné. Toto obmedzuje vozidlo v pohybe do strán, pretože zadné kolesá by mali tendenciu sa šmýkať miesto rolovania [8]. Plánovanie sa tak stáva komplexnejším.

Ešte predtým než sa hlbšie ponoríme do plánovania pohybu, vysvetlíme si s tým súvisiace pojmy.

2.3.1 Stupne voľnosti

Rôzne objekty majú rôzne parametre, s ktorými môžu manipulovať. Maximálny počet nezávislých parametrov potrebných na charakteristiku transformácie objektu v priestore označujeme termínom *stupne voľnosti* [8]. Ako príklad predpokladajme, že sa objekt pohybuje v rovine XY, a že môže byť natočený nejakým uhlom na tejto rovine. Objekt môže nadobudnúť ľubovoľnú súradnicu x, súradnicu y a nejaký uhol natočenia θ. Jeho stupne voľnosti sú preto tri.

2.3.2 Konfiguračný priestor

Aj pri plánovaní pohybu musí byť prostredie nejakým spôsobom reprezentované. Okrem prostredia musí byť naviac reprezentované aj vozidlo. V danom momente môže vozidlo nadobudnúť kombináciu stupňov voľnosti. Jedna takáto kombinácia stupňov voľnosti referuje termínu *konfigurácia*. Konfiguračný priestor predstavuje množinu všetkých konfigurácií, ktoré môže vozidlo nadobudnúť. Dimenzia konfiguračného priestoru je daná počtom stupňov voľnosti.

2.3.3 Holonomický a neholonomický systém

Vo všeobecnosti možno rozdeliť typy systémov používaných pri plánovaní pohybu do dvoch kategórií. Na *holonomické* a *neholonomické*. Niektoré systémy dokážu kontrolovať všetky svoje stupne voľnosti nezávisle ako napr. robotické rameno. Tieto systémy sa označujú termínom *holonomické*. U takýchto systémov je plánovanie jednoduché. Stačí ak plánovač nájde nekolidujúcu trasu bez ohľadu na pohybové obmedzenia pomocou plánovacieho algoritmu ako je napríklad A*.

Druhý typ systémov, ktorých stupne voľnosti nie sú nezávislé je *neholonomický* systém. Typickým príkladom je znova auto. Vieme, že auto nemôže jazdiť do strán alebo natočiť do ľubovoľného smeru. Stupne voľnosti auta sú závislé na jeho rýchlosti a uhlu natočenia kolies. U takých systémov nájdenie trasy pomocou algoritmu A* ešte nemusí znamenať uskutočniteľnú trasu. [5]

2.4 Algoritmy pre plánovanie pohybu

Plánovanie pohybu je iná perspektíva na riešenie problému hľadania trasy. Plánovač pohybu nesleduje uzly ale pohybuje s vozidlom v priestore medzi pózami nazývané konfigurácie. Pri každom presune medzi konfiguráciami sa díva či nekoliduje s prekážkou a naviac zohľadňuje fyzické parametre podvozku. V tejto sekcii si predstavíme niektoré z vybraných algoritmov vhodných na plánovanie pohybu vozidla.

2.4.1 Car grid search

Páni Barraquand a Latombe vyvinuli jednoduchý plánovač Car grid search [3] pre pohybujúce sa vozidla v ohraničenom priestore. Originálne má plánovač nadefinovaných šesť akcií $L_{\pm}$, $P_{\pm}$, $S_{\pm}$. Symbol L predstavuje natočenie kolies úplne do ľavej strany, P úplne do pravej strany a S znamená jazdu rovno bez otočenia kolies. Úplne natočiť kolesá v tomto prípade znamená natočiť ich do maximálneho možného uhla do požadovanej strany. Označenie „+" a „-" indikuje smer jazdy vpred a vzad. Tieto akcie predstavujú pre plánovač možnosti, ktorými môže posunúť vozidlo v každom kroku. Pseudokód je uvedený v algoritme 1.

Plánovač si udržiava dve štruktúry, strom T a zoznam *OPEN*. V strome T si uchováva už dosiahnuté konfigurácie pri vyhľadávaní a v zozname *OPEN* sú uložené ukazatele na konfigurácie v strome T, ktorých následníci neboli vygenerovaní. Ukazatele v zozname *OPEN* sú zoradené podľa cenového ohodnotenia. Na začiatku vyhľadávania je do zoznamu *OPEN* a koreňa stromu T vložená počiatočná konfigurácia q_{start}. V každej iterácií plánovač vyberá konfiguráciu zo zoznamu *OPEN* s najnižším cenovým ohodnotením a odstráni ju z *OPEN*. Následníci vybranej konfigurácie sú generovaní posunutím sa o predom daný fixný krok vpred. Ak novo vygenerovaná konfigurácia nekoliduje so žiadnou prekážkou jej ukazateľ je vložený do zoradeného zoznamu *OPEN* a do stromu T kde je naviac uložený aj spätný ukazateľ na predošlú konfiguráciu. [3].

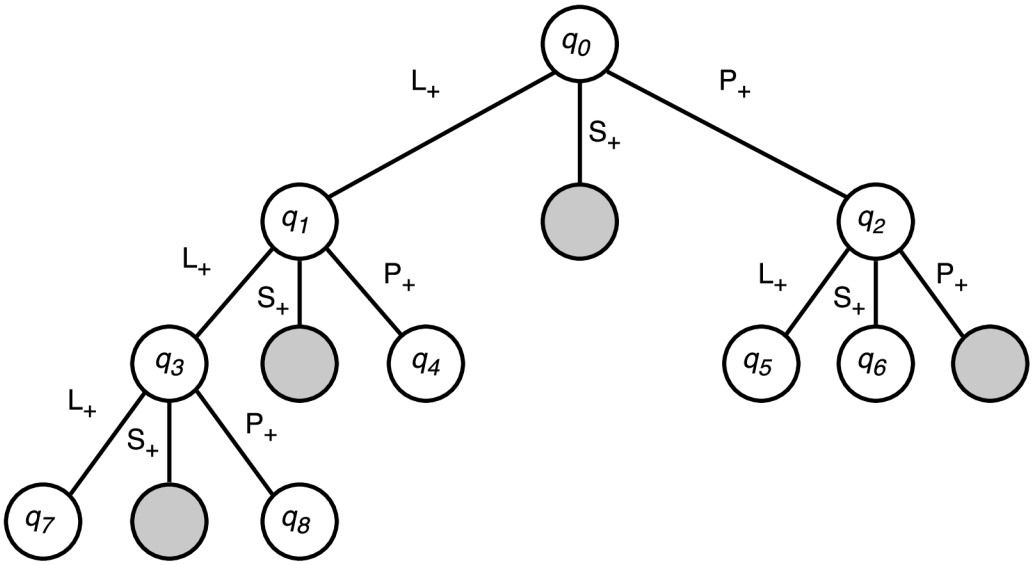
Algoritmus iteruje až dokým nenastane jedna z nasledujúcich podmienok:

- Zoznam *OPEN* je prázdny alebo počet záznamov v strome T dosiahol užívateľom zvolený limit *MAXTREESIZE* čo predstavuje, že plánovač nebol schopný nájsť postupnosť konfigurácií vedúcich do cieľového regiónu $G(q_{\text{goal}})$
- Plánovač úspešne dovŕšil konfiguráciu nachádzajúcu sa v cieľovom regióne $G(q_{\text{goal}})$

Algorithm 1 Car grid search

Input: Start configuration q_{start}, goal region $G(q_{\text{goal}})$
Output: A path from q_{start} to $G(q_{\text{goal}})$ or FAILURE

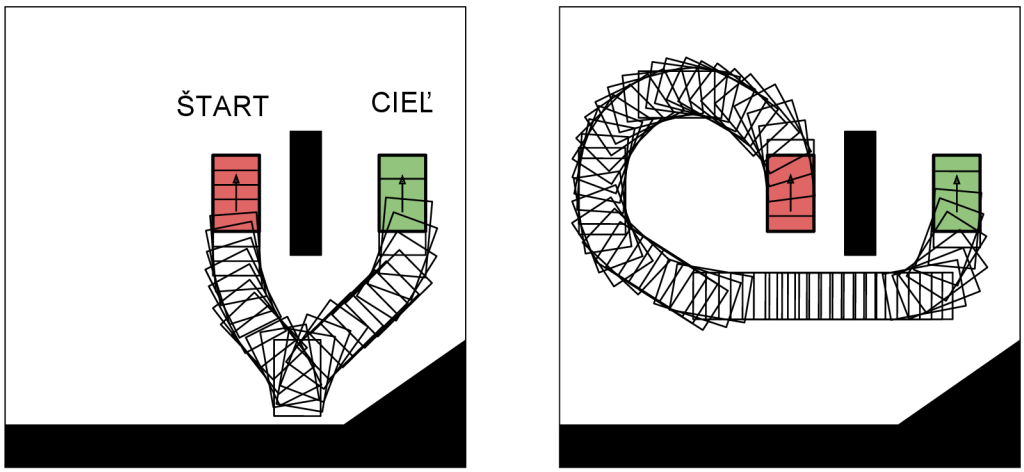
```
Initialize search tree T and list OPEN with start configuration q_start
while OPEN not empty and size(T) < MAXTREESIZE do
    q ← first in OPEN, remove from OPEN
    if q ∈ G(q_goal) then
        Report SUCCESS, return path
    end if
    if q is not near previously occupied configuration then
        Mark q occupied
        for all possible maneuvers do
            Integrate forward a fixed time to q_new
            if path to q_new is collision-free then
                Make q_new a successor to q in T
                Compute cost of path to new configuration q_new
                Place q_new in OPEN, sorted by cost
            end if
        end for
    end if
end while
```



Obr. 2.4: Príklad vyhľadávacieho stromu T. Uzly, ktorých trasa vedúca do konfigurácie koliduje s okolím sú vyplnené sivou farbou. Aby strom nebol príliš dlhý plánovač používa iba akcie L_+, P_+, S_+. [3].

Cenové ohodnotenie jednotlivých konfigurácií je funkciou počtu krokov od počiatočnej konfigurácie, počtu zmien natočenia kolies a počtu zmien smeru jazdy. Všetky tieto parametre majú priradenú váhu napr. a pre počet krokov, b pre počet zmien smeru kolies a c pre zmeny smeru jazdy. Výsledná cena konfigurácie je vážený súčet jednotlivých parametrov.

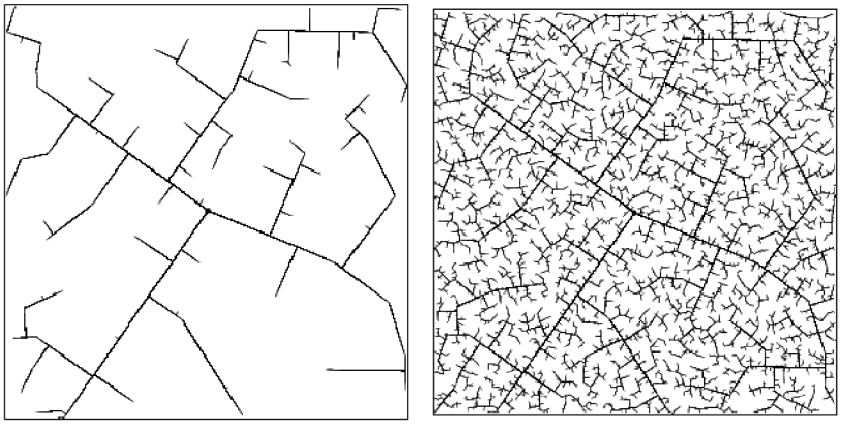
Pre $b = c = 0$ nájde plánovač najkratšiu cestu a pre $a = b = 0$ sa snaží minimalizovať počet zmien smeru jazdy. [3].



Obr. 2.5: Ukážka toho ako môže rovnaký problém vyriešiť plánovač použitím rôznych cenových ohodnotení. Na obrázku vľavo plánovač našiel najkratšiu cestu. Naproti tomu vpravo penalizuje zmeny smeru jazdy. Obrázky prevzaté z [3].

2.4.2 RRT - Rapidly random exploring trees

Ďalšia metóda používaná na plánovanie pohybu je metóda náhodných stromov (RRT). Táto metóda bola originálne dizajnovaná na plánovanie pohybu pod pohybovými obmedzeniami ako má napr. auto. Myšlienka náhodných stromov spočíva v tom, že sa v každej iterácií zväčšujú a graduálne tak pokrývajú konfiguračný priestor. Toto sa deje dovtedy, kým sa pri vyhľadávaní nenarazí na cieľovú konfiguráciu alebo bol dovŕšený počet krokov expanzie stromu. Vyhľadávanie môže byť jedno-stromové alebo viac-stromové. V prípade jedno-stromového vyhľadávania algoritmus začína expanziu stromu v počiatočnej konfigurácií postupne testuje či je možné ho prepojiť s cieľovou konfiguráciou. U viac-stromového riešenia, jeden strom narastá od počiatočnej konfigurácie a druhý od cieľovej konfigurácie. Vyhľadávanie končí úspešne ak sa podarí oba stromy prepojiť [8].



45 iterácií 2345 iterácií

Obr. 2.8: Ukážka pokrytia priestoru náhodného stromu RRT v rôznych iteráciách. Prevzaté z [8].

Metóda náhodných stromov je úplná s pravdepodobnosťou. Pravdepodobnosť, že RRT nájdu riešenie sa blíži k jednej ak sa počet iterácií blíži k nekonečnu. Na druhú stranu, sú neoptimálne. Existuje však upravená verzia RRT*, ktorá zaručuje optimálnosť.

2.5 Zhrnutie

Táto kapitola bola krátkym úvodom do problematiky plánovania. Prebrali sme metódy reprezentácie prostredia a prehľadávacie algoritmy primárne používané v hernom priemysle. Ako ďalšie sme pokryli aj komplexnejšie plánovanie so zohľadňovaním rôznych fyzických obmedzení, ktorých využitie je viac v robotike. Spôsobov na reprezentáciu prostredia a plánovacích algoritmov je okrem spomenutých mnoho. V tejto práci sa plánovanie do hĺbky nerozoberá a preto boli vybrané len niektoré.

Kapitola 3

Koordinovaný pohyb

Doposiaľ sme uvažovali pohyb len jedného vozidla. V tejto kapitole si priblížime pohyb skupiny vozidiel, ktorá je určitým spôsobom koordinovaná a udržuje si pohyb vo formácii. Pohyb vo formácii predstavuje pohyb v určitej organizovanej forme. Takýto typ pohybu je aplikovaný vo viacerých oblastiach. Používa sa v robotike kde viacero robotov rieši úlohu kooperatívnym spôsobom. Môže sa jednať o zberbu dát, monitorovanie zariadení, hliadkovanie apod. Ďalším odvetvím kde sú formácie uplatňované sú zložky ozbrojených síl. Pohyb vo formácii im totiž poskytuje taktický prístup v boji. V nasledujúcich sekciách sa pozrieme na to ako definovať a riadiť jednotky vo formácii.

3.1 Fixné formácie

Najjednoduchší typ formácií je ten, ktorý má pevne definovaný geometrický tvar. Každý člen formácie má definovanú relatívnu pozíciu voči referenčnému bodu. Referenčným bodom často býva líder skupiny ale môže to byť aj iný bod napr. ťažisko formácie. V prípade, že je to líder, tak ten sleduje trasu a zvyšní členovia nasledujú svoju relatívnu odchýlku voči nemu. Nie vždy je len pozícia dôležitým parametrom k udržaniu štruktúry formácie. V niektorých situáciách je vyžadované aby jednotky boli orientované v smere formácie. Princíp funguje rovnako. Líder predstavuje referenčnú orientáciu a zvyšné jednotky majú definovanú relatívnu orientáciu voči lídrovi. [6]

Každá dodatočná jednotka môže byť do formácie pridaná jednoduchým spôsobom. Ak vieme, že relatívna pozícia voči lídrovi je r_s, potom pozícia novo pridanej jednotky je

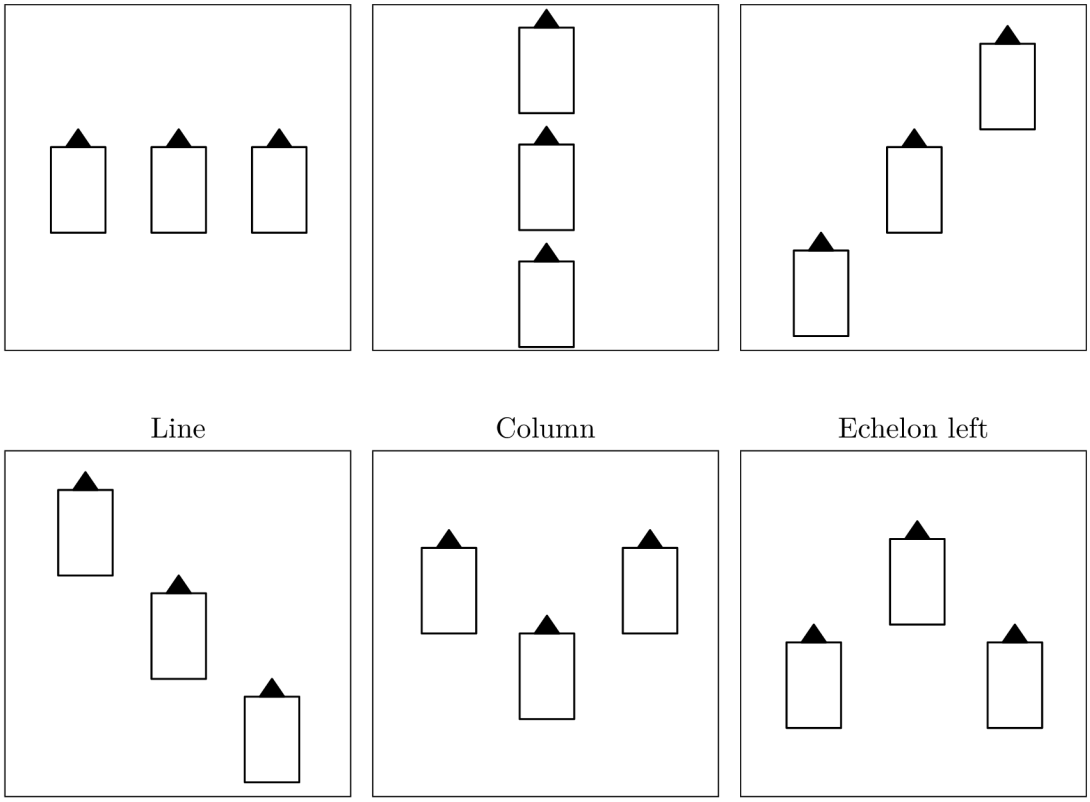
$$p_s = p_l + \theta_l \, r_s \tag{3.1}$$

kde p_s je výsledná pozícia vo formácií, p_l je pozícia lídra a θ_l je orientácia lídra. Podobne, definícia orientácie jednotky by bola

$$\omega = \omega_l + \omega_s \tag{3.2}$$

kde ω je finálna rotácia jednotky, ω_l predstavuje orientáciu lídra a ω_s relatívnu orientáciu jednotky voči lídrovi.

Štruktúra formácie môže byť rôzna. Ako inšpirácia sú na obrázku 3.7 znázornené najčastejšie vzory formácií používané v zložkách ozbrojených síl.



Obr. 3.7: Vzory formácií v ozbrojených silách.

3.2 Škálovateľnosť formácií

Z času na čas potrebujeme do formácie pripojiť jednotku. Štruktúra formácie potom závisí na počte jednotiek nachádzajúcich sa v nej. Príkladom môže byť tvar formácie defenzívneho kruhu kde jeho polomer je závislý od množstva jednotiek. Škálovateľné formácie umožňujú implementáciu bez predom stanoveného zoznamu relatívnych pozícií a orientácií jednotiek. Namiesto toho je pozícia jednotky počítaná dynamicky podľa celkového počtu jednotiek [6].

3.3 Riadenie formácie

Kombináciou geometrického vzoru formácie a riadiacich modulov jednotiek možno dosiahnuť požadovaný efekt pohybu vo formácii. Predtým sme si definovali geometrický tvar formácie relatívnymi odchýlkami v pozícií a orientácií. Ak by sme instantne menili pozíciu jednotky v každom kroku simulácie, malo by to nerealistický charakter. Tento problém možno obísť dvojitým riadením formácie. Miesto priameho nastavovania pozície a orientácie odošleme tieto parametre do riadiacich častí každej jednotky a sami sa musia zariadiť tak aby došli do cieľa. Jednotky môžu mať naviac definované ako sa správať pri vyhýbaní sa prekážkam.

Dvojité riadenie pozostáva z dvoch fáz. V prvej veliteľ skupiny riadi vzor formácie. Tým, že sa mení vzor menia sa aj pozície podriadených jednotiek. V druhej fáze sa preto ostatné

jednotky snažia dodržiavať vzor.

S týmto prístupom sa viaže aj jeden problém. Predpokladajme, že sa líder vyhýba prekážkam. Keďže líder riadi vzor formácie, celá skupina musí zmeniť smer pohybu aj keď to nemusí byť nutné. Naviac, po vyhnutí sa lídra prekážke môžu neskôr iné jednotky kolidovať touto prekážkou a budú sa jej snažiť vyhnúť čo spôsobí ešte väčší chaos. Takéto chovanie je neprirodzené, pretože jednotky napodobňujú správanie veliteľa pričom sú schopné sa vyhnúť prekážke vlastným spôsobom.

Riešením tohto problému je odstránenie väzby medzi lídrom a podriadenými jednotkami. Skutočný líder je nahradený mysleným, neviditeľným veliteľom, ktorý sa nemusí starať o vyhýbanie prekážkam. Vzor formácie je teraz ovládaný mysleným lídrom. Riadenie formácie sa tak odovzdáva do rúk riadiacich modulov všetkých jednotiek (druhá fáza v dvojitom riadení). Avšak, ak imaginárny líder prejde priamo cez prekážku, ostatní členovia už nemusia byť schopný nasledovať svoje pozície vo formácií. [6]

Rýchlosť formácie Pri vyhýbaní sa prekážkam je treba zohľadniť fakt, že ak bude prekážka príliš veľká, môže niektorým jednotkám dlho trvať kým obídu prekážku a nebudú sa stíhať zaradiť do formácie čím sa na nejakú chvíľu celá formácia desynchronizuje. Spomalením rýchlosti možno predísť tomuto problému. Rýchlosť formácie môže byť fixná a prispôsobená tak aby referenčný bod nepredbiehal jej jednotky napr. na polovičnú rýchlosť jednotiek. Lepším riešením je adaptívne prispôsobovať rýchlosť formácie v závislosti na tom ako jej jednotky majú problémy dosiahnuť požadovanú pozíciu v útvare.

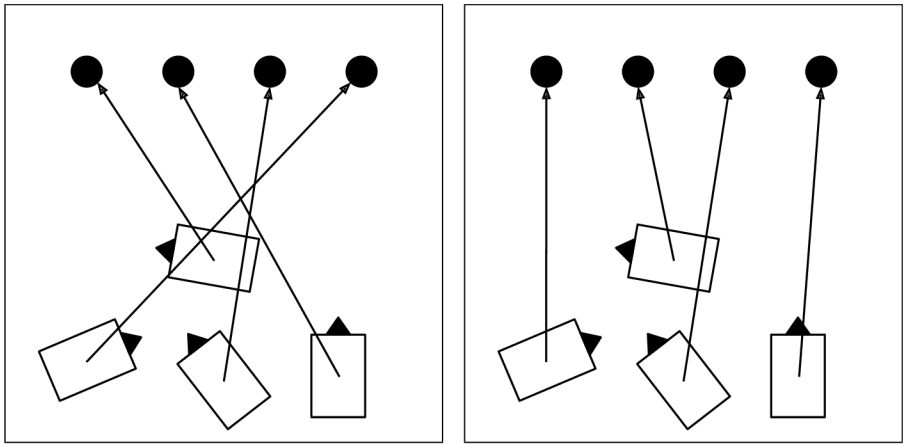
Mobilita jednotiek Dosť často nastane prípad kde sa viacero jednotiek pokúša o rovnakú činnosť. Tým sa ich trasy môžu prekrížiť čo vedie ku kolízií medzi vozidlami. Tento prípad typicky nastáva na začiatku sledovania trasy. Jednotky sú rozmiestnené v priestore a potrebujú sa čo najskôr zaradiť do formácie. To ako sa ich cesty krížia ovplyvňuje čas ako rýchlo sú toto schopné dokázať. Aby sa predišlo týmto komplikáciám, treba zvoliť inú formáciu redukujúcu počet krížiacich sa tras alebo inú stratégiu ako napr. priradiť každej jednotke najbližšiu pozíciu vo formácií. Situácia je znázornená na obrázkoch 3.10 (a) a (b) [13].

3.4 Formácie na báze chovaní

Princíp formácie na báze chovaní (angl. behavior-based) je založený na tom, že každá jednotka má nadefinovanú nejakú množinu základných riadiacich modulov. Výstup každého modulu je chovanie. Môže to byť sledovanie trajektórie, udržovanie vo formácií, vyhýbanie sa prekážkam a iným jednotkám, dopravenie sa do cieľovej pozície apod. Kombináciou týchto chovaní je ovplyvňovaný pohyb jednotky.

Všetky chovania bežia paralelne a preto majú pridelenú váhu. Váhou udávame dôležitosť jednotlivých chovaní. Ak má jednotka definované chovania napr. „vyhýbaj sa prekážkam" a „dôjdi do cieľovej pozície" a nastal by prípad, že sa jednotka blíži k prekážke, chovanie vyhnutiu sa prekážke môže mať väčšiu prioritu čím zabezpečíme, že jednotka nenarazí do prekážky.

Prístup na báze chovaní poskytuje možnosť formácií operovať autonómne v nepredvídateľných situáciách. Preto je vhodný do dynamických alebo neznámych prostredí. [7]



(a) (b)

Obr. 3.10: Ukážka toho ako zmena stratégie voľby umiestnenia jednotiek vo formácií môže obmedziť kolízie medzi nimi. Na ľavom obrázku majú jednotky umiestnenie pevne dané po celú dobu. Na pravej strane sú im dynamicky prideľované podľa najkratšej vzdialenosti.

3.5 Sledovanie veliteľa

Systém nasledovania veliteľa (angl. leader-following) je v podobný systému popísaného v sekcií 3.3. Jedna z jednotiek je považovaná za lídra skupiny a ostatní jeho nasledovatelia. Líder riadi formáciu ako celok tzn. sleduje trasu, vyhýba sa prekážkam apod. a jeho podriadené jednotky sledujú predom definované relatívne pozície voči nemu. Nevýhoda tejto metódy je, že líder sa pohybuje nezávisle na ostatných jednotkách. Inými slovami, ak nejaká z jednotiek zaostáva alebo nie je schopná si udržať svoju pozíciu vo formácií, líder nespomalí. Toto je treba riešiť spôsobom popísaným v 3.3. [7]

3.6 Virtuálna štruktúra

Ďalšia z metód riadenia formácie je prístup pomocou virtuálnej štruktúry (angl. virtual-structure approach). V rámci virtuálnej štruktúry sa formácia považuje za tuhé teleso. Virtuálna štruktúra obsahuje množinu umiestnení kde sa majú jednotky nachádzať. Tie sú dané relatívnymi odchýlkami v lokálnom súradnicovom priestore virtuálnej štruktúry. Štruktúra sa pohybuje po trase v nezmenenom tvare s určitou rýchlosťou a úlohou všetkých jednotiek je sledovať si svoju relatívnu odchýlku. Riadenie formácie tak spočíva v riadení virtuálnej štruktúry.

Výhodou tohto prístupu je že jednoduchým spôsobom umožňuje dynamicky v čase prekonfigurovať tvar formácie. [7]

3.7 Zhrnutie

V tejto kapitole sme si všeobecne objasnili myšlienku koordinovaného pohybu. Popísali sme si ako definovať formáciu, ako ju riadiť, aké typy riadenia formácii sa používajú a taktiež zopár vzorov formácii pre ilustráciu.

Kapitola 4

Návrh riešenia

V tejto kapitole si priblížime vlastný návrh na riešenie problematiky koordinovaného pohybu neholonomických vozidiel. V prevej časti sa zoznámime s existujúcimi riešeniami a popíšeme si ich prípadné nedostatky a potom návrh jednotlivých častí systému.

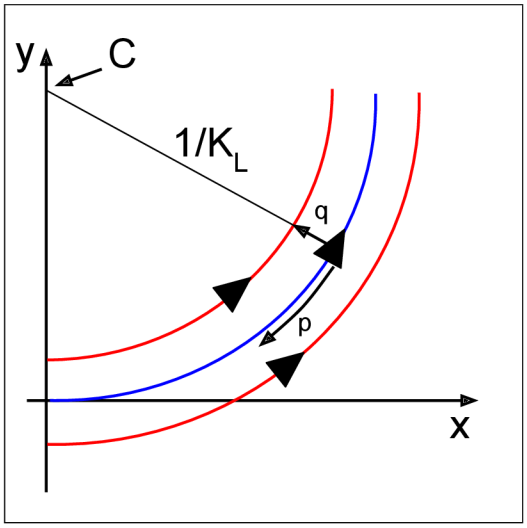
4.1 Existujúce riešenia

Ako prvý sa rozhodol implementovať systém koordinovaného pohybu Dave Pottinger. Vo svojich článkoch [11] a [12] detailne popisuje svoj prístup k riešeniu problému riadenia skupiny jednotiek. Zameral sa predovšetkým na strategické hry v reálnom čase (angl. real-time strategy games) kde pohyb vo skupine nesmie chýbať. Riešil takmer všetky problémy, s ktorými sa možno stretnúť pri riadení formácie ako prispôsobovanie rýchlosti, schopnosť udržovať si formáciu, vyhýbanie sa prekážkam, prechod úzkym miestom apod. Avšak, jeho riešenie je prispôsobené na pohyb postáv a nie vozidiel.

Páni Krontiris, Louis a Bekris [2] priniesli iný spôsob riadenia formácie. Ich riešenie spočíva v použití krivočiarej sústavy súradníc miesto klasickej priamočiarej (karteziánskej) sústavy. Reprezentácia formácie v krivočiarom súradnicovom systéme je znázornená na obrázku 4.1. Os x krivočiareho súradnicového systému reprezentuje trajektória veliteľa. Na obrázku 4.1 je zvýraznená modrou farbou. Veliteľ je referenčný objekt voči, ktorému je formácia definovaná. Zvyšné jednotky si udržujú svoju odchýlku p_i pozdĺž zakrivenej trajektórie veliteľa a odchýlku q_i kolmú na trasu veliteľa.

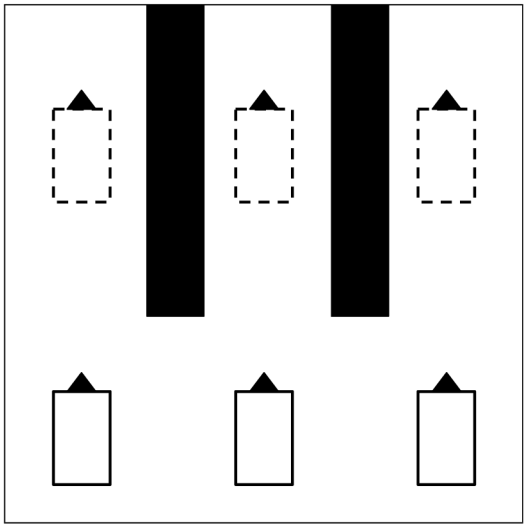
Ich práca bola navrhnutá pre simulátory s vojenskou tématikou. Podporuje aj zmenu vzoru formácie v čase ale neuvažuje s prekážkami a preto je použiteľná pri formáciách pohybujúcich sa v bezprekážkovom prostredí ako napr. formácia stíhačiek alebo lodí. Na podobnom princípe funguje aj práca [14] s rozdielom, že je aplikovaná na pozemných robotoch. Podobne, ani tam sa neriešia prekážky.

O ďalší rozvoj koordinovaného pohybu sa pokúsili autori práce [1]. V ich práci sa rieši aj prechod formácie úzkou prekážkou. K riešeniu pristúpili jednoducho. Na krídelných stranách formácie majú umiestnené dva senzory detekujúce prekážku vrhaním lúča vopred formáciu. Ak oba senzory v rovnakom čase detekujú prekážku, považuje sa to za úzku pasáž a formácia zmení vzor *Column* tak ako je to uvedené na obrázku 3.7. Každá jednotka má naviac definované chovanie ako sa samostatne vyhýbať prekážkam v prípade, že senzor detekuje blížiacu sa prekážku len z jednej strany. Tento prístup na detekciu úzkej pasáže má ale jeden problém. Ak by boli rozostupy medzi jednotkami veľké a ak uvažujeme, že šírka prekážky by



Obr. 4.1: Reprezentácia formácie v krivočiarom súradnicovom systéme.

bola menšia než tieto rozostupy, ani jeden z krídelných senzorov by nedetekoval prekážku. Pre formáciu by to ešte nemusel byť kritický prípad, že vozidlá narazia do prekážky. Problémom je, že formácia by sa mohla rozbiť. Prípad je znázornený na obrázku 4.2. Vozidlá prechádzajú okolo prekážok v nezmenenom vzore. Toto robí dojem, že vozidlá nepracujú ako skupina. Očakávaný efekt by bol zmena na vzor *Column*.



Obr. 4.2: Situácia kedy metóda detekcie úzkej pasáže v [1] môže zlyhať.

Práci na riadenie koordinovaného pohybu je mnoho, obzvlášť v robotike. V tejto oblasti to má veľmi dobre zhodnotené vo svojej práci [7] Kiattisin Kanjanawanishkul. Pre každý typ formácií t.j. formácie na báze chovania, sledovanie veliteľa aj virtuálnu štruktúru má množstvo odkazov na rôzne riešenia.

4.2 Plánovanie pohybu

Na plánovanie pohybu bude použitý algoritmus Car grid search. Jeho princíp je popísaný v sekcii 2.4.1. Účelom plánovacieho algoritmu bude nájsť postupnosť konfigurácií vedúcich do cieľa. Túto postupnosť si možno predstaviť ako trajektóriu. Nájdená trajektória bude neskôr slúžiť ako navigácia formácie k cieľu.

Vstupnými parametrami do plánovača budú počiatočná pozícia, počiatočná orientácia a konečná pozícia. Z týchto parametrov si plánovač vyrobí iniciálnu konfiguráciu, ktorú vloží do zoznamu *OPEN* a stromu T. Konfiguračný strom T bude implementovaný ako pole kde prvky pola tvoria uzly konfigurácií. Každý uzol si bude uchovávať index v poli na svojho predchodcu. U každej konfigurácie bude uvedená pozícia, orientácia a uhol natočenia volantu.

4.2.1 Zoznam OPEN

Jadrom plánovača je hlavný cyklus. V každej iterácií sa zoznamu *OPEN* vyberá prvok s najnižším cenovým ohodnotením. Algoritmus si tak vyžaduje aby bol zoznam *OPEN* zoradený. To, akou štruktúrou bude implementovaný ovplyvňuje výkon algoritmu. Zvyčajne sa v takých prípadoch používa špeciálna štruktúra nazývaná prioritná fronta. Jej použitím sa minimalizuje čas na nájdenie najlepšie ohodnoteného prvku. Môže byť implementovaná viacerými spôsobmi. Naivná implementácia je udržovať si nezoradený zoznam a vždy pri operácii výberu z fronty prejsť cez celý zoznam a nájsť prvok s minimálnym ohodnotením. Ďalšia možnosť je použitie nejakého zoraďovacieho algoritmu. Za každým pridaním prvku sa zoznam zoradí. V jazyku C# je trieda reprezentujúca zoznam *List*. Pre zoradenie prvkov má definovanú metódu Sort(). V dokumentácií funkcie sa uvádza, že v najhoršom prípade má metóda zložitosť $O(n^2)$ kde n je počet prvkov v zozname.

Vhodnou alternatívou k predchádzajúcim dvom možnostiam je Binary heap pretože má lepšie vlastnosti. Binary heap si udržuje svoje prvky v stromovej štruktúre. Každý prvok môže mať maximálne dvoch potomkov. Štruktúra zároveň umožnuje implementovať strom ako pole kde index ľavého potomka $2i$ a index pravého potomka $2i+1$ kde i je index rodiča. Pridávanie aj odoberanie prvku má zložitosť $O(\log n)$.

Metóda	**Vkládanie prvku**	**Odstránenie prvku**	**Výber najlepšieho prvku**
Naivna implementácia	$O(1)$	$O(n)$	$O(n)$
List.Sort()	$O(1)$	$O(n)$	$O(n^2)$
Binary Heap	$O(\log n)$	$O(\log n)$	$O(1)$

Tabuľka 4.1: Porovnanie zložitostí vyššie spomenutých metód na implementáciu prioritnej fronty. Zložitosti sú uvedené pre najhorší prípad.

Podľa tabuľky 4.1 je vidieť, že najvhodnejšia štruktúra na implementáciu prioritnej fronty je Binary heap a preto bude použitá.

4.2.2 Princíp výberu najlepšej konfigurácie

Potom čo plánovač vyberie konfiguráciu s najnižšou cenou pokúsi sa vykonať všetky manévre zo zoznamu manévrov. Zoznam manévrov bude udávať užívateľ a každý manéver

bude udávať mieru natočenia kolies vozidla. Manéver bude číslo v rozsahu $[-1, 1]$ kde -1 je úplné natočenie kolies do ľavej strany a 1 úplne do pravej strany. Manéver pohybu do zadu pre jednoduchosť nebude uvažovaný. Užívateľ takisto definuje maximálny uhol natočenia kolies vozidla, s ktorým bude plánovač pracovať. Vynásobením manévru a maximálneho uhla natočenia kolies sa získa uhol natočenia kolies pri danom manévri čo sa deje v pri riadení vozidla. Manéver je potom odoslaný do riadiacich častí vozidla kde sa spočíta nová konfigurácia pri posunutí o fixný krok, ktorý bude tiež vopred definovaný užívateľom. Nová konfigurácia sa bude počítať presne tak ako je to uvedené v sekcií 4.3.2. Ak nová konfigurácia koliduje s okolím, plánovač ju zahodí.

U všetkých novo vygenerovaných nekolidujúcich konfigurácií sa testuje či sa nachádzajú v cieľovom regióne. V prípade, že áno, plánovač vráti plán, ktorý získa prechodom po spätných ukazateľoch na predchádzajúce konfigurácie až po počiatočnú konfiguráciu. Inač, sa spočítajú ich príslušné cenové ohodnotenia a vložia do zoznamu *OPEN*. Cieľový región bude reprezentovaný kružnicou a jeho polomer bude zadaný užívateľom. Pôvodný algoritmus plánovača síce má informáciu o cieľovom regióne ale nijak ju pri určovaní ceny nepoužíva. Použitím vzdialenosti medzi konfiguráciou a cieľovým bodom môže plánovač heuristicky vyberať zo zoznamu *OPEN* konfiguráciu, ktorá najpravdepodobnejšie vedie k cieľu a obmedziť tak prudký nárast konfiguračného stromu T.

Pri výpočte ceny nemôžeme miešať dohromady vzdialenosť spolu s počtom krokov a zmien volantu pretože by nesedeli mierky. Preto zoraďovanie podľa ceny bude prebiehať na dvoch úrovniach. Prvou úrovňou bude radenie podľa počtu krokov a zmien volantu $f(k_j)$:

$$f(k_j) = a\,(m_i + 1) + b\,|s_i - s_j| \tag{4.1}$$

kde index i označuje je konfiguráciu odkiaľ vozidlo prišlo a index j novo vygenerovanú konfiguráciu. Hodnota m_i udáva počet konfigurácií, ktoré vozidlo prešlo. Hodnoty s_i a s_j sú stavy natočenia volantu v konfiguráciách k_i a k_j. Parametre a,b sú mierou penalizácie počtu krokov (konfigurácií).

Ak budú mať dva konfigurácie zhodné ohodnotenie prvej úrovne, zoradia sa podľa vzdialenosti k cieľu $g(k)$. Prednosť má tá konfigurácia, ktorá je bližšie k cieľu.

$$g(k) = ||P - G|| \tag{4.2}$$

kde P označuje vektor pozície konfigurácie k a G je vektor cieľovej pozície.

4.2.3 Detekcia kolízie

Vozidlo sa pohybuje buď po oblúkoch alebo po priamke. Problém detekcie kolízie s prekážkou sa stáva problémom prieseku priamky alebo oblúka s prekážkou. V tejto implementácií bude použitý jednoduchší prístup a to obklopením vozidla kružnicou s polomerom daným užívateľom. Polomer by mal byť zvolený tak aby umožniteľný plán bol nájdený, ale umožniteľná cesta cez tesné miesto nie. Test na kolíziu bude realizovaný ťahaním kružnice od konfigurácie vybranej na začiatku iterácie smerom k novej konfigurácii po úsečke.

4.2.4 Optimalizácia plánovača

Pretože veľkosť konfiguračného stromu exponenciálne rastie s expandovaním každej konfigurácie, mohol by byť príliš veľký ak by sa cieľová konfigurácia nachádzala veľmi ďaleko. Ako optimalizácia bude plánovač pohybu skombinovaný s plánovacím algoritmom A*. Najprv A* nájde trasu bez neholonomických obmedzení. V druhom kroku medzi každými dvoma

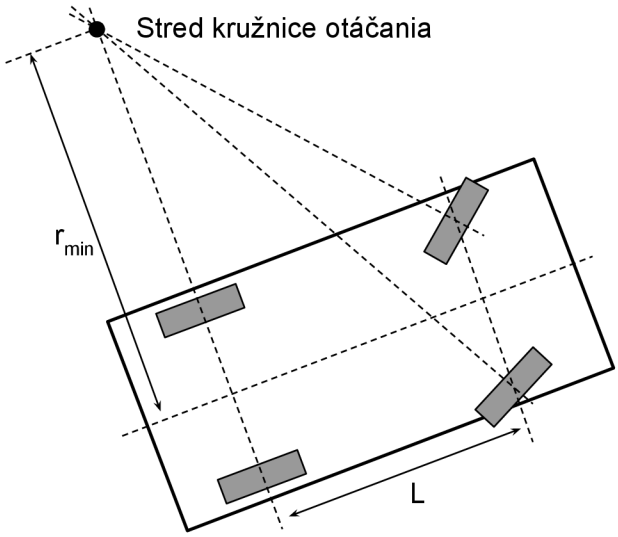
bodmi nájde plánovač pohybu uskutočniteľný plán. Tým, že plánovač hľadá trasu medzi bodmi trasy z A* sa zabezpečuje blízkosť cieľového bodu čím sa proces plánovania urýchli. Finálny plán vedúci k cieľu vznikne spojením všetkých plánov z druhého kroku.

4.3 Riadenie vozidla

V tejto sekcii bude popísané aký typ vozidla budeme vo zvyšku práce uvažovať ako aj jeho riadiace časti. Princíp bude vysvetlený na jednom vozidle a neskôr bude aplikovaný na všetkých vozidlách tvoriacich formáciu.

4.3.1 Použitý kinematický model vozidla

V tejto práci sa budeme sústrediť na pohyb neholonomických vozidiel. Konkrétne to budú vozidlá s Ackermannovým podvozkom čo je podobný typ podvozku ako majú dnešné osobné autá. Model takého podvozku je znázornený na obrázku 4.3.



Obr. 4.3: Kinematický model ackermannového podvozku [3]

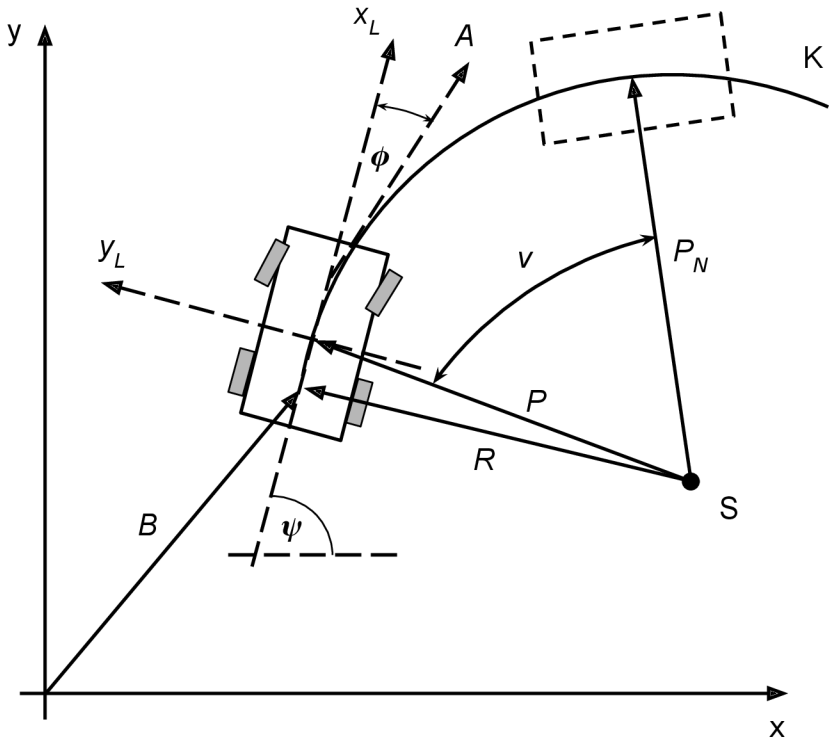
Obvykle u takých typov vozidiel sú ovládateľné len predné kolesá a zadné fixné. Predné kolesá sú naviac obmedzené maximálnym uhlom natočenia čo zároveň obmedzuje minimálny polomer otočnej kružnice $r_{\min}$, ktorú je schopné auto opísať keď ide do zatáčky. Aby kolesá nemali tendenciu sa šmýkať pri manévri do zatáčky, musí rotačná os každého kolesa prechádzať stredom otočnej kružnice. Preto oba predné kolesá majú rôzny uhol natočenia. Dôležitým parametrom udávaným aj výrobcami áut je rázvor kolies čo je vzdialenosť rotačných osí predných a zadných kolies. Na obrázku 4.3 je označený symbolom L.

Pri pohybe vozidlom nás bude zaujímať len pozícia a orientácia. Fyzikálne veličiny vplývajúce na pohyb ako trakčná sila, moment zotrvačnosti, trenie kolies nebudú brané v úvahu. Model tak možno zjednodušiť. Namiesto štyroch kolies budeme uvažovať dve „virtuálne" kolesá. Jedno sa bude nachádzať v strede rotačnej osi zadných kolies v pôvodnom štvorkolesovom modeli a jedno v strede osi predných kolies. Minimálny polomer otočnej kružnice $r_{\min}$ sa nemení.

4.3.2 Mechanika vozidla

Vozidlo sa môže pohybovať dvoma spôsobmi. Môže to byť priamočiary pohyb alebo zakrivený. Pri priamočiarom je pohyb jednoduchý. Vozidlo má danú rýchlosť a tou sa pohybuje po priamke. Trocha komplexnejší prípad je krivočiary pohyb kedy sa vozidlo pohybuje po kružnici. Obrázok 4.4 zobrazuje pohyb vozidla po kružnici spolu s veličinami potrebnými pre výpočet pozície vozidla na kružnici.

Pre lepšiu orientáciu pri výpočte bude smer vopred vozidla zarovnaný s osou x. Os y bude vždy kolmá na os smerujúcu vopred vozidlo. Zároveň smer osi y bude vždy na ľavo od vozidla.



Obr. 4.4: Pohyb vozidla po kružnici.

Na obrázku 4.4 osi x a y predstavujú globálne osi súradnicového systému. Osi lokálneho súradnicového systému vozidla sú označené x_L a y_L. Vektor A je smer natočenia kolies. Uhol medzi vektorom orientovaným vopred vozidla x_L a vektorom A je uhol natočenia kolies. Vozidlo je umiestnené na nejakej pozícií a natočené o uhol ψ v rámci globálnych súradníc. Cieľom je vypočítať pozíciu na kružnici K kde sa bude vozidlo nachádzať ak jeho rýchlosť je známa.

Polomer opisujúcej kružnice K sa riadi uhlom natočenia kolies. Získame ho pomocou nasledujúcej rovnice:

$$r = \frac{L}{\tan \phi} \tag{4.3}$$

kde L je rázvor kolies a $tan\ \phi$ je tangens uhla natočenia kolies.

Ďalší parameter nevyhnutný k výpočtu je umiestnenie stredu rotačnej osy zadných kolies. K tomu je treba vedieť ako ďaleko zadné kolesá od stredu vozidla. Táto odchýlka bude

udávaná užívateľom. Odčítaním odchýlky k zadným kolesám od pozície vozidla dostávame vektor B udržujúci pozíciu stredu osy zadných kolies. Keďže poznáme polomer sme schopní určiť vektor R. Smer tohto vektora bude ukazovať vždy od stredu kružnice do stredu osi zadných kolies a zároveň bude zarovnaný s lokálnou osou y_L. Jeho veľkosť bude rovná polomeru kružnice K získaného z rovnice 4.3. Odčítaním vektora R od B dostávame stred S opisujúcej kružnice.

$$S = B - R \tag{4.4}$$

Pomocou stredu kružnice si určíme vektor P. Ten bude smerovať od stredu S do ústredného bodu vozidla (angl. pivot point). V tomto prípade je ústredným bodom stred vozidla. Rýchlosť vozidla poznáme a vieme určiť uhol v o koľko sa má vozidlo posunúť na kružnici K. Získame ho cez nasledujúcu rovnicu:

$$v = \frac{s}{||P||} \tag{4.5}$$

kde s je rýchlosť vozidla a $||P||$ je dĺžka vektora P. Výsledný uhol v je v radiánoch a je treba ho konvertovať na stupne. Natočením vektora P o uhol v sa získa vektor P_N smerujúci od stredu k novej pozícii na kružnici K. Je to však iba smerový vektor a preto k dosiahnutiu novej pozície je treba pričítať vektor P_N k stredu kružnice K.

4.3.3 Ovládanie vozidla

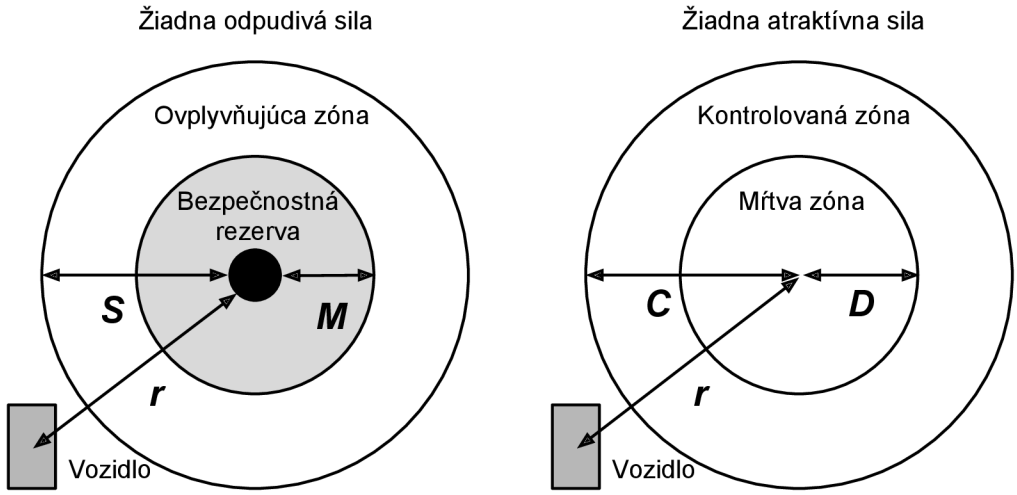
Myšlienka riadenia vozidla bude založená na Balchovej metóde [15] navigácie vozidiel. Každé vozidlo bude mať sadu modulov chovaní. Každý modul predstavuje jednu úlohu, ako napr. vyhnutie sa prekážkam alebo nasmerovanie k cieľu. Výstupom každého modulu bude vektor indikujúci, ktorým smerom by sa malo vozidlo vydať. Dĺžka vektora určuje dôležitosť prideleného modulu, ktorá sa bude v čase meniť. Každý vektor bude mať pridelenú váhu. Výsledný vektor kde sa vozidlo skutočne pohne je získaný váženým súčtom vektorov zo všetkých modulov.

Na navigáciu budú použité moduly chovania vyhýbanie sa prekážkam, vyhýbanie sa iným vozidlám, udržanie sa vo formácii.

Vyhýbanie sa prekážkam Modul pre vyhýbanie sa prekážkam bude vytvárať odpudzujúcu silu od všetkých prekážok v okolí vozidla. Okolie bude skenované kružnicou s polomerom daným užívateľom. Veľkosť sily bude závisieť na vzdialenosti vozidla od prekážky. Pokiaľ je vozidlo mimo ovplyvňujúcej zóny danú vzdialenosťou od prekážky S, ako je zobrazené na obrázku 4.5 vľavo, nebude generovaná žiadna odpudivá sila. Keď bude vzdialenosť vozidla r od prekážky menej ako S bude repulzívna sila lineárne zväčšovať svoju veľkosť až pokým vozidlo nedôjde po vzdialenosť bezpečnostnej medzery M. Pri vzdialenosti r menšej ako M by mala teoreticky byť sila nekonečne veľká. V tejto implementácií jej veľkosť bude po prekročení bezpečnostnej medzery rovná jednej. Nasledujúce vzťahy udávajú veľkosť a smer repulzívnej sily R v závislosti na vzdialenosti vozidla od prekážky r.

$$R_{veľkosť} = \begin{cases} 0 & pre \quad r > S \\ \frac{S-r}{S-M} & pre \quad M < r \leq S \\ 1 & pre \quad r \leq M \end{cases} \tag{4.6}$$

$$R_{smer} = O - V \tag{4.7}$$



Obr. 4.5: Na ľavom obrázku sú zobrazené parametre použité pri výpočte repulzívnej sily od prekážky, ktorá je vyplnená čiernou oblasťou. Na pravej strane sú parametre použité na výpočet atraktívnej sily k cieľu. Obrázky prevzaté z [15].

V rovnici 4.7 je vektor O zastupujúci pozíciu prekážky a V je vektor označujúci pozíciu vozidla. Výsledná odpudzujúca sila je daná súčtom individuálnych repulzívnych síl od všetkých prekážok.

Vyhýbanie sa iným vozidlám Vyhýbanie sa iným vozidlám bude prebiehať rovnakým štýlom ako vyhýbanie sa prekážkam s rozdielom, že odpudzujúca sila bude generovaná od vozidiel. Keďže odpudivá sila sa generuje v závislosti na tom či je v okolí iné vozidlo, mohla by byť sila aplikovaná aj v prípade, že vozidlu nič neprekáža v ceste. Tento prípad nastane vtedy ak sa iné vozidlo blíži zo zadu. Pre opatrenie tohto prípadu bude repulzívna sila od iného vozidla uplatnená len vtedy ak sa iné vozidlo nachádza v smere jazdy vozidla, na ktoré má byť repulzívna sila aplikovaná.

Udržanie pozície vo formácii Tento typ chovania bude pracovať na rovnakom princípe ako vyhýbanie sa prekážkam pričom nebude generovať repulzívnu silu ale atraktívnu silu. Vozidlo bude dostávať od riadenia formácie pozíciu kde sa má vo formácii nachádzať. Opäť, budú definované tri zóny a veľkosť atraktívnej sily bude závisieť na vzdialenosti vozidla r od pozície vo formácii. Schéma zón je znázornená na obrázku 4.5. Pokiaľ je vozidlo mimo kontrolovanej zóny, ktorej polomer je C, bude veľkosť atraktívnej sily rovná jednej. Po prekročení vzdialenosti r pod hranicu C bude veľkosť sily lineárne klesať k hranici mŕtvej zóny. Atraktívna sila má veľkosť rovnú nule ak sa vozidlo nachádza v mŕtvej zóne. Veľkosť a smer vektora atraktívnej sily A sa počíta pomocou nasledujúcich vzťahov kde r označuje vzdialenosť vozidla od cieľovej pozície

$$A_{veľkosť} = \begin{cases} 1 & pre \quad r > C \\ \frac{r-D}{C-D} & pre \quad D < r \leq C \\ 0 & pre \quad r \leq D \end{cases} \tag{4.8}$$

$$A_{smer} = G - V \tag{4.9}$$

V rovnici 4.9 symbol G je vektor označujúci cieľovú pozíciu a V je vektor určujúci pozíciu vozidla.

Po obdržaní síl zo všetkých modulov je následne ich váženým súčtom vypočítaná výsledná sila. Veľkosť výslednej sily by mohol byť v istých situáciách približne rovný nule. Nastáva to v prípade, že sa atraktívna sila k cieľu a repulzívna od prekážky navzájom vykrátia. Na elimináciu tohoto prípadu sa repulzívna sila od prekážky sčíta s jej kolmým vektorom. Normálové vektory sú však dva a jeden z nich bude vyberaný heuristicky. Heuristikou bude skalárny súčin normálového vektora s vektorom smerujúcim k cieľu. Ten normálový vektor, ktorého skalárny súčin je kladný bude použitý do súčtu s repulzívnou silou od prekážky.

Navigácia vozidla Po spočítaní výslednej sily je známe, ktorým smerom sa má vozidlo vydať. Tento smer sa predá do navigačnej časti vozidla. Úlohou navigačnej časti bude prispôsobovať rýchlosť, smer jazdy a natočenie kolies tak aby sa vozidlo šlo do požadovaného smeru.

Prvé čo potrebuje vozidlo zistiť je smer jazdy. Na determináciu smeru jazdy sa vektor požadovaného smeru transformuje do lokálneho súradnicového systému vozidla. Transformovaný vektor si označíme písmenom L. V sekcii 4.3.2 bolo definované, že vektor smerujúci vopred vozidla bude vždy zarovnaný s kladným smerom lokálnej osi x. To znamená, že ak súradnica lokálneho vektora L bude kladná požadovaný smer je pred vozidlom. Vozidlo sa v takom prípade bude pohybovať dopredu. Záporný smer bude značiť, že smer, opačný k orientácii vozidla a vozidlo sa bude pohybovať do zadu.

Smer natočenia kolies Ďalšie, čo musí vozidlo zistiť je do ktorej strany má natočiť kolesá. Na toto bude využitá súradnica y transformovaného vektora L. Opäť, podľa definície v sekcii 4.3.2 je lokálna os y vozidla kolmá na os x a ukazuje vždy na ľavo od vozidla. Podľa toho možno určiť stranu natočenia kolies. Kladná súradnica y vektora L bude znamenať, že vozidlo má natočiť kolesa doľava. V opačnom prípade ich natočí doprava. Takto sa bude natočenie kolies riadiť v prípade, že požadovaný smer je pred vozidlom. Pri cúvaní vozidla to bude presne naopak. Kladná súradnica L bude znamenať natočenie doprava a záporná doľava. Tým, že vozidlo cúva do opačnej strany sa zabezpečí to, že sa v istom momente požadovaný smer jazdy ocitne pred vozidlom. V tomto zlomovom momente začne vozidlo ísť dopredu. Inač by vozidlo malo tendenciu sledovať požadovaný smer cúvaním čo by nemalo žiadúci efekt.

V realite sa kolesá na vozidle nenatáčajú instantne. Preto bol zavedený ďalší parameter a to rýchlosť natáčania kolies. Ten bude udávať koľko stupňov je vozidlo schopné natočiť kolesá za sekundu. Toto bude simulovať, že vozidlo natáča kolesami plynule do požadovaného uhla.

Rýchlosť vozidla Vozidlo si bude prispôsobovať svoju rýchlosť podľa rozdiel v natočení kolies a požadovaným smerom kam sa má vozidlo pohnúť. Pri minimálnom rozdiele bude rýchlosť maximálna. V prípade veľkého rozdielu bude rýchlosť minimálna. Inač, čím bude rozdiel väčší, tým viac bude rýchlosť klesať exponenciálne až po minimálnu rýchlosť.

4.4 Riadenie formácie

Formácia bude riadená dvoj-úrovňovým riadením tak ako je to popísané v sekcii 3.3. Riadenie je celkovo kombináciou riadenia na báze chovaní (angl. behavior-based) a virtuálnej štruktúry (angl. virtual-structure approach).

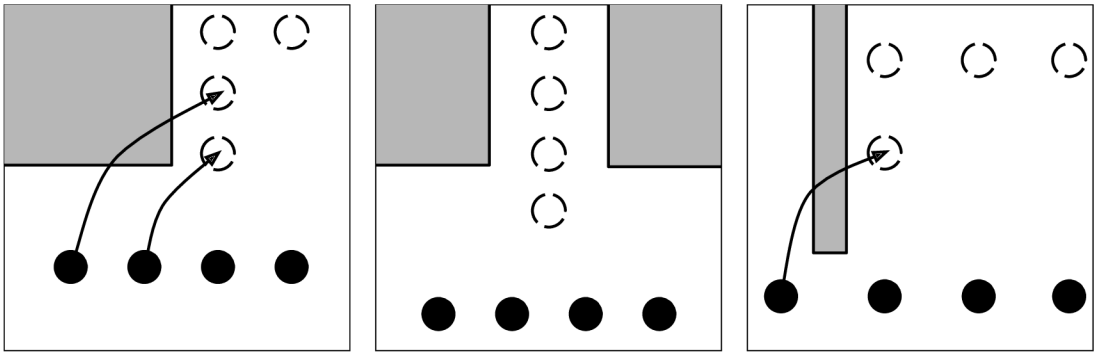
Prvá úroveň spočíva v riadení virtuálnej štruktúry. Toto je úlohou ovládača formácie, ktorý si uchováva zoznam pozícií kde sa majú príslušné vozidlá nachádzať a referenciu na korešpondujúce vozidlo. Virtuálna štruktúra bude reprezentovaná „virtuálnym" vozidlom. Po obdržaní pokynu k presunu formácie na nejaké miesto, ovládač formácie požiada plánovač o nájdenie trajektórie vedúcej k cieľovému bodu. Potom čo plánovač vráti trajektóriu ako postupnosť konfigurácií je táto postupnosť predaná virtuálnemu vozidlu. To, bude nasledovať trajektóriu a všetky pozície vozidiel vo formácii sa budú počítať relatívne voči nemu. Druhou úrovňou riadenia budú moduly chovania vozidiel vysvetlené v rámci sekcie 4.3.3.

4.4.1 Rýchlosť formácie

Virtuálne vozidlo bude mať definovanú maximálnu rýchlosť a minimálnu rýchlosť. Rýchlosti virtuálnemu vozidlu bude nastavovať ovládač formácie. Primárne, bude virtuálne vozidlo jazdiť maximálnou rýchlosťou čo je zároveň požadovaná rýchlosť celej formácie. Dosť často sa bude stávať, že vozidlá nebudú stíhať dôjsť do svojej pozície. V takom prípade musí ovládač formácie zmeniť rýchlosť. Rýchlosť bude prispôsobená na základe priemeru vzdialeností vozidiel od pozícií kde by sa mali nachádzať. Virtuálne vozidlo sa vždy pohybuje aspoň minimálnou rýchlosťou. Minimálna rýchlosť slúži ako prevencia k tomu aby sa jednotky nezasekli niekde v úzkych miestach prostredia.

4.4.2 Vyhýbanie sa prekážkam

Pri pohybe prostredím sa potrebujú jednotky aj vyhýbať prekážkam. Vo všeobecnosti môžu nastať tri prípady brániace hladkému prechodu formácie, ktoré sú znázornené na obrázku 4.6.



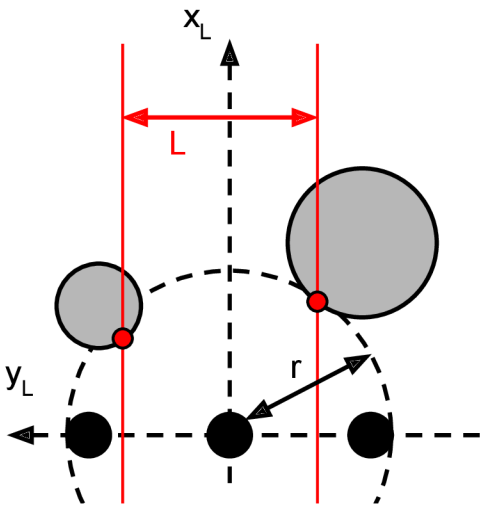
Obr. 4.6: Demonštrácia typov prekážok pri pohybe formácie. Kružnice vyznačené prerušovanou čiarou určujú očakávaný tvar formácie pre vyhnutie sa prekážke.

Ľavý obrázok zachytáva situáciu kedy hrozí prekážka z jednej krídelnej strany formácie. Vtedy majú niektoré jednotky voľnú cestu a niektorým stojí v ceste prekážka a musia sa tak prispôsobiť. Na obrázku v strede je znázornená situácia kedy formácia musí prejsť úzkym

miestom. V takej situácii sa očakáva, že formácia zmení vzor tak, že jednotky sa zaradia za seba pozdĺž cesty. Tretí prípad na obrázku vpravo predstavuje situáciu kedy nestojí prekážka priamo v ceste nejakej jednotky ale mohla by rozdeliť formáciu. To sa stáva vtedy ak sú rozostupy väčšie než je šírka prekážky. Vtedy by sa mali jednotky tiež prispôsobiť tak aby zostali pohromade.

4.4.3 Detekcia prekážok

Zodpovednosť za detekovanie prekážok v okolí a adaptívnosť formácie má ovládač formácie. Celá formácia je obalená kružnicou, ktorej polomer je vždy vzdialenosť k najvzdialenejšej odchýlke od virtuálneho vozidla. Na obrázku 4.7 je označený ako r. Polomer kružnice určuje ako ďaleko sníma ovládač formácie na prítomnosť prekážky. U všetkých prekážok, ktoré aspoň čiastočne zasahujú do obklopujúcej kružnice je získaný najbližší bod na ich hranách. Každý z týchto bodov je následne transformovaný do lokálnych súradníc virtuálneho vozidla. Kladná časť lokálnej osi y_L virtuálneho vozidla predstavuje ľavú stranu a záporná pravú stranu. Virtuálne vozidlo je vždy orientované podľa aktuálnej konfigurácie na trajektórii, na ktorej sa nachádza a jeho lokálna os x_L smeruje vždy pred neho. Zo všetkých transformovaných bodov od prekážok sa zistí najkratšia kladná vzdialenosť na y_L ľavú stranu (ľavá hranica) a najkratšia záporná vzdialenosť pre pravú stranu (pravá hranica). Všetky vozidlá, ktoré majú odchýlku po osi y_L väčšiu ako ľavá hranica alebo menšiu ako pravá hranica sú zaradené na koniec formácie spätne pozdĺž trajektórie. Ako príklad sú na obrázku 4.7 sú najkratšie vzdialenosti vyznačené červenou farbou.



Obr. 4.7: Princíp detekcie laterálnej šírky L úzkeho miesta v prostredí, ktorou musí formácia prejsť. Na základe L vie ovládač formácie zistiť, ktoré jednotky musí usporiadať pretože im hrozí prekážka v ceste.

4.5 Zhrnutie

V tejto kapitole sme si priblížili návrh na riešenie problému koordinovaného pohybu jednotiek. Najprv sme si popísali riadenie vozidla založené na metóde potenciálových polí a potom ako riadiť skupinu takých vozidiel. Navrhnutý systém spája dva typy riadenia. Prvou úrovňou je virtuálna štruktúra a druhou sú moduly chovania na vozidle. Celá myšlienka riadenia formácie spočíva v ovládaní týchto úrovní. Najprv sa pomocou virtuálnej štruktúry zorganizujú relatívne pozície kde sa majú vozidlá nachádzať a potom sa tieto pozície následne predajú do riadiacich častí príslušných vozidiel.

Kapitola 5

Implementácia

V tejto kapitole budeme analyzovať implementačné detaily programu. Systém bol implementovaný v hernom engine Unity3D. Na vývoj bol použitý jazyk C#.

5.1 Popis tried

V tejto sekcii budú popísané jednotlivé triedy systému a ich parametre použité na implementáciu, ktoré sú dôležité z užívateľského hľadiska. U všetkých tried sú parametre v jednotkách súradnicového systému pokiaľ nie je uvedené inak. Diagram vybraných tried systému je uvedený na obrázku 5.1.

Formation Controller Trieda *Formation Controller* je jadrom celého systému. Udržuje si zoznam jednotiek tvoriacich formáciu, ktorým spravuje relatívne odchýlky voči virtuálnemu vozidlu. Ak sa blíži prekážka, a je isté, že niektoré jednotky ňou neprejdu tak ich zaradí na koniec pozdĺž trajektórie s rozostupom daným parametrom *TailSpacing*. Rovnako tak je táto trieda reguluje rýchlosť virtuálneho vozidla. Za normálnych okolností, kedy sú všetky jednotky na svojich miestach sa virtuálne vozidlo pohybuje rýchlosťou udávaným v parametri *Speed*. V prípade prekážky kedy sa organizácia formácie mení, môžu vozidlá zaostávať a preto má ešte jeden parameter a to *MinSpeed* čo je minimálna rýchlosť, akou virtuálne vozidlo môže ísť v prípade, že jednotky nestíhajú dôjsť do svojich pozícií. Princíp ovládania formácie je v sekcii 4.4.

Rozhranie triedy je celkom jednoduché. Trieda má dve metódy *AddSlot()* a *RemoveSlot()*, ktoré slúžia na pridávanie a odobranie jednotky z formácie. Vyžadujú si dva parametre a to číselný identifikátor miesta vo formácii a referenciu na riadiacu časť vozidla *VehicleController*, do ktorej odosiela relatívnu odchýlku vo formácii. Pomocou metódy *MoveTo()* je možné presunúť formáciu na nejaké miesto v prostredí. Nie vždy sa podarí nájsť uskutočniteľnú trasu k určenému miestu. Pre tento účel bola zavedená udalosť *OnPathPlanningFailure*, ktorá sa vyvolá v prípade, že ovládač nebol schopný nájsť trajektóriu vedúcu do cieľa. Užívateľ si tak môže tento prípade ošetriť vlastným spôsobom.

IFormationPattern Jedným zo spôsobov ako definovať vzor formácie je ručné zadávanie relatívnych pozícii. Ručný spôsob ale nie je veľmi priateľský nakoľko si užívateľ musí pamätať ktorému vozidlu prislúcha ktorá pozícia. Pre jednoduchšie vytváranie vzorov formácii bolo vytvorené rozhranie *IFormationPattern*. Namiesto ručného zadávania pozície tak

užívateľ zadá identifikátor miesta vo formácii a rozhranie spočíta príslušnú pozíciu automaticky. Výpočet pozície sa deje v metóde *GetOffset()*, ktorú si tiež musí užívateľ definovať. Takým spôsobom si možno vytvoriť rôzne vzory formácii. Rozhranie ponúka aj metódu *IsSlotCountSupported()* pre ošetrenie neadekvátneho počtu jednotiek vo formácii. Napríklad u formácie *Wedge* (viď obr. 3.7) je vhodné ak je počet jednotiek na oboch stranách vyvážený, t.z. že vzor *Wedge* by mal pozostávať z párneho počtu jednotiek. Iným prípadom užitia by mohlo byť obmedzenie počtu jednotiek na fixne stanovenú hodnotu.

ControlValues Je trieda reprezentujúca riadiace hodnoty modulov chovaní vozidiel formácie. Jednotlivé moduly chovania sú detailnejšie popísané v 4.3.3. Parametre triedy sú nasledovné:

- `ObstacleSeekRadius` - Polomer kružnice okolia, v ktorom vozidlo detekuje prítomnosť prekážok.
- `ObstacleInfluenceRadius` - Vzdialenosť od hrany prekážky predstavujúca ovplyvňujúcu zónu kde odpudivá sila začína lineárne stúpať (viď obrázok 4.5).
- `ObstacleSafetyMargin` - Veľkosť bezpečnostnej rezervy od prekážky, v ktorej je odpudivá sila maximálna (viď obrázok 4.5).
- `ControlledZoneRadius` - Vzdialenosť od cieľového bodu, v ktorej sa atraktívna sila smerujúca k cieľu lineárne zväčšuje (na obrázku 4.5 kontrolovaná zóna).
- `DeadZoneRadius` - Vzdialenosť od cieľového bodu, v ktorej atraktívna sila nemá na vozidlo vplyv (na obrázku 4.5 vyznačená ako mŕtva zóna).
- `TimeToTryUnstuck` - Čas v sekundách, po ktorom sa vozidlo snaží vyslobodiť zo zaseknutia s iným vozidlom.
- `TimeGoingReverse` - Čas v sekundách, ktorý musí ísť auto minimálne dozadu pri pokuse o vyslobodenie z kolízie s iným vozidlom.

Na vozidlo bol pridaný dodatočný modul chovania šum. Šum generuje silu v pseudonáhodnom smere s veľkosťou rovnej jednej. Jeho úlohou je vyslobodiť vozidlo z uviaznutia v lokálnom minime (stav kedy sa atraktívna a odpudzujúca sila vykrátia).

- `NoiseGenerationInterval` - Časový interval daný v sekundách, v ktorom sa generuje nová pseudonáhodná sila šumu.
- `NoisePersistanceTime` - Koľko sekúnd pretrváva vygenerovaná sila šumu. Po uplynutí tohto času sa resetuje na nulu.

GainValues Táto trieda reprezentuje váhy síl z modulov chovaní.

- `AvoidStaticObstacles` - Váha odpudzujúcej sily od prekážok.
- `AvoidTeamMate` - Váha odpudzujúcej sily od ostatných vozidiel.
- `Noise` - Váha pseudonáhodnej sily šumu
- `MoveToGoal` - Váha atraktívnej sily k cieľu.

VehicleController Úlohou tejto triedy je navigovať vozidlo. Ako už bolo spomenuté, pokyny vozidlo dostáva od ovládača formácie *FormationController*. Trieda má vlastnosť *ReferencePoint* kde ovládač formácie za každým nastavuje príslušnú relatívnu pozíciu od virtuálneho vozidla. Po nastavení bodu *ReferencePoint* sa generuje atraktívna sila do tohto bodu. Okrem atraktívnej sily sú v tejto triede implementované aj ostatné moduly chovania. Vozidlo je navigované do cieľového bodu *ReferencePoint* s rešpektovaním týchto síl tak ako je uvedené v sekcii 4.3.3.

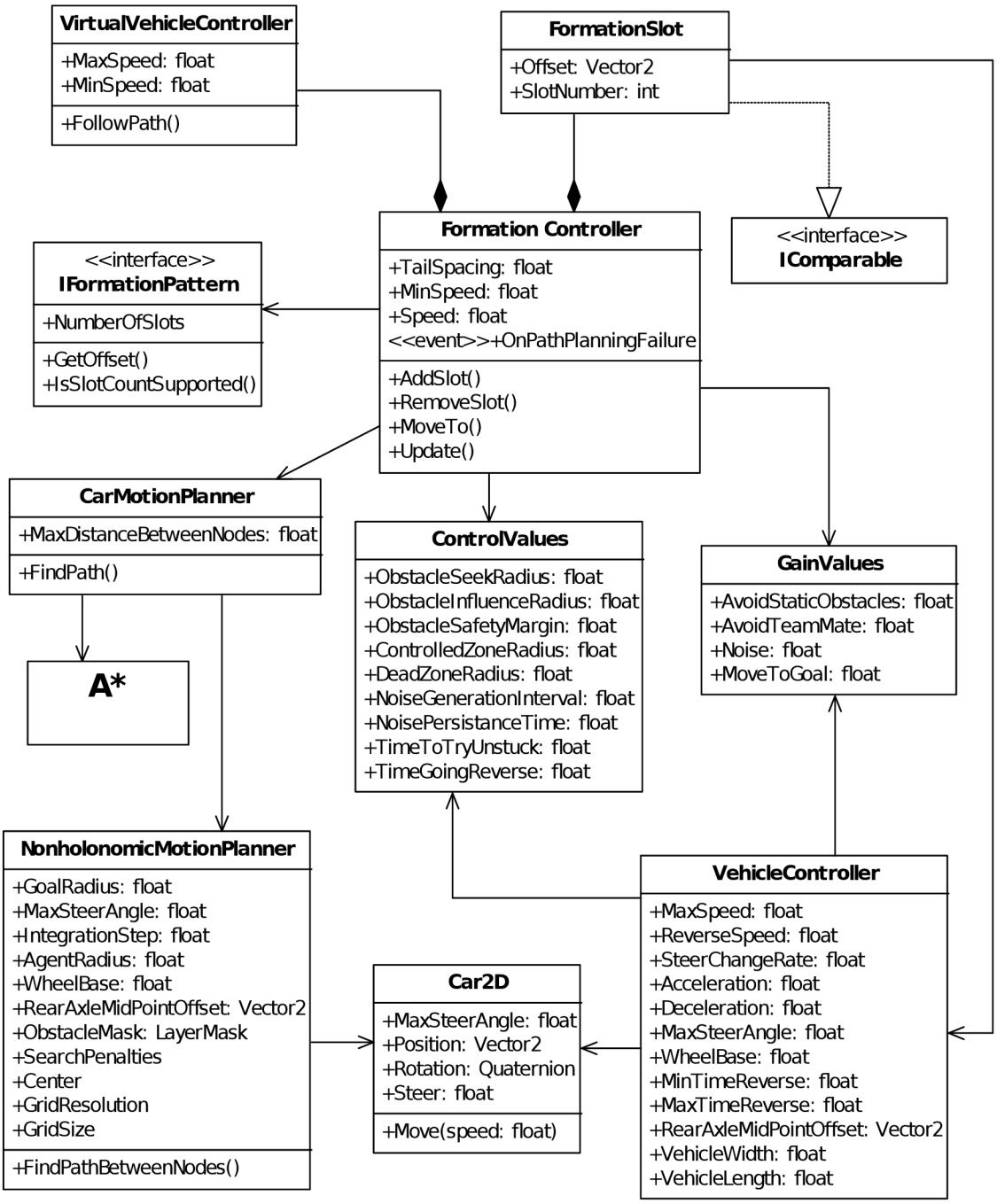
Aby natáčanie kolies nebolo instantné ale malo plynulý priebeh a vyzeralo viac realisticky, riadi sa exponenciálne čo znamená, že rýchlosť natáčania exponenciálne klesá so stúpajúcim sa uhlom medzi vektorom orientácie kolies a smerovým vektorom od vozidla k referenčnému bodu *ReferencePoint*. Rýchlosť natočenia udáva parameter *SteerChangeRate* v stupňoch za sekundu. Natočenie kolies je obmedzené uhlom *MaxSteerAngle* v stupňoch. V reálnom svete má vozidlo aj svoje rýchlostné limity. Pre tento účel je zavedený parameter *MaxSpeed* udávajúci maximálnu možnú rýchlosť vozidla dopredu. Pre cúvanie bola zvolená osobitná rýchlosť aby vozidlo nešlo prudko dozadu. Pre správnu funkcionalitu pohybu vozidla je ešte treba nastaviť rázvor kolies *WheelBase* (vzdialenosť predných a zadných kolies) a *RearAxleMidPointOffset* čo je ochýlka od stredu vozidla udávaná vektorom kde sa nachádza stred rotačnej osi zadných kolies. Trieda má naviac dva parametre *MinTimeReverse* a *MaxTimeReverse*, ktoré predstavujú minimálny a maximálny čas čo môže ísť vozidlo dozadu a slúžia na to aby vozidlo neoscilovalo medzi smermi jazdy dopredu a dozadu a zároveň aby nešlo permanentne dozadu.

NonholonomicMotionPlanner V tejto triede sa odohráva celý proces plánovania pohybu. Úlohou plánovača je zistiť už vo fáze plánovania či sa vozidlo dokáže do nejakého zadaného miesta v rámci svojich neholonomických obmedzení. Plánovač pri plánovaní myslene simuluje pohyb vozidlom a snaží sa tak nájsť trajektóriu vedúcu do cieľa. Pohyb vykonáva vo fixnom kroku *IntegrationStep* čo je vzdialenosť medzi jednotlivými konfiguráciami pozdĺž trajektórie. Taktiež, potrebuje mať informáciu o fyzických parametroch vozidla rovnako ako trieda *VehicleController*. Parametre *MaxSteerAngle*, *WheelBase* a *RearAxleMidPointOffset* by preto mali byť rovnaké ako v triede *VehicleController*.

Aby plánovač nevyhľadával trajektóriu dookola po jednej kružnici má definovanú mriežku, ktorá diskretizuje konfiguračný priestor. Pri procese plánovania si u každej konfigurácie zisťuje, do ktorého políčka v mriežke spadá a označí políčko ako obsadené. V prípade, že je už políčko obsadené, plánovač konfiguráciu zahodí a zabezpečuje tak, že vozidlo neprejde dvakrát tým istým miestom. Je treba poznamenať, že plánovač nie je precízny pretože vyhľadáva trajektóriu do cieľového bodu s určitou toleranciou. Táto tolerancia je daná polomerom kružnice cieľového regiónu *GoalRadius*.

Vozidlo je obalené kružnicou aby plánovač mohol testovať kolíziu s prostredím. Polomer tejto kružnice sa udáva v parametri *AgentRadius*. Detaily plánovača sú popísané v sekcii 4.2.

CarMotionPlanner Táto trieda je nadstavbou plánovača pohybu. Bola zavedená z dôvodu jeho optimalizácie aby nemusel vyhľadávať plán pohybu medzi veľmi vzdialenými bodmi. Prvé čo *CarMotionPlanner* nájde je trasa pomocou A* algoritmu. Následne je trasa optimalizovaná redukciou počtu bodov na trase. Parameter *MaxDistanceBetweenNodes* udáva aká môže byť maximálna vzdialenosť medzi jednotlivými bodmi trasy. Čím je tento parameter väčší, tým viac bude zredukovaný počet bodov po trase. Medzi každými dvoma bodmi je potom nájdený plán pohybu pomocou plánovača pohybu *NonholonomicMotionPlanner*. Plány od všetkých dvojíc bodov tvoria dohromady výslednú trajektóriu.



Obr. 5.1: Diagram tried systému. Na obrázku sú vybrané len dôležité triedy.

5.2 Zhrnutie

V tejto kapitole sme si predstavili ako je navrhnutý systém z predošlej kapitoly realizovaný a zároveň sme si objasnili korešpondenciu jednotlivých tried s návrhom. Popísané tu boli len najdôležitejšie triedy pričom samotná implementácia ich obsahuje viac.

Kapitola 6

Testovanie

V tejto kapitole bude implementovaný systém otestovaný. Všetky testovacie scény ako aj ich video ukážky možno nájsť na priloženom DVD.

6.1 Test jednotlivých častí

Ako prvé čo bolo treba otestovať boli jednotlivé časti systému. Bez overenia, že každá časť pracuje správne by nebolo prijateľné testovať systém ako celok. U všetkých prvkov systému sa nachádza mnoho parametrov. Pri testovaní bolo snahou ich nastaviť tak aby výsledný efekt bol žiadúci. Na testovanie bol použitý model vozidla s parametrami podľa nasledujúcej tabuľky:

Parameter	Veľkosť
Šírka	1,0
Dĺžka	2,0
Rázvor kolies	0,6
Maximálny uhol natočenia kolies	40
Odchýlka zadných kolies	(-0,3, 0.0)

Tabuľka 6.1: Parametre vozidla použitého pri testovaní.

6.1.1 Plánovač

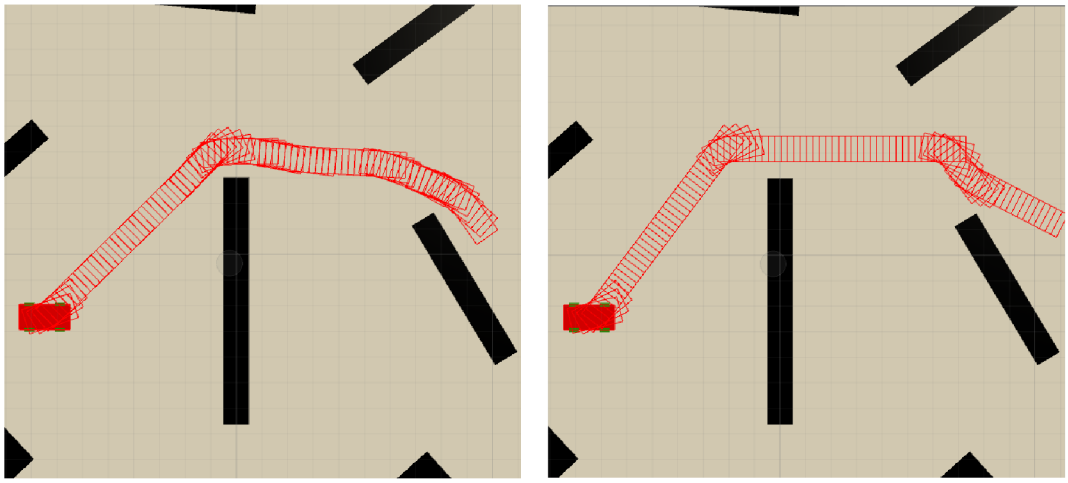
Najprv bol otestovaný plánovač, pretože je prerekvizitou k tomu aby celá formácia vedela pohybovať. Na jeho otestovanie bola vytvorená malá scéna s pár prekážkami. Pri testovaní bolo sledované ako váhy *SearchPenalties* spolu s veľkosťou mriežky a jej rozlíšenia vplývajú na tvar výslednej trajektórie. Zároveň bolo cieľom získať parametre, ktoré budú použité pri testovaní ďalších častí.

Veľkosť testovacej plochy bola 30×30 súradnicových jednotiek, na ktorej bol v prvom rade vytvorený graf pomocou knižnice A* Pathfinding[1]. Následne bol vytvorený objekt reprezentujúci plánovač, na ktorý bola pridaná komponenta *CarMotionPlanner*. Táto komponenta má dva závislé komponenty a to *NonholonomicMotionPlanner* a *Seeker*, ktoré sú pridávané na objekt automaticky. Po vytvorení objektu plánovača boli naň nastavené parametre vozidla podľa tabuľky 6.1. Dodatočne, bolo vozidlo presunuté do samostatnej kolíznej

[1] http://arongranberg.com/astar/)

vrstvy, ktorá bola následne vyradená z plánovača v parametri *ObstacleMask* aby nepovažoval vozidlo ako prekážku. Na koniec bola nastavená veľkosť mriežky na 30 čo je veľkosť plochy a jej rozlíšenie na 50.

Po otestovaní bolo zistené, že plánovač nájde trajektóriu s najlepším tvarom ak je fixný krok *IntegrationStep* menší. V tomto prípade to bolo 0,25. Hodnoty v rozmedzí 0,5–1,0 dávali ešte prijateľný výsledok ale pri väčších hodnotách fixného kroku začína byť trasa neprirodzená. U mriežky je to zase naopak. Lepšie výsledky boli namerané s jej vyšším rozlíšením. Čím vyššie rozlíšenie, tým väčšia bola šanca, že plánovač nájde trajektóriu. Taktiež bolo zistené, že plánovač nájde trasu rýchlejšie ak je vzdialenosť medzi uzlami A* trasy *MaxDistanceBetweenNodes* malá. V tomto prípade najviac vyhovovala vzdialenosť nastavená na 1. Čo sa týka penalizačných váh, trajektória má prirodzenejší tvar ak zatáčky neboli penalizované. Rozdiel v použití iných penalizačných váh je znázornený na obrázku 6.1.



Obr. 6.1: Ukážka tvaru trajektórie s rôznymi penalizačnými váhami. Na ľavej strane sú váhy nastavené *MotionStepsPenalty* = 1, *SteerChangesPenalty* = 0 čím plánovač hľadá najkratšiu trasu. Na pravom obrázku sa plánovač snaží minimalizovať natáčanie kolies s váhami *MotionStepsPenalty* = 1 a *SteerChangesPenalty* = 2.

6.1.2 Pohyb vozidla

Ďalšiu časť, ktorú bolo treba otestovať bolo samotné vozidlo. Zámerom tohto experimentu bolo získať vhodné parametre potenciálových polí v triedach *ControlValues* a *GainValues* tak aby správanie vozidla bolo čo najrealistickejšie. Na to bola opäť vytvorená jednoduchá scéna s jedným vozidlom, ktoré sa pohybovalo medzi prekážkami. Po experimentovaní vyplynulo, že pre testovaciu scénu sú najvhodnejšie parametre a váhy podľa tabuliek 6.2 a 6.3.

Je treba poznamenať, že parametre sa vždy líšia v závislosti na veľkosti prekážok a rozmeroch vozidla. Preto, je dobré s nimi vždy experimentovať a prispôsobiť ich na vlastný prípad užitia.

V ďalších experimentoch s viacerými vozidlami bude častým prípadom, že sa v okolí vozidla bude nachádzať iné vozidlo a zároveň aj prekážka. Z toho dôvodu bola ako predpríprava nastavená väčšia váha odpudivej sile z prekážky aby vozidlo s ňou prípadne nehavarovalo.

Parameter	Veľkosť
ObstacleSeekRadius	3,5
ObstacleInfluenceRadius	3
ObstacleSafetyMargin	1
DeadZoneRadius	1
ControlledZoneRadius	3
NoiseGenerationInterval	6,5
NoisePersistanceTime	4

Tabuľka 6.2: Parametre modulov chovaní.

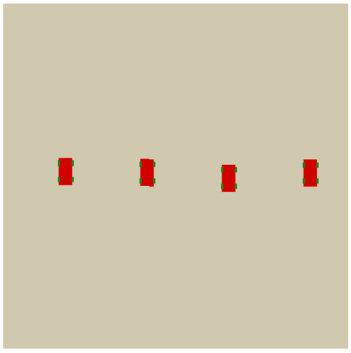
Váha	Veľkosť
AvoidStaticObstacles	1,1
AvoidTeamMate	0,7
Noise	0,1
MoveToGoal	0,5

Tabuľka 6.3: Váhy modulov chovania.

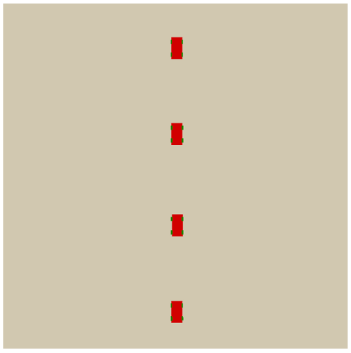
Príliš veľká bezpečnostná rezerva *ObstacleSafetyMargin* a ovplyvňujúca zóna *ObstacleInfluenceRadius* spôsobujú osciláciu vozidla zo strany na stranu, pretože vozidlo odpudzuje sila od prekážky až kým sa nedostane von z ovplyvňujúcej zóny. Potom sa snaží ísť za atraktívnou silou, ktorá ho vráti späť do ovplyvňujúcej zóny a celý proces sa opakuje. Modul chovania *Šum* spôsoboval to isté a preto mu bola nastavená najmenšia váha, pri ktorej sa oscilácia stále deje ale menej viditeľne. Účelom šumu bolo vyslobodiť vozidlo z lokálnych extrémov čo nastáva vtedy ak sa atraktívna a repulzívna sila navzájom vykrátia. V niektorých prípadoch bol šum úspešný, v niektorých sa vozidlo naopak zaseklo.

6.2 Test koordinovaného pohybu

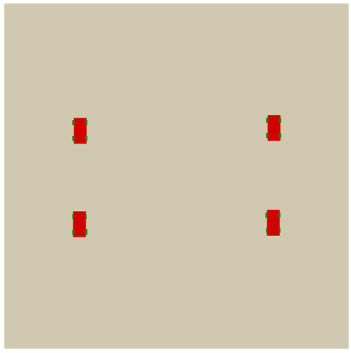
Finálnou etapou testovania bol pohyb vozidiel vo formácii. Vytvorené boli tri vzory *Line* s priečnymi rozostupmi 5 jednotiek, *Column* s pozdĺžnymi rozostupmi 7 jednotiek a *Square* s priečnymi rozostupmi 3 jednotky a pozdĺžnymi 5,5 jednotiek.



Obr. 6.2: Line



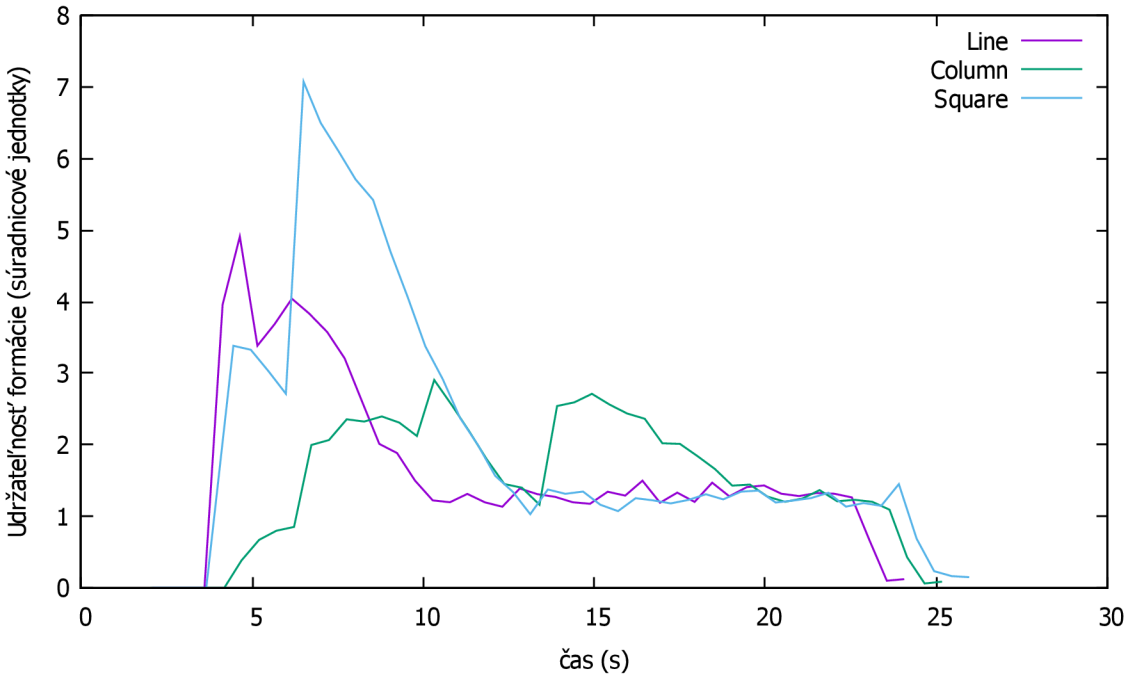
Obr. 6.3: Column



Obr. 6.4: Square

6.2.1 Udržateľnosť formácie

Prvým scenárom bolo testovanie bez prekážok. Tento scenár sa koncentroval na to ako sú jednotky schopné udržať svoju pozíciu vo formácii. Meranou metrikou bola priemerná hodnota absolútnych vzdialeností vozidiel od ich pozície vo formácii, ktorá bude ďalej označovaná pojmom *Udržateľnosť formácie*. Meranie bolo uskutočnené na rôznych vzoroch. Úkol formácie bol rovnaký, dostať sa do toho istého miesta s iným vzorom. Výsledok merania je uvedený na grafe 6.5, ktorý zachytáva ako sa vzdialenosť vozidiel od svojej pozície vyvíja v čase.



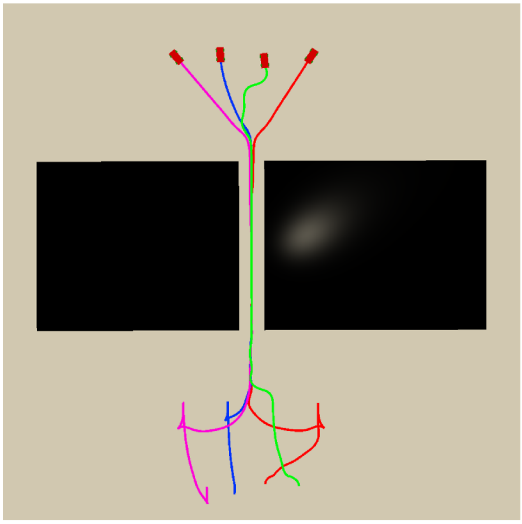
Obr. 6.5: Graf zobrazujúci zmenu priemernej vzdialenosti vozidiel od svojich pozícií v čase. Na začiatku sú vozidla ľubovoľne rozmiestnené a preto je vzdialenosť najväčšia. Najhoršie na tom bol vzor *Square*. Po čase sa to však ustáli u každého vzoru. V grafe si možno ešte všimnúť, že formácia dôjde do cieľa približne v rovnakom čase nezávisle na vzore.

Vzor	**Minimálna hodnota**	**Maximálna hodnota**	**Priemerná hodnota**	**Rozptyl**	**Smerodajná odchýlka**
Line	1,135	4,909	1,634	1,222	1,258
Column	0,385	2,906	1,474	0,771	0,878
Square	1,031	7,074	2,261	3,098	1,760

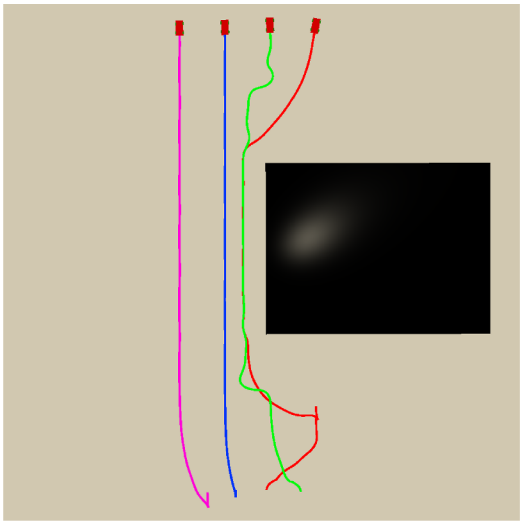
Tabuľka 6.4: Namerané hodnoty udržateľnosti formácie. Uvedené hodnoty boli merané až potom čo sa vozidlá dali do pohybu. Hodnoty na začiatku simulácie a po dôjdení do cieľa nemalo zmysel uvažovať pretože boli nulové. Z tabuľky sa dá odčítať, že vzor, ktorý si formácia dokázala udržať najlepšie je *Column*.

6.2.2 Vyhýbanie sa prekážkam

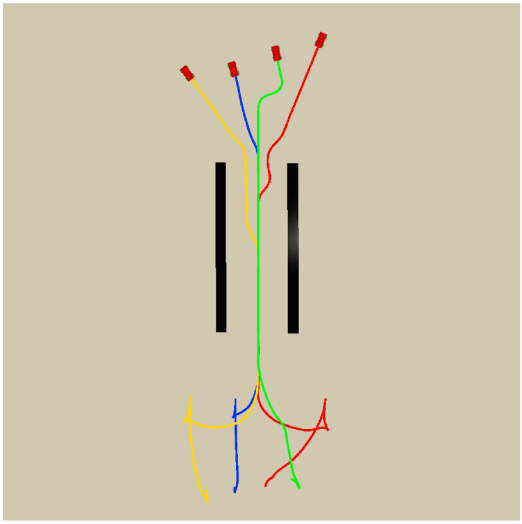
Na ďalší experiment bola nasadená formácia štyroch vozidiel. Ich úlohou bolo prispôsobiť sa prostrediu v určitej situácii. Experiment bol vykonaný na rozličných scenároch. V každom scenári bola formácia vo vzore *Line*. Priebeh všetkých scenárov je na obrázkoch 6.6 až 6.12. Na každom obrázku sú farebne vyznačené trajektórie vozidiel, ktoré prešli. Vo výsledku si formácia počínala celkom dobre. Bola schopná sa vyhnúť prekážkam až na prípad s pohyblivou prekážkou, s ktorou vozidlá kolidovali a ich pohyb nevyzeral prirodzene. Formácia zlyhala ešte v jednom prípade a to v scenári *Zmenšenie rozostupov* (obrázok 6.9). Očakávalo sa, že formácia zmenší rozostupy, namiesto toho sa vozidlá zaradili za seba.



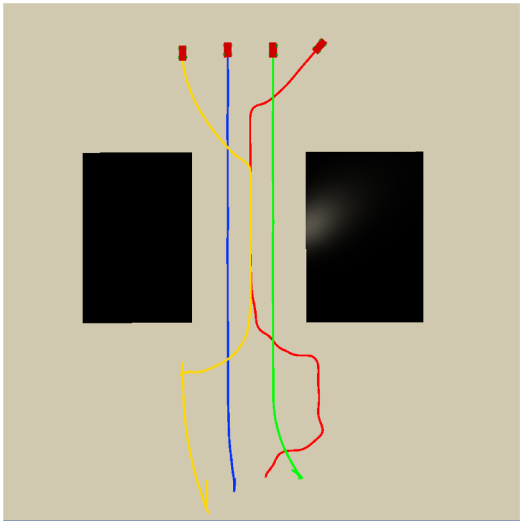
Obr. 6.6: Úzke miesto



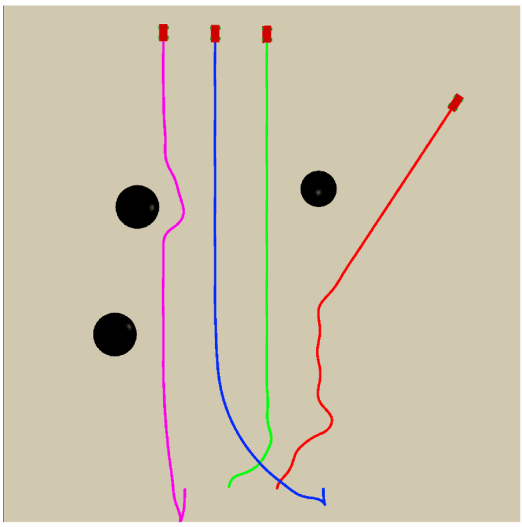
Obr. 6.7: Prekážka z jednej strany.



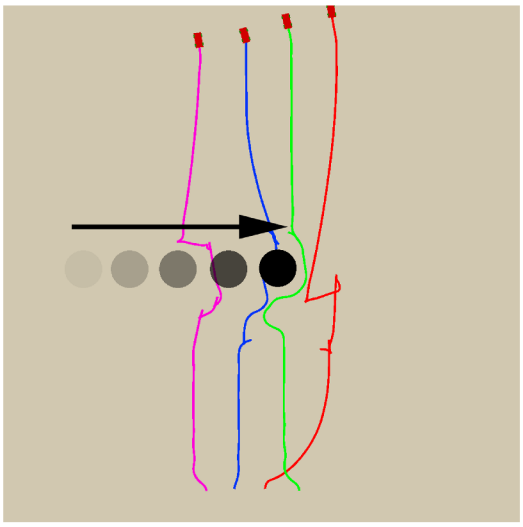
Obr. 6.8: Prekážka v rozostupoch.



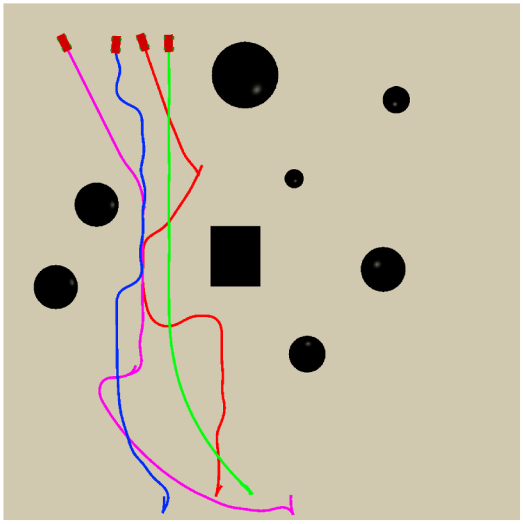
Obr. 6.9: Zmenšenie rozostupov.



Obr. 6.10: Odpojenie vozidla od formácie.



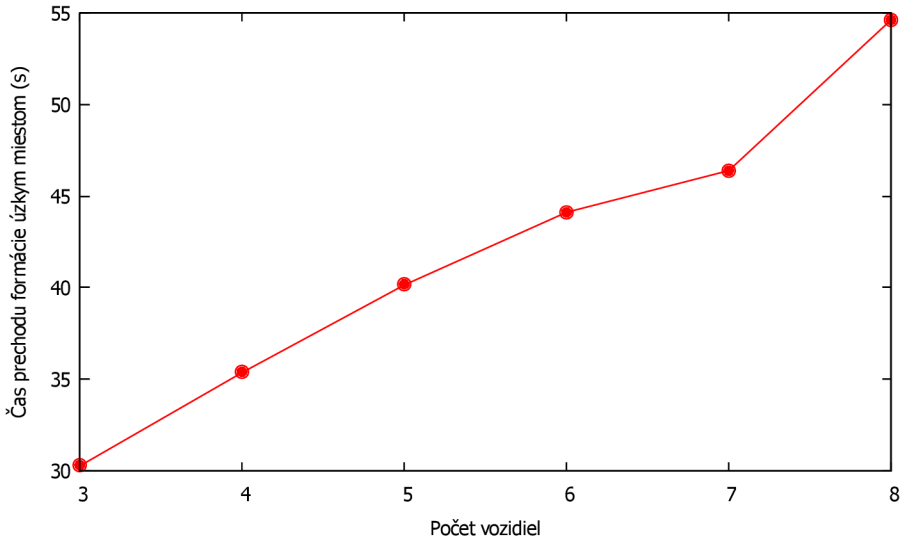
Obr. 6.11: Pohyblivá prekážka.



Obr. 6.12: Prechod formácie komplexnejším prostredím.

6.2.3 Rôzne počty jednotiek

Myšlienkou tohto experimentu bolo overiť vlastnosti škálovateľnosti formácie a zároveň zistiť ako sa skupina správa v danej situácii pri rôznom počte jednotiek. Na to bol použitý scenár prechodu úzkym miestom. Najkritickejším vzorom na tento scenár bol *Line* pretože jednotky boli široko rozprestrené. Meraný bol čas, za ktorý prejdu jednotky úzkym miestom. Výsledok merania je zobrazený na grafe 6.13.



Obr. 6.13: Na grafe je vidieť za aký čas je formácia schopná sa dostaviť do cieľa s rozličným počtom vozidiel pri prechode formácie úzkym miestom. Simulácia bola vyhodnotená na skupinách s počtom 3 až 8 vozidiel so vzorom *Line*.

Kapitola 7

Záver

Výsledkom tejto práce je systém na manažment formácie vozidiel s pohybovými obmedzeniami. Hlavným cieľom bolo vyvinúť metódu, s ktorou sa bude skupina vozidiel pohybovať koordinovane, bude adaptívna voči prostrediu a zároveň bude mať realistický dojem. Toto sa podarilo dosiahnuť dvoj-úrovňovým riadením formácie a kombinovaním metódy riadenia formácie na báze modulov chovaní (angl. behavior-based formation) a virtuálnej štruktúry (angl. virtual-structure approach). Systém bol otestovaný na rozličných scenároch. Z testovania vyplynulo, že formácia vie prispôsobiť v prostrediach s rôznymi typmi prekážok, je škálovateľná a dá sa k nej ľahko definovať nový vzor. Na druhú stranu, implementovaná metóda má nedokonalosť v tom, že sa vozidlá nevedia vyhnúť pohyblivej prekážke prirodzeným spôsobom.

Ako budúca práca sa predpokladá rozšírenie metódy do 3D. Dnes je väčšina počítačových hier a simulátorov vyvíjaných v 3D a preto aby mal systém lepšie využitie je nutné zapracovať na jeho 3D verzii. Ďalšia podstatná časť, ktorú je treba vylepšiť je riadenie vozidla. Dá sa povedať, že samotné riadenie má realistický nádych ale stále môže vyzerať v niektorých situáciach neprirodzene. Okrem toho, táto práca sa venovala iba homogénnym formáciám pozemných vozidiel. Vhodné by bolo urobiť systém viac generickým, ktorý by spravoval nehomogénne formácie s podporou iných typov jednotiek ako pechota, loďstvo či vzdušné sily.

Literatúra

[1] Abdulla M. Mamdouh, Ahmed Kaboudan, Ibrahim F. Imam: Realistic Behavioral Model for Hierarchical Coordinated Movement and Formation Conservation for Real-Time Strategy and War Games. [online], [cit. 2016-05-04].
URL http://www.iaeng.org/IJCS/issues_v39/issue_3/IJCS_39_3_03.pdf

[2] Athanasios Krontiris, Sushil Louis, Kostas E. Bekris: Simulating Formations of Non-Holonomic Systems with Control Limits Along Curvilinear Coordinates. [online], [cit. 2016-05-04].
URL http://www.cs.rutgers.edu/~kb572/pubs/simulating_curvilinear_coordinates.pdf

[3] Choset, H.: *Principles of robot motion : theory, algorithms and implementations.* Massachusetts : MIT Press, 2005, ISBN 0-262-03327-5.

[4] Giesbrecht, J.: Global Path Planning for Unmanned Ground Vehicles. [online], 2004 [cit. 2016-04-10].
URL http://www8.cs.umu.se/research/ifor/dl/Path%20planning/GetTRDoc.pdf

[5] Giesbrecht, J.: Local Navigation for Unmanned Ground Vehicles. [online], 12 2005 [cit. 2016-04-19].
URL www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA605022

[6] Ian Millington: *Artificial Intelligence for Games.* Morgan Kaufmann Publishers, 2006, ISBN 978-0-12-497782-2.

[7] Kiattisin Kanjanawanishkul: *Coordinated Path Following Control and Formation Control of Mobile Robots.* Dizertační práce, Fakultät fur Informations und Kognitionswissenschaften. Eberhard-Karls-Universität, Tubingen, 2010.
URL https://hsbiblio.uni-tuebingen.de/xmlui/bitstream/handle/10900/49446/pdf/diss_kanjanaw.pdf?sequence=1&isAllowed=y

[8] LaValle, S. M.: *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006, dostupné z: http://planning.cs.uiuc.edu/.

[9] Patel, A.: Map representations. *Amit's thoughts on Pathfinding* [online], [cit. 2016-04-12].
URL http://theory.stanford.edu/~amitp/GameProgramming/MapRepresentations.html

[10] Patel, A.: Heuristics. *Amit's thoughts on Pathfinding* [online], [cit. 2016-04-13].
URL http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html

[11] Pottinger, D.: Coordinated Unit Movement. [online], [cit. 2016-05-04].
URL http:
//www.gamasutra.com/view/feature/131720/coordinated_unit_movement.php

[12] Pottinger, D.: Implmenting Coordinated Movement. [online], [cit. 2016-05-04].
URL http://www.gamasutra.com/view/feature/3314/coordinated_unit_
movement.php?print=1

[13] Steve Rabin: *AI Game Programming Wisdom*. Charles River Media, 2002, ISBN 978-1584500773.

[14] T.D. Barfoot, C.M. Clark: Motion planning for formations of mobile robots. [online], [cit. 2016-05-05].
URL http://www.sciencedirect.com/science/article/pii/S0921889003001854

[15] Tucker Balch, Marie Hybinette: Social Potentials for Scalable Multi-Robot Formations. [online], 2000 [cit. 2016-05-08].
URL
http://www.eecs.ucf.edu/~gitars/cap6671-2008/Papers/balch00social.pdf

Prílohy

Zoznam príloh

Príloha A

Obsah DVD

Súčasťou práce je aj priložené DVD, ktorého obsah je nasledovný:

- `/tex/` — Zdrojové súbory tejto technickej správy pre LaTeX
- `/src/` — Zdrojové kódy programu v podobe projektu pre Unity3D.
- `/video/` — Video ukážky z testovacích scenárov.
- `/thesis.pdf` — Text tejto bakalárskej práce.
- `/README.txt` — Súbor so stručným návodom ako použiť programové rozhranie.