Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE

Markéta Popelová

Knihovna steering technik pro virtuální agenty

Kabinet software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Michal Bída
Studijní program: Informatika
Studijní obor: Obecná informatika

Praha 2011

Ráda bych poděkovala vedoucímu své bakalářské práce Mgr. Michalu Bídovi za cenné a konstruktivní připomínky, ochotu si vždy najít čas na mé dotazy a pomoc s platformou Pogamut. Další poděkování patří Bc. Editě Bromové za rady k textu. Nakonec děkuji svému drahému Jakubovi za náměty k inspiraci, poznámky k textu a péči a toleranci při tvorbě této práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracoval(a) 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 v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Knihovna steering technik pro virtuální agenty

Autor: Markéta Popelová

Katedra: Kabinet software a výuky informatiky

Vedoucí bakalářské práce: Mgr. Michal Bída, Kabinet software a výuky informatiky

Abstrakt: Jedním z možných přístupů k navigování autonomních agentů ve virtuálním prostředí je použití steering technik Craiga W. Reynoldse. Každá z nich má jeden úkol (např. vyhýbání se překážkám, následování jiného agenta, apod.) a jejich kombinací získáme bohaté možnosti navigace agentů. Předmětem této práce bylo upravit steeringy tak, aby umožňovaly řízení lidských virtuálních agentů, a prozkoumat, nakolik jsou k tomuto účelu vhodné. Kromě upravení a rozšíření původních steeringů C. W. Reynoldse byly navrženy i steeringy nové. Zároveň byly zkoumány možnosti kombinací steeringů pro řešení složitějších situací. Steeringy byly implementovány na platformě Pogamut v 3D prostředí UnrealEngine2Runtime. Vznikla tak knihovna steeringů, kterou mohou využívat i ostatní části platformy Pogamut, a grafická aplikace, která umožňuje spouštět v 3D prostředí agenty s různě nastavenými steeringy.

Klíčová slova: steering, navigace, virtuální agent, umělá inteligence

Title: Steering techniques library for virtual agents

Author: Markéta Popelová

Department: Department of Software and Computer Science Education

Supervisor: Mgr. Michal Bída, Department of Software and Computer Science Education

Abstract: The steering techniques of Craig W. Reynolds are an approach to navigation of autonomous agents in a virtual environment. Each steering has a single goal (e.g., to avoid obstacles, follow another agent, etc.). Combinations of these steerings provide rich possibilities of navigation. The goal of this thesis was to adjust steerings for navigation of human virtual agents and explore how appropriate are they for this purpose. In addition to adjusting the original steerings of C. W. Reynolds, new steerings were designed, as well as an algorithm of steering combination. The modified and newly designed steerings were implemented on the Pogamut platform in the 3D virtual environment UnrealEngine2Runtime. The created steering library may be used by other parts of Pogamut too. Also, an aplication was created with the purpose of running virtual agents in the virtual environment and setting parameters of their steerings.

Keywords: steering, navigation, virtual agent, artificial intelligence

Obsah

1. Úvod

Poslední dobou můžeme zaznamenat rostoucí oblibu 3D počítačových her a filmů s umělými bytostmi. S tímto trendem se objevují mnohé problémy, které musí tvůrci řešit. Jak zařídit, aby vypadaly počítačově vytvořené postavy realisticky? Jak se mají chovat, aby připomínaly reálné bytosti? Speciálně se zaměřme na to, jak řídit jejich pohyby – aby nevrážely do okolních objektů, reagovaly na jiné postavy apod. Šlo by každé postavě předem vymyslet přesnou dráhu pohybů. Výhodnější by však bylo, kdyby si postavy uměly své dráhy určovat samy.

Obdobným problémem se zabývají lidé z docela jiného odvětví, než je zábavní průmysl – vědci, kteří provádí simulace davu pomocí počítačových programů. Jejich výsledky se pak využívají při navrhování ulic a veřejných prostor tak, aby ideálně vyhovovaly chodcům, kteří se v nich budou pohybovat. I zde je potřeba umět řídit pohyby chodců tak, aby to odpovídalo pohybům reálných lidí. Jaké máme možnosti řešení?

Mohli bychom se pokusit sestrojit superpočítač, který bude modelovat neuronové sítě vnímání a rozhodování jednotlivých postav, a nechat ho spočítat dráhy všech postav v závislosti na mnoha parametrech daných situací. Kromě toho, že by bylo velmi náročné celý mechanismus navrhnout, by všechny výpočty trvaly patrně příliš dlouho. Navíc by výsledné chování nemuselo být vůbec věrné, neboť je pro nás lidský mozek stále příliš složitý.

Nebo bychom mohli natočit obdobnou reálnou situaci na kameru a z ní zrekonstruovat dráhy pohybů jednotlivých osob. Následně bychom nechali virtuální postavy, ať se pohybují přesně podle natočených drah. Tím bychom vytvořili jednu situaci, ale pro všechny další (v novém prostředí, s jinými postavami) by bylo třeba nových záznamů.

Naše řešení by tedy mělo být výpočetně poměrně jednoduché a přitom obecné, aby dovolovalo řešit různé druhy situací v různých podmínkách. Navíc by pohyby počítačových postav měly připomínat pohyby reálných lidí.

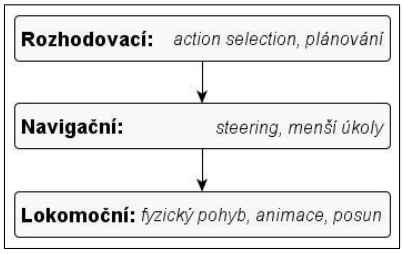
Tyto požadavky by mohlo splňovat řešení, které bude využívat steering techniky Craiga W. Reynoldse (dále je budeme nazývat *steeringy*). Ty byly původně navrženy pro řízení jednoduchých zvířecích virtuálních agentů [30] a později rozšířeny i pro obecné (živé i neživé, avšak pohybující se) virtuální agenty [34]. Jejich výhodami jsou výpočetní jednoduchost, předvídatelnost a obecnost (tzn. nejsou vytvořeny pro konkrétní situace). Často se řadí do reaktivního chování, neboť se většinou rozhodují jen podle aktuálního okolí agenta. Některé ale využívají i předplánované informace, čímž kombinují výhody obou přístupů: reaktivního chování i plánování.

Pro jednoduché zvířecí virtuální agenty (ptáky, ryby apod.) je tato metoda dobře prozkoumaná a s úspěchem používaná. Pro lidské virtuální agenty však tak dobře prozkoumaná není (alespoň ne všechny steeringy) a existují i názory, že pro lidské virtuální agenty tato metoda není úplně vhodná [4, 5]. Tato práce navrhuje, jak steeringy rozšířit tak, aby se pomocí nich dali řídit lidští virtuální agenti, a posuzuje vhodnost metody pro lidské virtuální agenty. Navržené řešení bylo implementováno a byl vytvořen nástroj pro pozorování agentů, kteří implementované steeringy používají. Část výsledků byla publikována v článku [24] a tato práce je jeho rozšířením.

1.1 Terminologie a úvod do problematiky

Virtuálním agentem myslíme typ autonomního agenta, který existuje ve virtuálním světě[1], má vlastní tělo a může se pohybovat. Rozhoduje se reaktivně podle stavu okolního světa, nemá tedy předem dané chování. Speciálně se zajímáme o tzv. *human-like inteligentní virtuální agenty* (*human-like IVAs, human-like intelligent virtual agents*), tedy virtuální agenty, kteří se vzhledem i chováním podobají lidem. Tyto agenty budeme někdy nazývat lidští agenti.

Chování agenta je velmi komplexní pojem. Zaměřme se na jednu jeho složku – pohyb. Vzhledem k této složce můžeme chování virtuálního agenta rozdělit do tří vrstev [34]: *rozhodovací* (*action selection*), *navigační* (*steering*) a *lokomoční* (*locomotion*), viz Obrázek 1.1. Rozhodovací vrstva se zabývá smyslem agentova pohybu a rozhoduje, jaké akce bude agent provádět, aby byly splněny určité cíle. Navigační vrstva dostává jednodušší úkoly, jak se má agent pohybovat. Např. že má dojít k určitému místu, vyhýbat se překážkám či ostatním agentům, následovat jiného agenta apod. Samotný fyzický pohyb vykonává vrstva lokomoční. Řeší, jak hýbat s tělem agenta, aby se správně posouvalo z místa na místo, otáčelo, atd.



Obrázek 1.1: Tři vrstvy pohybu virtuálního agenta.

Rozdělení do tří vrstev velmi hezky ukazuje ilustrační situace (vycházející z [34]) se stádem krav, kovbojem a šerifem. Jedna z krav se vzdálí od stáda. Šerif pověří kovboje, ať krávu přivede. Kovboj pobídne svého koně a vyrazí směrem ke krávě. Dává si pozor, aby po cestě nevrazil do ostatních kovbojů, krav, či ohrady. Až ke krávě dorazí, přižene ji zpět ke stádu a sám se vrátí k šerifovi. V tomto příkladu představuje šerif rozhodovací vrstvu: zjišťuje, jak se změnil stav okolního světa (utekla kráva) a co je třeba udělat (přivést krávu zpět). Daný úkol uloží kovbojovi, který odpovídá navigační vrstvě. Kovboj se stará o to, aby po cestě nenarážel do žádných překážek, aby dojel ke krávě a aby ji přivedl zpět. Každý z těchto úkolů je předmětem jednoho ze steeringů, které kovboj provádí. Za samotný pohyb kovboje (tedy za lokomoční vrstvu) je zodpovědný kůň, který ho nese. Kovboj mu dává příkazy (např. zatoč doprava, zrychli, zastav), které kůň plní.

Výhodou tohoto dělení je mimo jiné i jeho flexibilita. Rozhodneme-li se přenést model do jiného prostředí či na jiné objekty (které vykonávají lokomoci odlišně), stačí změnit jen lokomoční vrstvu a přizpůsobit příkazy navigační vrstvy. Proto když budou za sto let jezdit kovbojové na motorce, stále mohou používat stejné steeringy a plnit obdobné příkazy nadřízených, přičemž změní jen způsob, kterým ovládají své vozidlo.

Tato práce se zabývá především navigační vrstvou a jejím propojením s okolními vrstvami. *Steering* je algoritmus, který řeší navigaci virtuálního agenta vzhledem k jednomu cíli. Takovým cílem může být např. dojít ke krávě, jít podél zdi, nenarážet do lidí, atd. Chceme-li dosáhnout složitějšího chování, musíme zkombinovat více steeringů najednou.

Naše steeringy vycházejí ze steeringů, které navrhl Craig W. Reynolds [30, 34].

[1] V našem případě se jedná o virtuální 3D svět.

Jeho steeringy byly původně navrženy pro tzv. *boidy* (bird-like-objects), stvoření, pomocí kterých simuloval hejna ptáků či ryb, stád krav apod. Později rozšířil svůj model na obecnější virtuální agenty (chodící, létající či plovoucí živé i neživé pohyblivé agenty) a přidal několik dalších steeringů. Nejdříve si ukážeme společné vlastnosti steeringů obecných virtuálních agentů. Následně se zaměříme na to, v čem jsou požadavky odlišné pro agenty lidské.

1.1.1 Základní fungování steeringů

Abychom mohli virtuálního agenta navigovat, potřebujeme znát dvě základní informace o jeho stavu: *lokaci*, tedy jeho souřadnice v prostoru, a *vektor rychlosti*. Kromě těchto základních údajů můžeme vyžadovat různé doplňující informace, jako je vektor natočení, maximální velikost vektoru rychlosti, atd.

Dále potřebujeme znát různé informace z okolního světa – o překážkách, ostatních agentech a někdy i různé nadstandardní informace, jako je např. síť dostupných navigačních bodů v okolním světě. Jak můžeme tyto informace získat, záleží na realizaci světa. Někdy zná agent topologii celé mapy světa, ze které zjišťuje lokace okolních živých i neživých objektů. Někdy zná jen lokace ostatních agentů a informace o neživých objektech musí získávat jinak – např. pomocí paprsků vycházejících z jeho těla.

Steeringy, o které se zajímáme, jsou založeny na vektorovém principu. Pohyb agenta na úrovni navigační vrstvy vzniká úpravami vektoru rychlosti daného agenta. Takto upravený vektor rychlosti je předán lokomoční vrstvě, která zajistí, aby se agent pohyboval správně rychle a správným směrem. Steering jako algoritmus navigování agenta je specifický tím, že vektor rychlosti upravuje podle sil, které na agenta v danou chvíli působí. Např. stará-li se steering o vyhýbání se překážkám, pak chování steeringu odpovídá představa, že všechny blízké překážky působí na agenta odpudivou silou. Nebo, chceme-li, aby agent mířil k nějaké lokaci, pak daná lokace na agenta působí silou přitažlivou. Výsledkem steeringu v daném okamžiku je složení působících sil. Výsledná síla určuje nový vektor rychlosti agenta. Možností skládání sil je více. Tímto problémem se zabývá kapitola 4.

Celý tento proces probíhá periodicky. Čas ve virtuálním světě je rozdělen na tiky a v každém tiku je steering zavolán, aby vypočetl sílu, která na agenta působí – a podle které se pak agent bude pohybovat v následujícím tiku. Je poměrně podstatné, jak dlouho trvá jeden tik. Čím kratší dobu jeden tik trvá, tím jich proběhne za daný časový okamžik více. V každém z těchto tiků je možno pohyb trochu upravit. Proto čím kratší dobu jeden tik trvá, tím je snazší zařídit, aby bylo výsledné chování plynulé a přitom aby agent včas reagoval na změny v prostředí. Je-li hustota tiků řidší, steeringy musí být navrženy mnohem robustněji, aby bylo výsledné chování dostatečně přirozené.

1.1.2 Požadavky na steeringy lidských virtuálních agentů

Vytváříme-li steeringy pro lidské agenty, musíme počítat s tím, že se lidé pohybují značně jinak než třeba ptáci nebo krávy. Zvířecí agenti se mohou pohybovat zdánlivě náhodně, nicméně u lidských agentů by to vypadalo nepřirozeně. Zajímáme-li se o pohyb ve specifickém prostředí, např. ve městě, je dobré umět zohlednit, že

lidé chodí spíše po chodnících, než že by křižovali ulice[2]. Taktéž je nutno se zaměřit na plynulost pohybu: neměla by se často výrazně měnit rychlost ani směr. Je nutno rozhodnout, zda je v dané situaci přirozenější rychle zareagovat na změnu v prostředí, nebo zachovat plynulost.

Imitace chování lidských agentů má jeden značný handicap, který lze však brát i jako výzvu. Pozorovatel požaduje od lidského agenta mnohem věrohodnější chování než třeba od agenta představujícího exotické zvíře, mimozemšťana nebo jinou vymyšlenou bytost. Člověk nemusí být odborníkem na lidské chování, aby na první pohled poznal, že něco není v pořádku. S touto komplikací souvisí problém známý jako „uncanny valley“, na který prvně upozornil Masahiro Mori roku 1970 [17]. Tento problém poukazuje na to, že jsou často vytvořeny 3D postavy, které vypadají poměrně realisticky, pokud jsou zachyceny na fotce, avšak začnou-li se hýbat, působí velmi nepřirozeně. Dalo by se říci, že čím realističtější má postava vzhled, tím vyšší nároky jsou kladeny na její chování, začne-li se hýbat [20]. Proto má smysl předem zkoumat lidské chování a i v této práci byl kladen důraz na rozdíly lidského chování oproti chování vytvořeného původními Reynoldsovými steeringy.

Ve zbylé části této práce bude termínem agent automaticky myšlen human-like inteligentní virtuální agent.

1.2 Související práce

V roce 1987 publikoval C. W. Reynolds model boidů pro simulaci chování zvířecích stád a hejn [30]. Jedinci těchto stád jsou řízeni individuálně pomocí tří základních steeringových pravidel: aby nevráželi do překážek a okolních jedinců (*separation*), aby srovnávali svou rychlost (velikost i směr) podle okolních jedinců (*alignment*) a aby se snažili zůstávat blízko okolních jedinců (*cohesion*). Výsledné chování je natolik věrohodné, že bylo několikrát použito i v počítačových hrách a filmech, jako např. ve filmu Batman se vrací[39].

C. W. Reynolds svůj model boidů postupně rozšiřoval. Využíval k tomu mj. genetické programování. Pomocí něj zkoumal, jak co nejlépe řešit tyto problémy: koordinace pohybu živých tvorů ve skupině [31], vyhýbání se překážkám [32], průchod koridorem [33]. Rozšířený model steeringového chování autonomních agentů publikoval roku 1999 [34]. Uvedl v něm sadu steeringů, pomocí kterých lze řídit autonomní agenty. Tentokrát se nejedná jen o hejna či stáda, ale libovolné po zemi se pohybující, plovoucí či létající živé i neživé virtuální agenty. Tento model se stal poměrně oblíbeným a byl často využíván a reimplementován, např. [37].

Steering, kterému je věnováno asi nejvíce pozornosti, má za úkol předcházení kolizím s překážkami a ostatními agenty. Nejčastěji je označovaný jako Obstacle Avoidance, People Avoidance, či Collision Avoidance. Tímto steeringem se zabývali např. [46, 36, 45]. Předcházení kolizím s ostatními agenty je využíváno v simulacích davu (např. [9, 18, 14]), jejichž výsledky se používají např. při řešení krizových situací, evakuačních plánů, navrhování ulic, nákupních domů, v počítačových hrách apod.

Práce [20, 21] se zabývají teritoriálním chování. Nedávají si za cíl pouhou schopnost vyhýbat se ostatním agentům, ale složitější sociální interakce s nimi

[2]Pokud ovšem nemodelujeme opilce, neboť i to by mohlo být úkolem steeringu.

jako např. natáčení se během rozhovoru.

Dále existují práce, které nerozšiřují ani nevyužívají samotné steeringy, ale poskytují k nim obecné nástroje. Např. autoři [38] navrhují možnosti subjektivního a objektivního komplexního hodnocení kvality steeringů. Zaprvé předkládají sadu scénářů speciálních situací a kritérií, podle kterých hodnotit, jak dané steeringy situaci zvládly. Zadruhé navrhují metriky, podle kterých lze automaticky hodnotit kvalitu steeringů. Některé z navržených scénářů tato bakalářská práce využívá pro ohodnocení kvality implementovaných steeringů.

Dále existují práce, které souvisejí pouze s úzkou částí této bakalářské práce, například s jedním ze steeringů. Tyto související práce jsou uvedeny vždy u popisu dané části. Především je v *Kapitole 3* v závěru popisu každého ze steeringů uveden seznam podobných prací a případně i srovnání s těmito pracemi.

1.3 Cíle práce

Tato práce se zabývá implementací steering technik inspirovaných Craigem W. Reynoldsem v 3D prostředí Unreal Tournament 2004. K připojení k prostředí je využívána platforma Pogamut [23]. Výsledná knihovna bude distribuována jako rozšíření této platformy. Součástí práce je také evaluace jednotlivých řešení provedená pomocí grafické aplikace, která byla vytvořena v rámci této práce a která umožňuje porovnávat kvalitu jednotlivých steeringů. Hlavní cíle práce jsou:

- Navrhnout steeringy tak, aby umožňovaly řídit lidské virtuální agenty. Navrhnout vhodný algoritmus kombinování steeringů.
- Steeringy implementovat a vytvořit tak knihovnu steeringů jako rozšíření platformy Pogamut.
- Zhodnotit, zda jsou steeringy vhodným nástrojem pro řízení lidských virtuálních agentů.

Hlavní výsledky práce jsou: návrh steeringů a algoritmu jejich kombinování, knihovna steeringů, grafická aplikace, evaluace navržených steeringů.

1.4 Struktura dokumentu

Práce je rozdělena do 7 kapitol. *Kapitola 1* ukazuje motivaci pro tuto práci, uvádí čtenáře do problematiky a definuje hlavní cíle práce. *Kapitola 2* popisuje technické vlastnosti prostředí, které jsou pro implementaci důležité, a obecně připravuje půdu pro popis implementace steeringů. *Kapitola 3* obsahuje rozbor jednotlivých steeringů, jejich vlastnosti, implementaci i dílčí výsledky. *Kapitola 4* se zabývá kombinacemi steeringů. *Kapitola 5* stručně představuje grafickou aplikaci a její hlavní nástroje. *Kapitola 6* je věnována ukázkovým situacím, obecným výsledkům a použití různých kombinací steeringů. *Kapitola 7* shrnuje výsledky, nedostatky i význam celé práce. Uživatelská dokumentace grafické aplikace, programátorská dokumentace, návod k instalaci i ukázkové soubory jsou součástí přílohy na CD.

2. Obecné vlastnosti implementace navigační vrstvy

Předmětem této práce bylo navrhnout a implementovat navigační vrstvu, která využívá steeringy. Teoretické výsledky práce (myšlenky a algoritmy steeringů a jejich vylepšení či použitý algoritmus kombinování steeringů) jsou samozřejmě využitelné i mimo tuto implementaci, důležité však je, že steeringy byly navrhovány především tak, aby správně fungovaly v rámci této implementace. Ta je závislá na vlastnostech použitého virtuálního prostředí, a tak se může stát, že určitý steering by šlo navrhnout lépe, pokud bychom měli jiné možnosti z hlediska virtuálního prostředí (např. zjišťování informací o okolí). Jelikož je však nemáme, takovými možnostmi se práce dále nezabývá.

Vzhledem k těmto důvodům následuje nejdříve popis základních vlastností použitého virtuálního prostředí a až pak témata zabývající se samotnou navigační vrstvou.

Jako virtuální prostředí je použito 3D virtuální prostředí aplikace UnrealEngine2RuntimeDemo (UE2) [41]. Pro ovládání agentů v prostředí byla použita Java platforma Pogamut [23]. Pogamut obstarává komunikaci s virtuálním prostředím a pro naše potřeby plní funkci lokomoční vrstvy a případně i rozhodovací vrstvy. Pogamut komunikuje s prostředím UE2 pomocí textového protokolu, který je na straně UE2 interpretován rozšířením GameBotsUE2. GameBotsUE2 je implementováno v jazyce UnrealScript – nativním jazyce UE2.

Pro testování implementované navigační vrstvy byla využívána mapa města *EmohawkVille*, která obsahuje množství budov, zdí, stromů, ulic a jiných částí prostředí vhodných pro testování steeringů ve specifických situacích.

Kapitola je rozdělena na dvě části. V první části je popsána komunikace navigační vrstvy s okolím: získávání informací o okolí agenta (tzv. smysly), jaké příkazy dostává navigační vrstva od rozhodovací vrstvy a jaké příkazy posílá lokomoční vrstvě. V druhé části je popsána struktura už samotné vrstvy navigační: jakou roli v ní mají jednotlivé steeringy, co od těchto steeringů vyžadujeme a co mohou používat.

2.1 Komunikace se světem a ostatními vrstvami

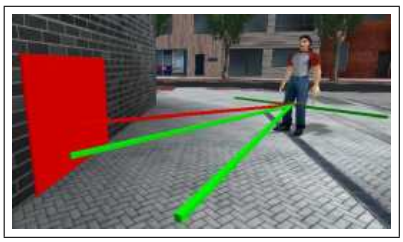
Pogamut poskytuje navigační vrstvě daného agenta vše potřebné k její existenci. Co všechno navigační vrstva potřebuje?

1. Získávat informace o agentovi a jeho okolí.
2. Posílat příkazy lokomoční vrstvě, např. posun směrem k lokaci L rychlostí v, zastavení, natočení k lokaci L, změna typu pohybu (chůze/běh).
3. Dostávat příkazy od rozhodovací vrstvy, např. zapni/vypni steering X, změň parametry steeringu Y, změn celkové parametry navigování.

2.1.1 Smysly

Pogamut umožňuje navigační vrstvě agenta přístup k informacím o okolním světě následujícím způsobem:

1. Neživé překážky – pomocí paprsků. Agent potřebuje vědět, kde se v jeho blízkém okolí vyskytují překážky. K tomu může používat tzv. paprsky, viz Obrázek 2.1. Agent se dozví, když paprsek koliduje s překážkou. Může se zeptat, jak hluboko se paprsek do překážky zanořil, a nechat si spočítat normálový vektor stěny překážky v místě kolize s paprskem (normála vede vždy ven z překážky).



Obrázek 2.1: Paprsky pro detekci neživých překážek. Červeně je znázorněn kolidující paprsek a tečná rovina stěny překážky v místě kolize s paprskem.

   Tento mechanismus má samozřejmě své nevýhody. Snadno se může stát, že se v agentově blízkém okolí bude vyskytovat překážka, která bude mít takový tvar, že do ní agent žádným paprskem nenarazí. Např. proto, že je překážka příliš úzká a vejde se mezi rozpětí agentových paprsků. Nebo že je příliš nízko či vysoko (paprsky jsou většinou rovnoběžné se zemí). Nebo se může stát, že má překážka složitý tvar a normálový vektor stěny v místě kolize může být úplně jiný než v místě o kousek vedle. V použitém světě se vyskytují všechny tyto typy překážek: úzké stromy, nízké lavičky, budovy s nerovnými zdmi. Nicméně virtuální prostředí jiný mechanismus pro detekci překážek nenabízí, při návrhu steeringů byla tedy snaha minimalizovat negativní vliv výše popsaných nebezpečí.

2. Virtuální agenti – přímo. Agent potřebuje znát informace o sobě (lokaci a vektor rychlosti) a někdy i o agentech v jeho okolí. K tomu nemusí používat paprsky, neboť mu Pogamut umožňuje přístup k seznamu všech viditelných agentů v okolí (tedy i sebe sama), jejich lokací a vektorů rychlosti. (Viditelní jsou ti agenti, kteří nejsou příliš daleko ani za nějakou neprůhlednou překážkou.) To je výhodné ze tří důvodů. Za prvé s detekcí úzkých překážek pomocí paprsků jsou problémy – a lidští agenti do tohoto typu překážek určitě patří. Za druhé se takto dají získat informace i o agentech vzdálenějších, než kam dosáhnou paprsky, tedy lze na tyto agenty reagovat dříve. Za třetí kromě jejich lokací jsou známy i jejich vektory rychlosti, a lze tedy využít informaci o tom, jak rychle a jakým směrem se pohybují.

Zde je podstatné si uvědomit, že navigační vrstva nevyžaduje žádnou další komunikaci s okolním světem, speciálně s ostatními agenty, ani nějaké globální řízení na úrovni více agentů. Navigační vrstva se stará vždy pouze o pohyb jediného agenta na základě výše popsaných informací o okolním světě a příkazů od vrstvy rozhodovací.

2.1.2 Rozhraní mezi navigační a lokomoční vrstvou

Lokomoční vrstva poskytuje navigační vrstvě sadu příkazů pro lokomoci agenta:

- Posun směrem k určité lokaci.
- Změna rychlosti.
- Zastavení a případně natočení k určité lokaci.
- Změna typu pohybu (chůze/běh).

Typ pohybu ovlivňuje jednak animaci, která se na agentovi přehrává, jednak reálnou rychlost agenta (rychlost posouvání). Problém je, že aktuálně existuje pouze jedna animace pro chůzi a jedna animace pro běh a nelze ovlivňovat rychlost přehrávání této animace. Pro každý z obou typů pohybu je jedna reálná rychlost posouvání (označme ji *základní rychlost* pro daný typ pohybu), která vypadá s příslušnou animací přirozeně. Pokud se agent posouvá výrazně pomaleji, při každém kroku jakoby sklouzne zpět. Naopak při příliš rychlém posouvání působí dojmem jako by lyžoval. Proto byla v rámci této práce vytvořena speciální škálovací funkce f, která dostane požadovanou rychlost pohybu a upraví ji tak, aby vhodně odpovídala přehrávané animaci. Na škálovací funkci f byly tyto požadavky:

- Nedovolit malé rychlosti chůze.
- Určit hraniční rychlost, která odděluje chůzi a běh.
- Zařídit plynulý přechod mezi chůzí a během.
- Omezit příliš velké rychlosti běhu.

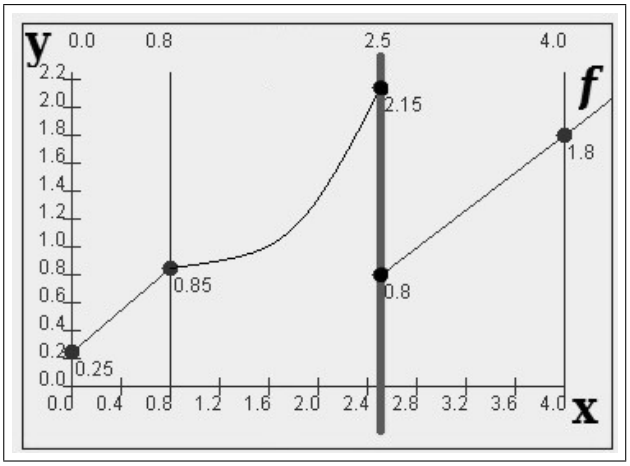
Dle těchto požadavkům byla vytvořena škálovací funkce znázorněná na Obrázku 2.2. Na ose x jsou vstupní hodnoty vektoru rychlosti vydělené základní rychlostí chůze (220 cm_{UT}/s). Na ose y jsou hodnoty multiplikativního faktoru, který se předává lokomoční vrstvě příkazem pro změnu rychlosti. Rychlost agenta je pak součinem tohoto faktoru a základní rychlosti pro daný typ pohybu.

2.1.3 Rozhraní mezi rozhodovací a navigační vrstvou

Navigační vrstva umožňuje rozhodovací vrstvě (de)aktivovat agentovi jednotlivé steeringy a měnit hodnoty jejich parametrů. Dále dovoluje měnit obecné parametry, které se nevážou k žádnému steeringu, mezi které patří `velocity multiplier`, který dovoluje zvětšovat či zmenšovat výslednou rychlost agenta.

2.2 Struktura navigační vrstvy

Jak bylo řečeno v úvodu, navigační vrstva dostává od rozhodovací vrstvy jednodušší úkoly, které má plnit. Za každý z těchto úkolů je zodpovědný jeden steering. Pro složitější chování se steeringy kombinují.



Obrázek 2.2: Škálovací funkce pro převod velikosti vektoru rychlosti.

V implementované navigační vrstvě má agent jednoho *steering managera*. Veškerá komunikace navigační vrstvy agenta s rozhodovací i lokomoční vrstvou probíhá přes steering managera. Rozhodovací vrstva mu posílá požadavky na aktivaci různých steeringů a nastavení hodnot jejich parametrů. V každém tiku je třeba zavolat steering managera, aby se postaral o pohyb v tomto tiku. To se děje jednou za 250ms. Steering manager postupně volá jednotlivé steeringy a od každého dostane jeden vektor, který představuje sílu působící na agenta podle úkolu, který steering plní. Steering manager všechny síly zkombinuje (viz kapitola 4) a vypočítá tak nový vektor rychlosti $\vec{v_t}$, kterým by se měl agent pohybovat v tomto tiku. Vekor $\vec{v_t}$ se vynásobí parametrem `velocity multiplier` a následně se upraví škálovací funkcí f, viz Obrázek 2.2. Upravený vektor označme $\vec{v_t'}$. Podle $\vec{v_t'}$ se určí příkazy pro lokomoční vrstvu:

- Je-li vektor $\vec{v_t'}$ nulový, pošle se příkaz pro zastavení.
- Podle velikosti $\left|\vec{v_t'}\right|$ a hranice mezi chůzí a během se nastaví typ pohybu.
- Pošle se příkaz pohybu ve směru $\vec{v_t'}$.
- Pošle se příkaz pro nastavení rychlosti dle velikost $\vec{v_t'}$.

Může se stát, že bude steering potřebovat, aby se agent zastavil. V takovém případě může vrátit tzv. „požadavek na zastavení“. Společně s tímto požadavkem může určit, jakým směrem se má agent natočit. Podrobněji je proces zastavení popsán v Kapitole 4.

Dále může steering využít tzv. „zpomalovací vektor“, který způsobuje zpomalení agenta (čím je vektor větší, tím se agent více zpomalí), či „rotační vektor“, který má směr kolmý na aktuální vektor rychlosti a jeho velikost je shodná s velikostí aktuálního vektoru rychlosti. Steering si může zvolit, zda se jedná o rotační vektor vpravo, či vlevo.

Síly v navigační vrstvě používají nově zavedenou jednotku N_{UT} (*Newton in Unreal Tournament*). Vzhledem k mechanismu kombinování sil a k zjednodušeným fyzikálním vlastnostem prostředí nelze přesně určit jaké zrychlení udělí hmotnému bodu určité váhy. Dá se ale říci, že pokud chceme, aby se agent stále pohyboval rychlostí 1 $\mathrm{cm}_{UT}/\mathrm{s}$, musíme na něj působit silou 1 N_{UT}. Jednotka vzdálenosti cm_{UT} přibližně odpovídá jednotce cm v reálném světě.

3. Popisy jednotlivých steeringů

Tato kapitola se zabývá popisem samotných steeringů, tedy hlavní částí práce. Jedná se o tyto steeringy a příslušné základní úkoly:

1. Obstacle Avoidance – vyhýbat se překážkám.
2. People Avoidance – vyhýbat se ostatním agentům.
3. Target Approaching – směřovat k určitému místu.
4. Path Following – projít pomyslným koridorem.
5. Wall Following – pohybovat se podél zdi.
6. Leader Following – následovat jiného agenta (vůdce).
   (a) základní – jít vždy za vůdcem (v zadané vzdálenosti).
   (b) formační – jít v určité pozici vůči vůdci (dané úhlem a vzdáleností).
7. Walk Along – dojít společně s jiným agentem k určitému místu.

Každý steering má dva typy chování: základní a rozšířené. Základní chování plní základní úkol. Pro steeringy 1. až 6. (a) platí, že je základní chování založeno na myšlenkách nějakého ze steeringů Craiga W. Reynoldse. Steeringy 6. (b) a 7. jsou nové. Rozšířené chování pak nabízí různá vylepšení, aby se agent choval přirozeněji, pohyboval plynuleji a působil lidštěji. Každému steeringu je věnována jedna podkapitola. Základní struktura podkapitoly je pro všechny steeringy stejná a je následující:

1. Vlastnosti steeringu.
   (a) Požadavky na základní chování.
   (b) Požadavky na rozšířené chování.
   (c) Možné hodnoty výsledného vektoru: co vrací steering jako svůj výsledek.
   (d) Parametry: tabulka parametrů steeringu.
   (e) Informace o okolí: jaké informace o okolí agenta steering potřebuje.
2. Základní chování: použitý algoritmus základního chování steeringu včetně vysvětlení a případných zdůvodnění.
3. Rozšířené chování: důvody pro jednotlivá vylepšení, diskuze možných řešení a algoritmy vylepšení. Většina vylepšení je doplněna ukázkou trajektorií drah pohybů agentů řízených steeringem bez daného vylepšení vs. s daným vylepšením. Tyto obrázky jsou vytvořeny pomocí speciálního nástroje grafické aplikace, podrobněji popsané v kapitole 5, představují tedy skutečná naměřená data odpovídající popisované situaci.[1]
4. Závěr: výhody i nevýhody navrženého steeringu, přehled souvisejících prací, které se zabývají stejným či podobným steeringem.

[1] Jedná se o pohled na město shora, z budov jsou tedy vidět pouze střechy a kvůli perspektivě pohledu není bohužel jasně vidět, jaký má tvar zeď ve výšce agenta.

3.1 Obstacle Avoidance

3.1.1 Vlastnosti steeringu

Požadavky na základní chování

Cílem steeringu Obstacle Avoidance je, aby se agent vyhýbal překážkám. Míří-li agent na překážku, měl by včas změnit směr pohybu, aby se překážce vyhnul. Zároveň by se měl vyhnout co nejplynuleji.

Požadavky na rozšířené chování

Rozšířené chování nabízí lepší řešení dvou typů situací. Prvním typem jsou tzv. *čelní kolize*, kdy jde agent směrem (téměř) kolmo na stěnu překážky. Druhým typem jsou kolize se stromy a jinými úzkými překážkami.

Možné hodnoty výsledného vektoru

K detekci překážek v blízkém okolí používá agent paprsky. Jestliže se alespoň jeden paprsek dostane do konfliktu s překážkou, vrací steering vektor odpovídající odpudivé síle od překážky. Pokud žádný paprsek nekoliduje, vrací steering nulový vektor.

Parametry

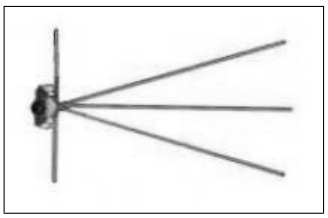
Parametry steeringu jsou popsány v Tabulce 3.1.

Název	Popis	Hodnoty
`Repulsive Force`	Velikost odpudivé síly od překážek.	0–2000 N_{UT}
`Force Order`	Řád síly. Ovlivňuje, jak roste velikost síly s klesající vzdáleností od překážky. Hodnota 1 představuje linární závislost. Hodnoty vyšší než 1 způsobují ostřejší reakce na velmi blízké překážky a slabší reakce na překážky vzdálenější.	1–10
`Front Collisions`	Nabízí lepší řešení *čelních kolizí*. Bez tohoto parametru se agent při *čelní kolizi* od překážky bezdotykově odrazí. S tímto parametrem se stočí podél překážky a pokračuje podél její stěny.	boolean
`Tree Collisions`	Vylepšuje základní chování o vhodnější řešení kolizí se stromy a jinými úzkými překážkami.	boolean

Tabulka 3.1: Parametry steeringu Obstacle Avoidance.

Informace o okolí

Steering používá 5 paprsků: 1 dlouhý přední ve směru agentova natočení, 2 dlouhé šikmé přední (od předního jsou odchýlené každý o 15°) a 2 krátké boční, viz Obrázek 3.1.

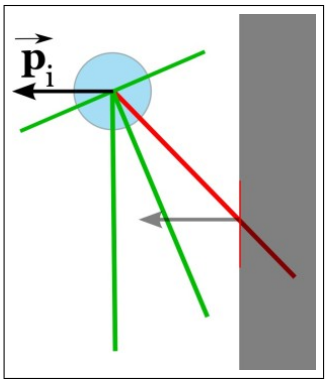


Obrázek 3.1: Paprsky.

3.1.2 Základní chování

Návratový vektor steeringu OBSTACLE AVOIDANCE je součet vektorů $\vec{p_1}, \ldots, \vec{p_n}$, kde n je počet kolidujících paprsků. Každý z těchto vektorů odpovídá odpudivé síle příslušné danému paprsku. Pro i-tý ($i \in \{1, \cdots, n\}$) kolidující paprsek se určí místo kolize paprsku s překážkou. Směr vektoru $\vec{p_i}$ je pak shodný se směrem normálového vektoru[2] stěny překážky v místě kolize s paprskem.

Výpočet vektoru $\vec{p_i}$ je znázorněn na Obrázku 3.2. Zde levý přední šikmý paprsek (červeně) koliduje s překážkou. Šedá šipka představuje směr normály stěny překážky v místě kolize s paprskem. Stejný směr má pak vektor $\vec{p_i}$ (černá šipka), který představuje odpudivou sílu od překážky.



Obrázek 3.2: Výpočet $\vec{p_i}$.

Velikost vektoru $\vec{p_i}$ se určí pomocí vzorce:

$$W \cdot F \cdot \left(\frac{2 \cdot D}{R}\right)^O,$$

kde W je konstanta určující důležitost daného paprsku (přední paprsek má hodnotu $W = 1$, šikmý přední 0.8 a boční 0.5), F je hodnota parametru `Repulsive Force`, D je délka části paprsku zanořené do překážky (tzn. rozdíl celkové délky paprsku a vzdálenosti agenta od místa kolize paprsku s překážkou), R je délka paprsku a O je hodnota parametru `Force Order`. Vzorec pro výpočet velikosti vektoru $\vec{p_i}$ je navržený tak, aby měl následující vlastnosti:

1. Čím je agent k překážce blíže, tím je vektor $\vec{p_i}$ větší, tedy tím více je od překážky odpuzován. Díky tomu se agent vyhýbá překážkám plynule.

2. Je-li agent vzdálený tak, že je paprsek zanořený přesně z poloviny, je velikost vektoru právě $W \cdot F$. To odpovídá jakési základní odpudivé síle pro daný typ paprsku. Ta má pro přední paprsek hodnotu F, pro šikmé přední paprsky $0,8 \cdot F$ a pro boční $0,5 \cdot F$.

3. Největší váhu má přední paprsek, dále šikmé přední paprsky a nakonec boční. Jde-li totiž agent k překážce čelně, je potřeba mnohem větší síla, aby směr svého pohybu změnil a aby vhodně zpomalil. Pokud však koliduje jen boční paprsek, pak se agent vyskytuje vedle překážky a nemusí měnit směr ani rychlost pohybu tak výrazně.

4. Je-li parametr `Force Order` roven 1, odpudivá síla od překážky roste lineárně s klesající vzdáleností. Pokud je řád vyšší, jsou reakce na vzdálenější překážky menší a na bližší překážky větší. Vzdálenějšími překážkami zde myslíme překážky vzdálené tak, že je paprsek zanořen z malé části. Naopak u bližších překážek je paprsek zanořen z velké části.

[2]Normálový vektor stěny překážky vede ve směru ven z překážky.

3.1.3 Rozšířené chování – řešení *čelních kolizí*

Nevýhody základního chování

V základním chování záleží na úhlu, který svírá vektor rychlosti agenta a normálový vektor stěny překážky. Jestliže je tento úhel menší než 177°, agent se vyhne překážce poměrně přirozeně: stočí se podél. Pokud je úhel mezi 177° a 180°, (agent jde kolmo na stěnu překážky), těsně před překážkou obrátí směr chůze a jakoby se od ní bezdotykově odrazí. Těmto situacím říkáme *čelní kolize*. Nejdříve se podívejme, proč dochází v základním chování k tomuto odrážení, a dále bude následovat popis, jak lze tyto situace řešit lépe.

Co se stane, když míří agent přímo kolmo na stěnu překážky? Pro jednoduchost uvažujme rovnou širokou zeď. V takovém případě kolidují právě tři paprsky: jeden přední a dva šikmé přední. Vektory $\vec{p_i}$ příslušné těmto paprskům budou mít směr přesně opačný, než je směr agentova vektoru rychlosti. Když se v daném tiku složí tyto vektory s vektorem rychlosti, nastane jeden ze tří případů:

1. Nový vektor rychlosti bude mít stejný směr jako ten původní, avšak menší velikost. Agent tedy bude v příštím tiku opět směřovat k překážce, ale půjde pomaleji.

2. Vektory se přesně odečtou a agent se zastaví.

3. Převáží vektor od překážky a agent se otočí o 180° a bude pokračovat ve směru od překážky.

První dva případy neřeší situaci příliš dobře. Pokud by složení vždy dopadlo prvním případem, agent by za několik tiků do překážky narazil, steering by tedy nesplňoval základní požadavky. Pokud by časem nastala druhá možnost, agent by se v danou chvíli zastavil. Pak by sice do překážky nenarazil, ale ani toto chování není přirozené[3]. Protože by byl stále příliš blízko překážce, v příštím tiku by ho odpudivá síla otočila stejně jako ve třetím případě.

Chceme-li, aby se agent překážce vyhnul, a máme-li na výběr pouze tyto tři možnosti, je třeba, aby v horizontu několika tiků došlo ke třetí možnosti. Podle toho jsou upraveny konstanty steeringu. Ideální průběh základního chování je takový, že agent začne nejdříve zpomalovat a pak se teprve otočí.

Tímto získáme chování, které vypadá jako by se agent od překážek bezdotykově odrážel. Rozbor možností se zaměřil na speciální situaci, kdy kolidují všechny tři paprsky a agent jde přímo kolmo na stěnu překážky. Pokud by kolidovaly jen některé z těchto tří paprsků (např. jen prostřední, třeba že by zeď nebyla tak široká), výsledek by byl stejný. Pokud by agent nešel přesně kolmo na překážku, ale pod úhlem blízkým kolmému ($\pm$ 3°), otočil by se o menší úhel než 180°, ale přesto by to vypadalo, jako by se odrazil. Toto nemusí být vždy nevhodné. Například jedná-li se o rybičku v akváriu, může vypadat poměrně přirozeně, pokud plave směrem ke stěně akvária a následně se od ní odrazí. Lidé se však od překážek většinou neodrážejí.

Mohlo by se zdát, že se nejedná o příliš časté situace. To ale není pravda, obzvláště na mapě města s kolmými ulicemi a pravoúhlými rohy budov a místností.

[3]Pokud překážka zrovna nepředstavuje nějakou hrozbu, které se agent lekl natolik, že není schopen pohybu. To však není zrovna typická situace.

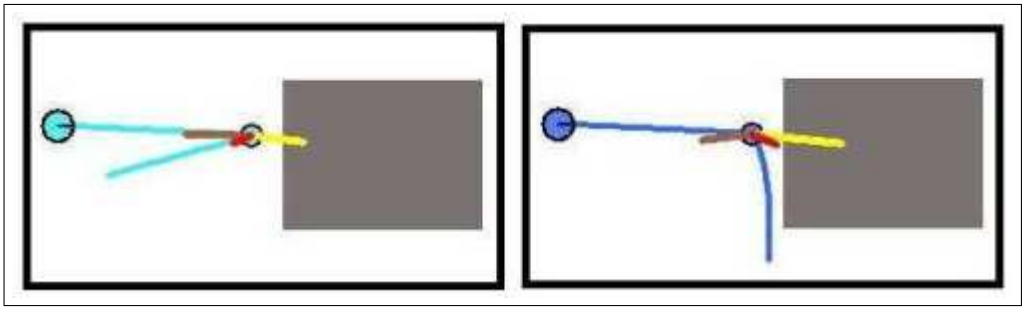
Proto vznikl parametr `Front Colllisions`, který nabízí řešení těchto *čelních kolizí*.

Řešení *čelních kolizí* pomocí parametru `Front Collisions`

Čelní kolize jsou ty situace, když koliduje přední paprsek nebo jeden z šikmých předních paprsků s překážkou a úhel vektoru rychlosti agenta a vektoru $\vec{p_i}$, kde i je index příslušného kolidujícího paprsku, je 177°– 180°. Parametr `Front Collisions` tyto situace detekuje a zařídí, že se agent od překážky neodrazí, ale stočí se a pokračuje podél ní.

K řešení *čelních kolizí* používá steering s parametrem `Front Collisions` rotační vektor $\vec{r}$, který je kolmý na vektor rychlosti agenta a míří doprava či doleva podle toho, zda jde agent k překážce spíše zleva či zprava. Pokud jde agent přímo kolmo na překážku, směr (levý/pravý) se zvolí náhodně. Velikost vektoru $\vec{r}$ se nastaví jako polovina velikosti vektoru $\vec{p_i}$ příslušný danému paprsku. K vektoru $\vec{p_i}$ se pak přičte vektor $\vec{r}$. Výsledný vektor se použije jako vektor příslušný danému paprsku. Pomocí této rotační síly se agent před překážkou stočí a plynule pokračuje podél ní.

V levé části Obrázku 3.3 je zobrazeno, jak se agent vyhne překážce, pokud není zapnutý parametr `Front Collisions`. V pravé části obrázku je zobrazena dráha agenta se stejnými parametry (počáteční pozice agenta, okolí atd.), avšak zapnutým parametrem `Front Collisions`. Hnědě je znázorněna síla steeringu Obstacle Avoidance, žlutě původní vektor rychlosti a červeně nový vektor rychlosti v daném tiku. Větším kolečkem je znázorněna startovní lokace agenta.



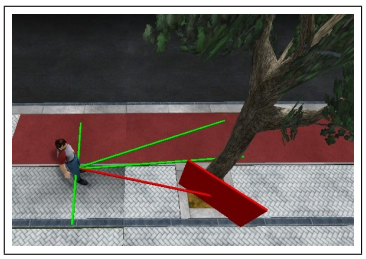
Obrázek 3.3: Vlevo: základní chování steeringu Obstacle Avoidance v případě hrozící čelní kolize s překážkou. Vpravo: stejná situace za použití parametru `Front Collisions`.

3.1.4 Rozšířené chování – řešení *kolizí se stromy*

Nevýhody základního chování

Druhé vylepšení se týká situací, které označujeme jako *kolize se stromy*. Obecně se jedná kolize s úzkými kulatými překážkami, tedy překážkami se zhruba kruhovým půdorysem a šíří ostře menší než vzdálenost předního a šikmého předního paprsku v nejvzdálenějším místě. V naší mapě světa jsou to nejčastěji stromy. Kromě tvaru překážky je podstatná poloha agenta vůči překážce. V problémových *kolizích se stromy* koliduje se stromem jediný paprsek – a to šikmý přední paprsek – a přední paprsek prochází vedle stromu z druhé strany, viz Obrázek 3.4.

Z Obrázku 3.4 vidíme, že by agentovi stačilo jen lehce změnit směr pohybu doleva a překážku obejít zleva. Nicméně vzhledem ke kulatému tvaru překážky míří směr vektoru kolmého na stěnu překážky napravo od agenta. Proto se v základním chování agent stočí doprava a vyhne se stromu zprava. Ve výsledku to navíc vypadá, že i když o stromu ví, míří chvíli přímo na něj – a až pak se horlivě snaží srážce vyhnout. Cílem parametru `Tree Collisions` je situaci *kolize se stromem* detekovat a zajistit, aby se agent vyhnul překážce ze správné strany.



Obrázek 3.4: *Ukázka kolize se stromem.*

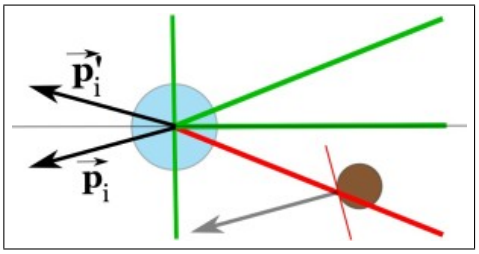
Řešení *kolizí se stromy* pomocí parametru `Tree Collisions`

Situace *kolize se stromem* je určená dvěma podmínkami:

1. S překážkou koliduje právě jeden paprsek – jeden z předních šikmých paprsků. Označme i index tohoto paprsku. Připomeňme, že $\vec{p_i}$ má směr normálového vektoru stěny překážky v místě kolize s paprskem.

2. Směr vektoru $\vec{p_i}$ je takový, že míří do stejné poloroviny ohraničené přímkou vektoru rychlosti agenta, jako míří vektor šikmého předního kolidujícího paprsku, viz Obrázek 3.4 a Obrázek 3.5.

Problém je, že $\vec{p_i}$ míří na druhou stranu, než kudy by se měl agent překážce vyhnout. Místo něj se tedy použije vektor $\vec{p_i'}$, který je s ním osově souměrný podle osy dané vektorem rychlosti agenta, viz Obrázek 3.5.

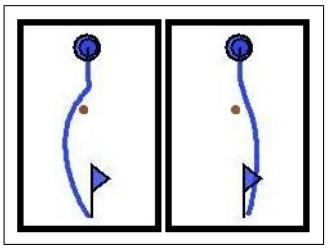


Obrázek 3.5: Nákres *kolize se stromem.*

Na Obrázku 3.6 vidíme, jak vypadá kolize se stromem s vypnutým (vlevo), resp. zapnutým (vpravo) parametrem `Tree Collisions`. Hnědá tečka představuje strom. Startovní pozice agenta (modré kolečko) je zvětšená pro názornost, ve skutečnosti je agent široký podobně jako strom, což zdůrazňuje potřebu parametru `Tree collisisons`.

3.1.5 Závěr

Výše navržený steering OBSTACLE AVOIDANCE se stará o to, aby se agent správně, včas a zároveň plynule vyhýbal překážkám. Na vzdálenější překážky reaguje mírněji, na bližší překážky reaguje výrazněji. Díky tomu mění směr pohybu postupně a plynule se překážce vyhne. Rozšířené chování nabízí alternativní a pro lidské agenty patrně přirozenější řešení dvou typů situací – *čelních kolizí* a *kolizí se stromy*.



Obrázek 3.6: Srovnání *kolize se stromem*. Vlevo: základní chování. Vpravo: s parametrem `Tree collision`.

Pokud se agent dozví o překážce včas, dokáže se jí poměrně přirozeně a plynule vyhnout. Největší nebezpečím je však to, že se o překážce dozví pozdě, nebo

vůbec. K tomu mohu vést dvě příčiny: žádný paprsek nezasáhne překážku (buď je překážka mezi paprsky, nebo je mimo dosah paprsků), nebo se agent pohybuje k překážce příliš rychle. S tím souvisí fakt, že je v použitém virtuálním prostředí poměrně nízká frekvence tiků (jeden tik trvá 250ms). Může se tedy stát, že v jednom tiku je agent od překážky ještě daleko, ale v následujícím tiku je už příliš blízko.

Výše navržený steering se snaží počítat se všemi zmíněnými úskalími a minimalizovat jejich následky. Vzhledem k tomu, že se může o překážce dozvědět poměrně pozdě, je třeba na ni reagovat dostatečně výrazně. Na druhou stranu se steering snaží o plynulé změny pohybu. Síly a konstanty jsou navrženy tak, aby byly co nejlépe splněny oba protichůdné požadavky.

Vyhýbání se překážkám je jedna ze zásadních schopností, které obvykle od virtuálních agentů vyžadujeme. Není tedy překvapivé, že se jedná o poměrně prozkoumané téma. Následuje srovnání výše navrženého steeringu Obstacle Avoidance s jinými pracemi, které se zabývají stejným úkolem.

Jedná-li se o statické překážky, častým řešením je využití předpočítaných dat o terénu, jako např. *navigation graph, navigation mesh, waypoint graph* apod. Z těchto dat lze obvykle získat informace o statických překážkách, nebo alespoň o cestách, které nevedou přes žádné statické překážky. Pak lze vypočítat průchod prostředím po těchto cestách bez rizika kolize se statickými překážkami, např. pomocí algoritmu A^* [16, 27, 28], Floyd-Warshall, Dijkrstra apod. Možné reprezentace prostoru jsou rozebrány v [40], algoritmy hledání cesty (tzv. *pathfinding)* jsou popsány v [29]. Toto řešení je velmi užitečnou alternativou ke steeringu Obstacle Avoidance. V této práci se jím zabývá steering Path Following.

V mnoha případech stačí následovat předpočítanou cestu, o které víme, že nevede přes překážky. Pokud se však agent od cesty odchýlí, vyskytne se na cestě nezaznačená překážka (třeba dynamická), nebo předpočítaná data o terénu nemáme k dispozici, je potřeba umět překážky detekovat a reagovat na ně. Což je přesně úkol steeringu Obstacle Avoidance.

Různé implementace steeringu Obstacle Avoidance se liší především tím, jak může agent detekovat překážky. Například Obstacle Avoidance [34, 35] C. W. Reynoldse využívá toho, že tvar všech překážek lze aproximovat koulí. Agent má před sebou pomyslný válec (ve směru jeho natočení, shodné šířky a parametrizovatelné délky) a počítá průnik tohoto válce s okolím. Díky tomu je schopný detekovat všechny překážky, do kterých by měl časem narazit (agent se pohybuje pouze dopředu).

V používaných virtuálních prostředích ale často nemáme možnost počítat průnik pomyslného válce před agentem s okolím. Bývá však možnost počítat průnik tzv. paprsků s okolím, nejčastěji paprsků vycházejících z jednoho místa v těle agenta. Tak je tomu i v našem případě. Náš steering Obstacle Avoidance je tedy spíše podobný jinému steeringu Craiga W. Reynoldse, který se nazývá Containment [35] a má za cíl udržovat agenta uvnitř určitého prostoru. Steering navržený v této práci se ale navíc snaží minimalizovat negativní následky detekce překážek pomocí paprsků v kombinaci s řídkou frekvencí tiků. Tyto vlastnosti společně s přirozenějším vyhýbáním se překážkám v některých výše popsaných situacích shrnují přínos steeringu Obstacle Avoidance navrženého v této práci.

3.2 People Avoidance

3.2.1 Vlastnosti steeringu

Požadavky na základní chování

Steering People Avoidance se stará o to, aby se agent správně vyhýbal ostatním agentům, aby do nich nevrážel, ani se k nim příliš nepřibližoval.

Poznámka

Mohlo by se zdát, že se jedná o obyčejné vyhýbání se překážkám (avšak živým), a že by tedy i toto zadání mohl řešit steering Obstacle Avoidance. Nicméně jsou zde dva základní rozdíly:

1. Virtuální agenti se mohou pohybovat. Je dobré uvažovat i informaci o jejich vektoru rychlosti. Např. běží-li dva agenti proti sobě, je potřeba výrazná a rychlá reakce, aby nedošlo ke srážce. Pokud jde agent směrem ke statické překážce, stačí slabší reakce. A nakonec se může stát, že se dráhy agentů mají křížit, ale v místě křížení bude jeden z nich mnohem dříve než ten druhý. Pak není třeba žádná reakce pro předejití srážky, neboť by k ní dojít ani nemělo. Se znalostí rychlosti lze tedy přirozeněji předejít kolizi.

2. Na detekci virtuálních agentů v okolí není potřeba používat paprsky, neboť máme k těmto informacím přímý přístup. To je výhodné, jelikož paprsky neposkytují informace o rychlosti, nemají takový dosah a navíc mají problém s detekcí úzkých překážek, mezi které lidští virtuální agenti určitě patří.

Tedy ač jsou požadavky na steeringy Obstacle Avoidance a People Avoidance na první pohled velmi podobné, každý steering vyžaduje vlastní řešení.

Požadavky na rozšířené chování

Rozšířené chování se týká situací, kdy se steering kombinuje s jinými (zde uvažujeme Target Approaching). Požadujeme, aby měli virtuální agenti schopnost obejít druhého agenta, když jim překáží v cestě, vhodně zpomalit či zrychlit a předejít tak srážce.

Možné hodnoty výsledného vektoru

Pokud nehrozí srážka s jiným agentem, vrací steering nulový vektor. Jinak vrací sílu, díky které se předejde srážce. V základním chování se jedná přímo o odpudivou sílu od agenta, v rozšířeném pak navíc o rotační, zpomalující, či zrychlující síly.

Parametry

Parametry steeringu jsou popsány v Tabulce 3.2.

Informace o okolí

Agent potřebuje znát lokace (vždy) a vektory rychlostí (v rozšířeném chování) agentů ve viditelném okolí (ti, co nejsou příliš daleko ani za pevnou překážkou).

Název	Popis	Hodnoty
`Repulsive Force`	Odpudivá síla od ostatních agentů.	0–1000 N_{UT}
`Distance`	Minimální vzdálenost od ostatních agentů.	50–2000 cm_{UT}
`Circumvention`	Je-li přepínač zapnutý, agent je schopný obejít jiného agenta, který mu překáží v jeho dráze.	boolean
`Projection`	V případě `Circumvention` si agent promítá svůj pohyb na `Projection` tiků dopředu a zkoumá, zda se do tehdy příliš nepřiblíží k jinému agentovi.	0–30
`Deceleration`	Je-li přepínač zapnutý, agent je schopný vhodně zpomalit, aby se přirozeně vyhnul srážce či jevu označovanému jako *vytlačování*.	boolean
`Acceleration`	Je-li přepínač zapnutý, agent je schopný vhodně zrychlit, aby se přirozeně vyhnul srážce či jevu označovanému jako *vytlačování*. Je-li navíc zapnutý i přepínač `Acceleration`, agent si vybere, zda zpomalí či zrychlí podle toho, co je v dané chvíli výhodnější.	boolean

Tabulka 3.2: Parametry steeringu People Avoidance.

3.2.2 Základní chování

Označme A agenta řízeného steeringem People Avoidance. Návratová hodnota steeringu se spočítá jako součet vektorů $\vec{p_1}, \ldots, \vec{p_n}$, kde n je počet jiných agentů bližších, než je parametr `Distance`. Každý z těchto vektorů odpovídá odpudivé síle od daného agenta. Pro i-tého ($i \in \{1, \ldots, n\}$) agenta, označme ho B_i, má vektor $\vec{p_i}$ směr od B_i k A a velikost:

$$F \cdot \frac{\max(0, D - |AB_i|)}{D},$$

kde F je hodnota parametru `Repulsive Force`, D je hodnota parametru `Distance` a $|AB_i|$ je vzdálenost agentů A a B_i. Vzorec pro výpočet velikosti vektoru $\vec{p_i}$ je navržený, aby splňoval následující vlastnosti:

1. Je-li druhý agent vzdálený alespoň na vzdálenost `Distance`, síla je nulová.
2. Čím je druhý agent blíže, tím je síla větší (nejvýše však `Force`). To mimo jiné napomáhá plynulému chování.

V základním chování vrací tedy steering součet odpudivých sil od agentů, kteří jsou příliš blízko. Ostatních agentů si nevšímá.

3.2.3 Rozšířené chování – *obcházení*

Nevýhody základního chování

Na Obrázku 3.7 je ukázkový příklad situace *obcházení*: agenti jdou za sebou, mají stejný směr vektoru rychlosti, avšak ten zadní jde rychleji. To může znamenat, že zadní více spěchá a potřebuje předního agenta předejít.

Obrázek 3.7:

Jak se v této situaci zachovají agenti, kteří používají steering People Avoidance v základním chování? Zadního agenta označme A, předního B. Agent A dojde až k agentovi B, pravděpodobně trochu zpomalí, avšak stále půjde za agentem B. Pokud byl jeho původní vektor rychlosti velký, může se stát, že odpudivá síla od agenta B nebude mít dostatečnou váhu, aby zabránila kolizi, tedy agent A do agenta B vrazí. Pokud bude naopak velká odpudivá síla, může převážit původní vektor rychlosti a otočit agenta A o 180°.

Jelikož vektory rychlosti obou agentů mají stejný směr a síla za People Avoidance od agenta B má směr přesně opačný, tak po složení není možné, aby se směr nového vektoru rychlosti nějak vychýlil.

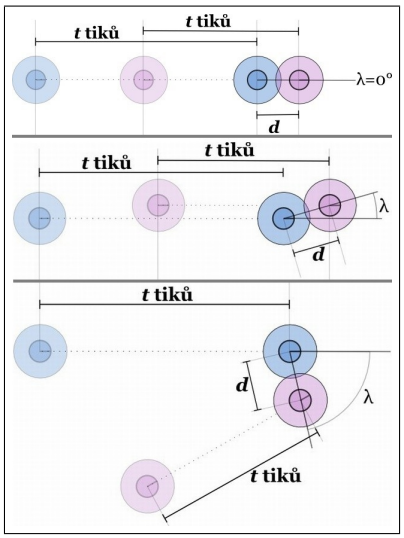
Situacím, kdy je potřeba agenta obejít, ale základní chování k tomu nestačí, říkáme *obcházení*[4]. Řešením těchto situací se zabývá parametr `Circumvention`, který navíc využívá volitelný parametr `Projection`.

Řešení *obcházení* pomocí parametru `Circumvention`

Prve je třeba přesně definovat situace *obcházení*. Mají společné to, že je potřeba použít rotační sílu, která agenta vychýlí ze směru pohybu. Čím jsou určeny? Předpokládejme, že se budou agenti i nadále pohybovat se stejným vektorem rychlosti jako nyní.[5] Určí se nejmenší počet tiků, za který si budou agenti blíže, než je hodnota parametru `Distance`. Daný počet tiků označme t. Pokud takové t neexistuje (agenti se tak blízko nepřiblíží), nebo je t vyšší než parametr `Projection`, není třeba nic dále řešit a parametr `Circumvention` chování steeringu nijak neovlivňuje. V opačném případě se spočítá rotační vektor $\vec{r}$, který se přičte k výslednému vektoru steeringu.

Označme L'_A lokaci agenta A za t tiků, L'_B lokaci agenta B za t tiků, $\vec{v_A}$ vektor rychlosti agenta A, $\vec{v_B}$ vektor rychlosti agenta B, d vzdálenost lokací L'_A a L'_B a nakonec označme $\vec{v_{AB}}$ vektor vedoucí od L_A k L_B. Nejdříve určíme stranu rotačního vektoru $\vec{r}$ (pravá/levá) a následně jeho velikost.

Strana rotačního vektoru $\vec{r}$ bude přesně opačná, než na jaké straně bude agent B vůči agentovi A za t tiků. Matematicky se to dá určit podle toho, zda leží lokace L'_B v levé či pravé polorovině[6] od přímky určené vektorem $\vec{v_A}$ a lokací L'_A. Pokud leží přímo na této přímce, strana se určí náhodně. Tři možné konfigurace situace *obcházení* jsou znázorněny na Obrázku 3.8.



Obrázek 3.8: Tři možné ukázky situace *obcházení*.

[4] Nemusí se jednat jen o ty situace, kdy jdou agenti přesně za sebou.

[5] Toto se sice pravděpodobně nestane, ale pro výpočet je to výhodné. Pokud se jejich směr změní, v daný tik se opět všechny síly všech agentů přepočítají a budou uvažovat nový vektor rychlosti. To zařídí poměrně plynulý pohyb, avšak zároveň včasné předcházení kolizí.

[6] Můžeme uvažovat 3D poloprostor, ale stačí uvažovat pouze první dvě složky vektorů. Agenti se totiž sice pohybují v 3D světě, ale třetí souřadnice je konstantní.

Velikost rotačního vektoru $\vec{r}$ se pak spočítá podle vzorce:

$$2 \cdot F \cdot K_1 \cdot \max(K_2, K_3), \text{ kde}$$

$$K_1 = \max\left(0, \min\left(1, \frac{P-t}{P}\right)\right),$$

$$K_2 = \max\left(0, \frac{D-d}{D}\right),$$

$$K_3 = \max\left(0, \frac{\frac{\pi}{2}-\lambda}{\frac{\pi}{2}}\right),$$

F je hodnota parametru `Repulsive Force`, P je hodnota parametru `Projection`, D je hodnota parametru `Distance`, d je vzdálenost lokací L'_A a L'_B a λ je úhel, který svírá $\vec{v_A}$ s vektorem $\vec{v_{AB}}$. Na $\vec{r}$ mají vliv tři faktory:

1. K_1: za jak dlouho má dojít ke srážce: čím kratší doba (malé t), tím by měla být velikost $\vec{r}$ větší. Dokud jsou agenti daleko od srážky, stačí pouze malé změny směru. Pokud je doba t rovna parametru `Projection`, pak je $K_1 = 0$, pokud je však nejvýše 1, pak $K_1 = 1$.

2. K_2: jak blízko si budou agenti za t tiků: čím blíže, tím by měla být velikost $\vec{r}$ větší. Pokud by měli být sobě poměrně vzdálení, opět stačí spíše malá změna pohybu. Pokud budou vzdálení přesně `Distance`, pak je $K_2 = 0$, čím méně, tím je K_2 vyšší, nejvýše však 1.

3. K_3: jaký úhel bude svírat vektor $\vec{v_A}$ s vektorem $\vec{v_{AB}}$, na Obrázku 3.8 znázorněný jako λ: čím menší, tím by měla být velikost $\vec{r}$ větší. Pokud půjdou téměř ve stejném směru, je potřeba směr pohybu změnit více, než když půjdou v podstatě každý jinam. Pokud bude úhel přesně π, pak je $K_3 = 0$, čím méně, tím je K_3 vyšší, nejvýše však 1.

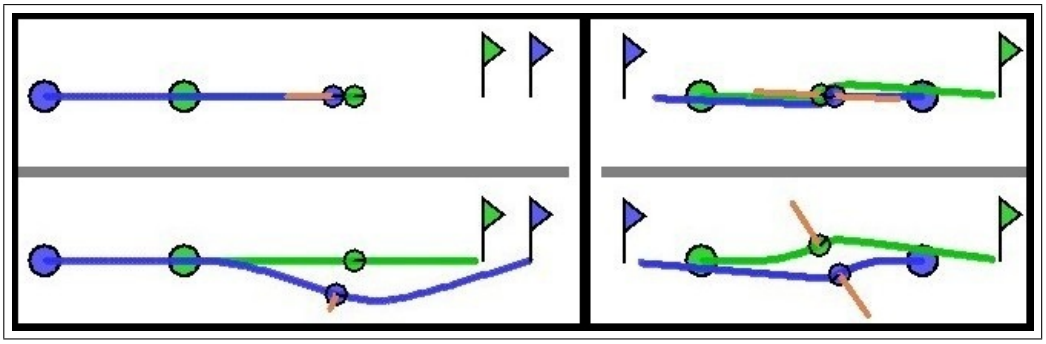
Všechny koeficienty K_1, K_2 a K_3 mají tedy hodnoty mezi 0 a 1. Čím je jejich hodnota vyšší, tím bude výsledná síla větší. Koeficienty K_2 a K_3 rozlišují, jak těsná by byla srážka, kdyby agent nikam neuhýbal. Stačí, aby z jednoho hlediska byla srážka těsná, a výsledná síla by měla být velká, i kdyby druhé hledisko nebylo podstatné. Proto obsahuje vzorec pro velikost rotační síly $\max(K_2, K_3)$.

Parametr `Circumvention` dává agentům schopnost obejít se i ve chvíli, kdy to základní chování neumožňuje. Je-li pomocný parametr `Projection` vyšší (např. 16), agent se zabývá srážce dostatečně dopředu, že jí dokáže předejít velmi plynule. Pokud se naopak agentům kříží dráhy, ale po celou dobu se nedostanou k sobě příliš blízko, steering jejich pohyb nijak neupravuje. Obrázek 3.9 ukazuje, jak se agenti zachovají v situaci *obcházení* s vypnutým (nahoře) či zapnutým (dole) přepínačem `Circumvention`.

3.2.4 Rozšířené chování – *vytlačování*

Nevýhody základního chování

Podívejme se na druhý typ problémových situací; ty nazýváme *vytlačování*. Představme si, že agenti A a B jdou každý ke svému cíli (např. mají oba steering

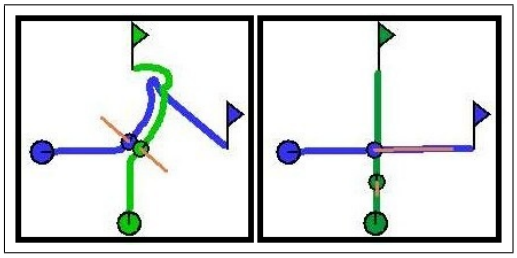


Obrázek 3.9: Dvě ukázky situace *obcházení*. Vlevo šli agenti za sebou a zadní (modrý) měl vyšší rychlost. V základním chování (vlevo nahoře) byl modrý agent zpomalen, ale nepodařilo se mu zeleného agenta obejít. Vidíme, že síla steeringu (oranžově) působí přesně proti směru pohybu. Za použití parametru `Circumevention` (vlevo dole) je síla steeringu rotační a modrý agent může předejít agenta zeleného. Vpravo šli agenti proti sobě. V základním chování (vpravo nahoře) do sebe vrazili. V rozšířeném chování (vpravo dole) se přirozeně obešli.

Target Approaching) tak, že se jim kříží dráhy pohybu. Uvažujme jen ty situace, kdy by se v místě křížení potkali zároveň. Vypadá přirozeně, když jeden z nich zpomalí (čímž toho druhého jakoby pouští) a druhý případně zrychlí (aby toho prvního příliš nezdržel). Ve výsledku se v místě křížení drah nepotkají ve stejnou dobu, tedy ani jeden z nich nebude muset měnit směr pohybu – neboť stačí změna rychlosti pohybu.

Jak se ve výše popsané situaci budou chovat agenti bez steeringu People Avoidance? Jsou-li parametry scény nastaveny přesně tak, že se v místě křížení budou agenti vyskytovat v jeden čas, je velká pravděpodobnost, že dojde k jevu, který je zde nazýván jako „vytlačování“. Agenti do sebe totiž vrazí a pokračují směrem představující průměr jejich původních cílů, přičemž ani jeden nebude chtít pustit toho druhého na původní směr.

Pokud budou mít oba agenti aktivovaný steering People Avoidance v základním chování, sice do sebe nevrazí, avšak přesto může dojít k jevu „vytlačování“, jako např. na Obrázku 3.10 vlevo. Pokud by byl navíc zapnutý přepínač `Circumvention`, jev vytlačování by byl dokonce významně zvětšený.



Obrázek 3.10: Situace *vytlačování*. Vlevo je základní chování steeringu People Avoidance. Síla steeringu (oranžově) působí přímo od druhého agenta. Kvůli tomu nemůže jít ani jeden ke svému cíli. Vpravo je rozšířené chování s parametry `Deceleration` a `Acceleration`. Modrý agent zrychlil (síla steeringu je ve směru pohybu) a zelený agent zpomalil (síla steeringu je proti směru pohybu), čímž přirozeně předešli kolizi.

Proto obsahuje steering PEOPLE AVOIDANCE parametry `Deceleration` a `Acceleration`, díky kterým se obdobným problémům předejde. Opět se využije i parametr `Projection`.

Řešení *vytlačování* pomocí parametrů `Deceleration` a `Acceleration`

Jak reální lidé poznají, že se jim kříží dráhy a v místě křížení by se potkali zhruba ve stejný okamžik? To je otázka například pro kognitivní vědce. Avšak pravděpodobně je člověk schopný odhadnout, jak rychle jde on i ten druhý a zda by se v místě křížení střetli, nebo by jím prošli každý v jinou dobu.

Parametry `Deceleration` a `Acceleration` to počítají následovně. Označme L_A, L_B lokace agentů A a B, $\vec{v_A}, \vec{v_B}$ jejich vektory rychlosti. Opět budeme předpokládat, že budou agenti pokračovat se stejným vektorem rychlosti, jako mají teď. Označme S lokaci křížení jejich budoucích drah (průsečík polopřímek určených lokacemi a vektory rychlostí). Dále označme t minimum z počtu tiků, za jak dlouho by dorazil k lokaci S agent A, a počet tiků pro agenta B. Lokace, na které by oba agenti dorazili za t tiků označme L_A^t, L_B^t. Je třeba předcházet „vytlačování“, pokud jsou splněny dvě podmínky:

1. Vzdálenost lokací L'_A a L'_B je menší než parametr `Distance`. Tedy agenti se setkají na lokaci S přibližně ve stejnou dobu (a dostanou se příliš blízko k sobě).

2. Počet tiků t je menší než parametr `Projection`, tedy situace nenastane za příliš dlouho.

Nechť jsou splněny obě podmínky. Pak je ještě potřeba rozhodnout, zda má agent zrychlit či zpomalit – a jak moc.

Pokud má být agent A na lokaci S dříve než agent B a agent A má zapnutý přepínač `Acceleration`, tak agent A zrychlí. Pokud by měl dorazit na lokaci S dříve naopak agent B a agent A má zapnutý přepínač `Deceleration`, tak zpomalí. Pokud by tam měli dorazit přesně ve stejném tiku, tak si agent A zvolí náhodně, zda zrychlí, či zpomalí.

Proč je potřeba řešit i zrychlování i zpomalování? Nemůžeme počítat s tím, že druhý agent má aktivní steering pro PEOPLE AVOIDANCE a natož zapnuté parametry `Deceleration` a `Acceleration`. Přesto se druhému agentovi chceme přirozeně vyhnout.

Jako základ výsledného vektoru $\vec{v}$ pro v případě `Deceleration`, resp. `Acceleration` se vezme tzv. zpomalovací vektor popsaný v Kapitole 2. Pokud má agent zrychlit, tak je $\vec{v}$ znegován. Následně se velikost $\vec{v}$ vynásobí hodnotou v rozsahu $\langle 0, 1)$ podle vzorce $K_1 \cdot K_2$, kde

$$K_1 = \frac{M - |L_A\ S|}{M}, \qquad K_2 = \frac{D - |L_A^t\ L_B^t|}{D},$$

M je vzdálenost, kterou by urazil agent A za `Projection` tiků, pokud by šel aktuální rychlostí, a D je hodnota parametru `Distance`. Na velikost zpomalovacího, resp. zrychlovacího vektoru $\vec{v}$ mají vliv dva faktory:

1. K_1: jak moc je místo střetu S daleko – hodnota na škále 0 až 1, přičemž čím je místo blíže, tím je hodnota větší.

2. K_2: jak moc jsou lokace L'_A a L'_B od sebe vzdálené – opět hodnota na škále 0 až 1, přičemž čím jsou si blíže, tím je hodnota větší.

Pokud se jedná o zrychlení, velikost vektoru se ještě vynásobí třemi, aby bylo zrychlení znatelné (což vyplývá z mechanismu kombinování steeringů, popsaného v Kapitole 4). Nakonec pokud je zapnutý i parametr `Circumvention`, jeho výpočet se použije až tehdy, když je vyjde vektor $\vec{v}$ nulový. Obrázek 3.10 ukazuje, jak se agenti zachovají v situaci *vytlačování* s vypnutými (vlevo) či zapnutými (vpravo) parametry `Deceleration` a `Acceleration`.

3.2.5 Závěr

Steering People Avoidance se stará o to, aby se agent včas, plynule a pokud možno přirozeně vyhýbal ostatním agentům. V základním chování se snaží především předejít srážkám s ostatními agenty. V rozšířeném chování se ostatním agentům navíc umí vyhýbat pomocí obcházení, zpomalování a zrychlování.

Steering People Avoidance je patrně nejčastěji zkoumaným a diskutovaným steeringem a tato bakalářská práce nepřináší nijak výrazně nový přístup. Podobné myšlenky lze najít například v těchto pracích: [14, 43, 42, 13, 11, 15]. Výhodou zde navrženého steeringu je, že výsledného chování dosahuje pomocí jednoduchých (myšlenkově i výpočetně) pravidel a sil, výsledné chování je poměrně přirozené a plynulé, zároveň je dobře parametrizovatelné a především je součástí širší navigační vrstvy, která zajišťuje kombinace s řešením jiných úkolů.

Na závěr následuje ukázka využití parametrů základního i rozšířeného chování. Výsledky testování tohoto steeringu totiž ukázaly, že různé nastavení parametrů steeringu vede k velmi zajímavým vlastnostem. Takto lze například ovlivnit, zda se agent chová spíše jako jedinec, nebo jako součást davu, viz Kapitola 6. Obdobně lze podpořit znázornění[7], jakou má agent povahu, stáří či náladu. Agent, který má působit jako dominantní, až arogantní člověk, který před sebe nikoho nepouští, ani ostatním neuhýbá, nebude používat parametry rozšířeného chování. Ostýchavý či dokonce humanofóbní jedinec bude mít vysoké hodnoty parametrů `Distance` a `Repulsive Force` (např. 600 cm_{UT}a 250 N_{UT}). Agent vyskytující se pouze mezi přáteli využije hodnoty parametru `Distance` spíše nižší (okolo 300 cm_{UT}, případně až 150 cm_{UT}). Agent představující zamyšleného člověka, který si nevšímá, že do něj někdo vráží, nebo si toho všímá pozdě, bude mít buď základní chování, či rozšířené chování s nízkou hodnotou parametru `Projection` (např. 3 tiky) a `Distance` (např. 150 cm_{UT}). Starý člověk nebude využívat parametr `Acceleration`, neboť mu jeho fyzické možnosti neumožňují zrychlovat, ale parametr `Deceleration` může využít poměrně dobře. Naopak spěchající agent (např. vytížený manager) raději zrychlí s nadějí, že ostatní zpomalí, aby do něj nevrazili. Pro něj tedy bude vhodné zapnout parametr `Acceleration`, ale nezapínat parametr `Deceleration`.

Podobných využití by šlo vymyslet mnoho dalších. Podstatné je, že si rozhodovací vrstva může nastavení zvolit v závislosti na povaze, náladě, či věku agenta. Díky tomu dovoluje steering People Avoidance nejen předcházet kolizím, ale i utvářet charakter agenta.

[7] Např. společně s vhodnými animacemi a texturou agenta. To však musí zajistit rozhodovací vrstva.

3.3 Target Approaching

3.3.1 Vlastnosti steeringu

Požadavky na základní chování

Pomocí steeringu Target Approaching se agent pohybuje směrem k zadané cílové lokaci. Jakmile k ní dorazí, zastaví se. Pokud by se však od ní následně vzdálil, začne ho cílová lokace přitahovat zpět.

Požadavky na rozšířené chování

V základním chování působí lokace na agenta konstantní přitažlivou silou. V rozšířeném chování lze specifikovat, jak se mění hodnota této síly v závislosti na vzdálenosti agenta od cílové lokace. V jednotlivých úsecích může být síla nulová, konstantní, či obecně lineární. Pokud je síla nastavena jako záporná, působí cílová lokace na agenta silou odpudivou.

Rozšířené chování navíc dovoluje zadat více lokací a ke každé určit velikost síly v závislosti na vzdálenosti agenta od cíle. Pak už se nejedná doslova o „cílové lokace“, ale přesto je tak budeme nazývat.

Možné hodnoty výsledného vektoru

V základním chování působí cílová lokace na agenta vždy konstantní přitažlivou silou, tedy výsledný vektor míří vždy od agenta k cílové lokaci a má stále stejnou velikost. Pokud je agent dostatečně blízko cíli, vrací steering požadavek na zastavení. V rozšířeném chování je výsledný vektor součet přitažlivých a odpudivých sil k jednotlivým lokacím.

Parametry

Parametry steeringu jsou popsány v Tabulce 3.3.

Název	Popis	Hodnoty
`Target Location`	Cílová lokace	Location
`Attractive Force`	Velikost přitažlivé síly k cílové lokaci.	0–1000 N_{UT}
`List of l,f`	Seznam lokací a příslušných sil.	List<l,f>

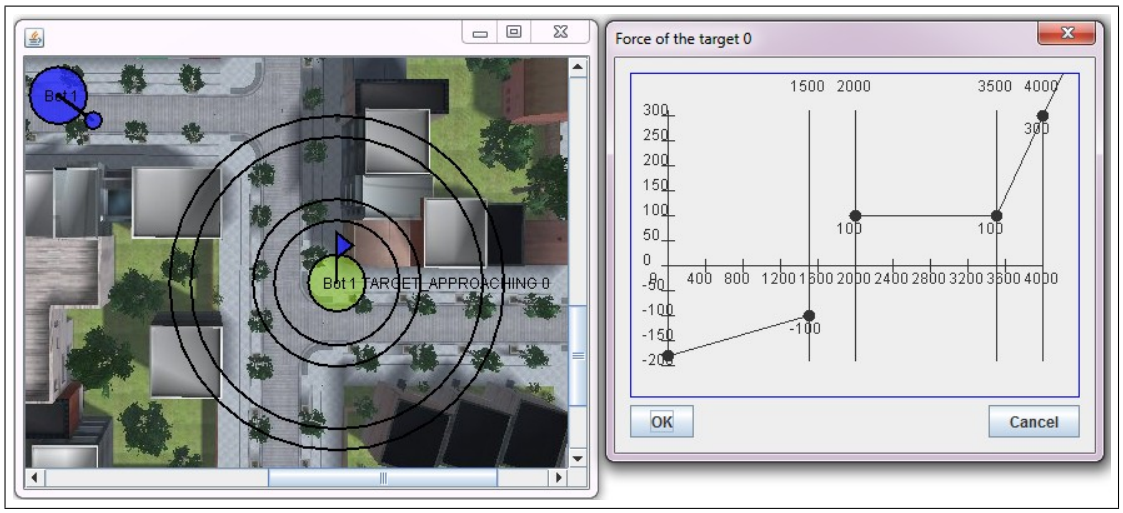
Tabulka 3.3: Parametry steeringu Target Approaching.

V rozšířeném chování se používá parametr `List of l,f`. Prvky tohoto seznamu jsou dvojice *Location* a *Function*, což je funkce určující velikost síly v závislosti na vzdálenosti agenta od cíle. Tato funkce je na jednotlivých úsecích lineární a je určena hodnotami v přechodech mezi úseky. Formálně se tedy jedná o seznam trojic (jedna trojice pro jeden přechod):

- *distance* – vzdálenost agenta od lokace *Location*
- *value* – hodnota síly ve vzdálenosti *distance*

- *continues* – boolean, který určuje, jak se vypočítá hodnota síly v následujícím úseku. Pokud je hodnota *continues* nepravda, síla v následujícím úseku je nulová.

Na Obrázku 3.11 je ukázka možného průběhu funkce *Function* pro jeden cíl.



Obrázek 3.11: Ukázka průběhu přitažlivé síly k cíli v závislosti na vzdálenosti od cíle v rozšířeném chování steeringu Target Approaching. Jedná se o obrázek z grafické aplikace. Vlevo je pohled na město z nadhledu. Modré kolečko představuje startovní lokaci agenta, zelené kolečko s modrou vlajkou jeho cílovou lokaci. Kruhy kolem cíle představují přechody mezi úseky funkce *Function*. Vpravo je průběh této funkce. Její zápis by představoval seznam pěti trojic: $[0, -180, true]$, $[1500, -100, false]$, $[2000, 100, true]$, $[3500, 100, true]$, $[4000, 300, true]$. Když je agent od cíle daleko, působí na něj přitažlivá síla k cíli, která se postupně zmenšuje. Dorazí-li na úroveň druhého největšího kruhu, tedy na vzdálenost 3500 cm_{UT}, bude na něj působit už jen konstantní síla velikosti 100 N_{UT}. V mezikruží ve vzdálenostech 2000 cm_{UT} až 1500 cm_{UT} na něj steering Target Approaching žádnou silou působit nebude. Pokud by se agent dostal dovnitř nejmenšího kruhu, působila by na něj dokonce odpudivá síla od cíle: čím by byl blíže k cíli, tím by ho cíl více odpuzoval.

Informace o okolí

Agent nepotřebuje znát žádné jiné informace o okolí, než svou vlastní lokaci.

3.3.2 Základní chování

Návratový vektor steeringu Target Approaching záleží na vzdálenosti agenta od cíle. V případě, že agent ještě nedorazil k cíli, má návratový vektor směr od lokace agenta k cílové lokaci dané parametrem `Target Location`. Jeho velikost je hodnota parametru `Attractive Force`. Pokud je agent již dostatečně blízko cíli (jeho vzdálenost od cíle je menší než 150 cm_{UT}), steering vrací požadavek na zastavení.

3.3.3 Rozšiřující chování

Návratový vektor rozšiřujícího chování steeringu TARGET APPROACHING se spočítá jako součet vektorů $\vec{p_1}, \ldots, \vec{p_n}$, kde n je velikost seznamu `List of l,f`, tedy počet cílů. Vektor $\vec{p_i}$ je vektor k i-tému ($i \in \{1, \ldots, n\}$) cíli. Jeho velikost se určí podle hodnoty funkce `Function` pro daný cíl. Je-li velikost kladná, směr vektoru $\vec{p_i}$ je od lokace agenta k lokaci i-tého cíle, jedná se tedy o sílu přitažlivou. Pokud je velikost záporná, jedná se o sílu odpudivou od lokace i-tého cíle a má směr od lokace i-tého cíle k lokaci agenta.

3.3.4 Závěr

Tento steering je velmi jednoduchý, avšak zároveň velmi užitečný. Základní chování dovoluje určit místo, které na agenta působí přitažlivou silou. V rozšířeném chování je možno určit takových míst více a navíc i definovat, jak je agent k danému místu přitahován či odpuzován v závislosti na jeho vzdálenosti od daného místa. Takto lze označit například místa, která představují jakési nebezpečí a agent se jim má vyhnout. Jedním z možných využití je umístění těchto nežádoucích míst doprostřed křižovatek ulic, díky čemuž se bude agent vyhýbat prostředkům křižovatek. Ukázku obsahuje Kapitola 6.

Základní chování tohoto steeringu neodpovídá žádnému ze steeringů Craiga W. Reynoldse. Nejbližší mu je steering SEEK [34, 35], ve kterém je však agent navigován vždy rovnou přímo k cíli. Namísto toho ve výše navrženém steeringu TARGET APPROACHING přitahuje cíl agenta (jakousi gravitační silou) a může chvíli trvat, než se stočí k cíli. Jeho pohyb je pak plynulejší, což je u lidských agentů podstatné. (Chceme-li, aby se agent stočil k cíli rychleji, nastavíme steeringu vyšší parametr `Attractive Force`.)

Rozšířené chování představuje určité zobecnění Reynoldsových steeringů SEEK, FLEE (agent směřuje od cíle) a ARRIVAL (agent směřuje k cíli, ale před cílem postupně zbrzďuje). Pro dosažení stejného výsledku jako ve steeringu ARRIVAL by bylo nutno, aby mohl steering TARGET APPROACHING používat zpomalovací vektor, což je jedno z možných rozšíření práce. Ve stávající verzi (vzhledem k mechanismu kombinací steeringů) lze agenta pouze velmi mírně zpomalovat.

Jiným z možných rozšíření je možnost definovat navíc stranu, ze které má agent k cíli dorazit. Tímto rozšířením se zabývá i práce [3].

3.4 Path Following

3.4.1 Vlastnosti steeringu

Požadavky na základní chování

Steering Path Following řídí agenta tak, aby prošel zadaným koridorem. Ten je určen *středovou cestou* (jakási osa koridoru) a vzdáleností od této cesty. Středová cesta je dána seznamem lokací. Můžeme ji chápat i jako grafovou cestu, jejíž vrcholy jsou lokace a hrany tvoří spojnice vždy dvou po sobě následujících lokací. V celé podkapitole budeme používat tuto grafovou terminologii. Od základního chování požadujeme, aby agent prošel koridorem pokud možno plynule a přímočaře, ale ne přímo po hranách cesty.

Poznámka

Proč nechceme, aby chodil agent přímo po hranách cesty? Ideově i výpočetně by to bylo velmi jednoduché. Toto chování má ale tři hlavní nevýhody:

1. Vždy při dosažení vrcholu změní agent směr pohybu. Dvojice hran s hodně odlišným směrem pak vedou k příliš ostrým změnám směru pohybu.

2. Jestliže se na cestě vyskytne ostrá zatáčka, agent dojde vždy až do vrcholu zatáčky. Přirozenější by bylo si cestu zkrátit po vnitřní straně zatáčky, nevyskytuje-li se tam nějaká překážka. Bylo by pěkné, aby steering uměl řešit i tento problém, není to však vyžadováno od základního chování.

3. Pokud by mělo koridorem procházet více agentů, navíc např. v obou směrech, měli by problém se sobě vyhnout a projít koridorem plynule.

Požadavky na rozšířené chování

Rozšířené chování nabízí dva parametry pro plynulejší pohyb. Dále zkoumá možnosti, jak zařídit, abych chodil agent spíše po vnitřních stranách zatáček, je-li to pro něj v danou chvíli výhodné (např. to má blíže). Zároveň musí být však agentovi umožněno projít zatáčku i po vnější straně (a nepůsobit na agenta příliš velkou silou). To může být potřeba například tehdy, pokud je na vnitřní straně zatáčky nějaká překážka.

Možné hodnoty výsledného vektoru

Návratový vektor může nabývat těchto hodnot:

1. Pohybuje-li se agent uvnitř koridoru ve směru cesty, není jeho pohyb nijak upravován. Steering tedy vrací nulový vektor.

2. Jestliže jde agent proti směru cesty, musí se otočit a pokračovat správným směrem. Steering vrací vektor od minulého vrcholu k následujícímu.

3. Jestliže by měl agent vyjít z koridoru, je směr jeho pohybu upraven tak, aby pokračoval uvnitř koridoru. Steering vrací sílu ke spojnici minulého a následujícího vrcholu.

4. Dojde-li agent na konec cesty, zastaví se. Steering vrací požadavek na zastavení.

5. V rozšířeném chování může být k návratovému vektoru přičítána usměrňovací síla ve směru aktuální hrany.

Parametry

Parametry steeringu jsou popsány v Tabulce 3.4.

Název	Popis	Hodnoty
`Path Force`	Síla (odpudivá síla od stěn koridoru).	0–1000 N_{UT}
`List of Locations`	Seznam vrcholů (lokací) cesty.	List<Location>
`Distance`	Vzdálenost od cesty.	0–1000 cm_{UT}
`Projection`	Agent si promítá svou pozici na `Projection` tiků dopředu a zkoumá, zda se za tuto dobu dostane ven z koridoru. Vhodné jsou hodnoty mezi 5 a 20. Vyšší hodnoty (okolo 15) umožňují plynulejší chování a včasnější reakce na změny cesty.	0–50
`Regulation`	Usměrňující síla. Vhodné hodnoty (cca 50 N_{UT}) zajistí plynulejší chování a často i průchod na vnitřní straně zatáček.	0–2000 N_{UT}

Tabulka 3.4: Parametry steeringu Path Following.

Informace o okolí

Agent nepotřebuje znát žádné jiné informace o okolí, než svou vlastní lokaci.

3.4.2 Základní chování

Steering Path Following si pamatuje dvojici vrcholů, mezi kterými se agent vyskytuje. Tuto dvojici nazvěme *minulý* a *následující vrchol*. Na začátku se nastaví minulý vrchol jako aktuální lokace agenta a následující vrchol jako první vrchol na cestě. Steering si pamatuje vzdálenost od následujícího vrcholu v minulém tiku. Výpočet návratového vektoru steeringu probíhá ve 4 fázích.

1. V první fázi výpočtu se provede kontrola, zda agent nedosáhl konce cesty. Pokud ano, výpočet steeringu končí s požadavkem na zastavení agenta.

2. Ve druhé fázi výpočtu je třeba zkontrolovat, zda agent nedosáhl následujícího vrcholu. Pokud ano, dvojice minulého a následujícího vrcholu na cestě se posune o 1 vrchol na cestě dál. Co to znamená dosáhnout následujícího vrcholu? Nabízí se dva možné přístupy:

   (a) Agent dosáhne následujícího vrcholu právě tehdy, když se dostane za úroveň následujícího vrcholu, tedy za přímku kolmou na spojnici minulého a následujícího vrcholu procházející následujícím vrcholem.

(b) Agent dosáhne následujícího vrcholu právě tehdy, když se dostane do určité vzdálenosti od tohoto vrcholu. Největší smysl pro tuto vzdálenost dává šířka koridoru (tedy vzdálenost od cesty).

Pokud vede cesta celou dobu téměř rovně, tyto dva přístupy dávají velmi podobné chování. Avšak v případě ostré zatáčky vede druhý přístup k dřívějšímu a výhodnějšímu dosáhnutí vrcholu (především pokud se používá rozšiřující parametr `Regulation`).

Na druhou stranu, pokud se steeringu nepodaří udržet agenta uvnitř koridoru a zrovna kolem následujícího vrcholu projde mimo koridor, začne se nesmyslně vracet k tomuto vrcholu. Proto je využita kombinace – sjednocení obou přístupů. Agent dosáhne vrcholu právě tehdy, když dosáhne vrcholu dle alespoň jednoho z nabízených přístupů.

3. Ve třetí fázi výpočtu je potřeba zkontrolovat, zda se agent nepohybuje proti směru cesty, tedy zda není vzdálenost od následujícího vrcholu větší než v minulém tiku. Pokud ano, steering vrací vektor směrem od minulého k následujícímu vrcholu, přeškálovaný na velikost:

$$F \cdot \frac{\alpha}{\pi},$$

kde F je hodnota parametru `Path Force` a α je úhel aktuální hrany a vektoru rychlosti agenta měřený v radiánech – čím je větší, tím se více odchýlil od správného směru cesty a steering vrací větší sílu.

4. V poslední fázi se kontroluje, zda agent nevybočuje mimo koridor, viz obrázek 3.12. Spočte se L'_A, předpokládaná lokace agenta za k tiků, pokud by se po tu dobu pohyboval s aktuálním vektorem rychlosti. V základním chování je k pevně konstanta 5, tedy se díváme na 5 tiků dopředu. Dále označme h přímku procházející aktuální hranou, tedy minulým a následujícím vrcholem. Pokud je lokace L'_A od h dál, než povoluje parametr `Distance`, je vypočítána síla $\vec{a_h}$, který agenta přitáhne dovnitř koridoru. Vektor $\vec{a_h}$ má směr od lokace v příštím tiku směrem k h. Velikost vektoru $\vec{a_h}$ se určí podle vzorce:

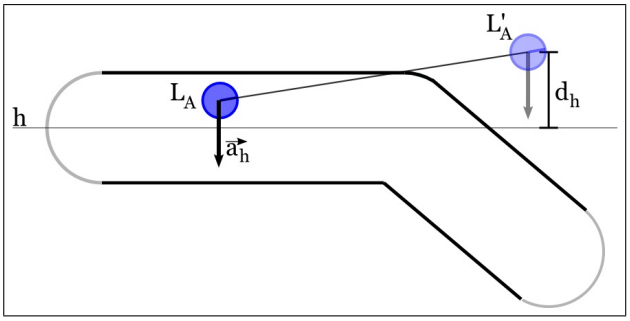
$$F \cdot \min\left(0, \frac{d_h - D}{D}\right) + 30,$$

kde F je hodnota parametru `Path Force`, D je hodnota parametru `Distance` a d_h je vzdálenost L'_A od h. Čím více by měl agent vybočit mimo koridor, tím bude síla větší. Pokud by se měl agent dostat jen těsně za hranici koridoru, síla bude alespoň 30, aby ho dokázala přitáhnout zpět.

3.4.3 Rozšiřující chování

Nevýhody základního chování

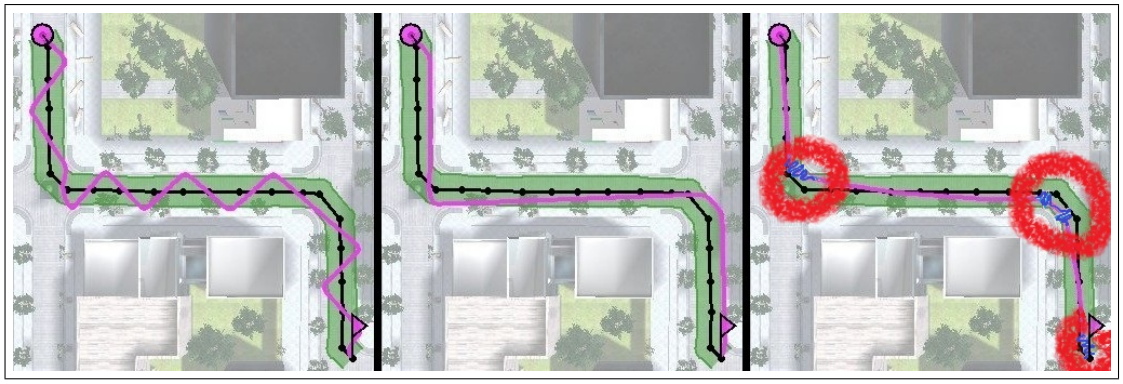
Základní chování zařídí, aby šel agent správným směrem po cestě uvnitř koridoru, avšak ne vždy je jeho pohyb úplně plynulý.



Obrázek 3.12: Výpočet vektoru $\vec{a_h}$, který představuje přitažlivou sílu ke středové cestě steeringu PATH FOLLOWING. L_A je aktuální lokace agenta, L'_A je předpokládaná lokace agenta za k tiků.

Vylepšení *plynulosti* pomocí parametru `Projection`

První vylepšení představuje parametr `Projection`. Ten nastavuje délku projekce agentova pohybu, tedy na kolik tiků dopředu se dívá, výše označovanou jako k. S delší projekcí se lépe předchází vybočení z koridoru. Pokud se však zvolí projekce příliš dlouhá, může to vést k příliš ostrým změnám pohybu či až natáčení agenta ze strany na stranu, pokud je koridor úzký. Naopak příliš krátká projekce (kratší než 5) vede ke klikatému pohybu – agent se nedívá příliš dopředu a jakmile zjistí, že by měl vybočit mimo koridor, přitáhne ho příliš velká síla dovnitř. Srovnání různých hodnot parametru `Projection` ukazuje Obrázek 3.13.



Obrázek 3.13: Srovnání různých hodnot parametru `Projection`. Postupně zleva: 1, 5 a 50. Vhodné hodnoty jsou 5 až 20. Základní hodnota je 5. Příliš malé hodnoty způsobují klikatou chůzi. Agent se totiž dozví příliš pozdě, že vybočuje z koridoru. Síla $\vec{a_h}$ je pak příliš velká, tedy ho stočí tak, že za chvíli bude vybočovat z koridoru na druhé straně. Příliš velké hodnoty parametru `Projection` mohou také způsobovat nepřirozené chování. V ostřejších zatáčkách (alespoň 90°) se agent zastaví a chvíli se na místě otáčí ze strany na stranu. Tato místa jsou na obrázku zvýrazněna červeně. Způsobuje to příliš dlouhá projekce, která se v jednom tiku dostane mimo koridor na jedné straně a v následujícím tiku na druhé straně. Riziko tohoto nepřirozeného chování je ještě vyšší, pokud je koridor úzký.

Možné rozšíření: Jedním z možných rozšíření by bylo využití informaci o tom, za jak dlouho by měl agent vybočit z koridoru. Pokud za dlouhou dobu, stačí upravit směr jeho pohybu málo. Pokud za krátkou, je třeba použít větší

sílu. Díky tomu by agent měnil směr pohybu ještě plynuleji. Nicméně to není příliš potřeba, pokud se navíc používá parametr `Regulation`.

Vylepšení *plynulosti* pomocí parametru `Regulation`

V tomto rozšířeném chování se k návratového hodnotě steeringu vždy (není-li agent na konci cesty) přičte tzv. „usměrňující síla“. Ta má směr od minulého vrcholu k následujícímu a její velikost je dána parametrem `Regulation`. Má-li parametr `Regulation` vhodnou hodnotu (např. 50), vylepšuje základní chování v těchto vlastnostech:

1. Tento parametr výrazně napomáhá plynulosti pohybu. V základním chování agent vždy dojde až téměř k hranici koridoru a teprve tehdy změní směr svého pohybu. Za použití usměrňující síly reaguje dříve a plynuleji.

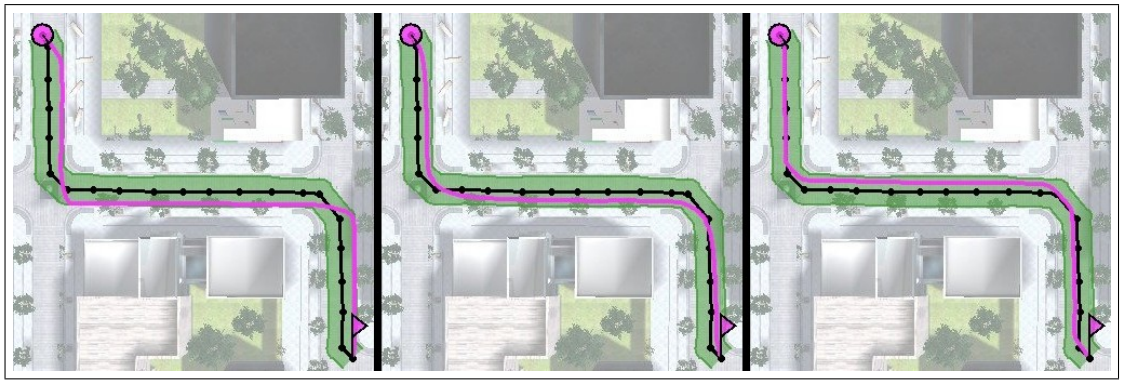
2. Toto rozšiřující chování dokáže vhodně upřednostňovat chůzi po vnitřní straně zatáček. Jakmile se agent dostane do blízkosti vrcholu ostré zatáčky, posune se následující vrchol o 1 dále na cestě. Usměrňující síla tedy začne působit ve směru spojnice nové dvojice minulého a následujícího vrcholu. Tato síla nesmí být příliš velká, aby agent na tuto změnu nezareagoval příliš výrazně. Zároveň pokud bude síla příliš malá, nebude změna znát. Pokud bude síla „přiměřená“ (např. 50), postupně se stočí ve směru následujícího vrcholu, tedy projde zatáčkou po vnitřní straně.

3. Mohlo by nastat nebezpečí, že tento parametr způsobí opět chůzi po hranách cesty. Nicméně je nutno si uvědomit, že usměrňující síla obecně nemá směr od aktuální lokace agenta k následujícímu vrcholu, avšak od minulého vrcholu k následujícímu. Proto půjde agent pouze rovnoběžně se spojnicí těchto vrcholů. Tedy není-li agent uprostřed cesty, nepůjde přímo po dané spojnici.

Obrázek 3.14 ukazuje srovnání různých hodnot parametru `Regulation`. Moc malé hodnoty návratový vektor příliš nezmění. Moc velké hodnoty mají dvě nevýhody. Zaprvé mohou způsobit ostré změny směru pohybu v době, kdy agent dosáhne nějakého vrcholu na cestě. Zadruhé je výsledný pohyb málo flexibilní. Jedním z požadavků základního chování bylo, aby se agent mohl pohybovat libovolně ve směru cesty uvnitř koridoru. V kombinaci s ostatními steeringy se totiž může stát, že bude potřeba, aby šel agent po druhé straně koridoru, vyhnul se překážce uvnitř koridoru, jinému agentovi apod. Steering Path Following sice může upřednostňovat jistý směr pohybu, ale neměl by tím znemožnit ostatní směry, které nejsou ve sporu se základními požadavky.

3.4.4 Závěr

Steering Path Following naviguje agenta pomyslným koridorem. Steering se stará o to, aby šel agent koridorem správným směrem, nevybočoval z koridoru a na konci koridoru se zastavil. Zároveň nechává agentovi dostatečnou volnost v pohybu uvnitř koridoru. Základní chování je implementováno přesně dle pravidel steeringu Path Following Craiga W. Reynoldse [34, 35]. Rozšířené chování pak



Obrázek 3.14: Srovnání různých hodnot parametru `Regulation`. Postupně zleva: 10, 50, a 200. Doporučená hodnota je 50. Příliš malé hodnoty ovlivňují pohyb jen mírně. Můžeme si všimnout, že ukázka zde vlevo je velmi podobná ukázce na Obrázku 3.13 uprostřed. Naopak příliš velké ho ovlivňují příliš prudce a výrazně. Na obrázku vpravo si můžeme všimnout, že trajektorie se více podobá lomené čáře. Vždy v místě dosažení následujícího vrcholu na cestě se výrazně změní směr pohybu ve směru následující hrany. Výsledný pohyb tedy není dostatečně plynulý.

nabízí dvě možnosti, jak zařídit, aby byl pohyb plynulejší a aby agent chodil spíše po vnitřní straně zatáček.

Jediné riziko představují překážky uvnitř koridoru. PATH FOLLOWING je většinou využíván tak, že zadaná středová cesta přes žádné překážky nevede. Mohou se ale vyskytovat mimo středovou cestu. Pro většinu překážek stačí, aby měl agent zároveň aktivní steering OBSTACLE AVOIDANCE. Přesto se může stát, že se agent o nějakou překážku zasekne (např. musí-li najednou projít úzkou chodbou).

Steering PATH FOLLOWING je z jednoho hlediska odlišný od ostatních steeringů implementovaných v rámci této práce. Ty ostatní spadají do definice *reaktivního chování*. Tedy nic dopředu neplánují a vždy se rozhodují jen podle aktuálního okolí. Většinou nepotřebují žádnou paměť, nebo jen velmi malou (např. pro 5 vektorů, jednu lokaci apod.). Často se pod pojmem steeringy myslí právě tyto reaktivní steeringy – a to především ty, jejichž úkolem je vyhýbání se živým i neživým objektům. Proto jsou steeringy někdy kritizovány, že ve složitějších situacích nevypadají realisticky ani uvěřitelně (především na úrovni lidského pohybu) a že se nedokáží vypořádat se slepými uličkami a složitějším prostředím [4, 5]. U kombinace steeringů OBSTACLE AVOIDANCE, PEOPLE AVOIDANCE a TARGET APPROACHING jsou tyto námitky do jisté míry oprávněné, avšak ne u steeringu PATH FOLLOWING. Pokud dostane „vhodnou“ cestu prostředím, je schopný navigovat agenta velmi přirozeně a plynule.

Daň za tyto výhody je zřejmá: „vhodnou cestu“ musí někdo naplánovat a spočítat. V našem případě se o tyto věci stará rozhodovací vrstva[8]. Výhodné je, pokud má k dispozici předzpracovaná data o prostředí, např. *síť navigačních bodů* (*navigation graph*). Její vrcholy (tzv. *navigační body*) a hrany jsou voleny tak, aby nevedly přes překážky, a daly se díky tomu používat ke snadnému průchodu sítí bez kolizí.

V typickém případě použití steeringu PATH FOLLOWING zná rozhodovací vrst-

[8]Stejně tak by to mohl být úkol přímo navigační vrstvy, ale toto dává rozhodovací vrstvě větší možnosti.

va startovní a požadovanou cílovou lokaci agenta a podle toho naplánuje v síti navigačních bodů cestu ze startu do cíle. Většinou vybírá cestu nejkratší, k čemuž může použít známé grafové algoritmy: Dijkstrův, Floyd-Warshallův či A*.[9] Nakonec předá rozhodovací vrstva steeringu PATH FOLLOWING seznam lokací navigačních bodů vybrané cesty. Technicky se nemusí jednat o lokace navigačních bodů, ale o libovolné lokace, nicméně je výhodné, pokud středová osa cesty nevede přes žádné překážky.

Existuje mnoho prací, které se zabývají následováním určité cesty. Nejčastěji navigují agenta přímo po zadané cestě. Jak bylo popsáno výše, to vede k mnoha problémům, například ostrým změnám směru a příliš předvídatelnému pohybu. Pro plynulejší pohyb se někdy používají různé techniky pro proložení několika bodů spojitě diferencovatelnou funkcí. Místo lomené čáry agent následuje tuto spojitě diferencovatelnou křivku, díky čemuž mění směr pohybu plynuleji. K tomuto účelu se používá např. *Catmull-Rom spline*, který zaručuje spojité první derivace (použito např. v [27]), nebo *Bézier spline*, který je sice o něco náročnější na výpočet, ale zaručuje spojité druhé derivace, tedy ještě hladší trajektorii (použito např. v [12]).

Ve srovnání s tímto přístupem sice nezaručuje výše navržený steering PATH FOLLOWING (v rozšířeném chování) spojité derivace výsledné trajektorie, avšak na pohled se pohybuje agent dostatečně plynule – a to za velmi malého úsilí. Zadanou cestu není třeba interpolovat ani jinak upravovat a výpočet v jednom tiku představuje pouze několik jednoduchých matematických výpočtů (vzdálenost bodu od přímky, průsečík dvou přímek apod.).

Obdobným problémem se zabývá i tzv. *Corridor Map Method* [8]. Ta navíc dovoluje uživateli zadat různou šířku koridoru v různých místech (díky tomu může agent plynule vejít do úzké chodby) a randomizovat výsledný tvar trajektorie. Obě tato vylepšení by mohla představovat možná rozšíření výše navrženého steeringu.

[9] Výběr „ideálního“ plánovacího algoritmu představuje oblíbený a diskutovaný problém, není to však předmětem této práce. Grafická aplikace této práce používá Floyd-Warshallův algoritmus jako hotovou komponentu platformy Pogamut.

3.5 Wall Following

3.5.1 Vlastnosti steeringu

Požadavky na základní chování

Cílem steeringu Wall Following je navigovat agenta podél zdi. Zdí myslíme stěnu libovolné dostatečně široké překážky, nejčastěji budovy. Pokud se v okolí agenta žádná zeď nevyskytuje, steering nemá ovlivňovat pohyb agenta. Pokud se v okolí zeď vyskytuje, měl by se k ní agent přiblížit na vhodnou vzdálenost a pokračovat podél ní. Steering tedy splňuje i základní vlastnosti steeringu Obstacle Avoidance, tedy že agent nevráží do překážek, ale navíc se snaží pokračovat podél nich.

Požadavky na rozšířené chování

Rozšířené chování chování dovoluje určit váhy jednotlivých složek výsledné síly a obsahuje jedno vylepšení pro plynulejší chůzi a lepší řešení *čelních kolizí*.

Možné hodnoty výsledného vektoru

Pokud se v okolí agenta nevyskytuje zeď, steering vrací nulový vektor. V opačném případě vrací steering sílu přitažlivou ke zdi či odpudivou od zdi (v základním chování), nebo navíc i rotační sílu (v rozšířeném chování).

Parametry

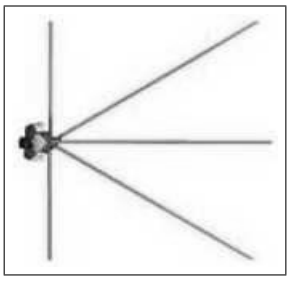
Parametry steeringu jsou popsány v Tabulce 3.5.

Název	Popis	Hodnoty
`Wall Force`	Velikost síly (ke zdi, od zdi).	0–1000 N_{UT}
`Force Order`	Řád síly. Ovlivňuje, jak roste velikost síly s klesající vzdáleností od překážky. Hodnota 1 představuje lineární závislost. Hodnoty vyšší než 1 způsobují ostřejší reakce na velmi blízké překážky a mírnější reakce na překážky vzdálenější.	1–10
`Attractive Weight`	Váha přitažlivé síly ke zdi.	0–10
`Repulsive Weight`	Váha odpudivé síly od zdi.	0–10
`Concave Weight`	Váha rotační síly za *silně konkávní roh*.	0–10
`Convex Weight`	Váha rotační síly od *silně konvexního rohu*.	0–10
`Just My Side`	S tímto parametrem si agent všímá pouze zdi, podél které jde, a ignoruje kolidování paprsků na druhé straně.	boolean
`Front Collisions`	Lepší řešení situace `čelní kolize`, analogicky jako u stejnojmenného parametru v Obstacle Avoidance.	boolean

Tabulka 3.5: Parametry steeringu Walll Following.

Informace o okolí

Steering používá 5 paprsků: 1 dlouhý přední ve směru agentova natočení, 2 dlouhé šikmé přední (od předního jsou odchýlené každý o 30°) a 2 krátké boční, viz Obrázek 3.15. Délky bočního a šikmého předního paprsku jsou nastaveny tak, aby přímka procházející jejich konci byla rovnoběžná s předním paprskem. To umožňuje plynulejší chůzi podél rovné zdi.



Obrázek 3.15: Paprsky steeringu Wall Following.

3.5.2 Základní chování

Základní chování je rozděleno na dvě části. První část řeší přitažlivou a odpudivou sílu od zdi. Tato část je dostatečná pro chůzi podél zdi, která je buď rovná, nebo jen mírně zahýbá. Pokud se zeď ostře stáčí od agenta (jedná se o tzv. *silně konkávní roh*) nebo k agentovi (tzv. *silně konvexní roh*), je třeba použít druhou část základního chování.

Rovná zeď a „neostré rohy“

Návratový vektor steeringu Wall Following je součet vektorů $\vec{p_1}, \ldots, \vec{p_n}$, kde n je počet kolidujících paprsků. Každý z těchto vektorů odpovídá odpudivé či přitažlivé síle příslušné danému paprsku. Pro i-tý ($i \in \{1, \cdots, n\}$) kolidující paprsek se určí místo kolize paprsku s překážkou. Vektor $\vec{p_i}$ je pak dán jako součet přitažlivé síly ke zdi a odpudivé síly od zdi. Odpudivá síla má směr shodný s normálovým vektorem stěny v místě kolize s paprskem. Přitažlivá síla má směr přesně opačný. Velikost přitažlivé síly ke zdi se určí podle vzorce

$$F \cdot \left(1 - \frac{R - D}{R}\right)^O$$

a velikost odpudivé síly od zdi se určí podle vzorce

$$F \cdot \left(\max\left(0, \frac{2 \cdot (R_{2/3} - D)}{R_{2/3}}\right)\right)^O,$$

kde F je hodnota parametru `Wall Force`, D je vzdálenost agenta od místa kolize paprsku se zdí (tedy délka nezanořené části paprsku), R je celková délka paprsku, $R_{2/3}$ jsou dvě třetiny délky paprsku a O je hodnota parametru `Force Order`.

Vzorec pro výpočet velikosti vektoru $\vec{p_i}$ je navržený, aby splňoval následující vlastnosti:

1. Pokud zasáhne paprsek zeď, působí na agenta přitažlivá síla ke zdi. Čím více je paprsek zanořený do zdi, tím je tato síla menší. Díky tomu si zeď agenta plynule jakoby přitáhne.

2. Pokud se agent zdi přiblíží natolik, že se paprsek zanoří z alespoň jedné třetiny, působí na něj zároveň síla odpudivá. Ta naopak roste se zvyšující se mírou zanoření paprsku.

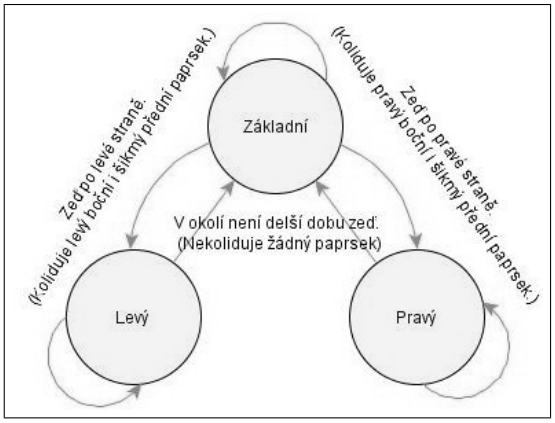
3. Ve vhodné vzdálenosti se síly vyruší. Agent je tedy schopný se plynule přiblížit zdi a jít podél ve vhodné vzdálenosti od ní.

Poznámka: pro přední paprsek se použije nulová hodnota přitažlivé síly, což snižuje pravděpodobnost, že agent vrazí čelem do zdi. Jinak by cestou čelem na zeď ještě přidal.

Ostré rohy

Pro hladkou zeď bez ostrých rohů stačí přitažlivé a odpudivé síly zdi. Pokud zeď mírně zahne směrem od agenta, přitažlivé síly paprsků zajistí, že se od ní agent nevzdálí. Pokud naopak zeď zahne mírně směrem k agentovi, odpudivé síly paprsků zabrání srážce agenta se zdí. Pokud zeď však zahne od agenta tak ostře, že s ní šikmý přední paprsek i boční paprsek ztratí kontakt, na agenta nepůsobí žádná síla, která by ho přitáhla zpět ke zdi. Tento typ rohu nazýváme *silně konkávní roh*. Pokud zeď zahne naopak ostře k agentovi, základní odpudivá síla nestačí, aby se agent stočil (dokonce by se většinou stočil na špatnou stranu). Tento druhý typ nazýváme *silně konvexní roh*.

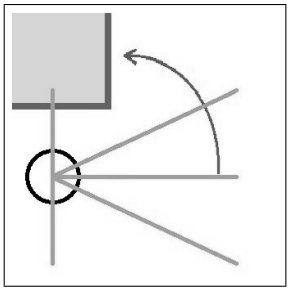
Prvně potřebujeme vědět, zda následuje agent zeď po levé či pravé straně. Zavedeme tedy pro agenta jednoduchý konečný stavový automat se třemi stavy: *základní*, *levý* a *pravý*. Na začátku je agent ve stavu *základní*. Z toho se může dostat do stavů *levý*, resp. pravý *pravý*, když narazí na zeď po levé, resp. pravé straně. (Pokud je zeď na obou stranách, tak si vybere stav pravý.) Mezi stavy *levý* a *pravý* žádný přechod nevede, může se však stát, že se agent dostane zpátky do stavu *základní*, pokud se mu zeď na delší dobu ztratí. Stavový automat znázorňuje Obrázek 3.16.



Obrázek 3.16: Stavový automat steeringu Wall Following.

Řešení *silně konkávních rohů*

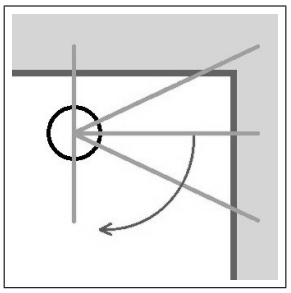
Označme S stav, ve kterém se agent vyskytuje. Zajímají nás pouze ty situace, kdy se jedná o stav *levý* či *pravý*, S tedy odpovídá straně, na které má agent zeď. O *silně konkávní roh* se jedná právě tehdy, když je agent ve stavu S a nekoliduje boční ani šikmý přední paprsek strany S. V takovém případě je třeba zahnout na stranu S, steering tedy vrátí rotační vektor r ve směru S. Situaci znázorňuje Obrázek 3.17



Obrázek 3.17: Silně konkávní roh.

Řešení *silně konvexních rohů*

Opět označme S stav, ve kterém se agent vyskytuje. Opět nás zajímají jen ty situace, kdy se jedná o stav *levý* či *pravý*. O *silně konvexní roh* se jedná právě tehdy, když je agent ve stavu S a koliduje přední paprsek a šikmý přední paprsek strany S. V takovém případě je třeba zahnout na druhou stranu než S, steering tedy vrátí rotační vektor r s opačným směrem, než je S. Situaci znázorňuje Obrázek 3.18



Obrázek 3.18: Silně konvexní roh.

Dodatek k řešení ostrých rohů

Zbývá ještě dodat, jak se může dostat agent do stavu *základní* z jiného stavu. K tomu se využívá čítač, který počítá, kolikrát za sebou se detekoval *silně konkávní roh*. Pokud vícekrát, než je maximální hodnota (nastavená na 10 tiků), agent se přepne do stavu základní. Toto je výhodné především tehdy, když je steering v kombinaci s jinými, kterým se může například hodit, aby šel agent chvíli podél zdi, ale časem se od ní oddálil a za chvíli pokračoval podél zdi na druhé straně, než byla ta předchozí.

3.5.3 Rozšířené chování – *čelní kolize*

Stejně jako u steeringu Obstacle Avoidance, i zde je potřeba řešit případy, kdy jde agent přímo čelem ke zdi, tedy situace nazvané *čelní kolize*. Pokud je agent v jiném než *základním* stavu, čelní kolize se bude jevit jako *silně konvexní roh* a agent se stočí na správnou stranu jen za pomoci základního chování. Pokud je však ve stavu *základní*, je třeba parametr `Front Collisions`, aby se agent správně stočil. Pro to se použije stejné řešení *čelních kolizí*, jako ve steeringu Obstacle Avoidance, viz sekce 3.1.

3.5.4 Rozšířené chování – *druhá strana*

Je-li agent ve stavu *levý* a objeví-li se po jeho pravé straně nějaká jiná překážka, přitažlivé (a následně odpudivé) síly k této překážce způsobí, že se agent podivně zapotácí, tedy bude chvíli měnit směr ze strany na stranu.

Parametr `Just My Side` způsobuje, že agent ignoruje přitažlivé a odpudivé síly bočního a šikmého předního paprsku druhé strany, než na jaké straně je zeď, podél které jde (což se pozná podle stavu). Díky tomu agent nevnímá překážky na druhé straně.

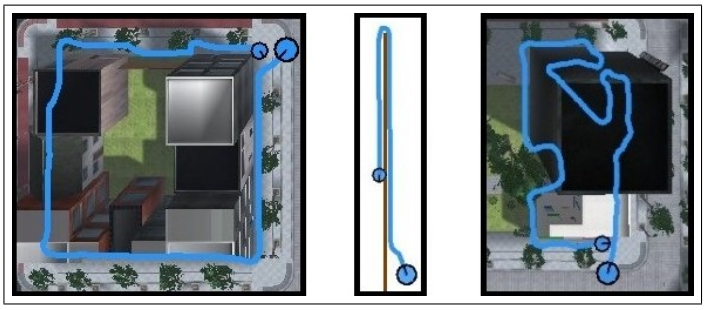
3.5.5 Rozšířené chování – váhy

Rozšířené chování dovoluje nastavit váhy jednotlivým složkám sil základního chování: přitažlivá síla ke zdi (`Attractive Weight`), odpudivá síla od zdi (`Repulsive Weight`, rotační síla u *silně konkávních rohů* (`Concave Weight`) a rotační síla u *silně konvexních rohů* (`Convexe Weight`). Pokud například chceme, aby se steering `Wall Following` spíše podobal steeringu `Obstacle Avoidance`, nastavíme nižší

parametr `Attractive Weight` a `Concave Weight`. Díky tomu nebude zeď agenta tolik přitahovat a agent se snáze odtrhne od zdi, pokud k tomu přispějí síly ostatních steeringů. To je velmi praktické v kombinaci s ostatními steeringy, např. steeringem `Target Approaching`, viz kapitola 6.

3.5.6 Ukázky

Obrázek 3.19 obsahuje několik ukázek steeringu Wall Following.



Obrázek 3.19: Vlevo: agent obchází komplex několika domů. Uprostřed: agent obchází plot. Vpravo: agent obchází budovu, narazí na vchod dovnitř (musí být dostatečně široký), obejde místnost zevnitř a pokračuje v obcházení budovy. Poznámka: z budov jsou tedy vidět pouze střechy a kvůli perspektivě pohledu není bohužel jasně vidět, jaký má tvar zeď ve výšce agenta.

3.5.7 Závěr

Výše navržený steering Wall Following naviguje agenta tak, aby se pohyboval podél zdi, nenarážel do ní, ani se od ní příliš nevzdaloval, i když zeď ostře zahne. Steering je navržen tak, aby se agent pohyboval co nejplynuleji, nicméně i přesto se může stát, že agent po několik tiků mírně mění své natočení ke zdi a od zdi. Typicky se toto kmitání rychle ustálí na plynulé chůzi, je zde však prostor pro možná vylepšení. Náročné jsou především zdi s mnoha výklenky a nerovnostmi.

Základní chování steeringu je založeno na myšlence stejnojmenného steeringu Craiga W. Reynoldse [34, 35], implementace se však různí. Steering C. W. Reynoldse využívá toho, že si agent může nechat spočítat vzdálenost jakéhokoliv bodu v prostředí (konkrétně své vlastní projekce za několik tiků) od nejbližší zdi. Oproti tomu výše navržený Wall Following umožňuje navigovat agenta podél zdi jen za pomoci paprsků vycházejících z těla agenta.

Pokud bylo možno zjistit, tento steering není v souvislosti s virtuálními agenty mnoho zkoumaný (více se jím zabývají v oblasti robotiky, jako např. [44]). Na první pohled není využití tohoto steeringu příliš široké. Avšak jakmile začneme steering kombinovat s jinými, můžeme si všimnout velmi zajímavých a výhodných vlastností: agent lépe obchází větší či složitější překážky než se steeringem Obstacle Avoidance, najednou začne chodit po chodnících, kde by za pomoci jiných steeringů chodil náhodně – a dokonce umožňuje vytvořit poměrně speciální avšak zajímavou situaci „tajného sledování“. Všechny tyto ukázky jsou součástí Kapitoly 6.

3.6 Leader Following

3.6.1 Vlastnosti steeringu

Požadavky na základní chování

Základním cílem steeringu Leader Following je navigovat agenta (*následovníka*), aby následoval jiného agenta (*vůdce*). Vůdce může mít aktivní libovolné steeringy, nebo se ani nemusí pohybovat pomocí steeringů. Klidně se může stát, že má aktivní také steering Leader Following a následuje nějakého třetího agenta.

Agent by se měl pohybovat za vůdcem v zadané vzdálenosti. Pokud je vůdce příliš daleko, měl by ho dohnat. Pokud je vůdce příliš blízko, měl by se od něj vzdálit. Pokud se vůdce zastaví a následovník je ve vhodné vzdálenosti, měl by se také zastavit a natočit se na něj.

Požadavky na rozšířené chování

Rozšířené chování nabízí jednak druhý typ steeringu Leader Following, tzv. *formační*, jednak vylepšení prvního typu, tzv. *základního* (popsaného v základním chování). Tato vylepšení základního typu se týkají přirozenějších reakcí na změny parametrů steeringu.

Formační typ dovoluje kromě vzdálenosti od vůdce určit zároveň polohu následovníka vůči vůdci (např. vpravo od něj, za ním, před ním, atd.). Pomocí toho se dají vytvářet jednoduché formace více agentů. Vylepšení formačního typu se zabývají plynulostí pohybu a schopností oběhnout vůdce, pokud se vyskytuje mezi následovníkem a jeho určenou pozicí.

Možné hodnoty výsledného vektoru

V základním chování představuje návratový vektor přitažlivou sílu k vůdci, odpudivou sílu od vůdce či požadavek na zastavení. V rozšířeném chování se může jednat navíc o zpomalovací sílu, sílu k pozici a rotační sílu.

Parametry

Parametry steeringu jsou popsány v Tabulce 3.6.

Informace o okolí

Agent potřebuje znát lokaci svého vůdce, svůj vektor rychlosti a svou lokaci. Ve formačním typu navíc vyžaduje vektor rychlosti vůdce a dobu trvání jednoho tiku.

Název	Popis	Hodnoty
`Leader Force`	Velikost síly (k vůdci, od vůdce).	0–1000 N_{UT}
`Leader`	Vůdce, kterého má agent následovat.	Agent
`Distance`	Vzdálenost od vůdce.	30–2000 cm_{UT}
`Force Distance`	V této vzdálenosti od vůdce má přitažlivá síla k vůdci hodnotu `Leader Force`. Se zvyšující se vzdáleností roste síla lineárně.	1–2000 cm_{UT}
`Type`	Typ steeringu. Základní typ (agent následuje vůdce nehledě na natočení vůdce) či formační (agent jde v určití pozici vůči natočení vůdce – tato pozice je daná parametry `Distance` a `Angle`).	
`Angle`	Úhel vůči natočení vůdce. Např. hodnota 90° znamená vpravo od vůdce, -90° vlevo , 180° za vůdcem, 0° před vůdcem. Pouze u formačního typu.	-180°–180°
`Memory`	Paměť vektorů rychlosti vůdce. Napomáhá plynulejšímu pohybu. Pouze u formačního typu.	boolean
`Memory Size`	Velikost paměti – počet zapamatovaných vektorů rychlosti vůdce. Pouze u formačního typu.	0–20
`Circumvention`	Umožňuje oběhnout vůdce, pokud se vyskytuje mezi agentem a jeho určenou pozicí. Tím předchází srážkám s vůdcem. Pouze u formačního typu.	boolean
`Deceleration`	Schopnost zpomalení, pokud je vůdce příliš blízko. Pouze u základního typu.	boolean

Tabulka 3.6: Parametry steeringu Leader Following.

3.6.2 Základní chování

Pokud agent vidí svého vůdce, mohou nastat tyto situace:

1. Vůdce se pohybuje a je blíže než `Distance`. Pak steering vrací odpudivou sílu od vůdce, jejíž velikost je určena vzorcem

$$F \cdot \left(\frac{d}{D}\right)^2,$$

   kde F je hodnota parametru `Leader Force`, d je vzdálenost agenta od vůdce a D je hodnota parametru `Distance`. Čím je agent blíže k vůdci, tím je odpudivá síla větší, nejvýše však `Leader Force`. Umocnění podílu zmenšuje sílu v malých vzdálenostech od vůdce.

2. Vůdce se pohybuje a je dále než `Distance`. Pak steering vrací přitažlivou sílu k vůdci, jejíž velikost je určena vzorcem

$$F \cdot \frac{d - D}{D_F},$$

   kde F je hodnota parametru `Leader Force`, d je vzdálenost agenta od vůdce, D je hodnota parametru `Distance` a D_F je hodnota parametru `Force`

`Distance`. Ve vzdálenosti D od vůdce je přitažlivá síla nulová a dále se zvětšuje lineárně s rostoucí vzdáleností agenta od vůdce tak, že ve vzdálenosti $D_F + D$ od vůdce je velká přesně `Leader Force`.

3. Vůdce se pohybuje a je vzdálený přesně `Distance`. Pak steering vrací nulový vektor.

4. Vůdce se zastavil a je vhodně vzdálený (tzn. $[D - 80,\ D + 80]$, kde D je `Distance`). Pak steering vrací požadavek na zastavení a natočení na lokaci vůdce.

5. Vůdce se zastavil, ale není vhodně vzdálený. Pak podle toho, zda je moc blízko, či daleko, vrací steering odpudivou sílu od něj, či přitažlivou sílu k němu jako v bodech 1. a 2.

Pokud agent vůdce nevidí, ale někdy předtím ho viděl, použije jeho poslední lokaci a vektor rychlosti. Pokud agent vůdce nikdy dříve nespatřil, začne se otáčet a čeká, až ho spatří (steering vrací rotační vektor).

3.6.3 Rozšiřující chování – rozšíření základního typu

Nevýhody základního chování

Představme si, že jde agent ve vhodné vzdálenosti za vůdcem, ale najednou je vůdci příliš blízko. To může být způsobeno buď tím, že vůdce najednou zpomalil, nebo že se změnil (zvětšil) parametr `Distance`. Pak se může pak stát, že odpudivá síla od vůdce ho otočí směrem zpět, aby došel na požadovanou vzdálenost od vůdce. Nicméně pokud se mezitím vůdce pohybuje v původním směru, agent se za krátkou chvíli zase otočí zpět a pokračuje za ním, což vypadá poměrně zmateně.

Je zde zřejmý rozpor mezi plynulostí pohybu a včasnou reakcí na změny. Kdyby agent místo toho zpomalil (ale vůdce by šel pořád stejně rychle), jejich vzdálenost by se časem přirozeně zvětšila. Výsledek by byl plynulejší, ač by chvíli trvalo, než by se agent dostal do vhodné vzdálenosti od vůdce.

Na druhou stranu kdyby vůdce najednou třeba zastavil, pouhým zpomalením by se agent do vhodné vzdálenosti nedostal. Je tedy nutné umět rozlišit, kdy je výhodnější zpomalit, a kdy je potřeba, aby se agent otočil a došel do vhodné vzdálenosti. Takové řešení nabízí parametr `Deceleration`.

Řešení vracení pomocí parametru `Deceleration`

Parametr `Deceleration` ovlivňuje situace, kdy je agent k vůdci blíže, než je hodnota parametru `Distance`. Rozlišuje, kdy stačí vrátit zpomalovací vektor, a kdy je nutno vrátit vektor představující odpudivou sílu od agenta. Tím kritériem je rychlost pohybu vůdce. Pokud se vůdce pohybuje alespoň tak rychle jako jeho následovník, stačí aby následovník zpomalil a časem se tak přirozeně od vůdce vzdálil.[10] V takovém případě vrací steering tedy zpomalovací vektor. Čím je rozdíl

[10] Samozřejmě pouze za podmínky, že se do té doby něco nezmění, třeba že vůdce nezpomalí. Ale v daném tiku by se rozhodoval znovu a vhodně by na to zareagoval.

požadované a reálné vzdálenosti větší, tím je zpomalovací vektor větší (v největším případě způsobí zastavení agenta).

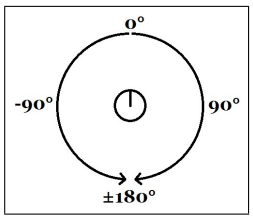
Pokud se vůdce pohybuje pomaleji než jeho následovník, vrací steering součet zpomalovacího vektoru (jako v předchozím případě) a odpudivé síly od vůdce, která je tím větší, čím více se liší rychlost vůdce a následovníka.

Pokud tedy například vůdce zastaví, agent je schopný dojít (resp. doběhnout) na místo ve vhodné vzdálenosti od vůdce. Pokud se ale vůdce pohybuje stále stejně rychle, jeho následovník plynule zpomaluje do té doby, než se vzdálenost mezi nimi zvětší na požadovanou.

3.6.4 Rozšiřující chování – formační typ

Základní vlastnosti formačního typu

Formační typ steeringu Leader Following dovoluje určit pozici, kde se má následovník vůči vůdci vyskytovat. Parametr `Distance` určuje vzdálenost od vůdce a parametr `Angle` určuje úhel mezi vektorem natočení vůdce (pokud se vůdce pohybuje, tak je shodný s jeho vektorem rychlosti) a vektorem od vůdce k požadované pozici následovníka. Pokud má být následovník vpravo od vůdce, hodnoty jsou kladné, vlevo záporné, viz obrázek 3.20.



Obrázek 3.20: Hodnoty parametru `Angle` steeringu Leader Following formačního typu.

Jak se určí návratový vektor steeringu Leader Following formačního typu? Nejdříve je nutno určit, na jaké lokaci by se měl následovník v příštím tiku vyskytovat a dle toho spočítat přitažlivý vektor k této lokaci. Tato lokace závisí na lokaci a natočení vůdce v příštím tiku. Označme L_L aktuální lokaci vůdce, $\vec{v_L}$ aktuální vektor rychlosti vůdce, L_F aktuální lokaci následovníka, t dobu trvání jednoho tiku, L'_F předpokládanou lokaci následovníka v příštím tiku a L'_L předpokládanou lokaci vůdce v příštím tiku, kterou spočítáme podle vzorce:

$$L'_L = L_L + t \cdot \vec{v_L}.$$

Následně určíme pomocný vektor $\vec{p}$, který bude představovat vektor od nové lokace vůdce k požadované pozici následovníka. Po složkách má hodnoty:

$$(\vec{p})_x = (\vec{v_L})_x \cdot \cos\alpha - (\vec{v_L})_y \cdot \sin\alpha$$

$$(\vec{p})_y = (\vec{v_L})_x \cdot \sin\alpha + (\vec{v_L})_y \cdot \cos\alpha$$

$$(\vec{p})_z = 0$$

přičemž α je hodnota parametru `Angle`. Následně se spočítá

$$L'_F = L'_L + \vec{p}.$$

Na tuto pozici by se měl následovník dostat ideálně v následujícím tiku. Pokud je daleko, tak samozřejmě nemá smysl se o to snažit za každou cenu – stačí, aby vůdce dohnal postupně.

Návratový vektor bude mít směr

$$\vec{v} = L'_F - L_F$$

a velikost

$$\frac{F \cdot |\vec{v}|}{d},$$

kde F je hodnota parametru `Leader Force` a d je hodnota parametru `Force Distance`. Pokud je vzdálená přesně `Force Distance`, má $\vec{v}$ velikost přesně `Leader Force`. Čím je pozice dále, tím je $\vec{v}$ větší. Chceme-li, aby agent rychle reagoval na změny pozice a dařilo se mu jít na zadané pozici, zvýšíme parametr `Leader Force` či snížíme parametr `Force Distance`[11].

Nevýhody základních vlastností formačního typu – plynulost

Jestliže vůdce změní na chvíli směr svého vektoru rychlosti (např. ho na chvíli rozhodí nějaká překážka), pro jeho následovníka to znamená velkou změnu pozice. Jak se následovník rychle snaží dostat na nové místo a zakrátko se zase vrací, vypadá to, že zmateně pobíhá. Je-li následovníků dokonce více, situace působí ještě komičtěji.

Problém je, že se následovník snaží ihned dostat na pozici určenou novým natočením vůdce. Nicméně pokud je to natočení pouze dočasné, bylo by lepší ho ignorovat. Jak ale rozlišit, zda se jedná o dočasnou změnu, nebo konečnou, na kterou je třeba rychle zareagovat? Tímto problémem a možností plynulejšího pohybu se zabývá parametr `Memory` s pomocným parametrem `Memory Size`.

Plynulejší pohyb pomocí parametru `Memory`

Pokud je zapnutý přepínač `Memory`, agent si pamatuje posledních n vektorů rychlosti vůdce, kde n je hodnota parametru `Memory Size`. Do vzorce pro výpočet požadované lokace v příštím tiku se pak místo $\vec{v_L}$ (aktuální vektor rychlosti vůdce) dosadí průměr n posledních vektorů rychlosti vůdce.

Pokud tedy vůdce najednou změní směr pohybu a v příštím tiku se vrátí k původnímu směru, pro následovníka to bude znamenat mnohem menší změnu požadované lokace, neboť průměrný vektor rychlosti se tolik nezmění.

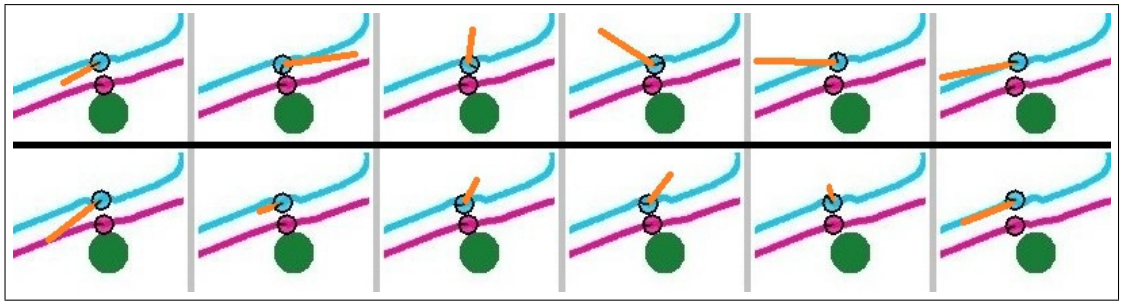
Pohyb následovníka je pak mnohem hladší a plynulejší, neboť krátkodobé výchylky vůdcova pohybu neovlivňují ostře pohyb následovníka. Na druhou stranu ovlivní tyto výchylky pohyb následovníka na delší dobu (ač mírně) a zároveň vyšší hodnoty parametru `Memory Size` způsobují pomalejší reakce na změny vůdcova pohybu. Z hlediska rozhodovací vrstvy je nutno si vybrat, jak rychlé reakce na změny potřebuje, oproti tomu, jak plynule se má následovník pohybovat.

Obrázek 3.21 ukazuje srovnání formačního typu bez a s pamětí.

Nevýhody základních vlastností formačního typu – kolize

Obdobně jako u steeringu `People Avoidance`, i zde jsou problémové ty situace, kdy se mezi následovníkem a jeho požadovanou lokací vyskytuje jiný agent – zde vůdce. Pak je potřeba nějaké rotační síly, která agenta vychýlí ze směru k požadované lokaci, aby vůdce oběhl. Typicky to nastává tehdy, když je změněn parametr `Angle` ze 180° na 0°. Schopnost oběhnout vůdce řeší parametr `Circumvention`.

[11] V podstatě by stačil pouze parametr `Leader Force`, avšak tato dvojice parametrů by měla být uživatelsky přívětivější. Uživatel rozhoduje, v jaké vzdálenosti od vůdce na něj působí jaká síla, a má možnost hýbat buď tuto vzdálenost či velikost síly.



Obrázek 3.21: Vůdce (růžový) se vyhýbá stromu (zelené kolečko). Modrý agent ho následuje s hodnotou parametru `Angle` 90°. Oranžově je znázorněna síla steeringu Leader Following, která zde představuje sílu k požadované pozici vpravo od vůdce. Nahoře je průběh scény bez parametru `Memory`. Krátkodobá změna směru vůdce způsobí, že se modrý agent otočí dokolečka (pozorujte natočení modrého agenta znázorněné černou čárkou). Dole je průběh stejné scény za použití parametru `Memory` s pamětí velkou 5 vektorů rychlosti. Díky paměti se následovník neotočí dokolečka jako v prvním případě, jen se postupně plynule natočí doprava a pak zase zpět. Výsledné chování vypadá plynuleji, ač trajektorie pohybu může být méně přímočará. To je způsobeno tím, že následovník reaguje na jedno odlišné natočení vůdce sice méně prudce, ale delší dobu.

Obcházení pomocí parametru `Circumvention`

Myšlenka řešení obcházení pomocí toho parametru je obdobná jako u steeringu `People Avoidance`, avšak uzpůsobená specificky pro formační typ steeringu `Leader Following`.

Které situace parametr řeší? Když dráha následovníka k požadované lokaci se přiblíží lokaci vůdce na příliš blízko, tedy blíže než

$$D_2 = \max\left(150, \frac{D}{2}\right),$$

kde D je hodnota parametru `Distance`.

Označme $\overline{L_L}$ průměr aktuální lokace vůdce a předpokládané lokace vůdce v příštím tiku. Prvotní podmínka je, že vzdálenost agenta k požadované lokaci musí být delší než vzdálenost agenta k $\overline{L_L}$. Dále označme P nejbližší bod k $\overline{L_L}$ na úsečce od aktuální lokace agenta k požadované lokaci.

Pokud prochází přímka od aktuální lokace agenta k požadované lokaci přímo lokací $\overline{L_L}$, zvolí se směr rotačního vektoru (doprava/doleva) náhodně a koeficient K jako 1. V opačném případě se zvolí ten směr, která dává větší smysl (pokud je $\overline{L_L}$ od přímky vlevo, bude směr doprava, a naopak). Koeficient K se pak určí jako

$$\max\left(0, \frac{D_2 - d_P}{D_2}\right),$$

kde d_P je vzdálenost bodu P od lokace $\overline{L_L}$. Velikost rotačního vektoru se vynásobí koeficientem K. Návratový vektor se pak určí jako přitažlivá síla k požadované lokaci + rotační vektor vynásobený konstantou

$$\frac{F}{d},$$

kde F je hodnota parametru `Leader Force` a d je vzdálenost agenta od vůdce.

3.6.5 Závěr

Steering Leader Following naviguje agenta, aby následoval jiného agenta v zadané vzdálenosti (základní typ), nebo navíc i na zadané pozici (formační typ). Základní typ má jedno vylepšení, díky kterému agent přirozeně zpomalí, zvětší-li se najednou požadovaná vzdálenost od vůdce. Formační typ obsahuje vylepšení schopnosti následovníka oběhnout vůdce, když se vůdce vyskytuje mezi následovníkem a jeho požadovanou pozicí. Dále obsahuje mechanismus, jak zařídit o něco plynulejší pohyb následovníka.

Hlavním problémem steeringu Leader Following je rozpor mezi snahou udržovat vzdálenost, resp. pozici vůči vůdci, a neměnit příliš často směr ani rychlost pohybu. Nejvýrazněji je tento rozpor vidět ve formačním typu s úhly $\pm 90°$, především pokud je okolní prostředí bohaté na různé překážky. Následovník pak buď často mění směr pohybu (může ho to až úplně otočit jako na Obrázku 3.21), nebo nejde vedle vůdce, ale mírně za ním. Pokud si rozhodovací vrstva (resp. uživatel) zvolí, zda je pro ni důležitější plynulost, či včasné reakce, může využít prostředky základního i rozšířeného chování pro daný účel: vyšší parametr `Leader Force`, resp. nižší `Force Distance` (např. síla 240 N_{UT}a vzdálenost 50 cm_{UT}) způsobí rychlejší a úspěšnější reakce na změny, oproti tomu nižší parametr `Leader Force`, resp. vyšší `Force Distance` (např. síla 200 N_{UT}a vzdálenost 100 cm_{UT}a více) a parametr `Memory` slouží k plynulejšímu pohybu.

Jednou z výhod tohoto steeringu je, že může ušetřit výpočetní náročnost simulace. Jak totiž agent následuje svého vůdce, tak v podstatě kopíruje jeho dráhu (pokud není příliš daleko). Stačí tedy zařídit, aby se vůdce správně pohyboval (např. pomocí steeringu Path Following) a následovníkovi pak stačí už jen steering Leader Following, či případně People Avoidance, pokud je následovníků více a nemají do sebe vrážet.

Základní chování odpovídá stejnojmennému steeringu Craiga W. Reynoldse [34, 35]. Rozšířené chování dovoluje vytvářet jednoduché formace. Tématem formací se zabývají například i tyto práce [6, 10, 2]. Nejbližší je práce [6], kde jsou základem formace taktéž autonomní agenti řízení poměrně jednoduchými pravidly a silami. Oproti jiným přístupům je výhodné, že formace není striktně rigidní a přizpůsobuje se okolí. Zároveň jakmile to je možné, vytvoří agenti původní tvar formace. Výhodou tohoto přístupu je taktéž výpočetní nenáročnost. Oproti tomu jiné dva používané přístupy jsou: zaprvé formace řízená centrálně (např. [10]), zadruhé formace pohybující se po předem vypočítané cestě; tedy již při plánování této cesty se počítá s tím, že má sloužit pro celou skupinu agentů (např. [2]).

3.7 Walk Along

3.7.1 Vlastnosti steeringu

Požadavky na základní chování

Steering Walk Along naviguje agenta tak, aby společně s jiným agentem (*partnerem*) došli na určité místo. Společně znamená, že by měli jít vedle sebe v zadané vzdálenosti, případně na sebe čekat či jeden druhého dohnat, když je to potřeba.

Požadavky na rozšířené chování

Rozšířené chování nabízí dvě vylepšení. První vylepšení se stará o to, aby se agenti obešli, pokud se nemohou dostat vedle sebe. Druhé vylepšení se týká toho, co má agent udělat, pokud je zhruba mezi partnerem a cílem a zároveň poměrně daleko od partnera. V základním chování doběhne k partnerovi. V rozšířeném chování umí na partnera počkat a pokračovat s ním, až když k němu partner dojde.

Možné hodnoty výsledného vektoru

V základním chování představuje výsledný vektor kombinaci sil k cíli, k partnerovi a od partnera. V rozšířeném chování se pak může jednat navíc o speciální sílu, která udržuje partnery ve vhodné vzdálenosti od sebe, či požadavek na zastavení.

Parametry

Parametry steeringu jsou popsány v Tabulce 3.7.

Název	Popis	Hodnoty
`Partner Force`	Velikost síly.	0–1000 N_{UT}
`Target`	Cílová lokace.	Location
`Partner`	Partner, se kterým má dojít k cíli.	Agent
`Distance`	Ideální vzdálenost od partnera.	0–2000 cm_{UT}
`Give Way`	Tento parametr lépe řeší *odpuzování* a obecně napomáhá, aby se nedostali agenti příliš blízko k sobě a zároveň napomáhá tomu, aby šli vedle sebe.	boolean
`Wait for Partner`	S tímto parametrem agent na partnera čeká, pokud je druhý daleko a jsou splněny speciální podmínky. Bez tohoto parametru by agent běžel k partnerovi.	boolean

Tabulka 3.7: Parametry steeringu Walk Along.

Informace o okolí

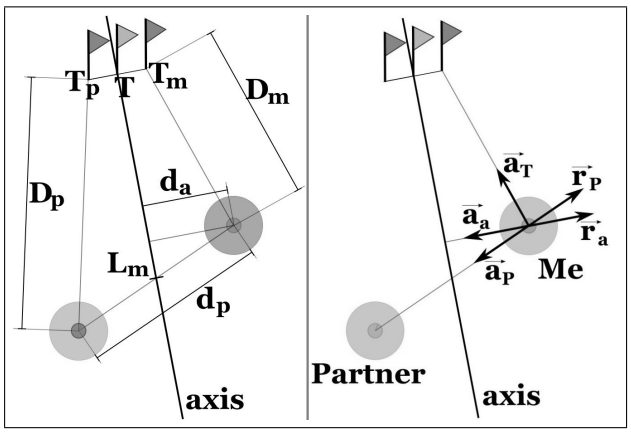
Agent potřebuje znát lokaci partnera, svou lokaci a svůj vektor rychlosti.

3.7.2 Základní chování

Návratový vektor steeringu WALK ALONG (v základním i rozšířeném chování) je součet tří sil: $\vec{v_1}$, $\vec{v_2}$ a $\vec{v_3}$. První vždy zajišťuje, aby došli partneři k cíli a zbylé dva aby si mezi sebou udržovali vhodnou vzdálenost. V základním chování jsou tyto vektory určeny poměrně jednoduše: $\vec{v_1} = \vec{a_T}$ (přitažlivá síla k cíli), $\vec{v_2} = \vec{a_P}$ (přitažlivá síla k partnerovi) a $\vec{v_3} = \vec{r_P}$ (odpudivá síla od partnera). V závislosti na poloze obou partnerů a cíle mají tyto vektory různou velikost. K určení velikostí se nám bude hodit zavést si značení několika lokací, vzdáleností a sil, viz Obrázek 3.22.

Značení

Prvně zavedeme tři specifické cíle: T (lokace určená parametrem `Target`), T_m (*my target*) a T_p (*partners target*). Každý z partnerů půjde ke svému vlastnímu cíli.



Obrázek 3.22: Nákres partnerů a jejich cílů, vzdáleností (vlevo) a sil (vpravo) steeringu WALK ALONG pro modrého agenta.

Pro tyto tři cíle potřebujeme lokaci L_m (*middle location*), která je umístěna přesně uprostřed úsečky spojující lokaci agenta a lokaci partnera. Dále označme *axis* spojnici L_m a `Target`. Lokace T_m a T_p leží na přímce, která prochází `Target` a má směr kolmý na *axis*. Vzdálenost T_m i T_p od `Target` je polovina hodnoty `Distance`. Tomuto popisu odpovídají dvě lokace (nalevo a napravo od *axis*): T_m je ta, co je blíže lokaci agenta, a T_p je ta, co je blíže lokaci partnera. Pokud jsou vzdálenosti náhodou přesně stejné, T_m a T_p se rozdělí náhodně.

Dále označme D_m (*my distance*) vzdálenost lokace agenta (*Me*)a cíle T_m, D_p vzdálenost lokace partnera od cíle T_p, d_p vzdálenost lokací obou partnerů a d_a vzdálenost lokace agenta od přímky *axis*.

Nakonec označme $\vec{a_T}$ (*attractive force to my Target*) vektor od lokace agenta k T_m, $\vec{a_P}$ (*attractive force to Partner*) vektor od lokace agenta k lokaci partnera, $\vec{r_P}$ (*repulsive force from Partner*) vektor přesně opačný, $\vec{a_a}$ (*attractive force to axis*) vektor od lokace agenta k přímce *axis* a $\vec{r_a}$ (*repulsive force from axis*) vektor přesně opačný vektoru $\vec{a_a}$.

Výpočet návratového vektoru

Vektor $\vec{v_1}$ má směr $\vec{a_T}$. Jeho velikost se určí vzorcem:

$$F \cdot 2^{\min(1.5,\ D_m - D_p)} + c,$$

kde F je hodnota parametru `Partner Force`, D_m je vzdálenost lokace agenta od T_m, D_p je vzdálenost lokace partnera od T_p a c je konstanta s hodnotou 30, pokud $D_m > D_p$, či hodnotou 0 jinak.

Vzorec je zvolený tak, aby byl vzdálenější partner přitahován k lokaci výrazně (alespoň silou `Partner Force` + c) a bližší partner silou poměrně malou. Díky tomu dokáže bližší partner zpomalit, aby ho vzdálenější došel, ale zároveň jsou schopní jít plynule k cíli.

Dále podle toho, zda d_p (vzdálenost partnerů) větší či menší než `Distance`, působí na agenta přitažlivá síla k partnerovi ($\vec{v_2}$) či odpudivá síla od partnera ($\vec{v_3}$). Pokud je přesně rovna `Distance`, nepůsobí ani jedna z těchto sil.

Vektor $\vec{v_2}$ má směr $\vec{a_P}$. Jeho velikost se určí vzorcem:

$$F \cdot \frac{\max(0, d_p - D)}{D},$$

kde F je hodnota parametru `Partner Force`, d_p je vzdálenost lokací partnerů a D je hodnota parametru `Distance`. Čím je partner dále, tím je tato síla větší.

Vektor $\vec{v_3}$ má směr $\vec{r_P}$. Jeho velikost se určí vzorcem:

$$F \cdot \frac{\max(0, D - d_p)}{D},$$

kde F je hodnota parametru `Partner Force`, d_p je vzdálenost lokací partnerů a D je hodnota parametru `Distance`. Čím je partner blíže, tím je tato síla větší, nejvýše však `Partner Force`.

Díky této trojici sil jsou agenti schopní společně dojít k cíli. Pokud jsou od sebe příliš vzdálení, nejdříve se seběhnou. Pak pokračují k cíli a snaží se udržovat mezi sebou vzdálenost `Distance`. Pokud se jeden z nich někde zdrží (např. kvůli překážce), druhý zpomalí (pokud je první nedaleko), či za prvním dojde (pokud je daleko). Když dojdou k cíli, zastaví se (steering vrátí požadavek na zastavení).

3.7.3 Rozšiřující chování – *odpuzování*

Nevýhody základního chování

Základní chování funguje dobře, pokud už jdou agenti vedle sebe a okolí jim poskytuje dostatek volného prostoru pro chůzi vedle sebe směrem k cíli. Nicméně pokud se agenti dostanou za sebe (tedy jejich lokace a `Target` leží zhruba na jedné přímce), trvá poměrně dlouho (např. v řádu jednotek sekund), než se dostanou vedle sebe, i když mají dostatek volného prostoru okolo. Proč tomu tak je?

Jakmile se dostanou za sebe, odpudivá síla od partnera má velmi podobný směr jako přitažlivá síla k cíli pro předního partnera a zhruba opačný směr pro zadního partnera. Místo aby zadní partner zrychlil a došel vedle předního, tak zpomalí. Přední partner by měl zpomalit a počkat na toho zadního, ale odpudivá síla od partnera ho donutí naopak zrychlit.

Tyto situace nazýváme *odpuzování*, neboť se agenti nevhodně odpuzují, aby do sebe nevrazili, čímž je znemožněno, aby šli bok po boku vedle sebe. Jedná se o velmi podobný problém jako v situaci *obcházení* v případě steeringu People Avoidance. Steering Walk Along je ale specifičtější a problém *odpuzování* lze řešit jednodušeji. Takové řešení nabízí parametr `Give Way`.

Řešení *odpuzování* pomocí parametru `Give Way`

Je-li zapnutý přepínač `Give Way`, steering nezkoumá, zda si nejsou partneři příliš blízko, ale zda není agent příliš blízko přímce *axis*.

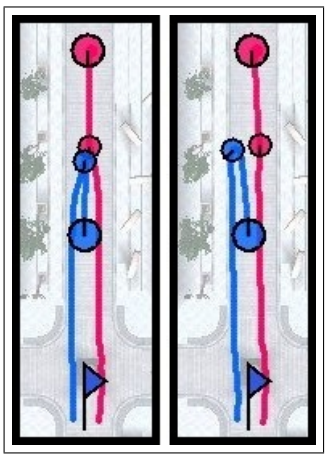
Místo původní síly $\vec{r_P}$ se jako $\vec{v_3}$ použije síla $\vec{r_A}$. Pokud je $\vec{d_a}$ (vzdálenost od *axis*) větší jak polovina hodnoty `Distance`, síla $\vec{r_A}$ je nulová. V opačném případě představuje odpudivou sílu od přímky *axis*. Směr vektoru $\vec{r_A}$ je kolmý na přímku *axis* a směrem k agentovi, viz Obrázek 3.22 (vpravo). Velikost vektoru $\vec{r_A}$ se určí podle vzorce:

$$F \cdot \frac{D - 2 \cdot d_a}{D},$$

kde F je hodnota parametru `Partner Force`, D je hodnota parametru `Distance` a d_a je vzdálenost lokace agenta od přímky *axis*.

Čím je agent blíže k přímce *axis*, tím ho tato přímka více odpuzuje. Pokud jsou agenti vedle sebe čelem ke svým cílům, tato síla odpovídá původní síle r_P. Pokud se ale dostanou za sebe, posunou se každý na svou stranu a uvolní tak místo druhému. Takže zadní může pohodlně dojít vedle předního partnera a pokračovat po jeho boku.

Obrázek 3.23 ukazuje význam parametru `Give Way`. Vlevo je situace v základním chování. Zadní agent (růžový) by měl dojít vedle svého partnera (modrý), ale nedaří se mu to. Mezi ním a cílem je totiž modrý partner. Síla k cíli a síla od modrého partnera se pak téměř odečtou (a převáží ta od partnera). Kvůli tomu jim jednak bude chvíli trvat, než se dostanou vedle sebe, jednak se dostanou příliš blízko k sobě. Vpravo je stejná situace za použití parametru `Give Way`. Na růžové agenta teď nepůsobí síla přímo od modrého agenta, ale od přímky *axis*. Díky tomu partnera obejde a navíc mu partner uhne.



Obrázek 3.23: `Give Way`.

3.7.4 Rozšiřující chování – *čekání*

Nevýhody základního chování

Situace nazývané *čekání*, nastávají tehdy, když je
agent mezi partnerem a cílem `Target`, je poměrně daleko od partnera a blízko přímce *axis*. Jak se v takové situaci zachová agent se základním chováním? Doběhne k partnerovi, zhruba v půli cesty se spolu setkají a pokračují k cíli společně. Agent tedy běží pryč od cíle a následně se podobnou cestou vrací zase zpátky. Poměrně přirozené by bylo, aby počkal tam, kde je, než k němu dojde ten druhý, a pak by pokračovali společně. Takové řešení nabízí parametr `Wait for Partner`.

Řešení *čekání* pomocí parametru `Wait for Partner`

Situace *čekání* je definovaná těmito podmínkami:

1. Vzdálenost partnerů je ostře vyšší než hodnota parametru `Distance`.
2. Hodnota ΔD je větší než trojnásobek `Distance`, kde ΔD je rozdíl vzdálenosti agenta od svého cíle a partnera od svého cíle.

Jestliže je zapnutý parametr `Wait for Partner` a jsou splněny podmínky situace *čekání*, použije se následující výpočet:

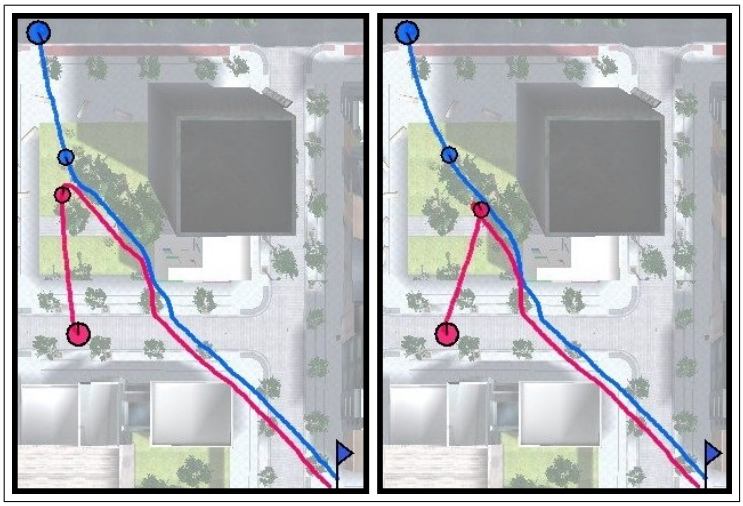
Pokud je d_a (vzdálenost od $axis$) větší než polovina `Distance`, síla $\vec{v_2}$ je součet přitažlivé $\vec{a_P}$ (síly k partnerovi) a $\vec{a_A}$ (přitažlivé síly k přímce $axis$), jejichž velikost je dána společným vzorcem:

$$F \cdot \frac{d_a}{D},$$

kde F je hodnota parametru `Partner Force`, D je hodnota parametru `Distance` a d_a je vzdálenost lokace agenta od přímky $axis$. Čím je agent dále od spojnice, tím je tato síla větší. Přitahuje ho jak ke spojnici, tak k partnerovi.

Pokud je d_a nejvýše polovina `Distance`, steering vrací požadavek na zastavení a natočení na partnera.

Obrázek 3.24 ukazuje použití parametru `Wait for Partner`. Vlevo běží agenti přímo k sobě. Vpravo se modrý agent přiblíží k místu, kudy bude muset zelený agent projít, a tam na něj počká. Těsně než zelený agent dorazí, mu vyjde modrý agent naproti a dále pokračují spolu.



Obrázek 3.24: Vlevo: základní chování. Vpravo: stejná situace s parametrem `Wait for Partner`.

3.7.5 Závěr

Steering Walk Along naviguje agenta tak, aby došel s jiným agentem ke společnému cíli. Aby steering fungoval správně, je třeba, aby měl i druhý agent aktivní steering Walk Along se stejným nastavením parametrů. I tak se jim ale nemusí podařit dojít k cíli. Například kvůli tomu, že se jeden z agentů zasekne o nějakou překážku. Dále je potřeba, aby na sebe agenti viděli. Pokud se nevidí, tak se „rozhlíží“ po okolí otáčením dokola.

Pokud bylo možno zjistit, nikdo se tímto steeringem dosud nezabýval. Není patrně tak všeobecně užitečný jako steeringy Obstacle Avoidance či People Avoidance, ale na druhou stranu je dobrým příkladem toho, jak se dá vektorový přístup steeringů smysluplně použít i pro vysokoúrovňovější úkoly než jen průchod prostředím bez kolizí. Steering Walk Along vytváří konkrétní vztah mezi dvěma jedinci a navíc dovoluje do jisté míry specifikovat povahu tohoto vztahu i samotných jedinců. Většinou je vhodné použít oba parametry rozšířeného chování. Jsou však případy, kdy je smysluplné jeden z parametrů nepoužít (nezapnout). Například starší pán, líný člověk, příliš sebevědomý člověk, apod. nemusí mít chuť uhýbat tomu druhému. Pro takového jedince je lepší nepoužít parametr `Give Way`. Obdobně vypnut parametru `Wait for Partner` je vhodné pro dvojici přátel/milence/atd., kteří se dlouho neviděli a těší se na sebe natolik, že se seběhnou, i když to pro jednoho z nich bude znamenat zbytečnou zacházku.

4. Kombinace steeringů

Předchozí kapitoly se zabývaly tím, jakou mají steeringy funkci, jak získávají informace o světě a jak tyto informace využívají, aby byla spočítána síla, která má ovlivnit pohyb agenta. Zbývá už tedy jen to poslední – jak požadavky všech steeringů skloubit dohromady. Tato kapitola nabízí diskuzi několika možných přístupů, jejich porovnání a podrobnější popis vybraného řešení. Následující kapitola se zabývá evaluací nejzajímavějších kombinací implementovaných steeringů.

Připomeňme, že každý steering vrací jeden vektor. Je tedy potřeba zkombinovat $n+1$ vektorů: n návratových vektorů aktivních steeringů a původní vektor rychlosti. Touto kombinací vznikne nový vektor rychlosti, který se předá lokomoční vrstvě. Jaké máme požadavky na algoritmus kombinování steeringů?

1. Měl by přispívat k přirozenému chování. To je velmi obecný a náročný požadavek. Konkrétně je snaha, aby byly splněny nejdůležitější požadavky všech zúčastněných steeringů (které mohou být ale protichůdné!) a aby byl zvolen vhodný kompromis mezi rychlými reakcemi na požadavky steeringů a plynulým pohybem.

2. Algoritmus by měl být pokud možno dostatečně jednoduchý. Díky tomu bude lépe předvídatelný a bude snazší odhadnout typy situací, které navigační vrstva řeší špatně a které je potřeba vyřešit na vyšší úrovni. Sem mohou patřit situace, kdy mají steeringy protichůdné požadavky.

4.1 Diskuze výběru algoritmu kombinací

Velkou výhodou celého řešení navigační vrstvy pomocí steeringů je, že se požadavky steeringů dají velmi snadno kombinovat. Jelikož jsou požadavky ve formě vektorů, lze požadavky přirozeně sčítat a násobit, či dělit konstantami. Otázkou ovšem je, jak tyto operace využít a zda mají mít na výsledek vliv všechny steeringy. Nabízí se dva hlavní přístupy:

1. Složení všech $n + 1$ vektorů. Složení pak může znamenat součet, průměr, vážený průměr, apod.

2. Stanovení priorit steeringů a složení jen původního vektoru rychlosti a vektoru steeringu s nejvyšší prioritou.

Oba tyto přístupy mají své nevýhody. Skládání všech vektorů může způsobit jakési průměrné řešení, které nevyhovuje ani jednomu požadavku. Např. táhne-li jeden steering agenta vlevo od překážky a druhý vpravo, ve výsledku půjde agent rovně na překážku.

Naopak prioritní přístup uvažuje pouze požadavky steeringu s nejvyšší prioritou, a i kdyby existovalo kompromisní řešení přijatelné pro všechny, tento prioritní přístup si ho patrně nevšimne.

V této práci byl vybrán první přístup. K volbě vedl předpoklad, že vhodná kompromisní řešení budou mnohem častější než průměrná řešení nevyhovující nikomu. První přístup byl doplněn možností stanovit priority ve formě vah.

Dále se podívejme, jak se dají jednotlivé vektory složit. Začneme u obyčejného sčítání vektorů, které budeme postupně vylepšovat.

1. **Součet vektorů**. Pro mnoho situací stačí, aby se všechny vektory obyčejně sečetly. Příkladem může být kombinace steeringů OBSTACLE AVOIDANCE a TARGET APPROACHING. Výsledek bude tvořit součet 3 vektorů, přičemž jeden z nich bude často nulový a druhý stále stejně veliký.

2. **Průměr vektorů**. Obyčejný součet má jedno úskalí. Pokud mají sčítané vektory podobný směr, výsledný vektor rychlosti bude příliš velký. Agent pak půjde rychleji (či se rozeběhne). Přitom ani jeden ze steeringů pravděpodobně nechtěl agenta přinutit k běhu. Proto má smysl výsledný vektor rychlosti podělit číslem $k := n + 1$ (počet sčítaných vektorů).

3. **Průměr nenulových vektorů**. Průměr všech vektorů má také nevýhody. Některé steeringy (např. OBSTACLE AVOIDANCE) často vrací nulový vektor. Tento vektor bychom neměli započítávat do čísla k, kterým se velikost výsledného vektoru rychlosti dělí. I pokud vrací steering velmi malý vektor rychlosti, tak to neznamená, že chce agenta zpomalit, nýbrž spíše velmi mírně ovlivnit směr jeho pohybu. Třetí přístup tedy do hodnoty k započítává jen ty vektory, které nevracejí nulové či velmi malé hodnoty.

4. **Vážený průměr nenulových vektorů**. Chceme-li některé steeringy upřednostnit před jinými, můžeme každému steeringu určit jeho váhu a pak místo obyčejného průměru nenulových vektorů použít vážený průměr nenulových vektorů.

V práci bylo vybráno poslední řešení. Následuje jeho podrobnější popis:

4.2 Vybraný algoritmus

Vektor rychlosti v tomto tiku se spočítá jako:

$$\vec{v_t} = \frac{a \cdot \vec{v_{t-1}} + \sum_{i=1}^{n} b_i \cdot \vec{s_i}}{k}, \quad \text{kde} \qquad k = a + \sum_{i \in \{1,\ldots,n\}, s_i \neq 0} b_i.$$

- $\vec{v_t}$ – vektor rychlosti v tomto tiku
- $\vec{v_{t-1}}$ – vektor rychlosti v minulém tiku
- $\vec{s_i}$ – návratový vektor i-tého aktivního steeringu, kde $i \in \{1, \ldots, n\}$ a n je počet aktivních steeringů
- a – váha původního vektoru rychlosti
- b_i – váha i-tého steeringu, kde $i \in \{1, \ldots, n\}$ a n je počet aktivních steeringů
- k – součet váhy a a vah steeringů, kterým odpovídají nenulové vektory s_i

Váhy b_i jsou volitelné parametry steeringů. Poměrně značný význam má a, váha původního vektoru rychlosti. Čím má tento vektor vyšší váhu, tím je chování plynulejší. Navigační vrstva využívá speciální výpočet této váhy – podle velikosti původního vektoru rychlosti.

$$a = \max\left(1, \min\left(2,\ 3 - \frac{v_{t-1}}{WALK}\right)\right).$$

Konstanta `WALK` určuje základní rychlost chůze a odpovídá 220 $\mathrm{cm}_{UT}/\mathrm{s}$. Pokud se agent pohybuje normálně rychlou chůzí, je váha $a = 2$. Pokud jde však rychleji (či přímo běží), nechceme, aby pokračoval v běhu, pokud na tom nemá zájem jiný vektor. Tento mechanismus tedy zařídí, že přirozeně zpomalí na chůzi.

4.3 Vybraný algoritmus – rozšíření

Následuje popis dvou vylepšení základního algoritmu kombinování steeringů.

4.3.1 Možnost navýšení rychlosti

První vylepšení se týká navyšování rychlosti. Na ukázku si představme situaci, kdy je aktivní pouze steering Obstacle Avoidance. Když se blíží agent překážce, tento steering agenta jednak zpomalí, jednak odkloní od překážky. Jakmile se ale agent vyhne překážce, Obstacle Avoidance vrací nulový vektor a agent pokračuje tak pomalu, jak ho zpomalilo vyhýbání se překážce. V tuto chvíli by však bylo přirozenější, aby agent opět zrychlil na nějakou základní rychlost.

Každý steering má tedy možnost zvolit, zda chce či nechce, aby se rychlost navýšila. Pokud všechny steeringy chtějí, aby se rychlost navýšila, a agent jde pomaleji, než je základní rychlost chůze, rychlost je navýšena. Pokud je alespoň jeden steering proti, rychlost zůstane nenavýšena.

4.3.2 Možnost zastavení a natočení

Steeringy mají velké možnosti, pokud se má agent pohybovat. Jsou ale chvíle, kdy může chtít steering agenta zastavit (typicky, když dojde k cíli). Je otázka, zda má mít steering právo agenta zastavit a jakou to má mít prioritu oproti požadavkům ostatních aktivních steeringů.

Bylo vybráno řešení, které je určitým kompromisem. Steering má možnost vrátit tzv. „požadavek na zastavení“. Pokud je tento steering jediný aktivní steering, který vrací nenulový vektor, agent se zastaví. Jako výsledek tohoto steeringu se použije tzv. „stop-vektor“, jehož směr je přesně opačný než směr původního vektoru rychlosti $\vec{v_{t-1}}$ a velikost je $a \cdot |\vec{v_{t-1}}|$.

Pokud je aktivní jiný steering a vrací nenulový vektor, agent se pravděpodobně nezastaví. To je ovšem poměrně přirozené chování. Například když agent dojde k cíli, zastaví se. Pokud pak kolem něj projde jiný agent příliš blízko, první agent mu uhne. A pak se opět vrátí.

Kromě možnosti zastavení má steering možnost určit směr, kam se agent po zastavení natočí. To může být poměrně podstatné. Např. když se vůdce i následovník zastaví, je potřeba, aby následovník viděl na vůdce, aby si všiml, když se vůdce opět začne pohybovat.

Zde je mírný technický problém, že když se agent zastaví a následně někam natočí, nepřehrává se na něm žádná animace typu otáčení s přešlapováním. Postavička zůstává v základní pozici, ve které je otáčena. Výsledek nevypadá moc přirozeně. Nicméně toto je spíše otázka lokomoční vrstvy.

Některé steering taktéž využívají tzv. „zpomalovací vektor“. Ten má stejný směr jako „stop-vektor“ a velikost velkou nejvýše jako „stop-vektor“. Po složení

s aktuálním vektorem rychlosti vynásobeným váhou a vznikne vektor stejného směru, avšak menší velikosti.

4.4 Shrnutí

V této kapitole byl popsán algoritmus kombinování steeringů (vážený průměr nenulových steeringů) a jak se z nich vypočítá vektor rychlosti pro následující tik. Zároveň byla popsána tři vylepšení, jak může steering vyslat požadavek, aby se zvýšila rychlost agenta (na standardní rychlost chůze), aby se agent zastavil a případně ještě natočil na daný směr, či aby zpomalil.

Tak vznikne vektor rychlosti, který se již vynásobí parametrem steeringu `velocity multiplier`, přeškáluje rychlostní funkcí, podle výsledku se určí typ pohybu (chůze či běh) a přeškálovaný vektor rychlosti se předá lokomoční vrstvě, která daný pohyb vykoná.

Proces nastavování konstant pro chůzi, běh, velikosti návratových vektorů steeringů a dalších konstant, které na to mají vliv, byl poměrně složitý. Steering se totiž chová rozdílně, pokud je aktivní jako jediný, či pokud je v kombinaci s jinými steeringy. Bylo nutno navrhnout mechanismus skládání i jednotlivé steeringy tak, aby steeringy fungovaly rozumně, pokud jsou v kombinaci sami, i pokud jich je více. Ve výsledku steeringy fungují velmi rozumně i s vahami jedna.

Nevýhodou tohoto přístupu je, že steering neví, které ostatní steeringy jsou v danou chvíli aktivní. To může způsobovat problémy. Například jde-li agent přímo kolmo na překážku, steering Obstacle Avoidance se náhodně rozhoduje, na jakou stranu agenta stočí. Takto se rozhodne např. pro pravou stranu. Avšak ve stejném tiku může jiný steering vracet sílu doleva. To může způsobit, že se agent nestočí ani na jednu ani na druhou stranu. Kdyby byl steering Obstacle Avoidance věděl o požadavcích ostatních steeringů, nemusel volit stranu náhodně – a výsledný pohyb by lépe splňoval požadavky všech steeringů.

To by šlo vylepšovat různými metodami jako jsou tzv. „inverzní steeringy“ [1] apod. Nicméně výše popsané řešení si udržuje výhodu jednoduchosti, tedy i předvídatelnosti, která je podstatná. Vzhledem k tomu se zdá výše navržené řešení jako vhodný kompromis mezi poměrně jednoduchým, avšak dostatečně funkčním řešením kombinování steeringů.

Jedním z možných vylepšení by bylo oddělit výpočet směru vektoru rychlosti a jeho velikosti. Pokud chce steering v aktuálním řešení agenta zpomalit, musí použít „zpomalovací vektor“.

Možnostmi kombinování steeringů se zabývají i tyto práce: [25, 26]. Uvažují podobné možnosti kombinování požadavků, jako ty, které byly zmíněny v této kapitole, nicméně nechají uživatele (rozhodovací vrstvu apod.) si vybrat způsob, kterým budou požadavky kombinovány. Uživatel má tedy větší výběr. Na druhou stranu musí vědět, jak si zvolit (případně může existovat nějaká komponenta, která to za něj rozhodne). V této práci byla z důvodů snazšího používání vybrána pouze jedna varianta kombinování, v případě rozšíření by však nebylo náročné doplnit i jiné varianty.

5. Grafická aplikace

Kromě samotné implementace navigační vrstvy (jejíž výsledek je nazýván knihovna steeringů) byla vytvořena grafická aplikace pro spouštění virtuálních agentů ovládaných steeringy. Uživatel se může podívat, jak steeringy fungují, a testovat různá nastavení parametrů. Aplikace navíc obsahuje speciální nástroj pro rekonstrukci průběhu pohybů, který napomáhá hlubšímu porozumění vlastností steeringů. Grafická aplikace má tři komponenty:

1. **Základní ovládání**, viz Obrázek 5.1. Tato komponenta umožňuje:
   (a) Přidávat do scény agenty.
   (b) Měnit nastavení agentů (počáteční pozice, počáteční natočení, atd.).
   (c) Měnit nastavení steeringů (parametry a váhy).
   (d) Přidat ke scéně vlastní popis.
   (e) Uložit či načíst scénu.
   (f) Spustit, pozastavit, či zastavit běh scény a uložit průběh scény, který pak lze zobrazit v komponentě **Trajektorie**.
   (g) Měnit parametry i za běhu a sledovat, jak se změny projeví.
2. **Mapa města**, viz Obrázek 5.2. Základem této komponenty je obrázek mapy virtuálního světa (města) – v pohledu shora dolů (z ptačí perspektivy). Zobrazuje a umožňuje pohodlně měnit tyto údaje:
   (a) Počáteční lokace a rotace agentů.
   (b) Cílové lokace steeringů Target Approaching, Path Following a Walk Along.
   (c) Cestu steeringu Path Following.
   (d) Soustředné kruhy kolem cíle (zobrazující vzdálenosti od cíle) steeringu Target Approaching. (Ty jsou jen zobrazovány.)
3. **Trajektorie**, viz Obrázek 5.3. Základem této komponenty je opět obrázek s ptačím pohledem na město. Na něm se vykresluje průběh trajektorií jednotlivých agentů z odehrané scény. Součástí komponenty je i časová osa, pomocí které lze sledovat, kde se který agent vyskytoval v daném tiku, kam byl natočený a které síly na něj působily. Komponenta umožňuje:
   (a) Načíst výsledky běhu jedné či více scén.
   (b) Nastavit, co vše je vykreslováno. Na mapě může být vykreslováno:
      i. Základní nastavení scény – znázorněno pomocí obdobných grafických značek, jako používá komponenta **Mapa města**.
      ii. Dráhy pohybů agentů.
      iii. Aktuální pozice a natočení agenta v daném tiku.
      iv. Síly, které na agenta v daném tiku působily (původní vektor rychlosti a síly aktivních steeringů) a jak se složily (nový vektor rychlosti po přeškálování).

(c) Vykreslit na mapě informace o všech či pouze některých agentech.

(d) Zobrazit kompletní výpis parametrů scény.

První dvě komponenty slouží především pro pohodlné nastavování parametrů scény. Pomocí třetí komponenty lze pak zpětně analyzovat důvody chování agentů s daným nastavením steeringů. Zároveň se dá ověřit, zda byly splněny základní či rozšiřující požadavky steeringů, jako např.:

1. Obstacle Avoidance a People Avoidance: zda nedošlo ke kolizi a jak plynule se agent překážce či druhému agentovi vyhnul. To, že agent nešel plynule se pozná tak, že je trajektorie klikatá a/nebo se často mění natočení agenta.

2. Path Following: zda agent nevybočil mimo koridor, s jakým předstihem předcházel vybočení z koridoru a jak plynule se pohyboval.

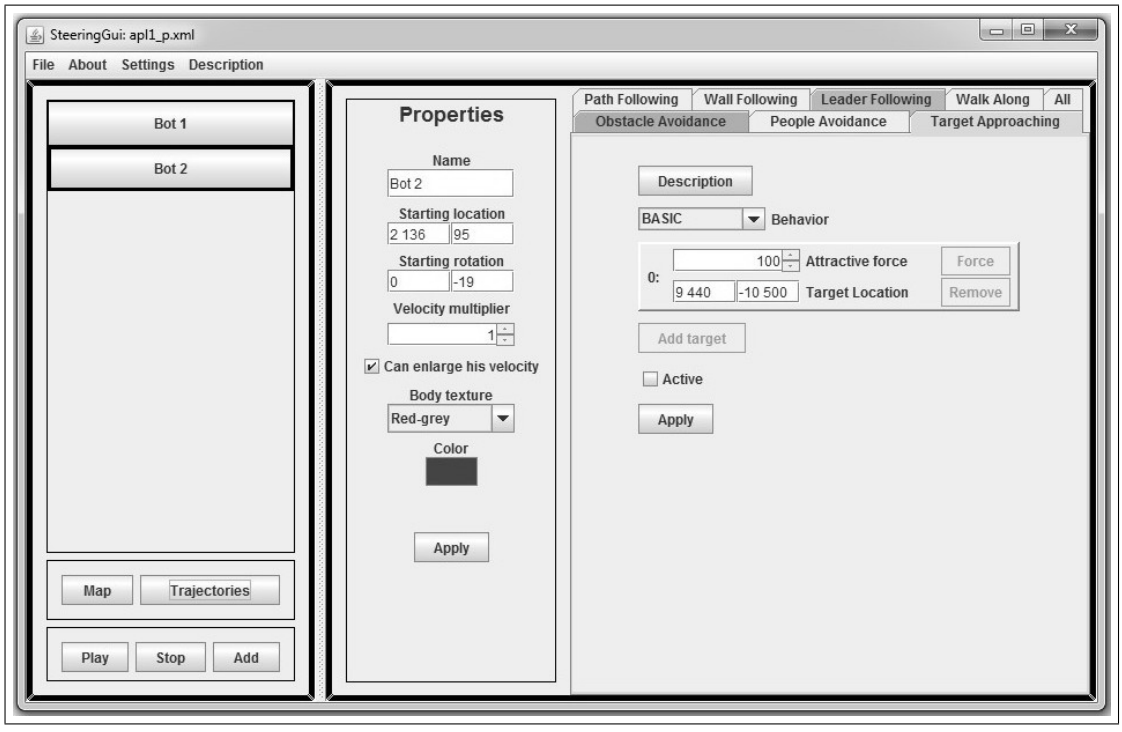
3. Leader Following: zda se agent pohyboval ve správné pozici vůči vůdci, jak rychle vs. plynule reagoval na změny pohybu vůdce.

4. Walk Along: zda šli agenti vedle sebe a došli k cíli.

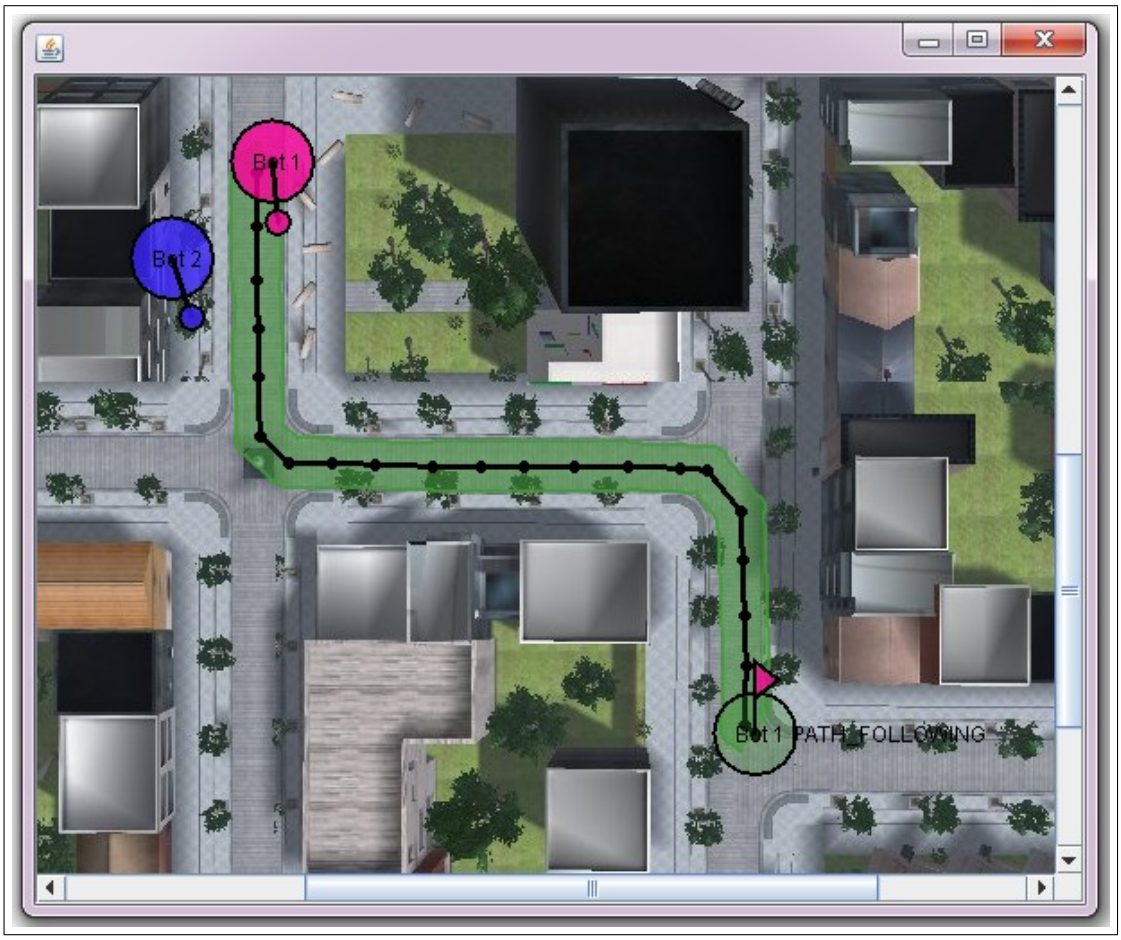
Podstatná výhoda třetí komponenty je ta, že ukazuje výsledky steeringů a dovoluje rozebírat průběh scény.

Poznámka

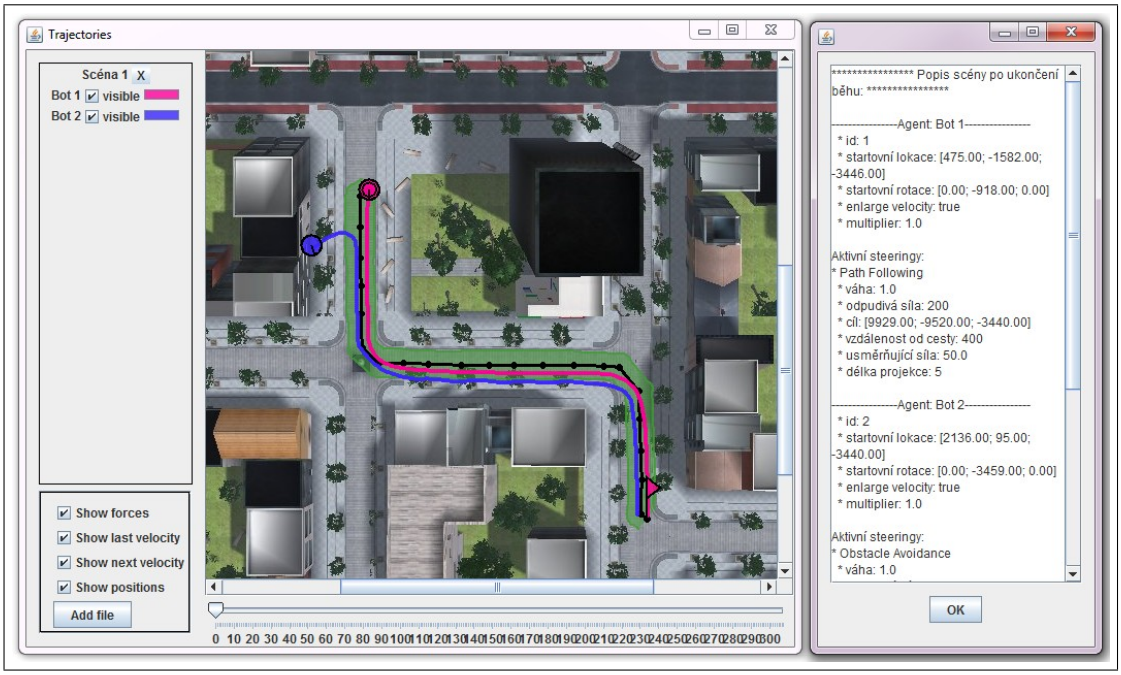
Grafická aplikace nezajišťuje otevření okna s pohledem do virtuálního prostředí, to je třeba spustit zvlášť. Postup je popsán v příloze na CD.



Obrázek 5.1: Základní ovládání – první komponenta grafické aplikace.



Obrázek 5.2: **Mapa města** – druhá komponenta grafické aplikace.



Obrázek 5.3: **Trajektorie** – třetí komponenta grafické aplikace.

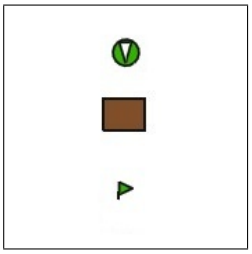
6. Scény

Tato kapitola představuje evaluaci implementované navigační vrstvy. Její součástí jsou ukázkové nastavení scén pro různá zadání. *Scénou* je myšlen seznam agentů, jejich parametrů (iniciální lokace a rotace atd.) a nastavení aktivních steeringů. Jedno zadání může řešit více různých scén. Tato kapitola obsahuje vždy jedno či více nejlepších řešení a případné srovnání, v čem je které lepší. Pokud je napsáno, že je steering použit v plném rozšířeném chování, myslí se tím, že jsou zapnuté všechny parametry pro řešení speciálních situací.

Některá zadání jsou scénáře převzaté z nástroje SteerBench [38], který slouží k hodnocení kvality implementovaných steeringů. Otestování těchto scénářů dovoluje srovnání této práce s případnými jinými, které by scénáře z nástroje SteerBench testovaly. Názvy zadání scénářů z nástroje SteerBench jsou označena hvězdičkou. Vedle zadání je příslušný nákres situace, což jsou obrázky z [38], drobně upravené pro možnosti našeho města. Všechny uvedené scény jsou součástí přílohy na CD, kde k nim lze najít přesné nastavení všech parametrů.

6.1 Solitérní scény – řešení kolizí

6.1.1 K cíli kolem malé překážky*



Obrázek 6.1: Nákres 6.1.1.

Zadání

Agent má dojít k cíli za malou překážkou.

Řešení

Target Approaching v základním chování a Obstacle Avoidance v plném rozšířeném chování.

Obrázek 6.2: Trajektorie 6.1.1.

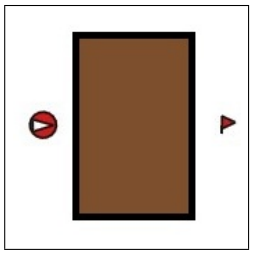
Zhodnocení

Agent překážku poměrně plynule obejde. Jediný nepřirozený moment může nastat ve chvíli, kdy jde vedle překážky (zde vpravo) a začne mu kolidovat boční paprsek (zde levý). Pak se stočí lehce směrem od překážky (zde vpravo) a chvíli trvá, než se stočí zase zpět k cíli. Přirozenější by bylo, aby ukročil stranou, ale dále se pohyboval v původním směru.

Poznámka

Pro toto zadání by šly použít i jiné kombinace steeringů (např. Path Following), není to však potřeba.

6.1.2 K cíli kolem velké překážky



Obrázek 6.3: Nákres 6.1.2.

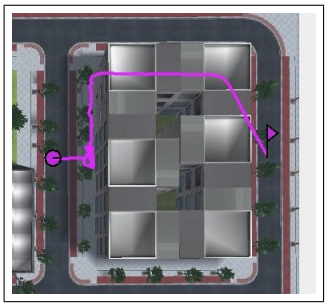
Zadání

Agent má dojít k cíli za velkou překážkou.

Řešení A

Target Approaching v základním chování a Obstacle Avoidance v plném rozšířeném chování.

Zhodnocení A



Obrázek 6.4: Trajektorie 6.1.1 A.

Agent nejde úplně plynule. Ve fázích, kdy jsou síly použitých steeringů hodně odlišné, až úplně opačné, jde agent velmi pomalu, případně často mění směr od budovy a k budově. Pokud není zeď budovy rovná (např. obsahuje drobné výklenky), je velká pravděpodobnost, že se agentovi nepodaří úkol splnit a v některém z výklenků se zasekne. Jinak k cíli většinou dojde, ač se může v průběhu několikrát vrátit, jako zde na ukázce (trajektorie je v daném místě tlustší).

Řešení B

Target Approaching v základním chování a Wall Following v plném rozšířeném chování.

Zhodnocení B

Obrázek 6.5: Trajektorie 6.1.1 B.

Pokud se místo Obstacle Avoidance použije steering Wall Following, agent se s obcházením zdi vypořádá mnohem lépe. Jde plynuleji a většinou se nevrací. Problém může nastat ovšem v místě, kde by se měl od překážky odtrhnout a pokračovat k cíli. Jako vidíme na ukázce, pokud převáží přitažlivá síla zdi, agent pokračuje v obcházení zdi.

Řešení C

Target Approaching v základním chování a Wall Following v plném rozšířeném chování se zmenšenými odpudivými silami a zvětšenými přitažlivými silami.

Zhodnocení C

Obrázek 6.6: Trajektorie 6.1.1 C.

Zde s výhodou využijeme možnost nastavit steering `Wall Following`, aby se choval více jako `Obstacle Avoidance`. Snížíme váhy přitažlivých sil ke zdi (parametry `Attractive Weight` a `Convex Weight` nastavíme na 0.8) a zároveň zvýšíme váhy odpudivým sil

(parametry `Repulsive Weight` a `Concave Weight` nastavíme na 1.2), aby se snížila pravděpodobnost, že agent půjde příliš blízko zdi či do ní vrazí, když bude cíl přesně ve směru za zdí. Vidíme, že agent plynule budovu obejde a ve správném místě se stočí směrem k cíli.

Řešení D

Path Following v plném rozšířeném chování (a případně Obstacle Avoidance v plném rozšířeném chování, kdyby bylo riziko, že se na cestě vyskytují další překážky.)

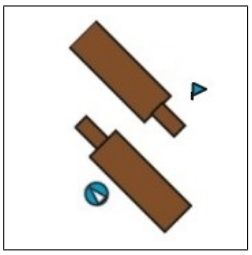
Zhodnocení D

Obrázek 6.7: Trajektorie 6.1.1 D.

Výhodou předchozích řešení bylo, že nebylo třeba nic plánovat, nebyla potřeba téměř žádná paměť a chování by se dalo označit za reaktivní. Pokud však máme k dispozici síť navigačních bodů, můžeme si nechat spočítat nejkratší cestu ze startu do cíle a tu předat steeringu Path Following. Výhodou takové cesty bude, že nepovede přímo přes žádné statické překážky. Na ukázce vidíme, že agent obešel budovu přirozeně a bez obtíží. Nevýhodou tohoto řešení je, že vyžaduje předpracovaná data o mapě (síť navigačních bodů) a plánování cesty. Pokud nebude síť navigačních bodů dostatečně hustá, může se také stát, že agent bude agent zbytečně a nepřirozeně obcházet volný prostor.

6.1.3 K cíli skrze „S-zatáčky“*

Zadání



Obrázek 6.8: Nákres 6.1.3.

Agent má dojít k cíli skrze zatáčku ve tvaru písmene „S“. V našem případě ji ohraničují pult a stěna místnosti.

Řešení

Target Approaching v základním chování a Obstacle Avoidance v plném rozšířeném chování.

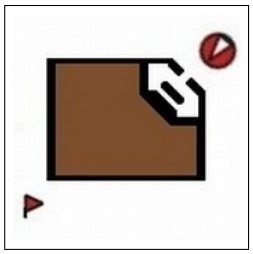
Zhodnocení

Obrázek 6.9: Trajektorie 6.1.3.

Agent projde zatáčkami poměrně plynule. Mohlo by se však stát, že by se o jednu z překážek zasekl. (Ve městě není mnoho podobných překážek, situaci tedy nelze průkazněji otestovat.)

6.1.4 K cíli kolem „U-překážky“

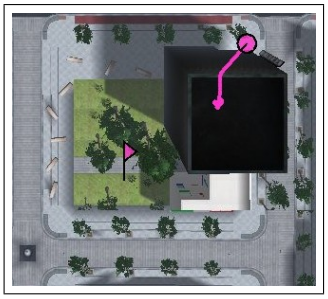
Zadání



Obrázek 6.10: Nákres 6.1.4.

Agent má dojít k cíli za konkávní překážkou ve tvaru písmene „U“. V našem případě se jedná o velmi složitou situaci, kde mezi startem a cílem je budova s vchodem do místnosti. Pokud by místo celé místnosti byl v budově jen malý výklenek, agent by řešil situace mnohem lépe.

Řešení A



Obrázek 6.11: Trajektorie 6.1.4 A.

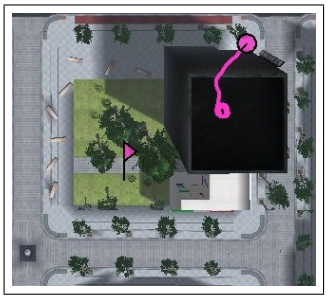
Target Approaching v základním chování a Obstacle Avoidance v plném rozšířeném chování.

Zhodnocení A

Agent vejde do místnosti, kde se zasekne, či otáčí dokola v nejzazším místě místnosti. K cíli tedy rozhodně nedojde.

Řešení B

Target Approaching v základním chování a Wall Following v plném rozšířeném chování.



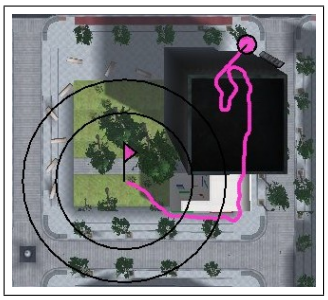
Obrázek 6.12: Trajektorie 6.1.4 B.

Zhodnocení B

Zde je již větší naděje, že agent situaci zvládne. Agent vejde do místnosti a začne ji obcházet. Bohužel jakmile se dostane ke stěně s dveřmi, síla k cíli ho stočí opět dovnitř místnosti. Tedy ani tato kombinace nestačí pro úspěšné splnění zadání.

Řešení C

Target Approaching v rozšířeném chování a Wall Following v plném rozšířeném chování. Síla k cíli začne působit až od určité vzdálenosti k cíli.



Obrázek 6.13: Trajektorie 6.1.4 C.

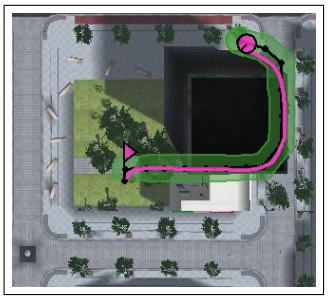
Zhodnocení C

Ze zkušeností s předchozího řešení víme, že pokud působí na agenta síla k cíli ve chvíli, kdy vejde do místnosti, není schopen z ní vyjít. Steering `Target Approaching` dovoluje trik, kterým se to dá vyřešit. Nastavíme přitažlivou sílu k cíli až od vzdálenosti mimo místnost. Agent tedy vejde do místnosti, obejde ji, vyjde ven (to se mu nemusí podařit, jsou-li dveře příliš úzké a zasáhnou-li paprsky pokračování zdi za dveřmi), obejde budovu zvenčí – a až tehdy na něj začne působit přitažlivá síla, která ho přitáhne k cíli. Agent projde situaci velmi

zdárně. Nicméně toto řešení má tři zřejmé nevýhody. Zaprvé někdo musí nastavit vzdálenost působení přitažlivé síly, k čemuž potřebuje vědět, kam až zasahuje místnost. Zadruhé reálný člověk by pravděpodobně do místnosti vůbec nevešel. Zatřetí pokud by ho zeď budovy nedovedla k místu, kde na něj začne působit přitažlivá síla k cíli, pravděpodobně by k cíli vůbec nedošel.

Řešení D

PATH FOLLOWING v plném rozšířeném chování (a případně OBSTACLE AVOIDANCE v plném rozšířeném chování, kdyby bylo riziko, že se na cestě vyskytují další překážky.)



Obrázek 6.14: Trajektorie 6.1.4 D.

Zhodnocení D

Máme-li možnost spočítat cestu od startu do cíle, je toto nejlepší možné řešení. Agent vůbec nevejde do místnosti a projde co nejkratší cestou k cíli. Díky steeringu `Path Following` jde navíc plynule.

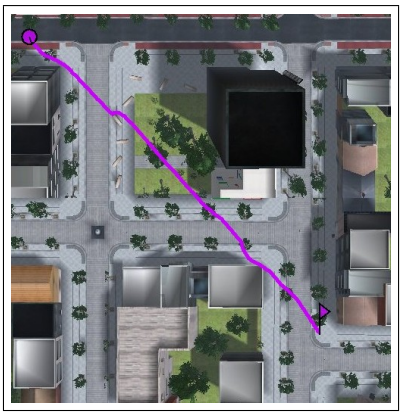
6.1.5 Složitější průchod městem

Zadání

Agent má projít městem k cíli, aniž by vrážel do překážek. Dále se hodnotí podobnost s lidským chováním.

Řešení A

TARGET APPROACHING v základním chování a OBSTACLE AVOIDANCE v plném rozšířeném chování.

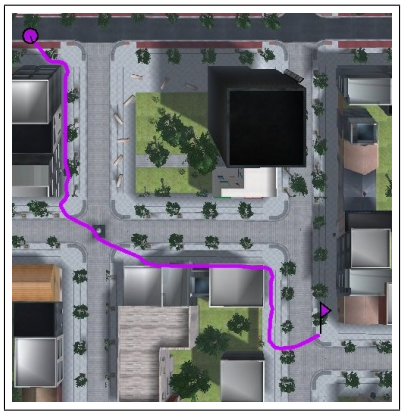


Obrázek 6.15: Trajektorie 6.1.5 A.

Zhodnocení A

Pokud se agent nedostane do situace, ve které by se mohl zaseknout (nějaká „U-překážka“), a nemusí obcházet velké budovy, projde poměrně přímou cestou k cíli a správně se vyhýbá překážkám. Ignoruje přitom, zda jde po chodníku, skrze ulici či parkem. Nicméně jeho chování je velmi dobře předvídatelné.

Řešení B



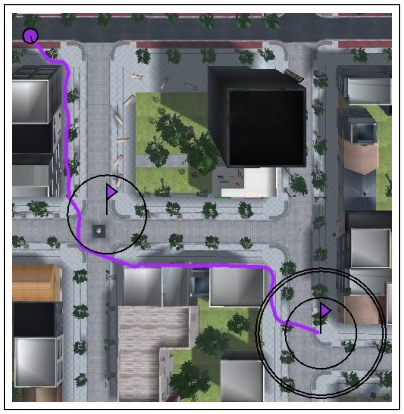
Obrázek 6.16: Trajektorie 6.1.5 B.

Target Approaching v základním chování a Wall Following v plném rozšířeném chování.

Zhodnocení B

Agent se lépe vypořádá s náročnějšími překážkami (větší budovy či výklenky). Především ale projde městem tak, že chodí převážně po chodnících, čímž velmi dobře připomíná reálného člověka. Tuto cestu po chodnících si nemusí nijak plánovat, ani k tomu nepotřebuje mít zaznačené, kde chodníky jsou a kde ne, naopak k celému tomuto chování mu stačí informace o okolních zdech a sčítání sil. Toto je výhodou samozřejmě pouze na mapě, kde jsou kolem budov chodníky, tedy typicky ve městech. Tato kombinace představuje zároveň určité nebezpečí – ve chvíli, kdy je dosti odlišný směr k cíli a síla steeringu `Wall Following`, agent se může chovat nevhodně. První nebezpečí je, že je cíl za zdí a agent jde příliš blízko zdi. Druhé nebezpečí je, že v osudnou chvíli, kdy by se měl od zdi odtrhnout či nezabočit za roh, převáží síla steeringu `Wall Following` a agent se začne vzdalovat od cíle.

Řešení C



Obrázek 6.17: Trajektorie 6.1.5 C.

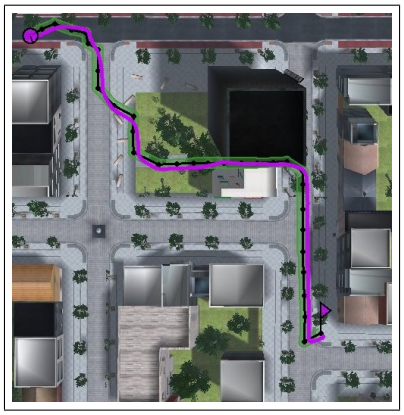
Target Approaching v rozšířeném chování a Wall Following v rozšířeném chování. Síla k cíli má jeden zlom ve vzdálenosti od cíle shodné se vzdáleností zdi od cíle. Od této vzdálenosti blíže se začne síla zvětšovat, aby cíl přitáhl agenta a převážil nad přitažlivou silou ke zdi. Doprostřed křižovatek jsou umístěny cíle s konstantní odpudivou silou dosahující kus od zdi. To zvyšuje pravděpodobnost, že agent nepůjde skrze křižovatku, ale obejde ji kolem po chodníku. Steering Wall Following má snížené váhy `Concanve Weight` a `Attractive Weight` a zvýšené váhy `Convex Weight` a `Repulsive Weight`, aby ho zeď příliš nepřitahovala a nezabraňovala mu tak chůzi k cíli.

Zhodnocení C

Oproti předchozímu nastavení je menší riziko, že půjde agent příliš blízko zdi i že půjde ke zdi ve chvíli, kdy má jít od ní. Na druhou stranu je zde vyšší riziko, že se agent odtrhne od zdi příliš brzy a bude pokračovat přímo k cíli. Dalo by se říci, že se jedná o určitý kompromis mezi řešením A a řešením B a z každého si bere „to lepší“. Pro řešení B i C platí, že pokud nekoliduje žádný z jeho paprsků (například na začátku), agent zeď a chodníky nevnímá a jde přímo k cíli. Jakmile však se dostane do blízkosti zdi, je pak schopný jít plynule podél ní.

Řešení D

Path Following v rozšířeném chování a Obstacle Avoidance v plném rozšířeném chování.

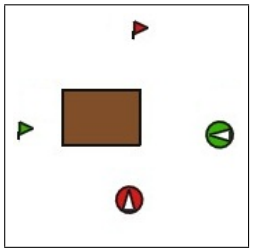


Obrázek 6.18: Trajektorie 6.1.5 D.

Zhodnocení D

Pokud je cesta mezi startovní a cílovou pozicí dobře zmapovaná sítí navigačních bodů, představuje tato kombinace opět velmi vhodné a stabilní řešení. Agent dojde k cíli plynule a navíc je výhoda, že cesta přes navigační body obvykle přirozeně obchází překážky, tedy není tolik překážek, kterým by se musel agent vyhýbat pomocí steeringu `Obstacle Avoidance`. Nevýhodou je, že toto řešení vyžaduje (dobře) zmapovanou oblast a agent si musí pamatovat cestu.

6.2 Skupinové scény – řešení kolizí[1]



Obrázek 6.19: Nákres 6.2.1.

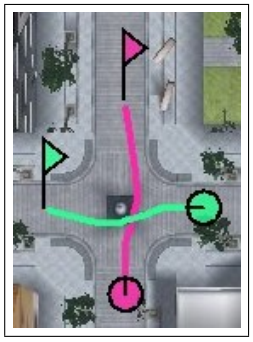
6.2.1 2 agenti proti sobě + překážka*

Zadání

Dva agenti jdou naproti sobě, poblíž místa hrozícího střetu se navíc vyskytuje překážka. Agenti se mají vyhnout sobě i překážce.

Řešení

Target Approaching v základním chování, Obstacle Avoidance v plném rozšířeném chování a People Avoidance v plném rozšířeném chování.



Obrázek 6.20: Trajektorie 6.2.1.

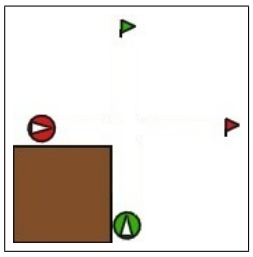
Zhodnocení

Agenti se plynule vyhnou sobě i překážce. Uhnou si navzájem, případně zpomalí, pokud se tak dá lépe předejít srážce.

6.2.2 2 agenti + překvapení*

Zadání

Dva agenti jdou každý z jedné strany budovy a do poslední chvíle na sebe nevidí. Přesto by do sebe neměli vrazit a měli by dorazit každý ke svému cíli.

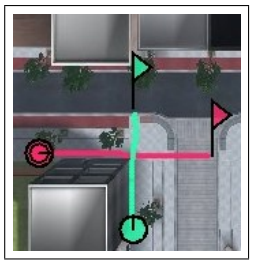


Obrázek 6.21: Nákres 6.2.2.

[1] Scény se 2 – 5 agenty.

Řešení

Target Approaching v základním chování a People Avoidance v plném rozšířeném chování (+ případně Obstacle Avoidance v plném rozšířeném chování).



Obrázek 6.22: Trajektorie 6.2.2.

Zhodnocení

I přesto, že se agenti spatří až na poslední chvíli (asi 2 tiky před srážkou), jsou schopní se sobě vyhnout, neboť jeden z nich rychle zpomalí a druhý výrazně zrychlí. Je určité riziko, že se oba rozhodnou pro to samé (např. zrychlit), avšak to se může stát i reálným lidem, jedná se tedy o pozitivum tohoto chování.

Poznámka

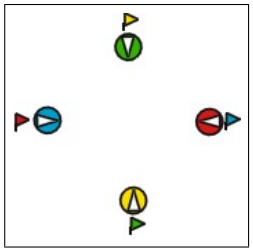
Pro srovnání Obrázek 6.23 ukazuje, jak situace dopadne, není-li použit steering People Avoidance, ale jinak je nastavení scény stejné. Vidíme, že se agenti přesně v místě překvapení srazí a vytlačují se až k protější zdi, neboť ani jeden nechce pustit toho druhého jeho směrem.

Obrázek 6.23: Trajektorie 6.2.2 2.

6.2.3 4 agenti proti sobě*

Zadání

Čtyři agenti jdou ze čtyř stran proti sobě tak, že se mají setkat v téměř stejnou chvíli na jednom místě. Neměli by do sebe vrazit a místo toho by se sobě měli plynule a přirozeně vyhnout a dorazit každý ke svému cíli.



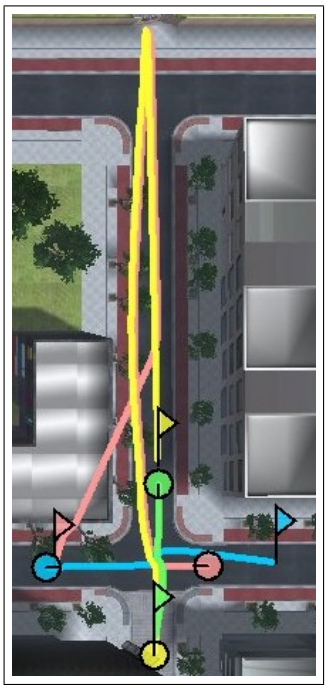
Obrázek 6.24: Nákres 6.2.3.

Řešení A

Target Approaching v základním chování.

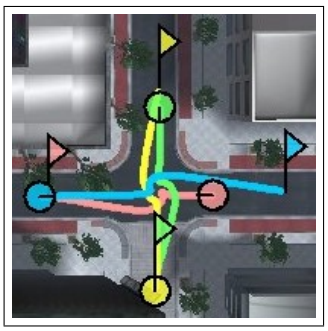
Zhodnocení A

Může se stát, že agenti projdou bez kolizí i když se jim dráhy kříží. Nicméně pokud je situace nastavena tak, že se setkají v místě křížení drah ve stejný čas, dojde ke kolizi a případně i jevu *vytlačování*, jak je vidět na této ukázce, kde je tento jev velmi výrazný (vytlačovaní agenti došli až první zdi, od které se mohli odrazit a vrátit se zpět). Toto řešení nijak nepředchází kolizím. Má smysl ho používat pouze tehdy, pokud je pravděpodobnost srážek agentů malá a potřebujeme výpočetně co nejméně náročné řešení.



Obrázek 6.25: Trajektorie 6.2.3 A.

Řešení B



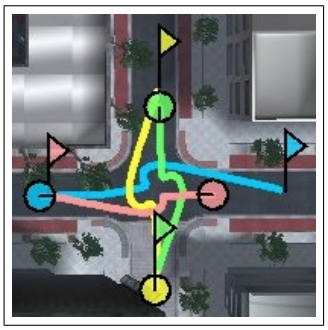
Obrázek 6.26: Trajektorie 6.2.3 B.

Target Approaching v základním chování a People Avoidance v základním chování.

Zhodnocení B

Toto nastavení má několik možností, jak může dopadnout (záleží na drobných odchylkách startovních lokací apod.). Agenti mohou mít problém, že jdou přímo proti sobě a nebudou se umět obejít (chybí jim parametr `Circumvention`), stejně tak může dojít k *vytlačování*. Ale při symetrickém rozložení lokací agentů těsně před střetem může dojít k velmi zajímavému řešení, jako na této ukázce. Agenti se těsně před místem střetu stočili každý o 90° doleva, vytvořili malý kruh, prošli každý polovinu obvodu kruhu a následně vyšli z kruhu každý ke své lokaci. Kromě toho, že vizuální podoba tohoto „mlýnku“ je velmi estetická, tak je překvapivé, že takto rafinovaný útvar vznikl pouze ze základních sil těchto steeringů. Řešení je v podstatě chytré, jako kdyby byl místo křižovatky kruhový objezd. Nicméně je nutno uznat, že lidé se na obyčejných křižovatkách takto obvykle nevyhýbají.

Řešení C

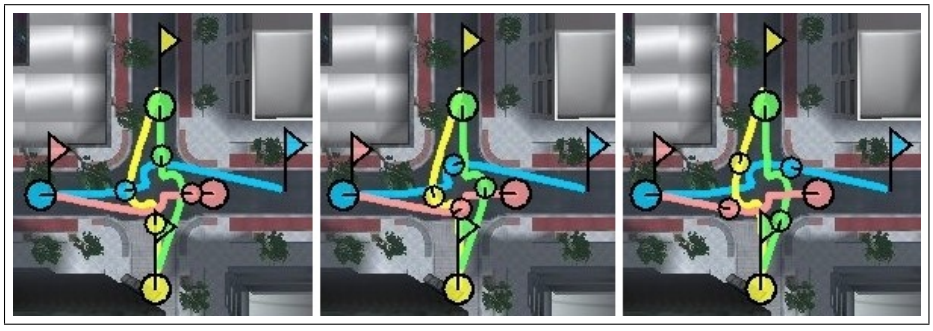


Obrázek 6.27: Trajektorie 6.2.3 C.

Target Approaching v základním chování a People Avoidance v základním chování + aktivní parametr `Circumvention`.

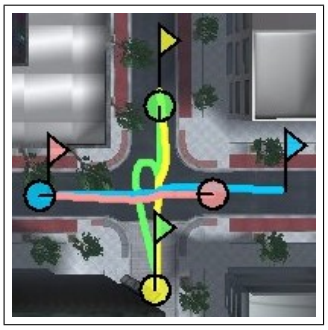
Zhodnocení C

Ukázková situace dopadla velmi podobně jako v předchozím případě, tentokrát agenti utvořili kruh větší a dříve, což odpovídá vlastnostem parametru `Circumvention`. Na Obrázku 6.28 je vidět průběh této scény. Zhodnocení je v podstatě velmi podobné jako v předchozím případě, snad jen s tou výjimkou, že zde je sníženo riziko kolizí.



Obrázek 6.28: Průběh řešení C. Agenti se po kruhu obejdou a pokračují ke cíli.

Řešení D



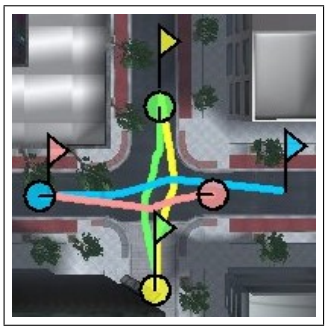
Obrázek 6.29: Trajektorie 6.2.3 D.

TARGET APPROACHING v základním chování a PEOPLE AVOIDANCE v základním chování + aktivní parametry `Deceleration` a `Acceleration`.

Zhodnocení D

Ukázková situace dopadla přesně podle vlastností steeringu PEOPLE AVOIDANCE v základním chování + s aktivními parametry `Deceleration` a `Acceleration`. Díky tomuto parametru dva protilehlí agenti počkali, až si druzí dva vymění místa, a až pak si vyměnili místa sami. To je poměrně přirozené. Nicméně při výměně míst šli oba agenti ve dvojici přesně proti sobě. Ukázková situace potvrzuje, že to může dopadnout dvěma způsoby: buď do sebe vrazí, ale těsně po srážce se stočí každý na svou stranu a pokračují k cíli (první pár, modrý a červený agent), nebo se jeden z nich obrátí na druhou stranu a až časem se stočí zpět a dojde ke svému cíli (druhý pár, zelený a žlutý agent). Kolize způsobila absence parametru `Circumvention`. Toto řešení není příliš vhodné.

Řešení E

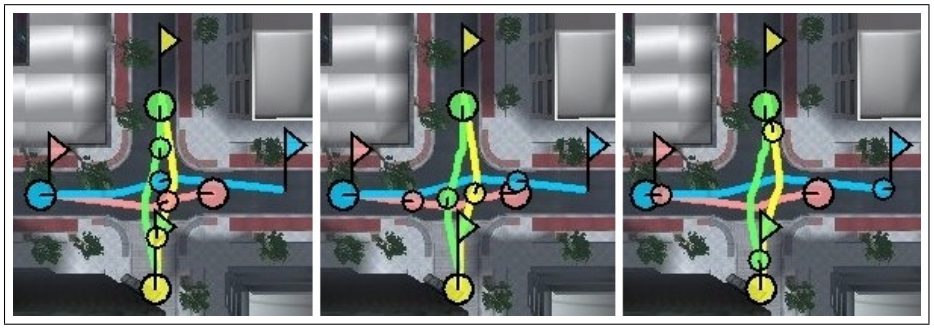


Obrázek 6.30: Trajektorie 6.2.3 E.

TARGET APPROACHING v základním chování a PEOPLE AVOIDANCE v plném rozšířeném chování.

Zhodnocení E

Toto je pravděpodobně nejlepší nastavení pro dané zadání. Na Obrázku 6.31 vidíme průběh celé situace. Zelený a žlutý agent zpomalují, aby nechali projít modrého a červeného agenta. Ti se plynule vyhnou každý z jedné strany a pokračují ke svým cílům. Následně opět zrychlí první dva, kteří se taktéž plynule vyhnou a dojdou volnou cestou ke svým cílům.



Obrázek 6.31: Průběh řešení E. Dva protilehlí agenti počkají, až se vymění druhá dvojice, a následně se vymění oni sami.

Poznámka

Situace s více agenty jsou obecně hůře předvídatelné. Například existuje určité riziko, že se složí síly od všech ostatních agentů tak, že se agent obrátí a chvíli půjde pryč od místa křížení. Ač by se to dalo vykládat různými způsoby (třeba že se ostatních lekl, odradil ho velký dav apod.), tak to není žádoucí chování a poskytuje prostor pro další rozšíření a vylepšení (např. kontrola na úrovni jednotlivých steeringů, že pokud je více důvodů pro jeden směr a velikost rychlosti, či především o zpomalení, tak se nevezme jejich součet, ale jen jeden reprezentant apod.).

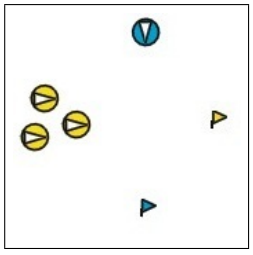
Závěr

Z testů pro toto zadání i jemu podobné vychází, že plné rozšířené chování steeringu `People Avoidance` je ideální nastavení, pokud chceme přirozeně předcházet kolizím agentů. Výjimku z tohoto pravidla tvoří situace, kdy chceme specifikovat povahu jedince, čímž se zabývají následující zadání.

6.2.4 Malá skupina + 1 křížem*

Zadání

Malá skupina agentů (v našem případě 3) jde jedním směrem a jiný agent jde směrem kolmo k nim.



Obrázek 6.32: Nákres 6.2.4.

Řešení A

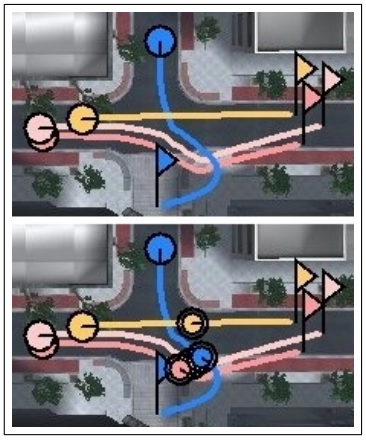
Target Approaching v základním chování a People Avoidance v základním chování.

Zhodnocení A

Tento případ může dopadnout několika způsoby. Pokud se jedná o takto malou skupinu 3 agentů, čtvrtý agent jimi může projít docela přirozeně. Případně dojde k tomu, že si navzájem budou trochu uhýbat, což může způsobit až jev dříve nazývaný jako *vytlačování*. Je podstatné, že v tomto nastavení nebude mít skupina žádnou převahu: stejně tak může skupina přinutit čtvrtého agenta k jinému směru, jako může čtvrtý agent rozhodit agenty ze skupiny.

Řešení B

Target Approaching v základním chování a People Avoidance v základním chování + všichni s aktivním parametrem `Circumvention`.



Obrázek 6.33: Trajektorie 6.2.4 B.

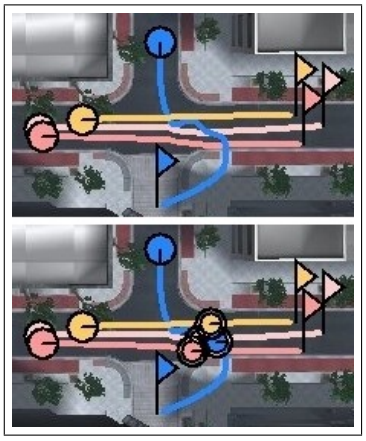
Zhodnocení B

Situace může dopadnout opět různě. Agenti se mohou vyhnout přirozeněji než v předchozím případě, je však mnohem větší pravděpodobnost jevu *vytlačování*. Je zde ještě znatelnější, jak může singulární

agent zmanipulovat část skupiny, jak je vidět na ukázce.

Řešení C

Target Approaching v základním chování a People Avoidance v základním chování + čtvrtý agent s aktivním parametrem `Circumvention`.



Obrázek 6.34: Trajektorie 6.2.4 C.

Zhodnocení C

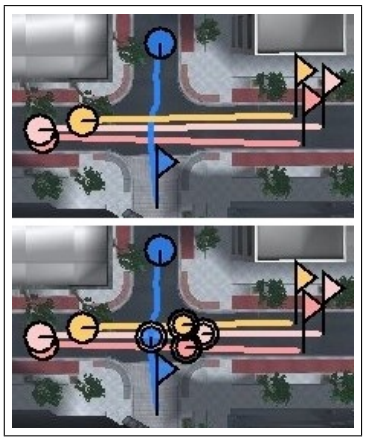
Zde je zajímavé, že skupina získá větší váhu. Už se patrně nestane, že by singulární agent vytlačil část skupiny, naopak skupina může „strhnout“ agenta mezi sebe a na chvíli mu vnutit jejich směr. To je poměrně přirozené i pro lidské agenty: pokud se někdo vrhne do davu, dav ho strhne s sebou.

Řešení D

Target Approaching v základním chování a People Avoidance v základním chování + čtvrtý agent s aktivními parametry `Circumvention`, `Deceleration` a `Acceleration`.

Zhodnocení D

Toto řešení dává nejlepší výsledky. Ve většině případů ke srážce nedojde a singulární agent si počká, až skupina přejde a až pak teprve projde. V případě, že si myslí, že by to stihl projít rychle před nimi, tak naopak zrychlí a projde dříve on. To vypadá velmi přirozeně: člověk také většinou odhadne situaci a pokusí se projít ještě před davem, nebo až po něm, místo aby se vrhal do něj. Zde ovšem záleží na povaze agenta, zda se rozhodne pro D či C. To dává dobré možnosti rozhodovací vrstvě vyjadřovat povahu agenta.



Obrázek 6.35: Trajektorie 6.2.4 D.

Poznámka

Ještě by se daly zkoumat další kombinace, pro které zde již není místo. Nejzajímavější z nich je kombinace, kdy mají všichni agenti plné rozšířené chování. Pak má opět stejnou váhu singulární agent i skupina a může se stát, že celá skupina se zastaví, aby pustila singulárního agenta.

Závěr

Mají-li se chovat agenti jako jednotlivci, je rozumné použít plné rozšířené chování steeringu People Avoidance. Mají-li se chovat spíše davově, je vhodné pro ně použít základní chování steeringu People Avoidance.

6.3 Jiné sociální interakce

První dvě sekce se zabývaly předcházením kolizí s živými i neživými objekty. To je v podstatě nejčastější úkol, na který se steeringy obecně dosud využívají. Zajímavé ovšem je, že přístup steeringů se dá velmi hezky použít i pro jiné úkoly, nazvěme je „sociální“. Tato sekce nabízí několik ukázek.

6.3.1 Sociální aspekty steeringu People Avoidance

Vzdálenost

Parametr `Distance` může pomoci k vytváření vztahů mezi agenty. Například pokud má agent výrazně nižší hodnotu `Distance` než ostatní agenti (např. 150 cm_{UT} vs. 300 cm_{UT}), může působit jako „vtíravý“ člověk, od kterého se ostatní obracejí zády. Naopak pokud má hodnotu parametru `Distance` výrazně vyšší (např. 500 cm_{UT}), může působit jako člověk, co se straní ostatních.

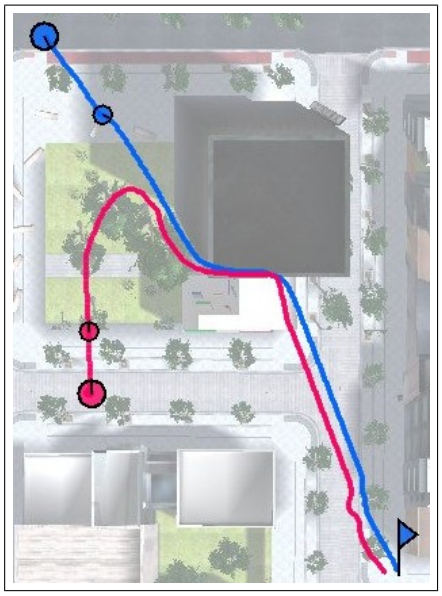
Změna rychlosti

Parametry `Deceleration` a `Acceleration` způsobují, že agent zpomaluje či zrychluje, aby nedošlo ke srážce s jiným agentem. Pokud jeden z agentů tyto parametry nepoužívá, může působit jako dominantní člověk, který nikoho před sebe nepouští, ani nezrychluje. Pokud ostatní agenti tento parametr používají, budou zrychlovat, resp. zpomalovat, aby se s dominantním agentem nesrazili, tedy budou dělat všechno proto, aby mohl volně projít. Zároveň lze využít pouze jeden z parametrů a tím rozlišit, zda chce/nechce agent zpomalovat či zrychlovat.

6.3.2 Dvojice

Zadání

Dvojice agentů má dojít ke společnému cíli. Zajímají nás možnosti, jak rozlišit různé aspekty jejich vztahu: jak si jsou blízcí, zda je jeden z nich dominantní, atd.



Obrázek 6.36: Trajektorie 6.3.2 A.

Řešení A

První agent: Target Approaching, druhý agent: Leader Following formační typ v rozšířeném chování s úhlem 90°.

Zhodnocení A

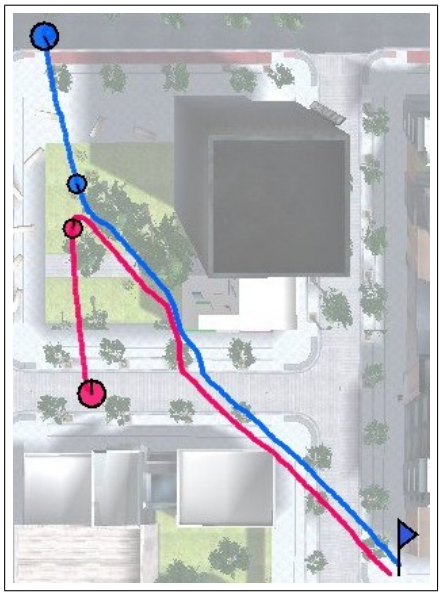
Je znatelné, že je první agent vůdce. Pokud se od sebe vzdálí, vůdce se neohlíží na druhého a jde přímo k cíli, přičemž následovník se ho snaží dohnat. Vše se řídí podle vůdce. To je vhodné pro vztah dvojice, kde jeden je znatelně dominantní a druhý submisivní.

Řešení B

Walk Along v základním chování + aktivní parametr `Circumvention`.

Zhodnocení B

Vztah agentů je mnohem vyrovnanější. Jsou-li od sebe daleko, nejdříve se seběhnou a až pak jdou k cíli. Zpozdí-li se jeden, druhý bez něj nepokračuje. Parametr `Distance` ovlivňuje, zda vypadají jako osoby blízké (půjdou blízko u sebe, tzn. např. hodnoty 100 cm_{UT} – 250 cm_{UT}), nemají tak blízký vztah (tzn. např. hodnoty 250 cm_{UT}– 400 cm_{UT}), nebo se vyloženě nemají rádi, ale z jakéhosi důvodu musí jít spolu (např. hodnoty vyšší než 400 cm_{UT}).



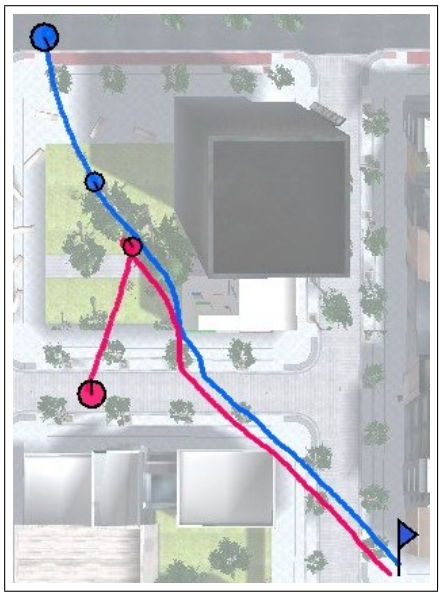
Obrázek 6.37: Trajektorie 6.3.2 B.

Řešení C

Walk Along v plném rozšířeném chování.

Zhodnocení C

Toto řešení je jen lehce modifikováno oproti předchozímu. Rozdíl je v situaci, kdy se dostanou daleko od sebe, ale jeden bude výrazně blíže k cíli. Pak tento agent nepoběží přímo k prvnímu agentovi, ale jen se k němu přiblíží a případně se zastaví a počká na něj. Předchozí řešení je vhodnější např. pro dvojici agentů, kteří se nemohou dočkat, až se uvidí (např. milenci, kteří se dlouho neviděli, nebo malé děti). Oproti tomu toto řešení je vhodné pro obyčejnější situace setkání dvou agentů, případně speciálně pro starší agenty.



Obrázek 6.38: Trajektorie 6.3.2 C.

6.3.3 Formace

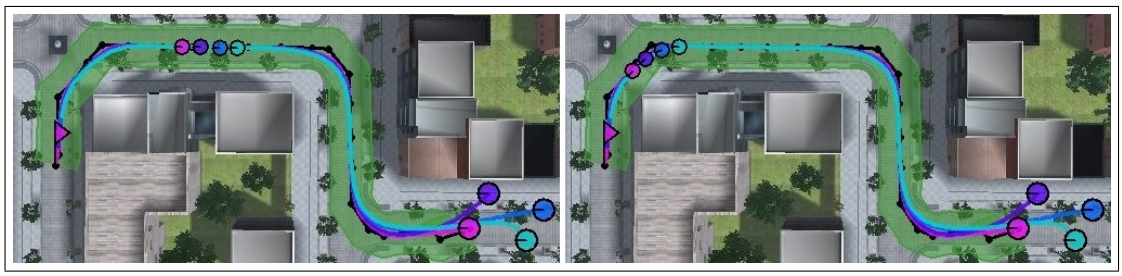
Zadání

Malá skupina agentů prochází určitou dráhu v zadané formaci. Jednotlivá řešení obsahují ukázkové formace. Hodnocena je plynulost, schopnost udržet formaci a schopnost dojít k cíli.

Řešení A

Formace „had“: každý agent má steering Leader Following základní typ v rozšířeném chování tak, že jeho vůdcem je vždy jeden před ním v řadě, při-

čemž první agent nemá steering LEADER FOLLOWING, ale libovolnou kombinaci steeringů, díky které projde zadanou dráhu k cíli.



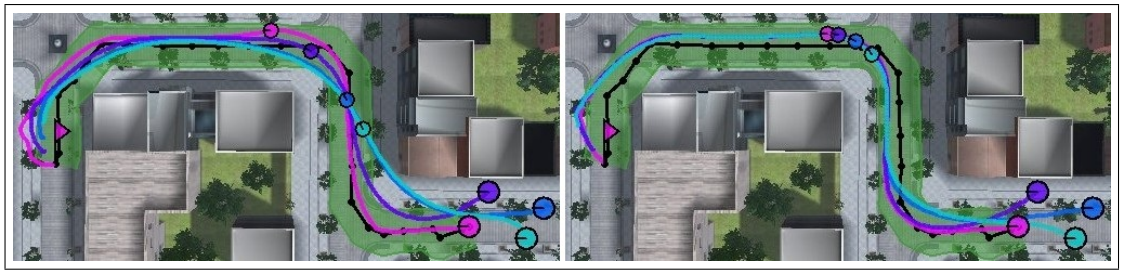
Obrázek 6.39: Průběh řešení A.

Zhodnocení A

Na volném prostranství nemají agenti problém s plynulostí, udržením formace, ani schopností dojít k cíli. Pokud nejsou na začátku agenti příliš daleko od sebe a první agent nejde příliš rychle, jdou agenti téměř v identické dráze jako první vůdce. Díky tomu (není-li na cestě dynamických překážek) stačí, aby se první agent uměl vyhýbat překážkám a ostatní ho jen následují.

Poznámka

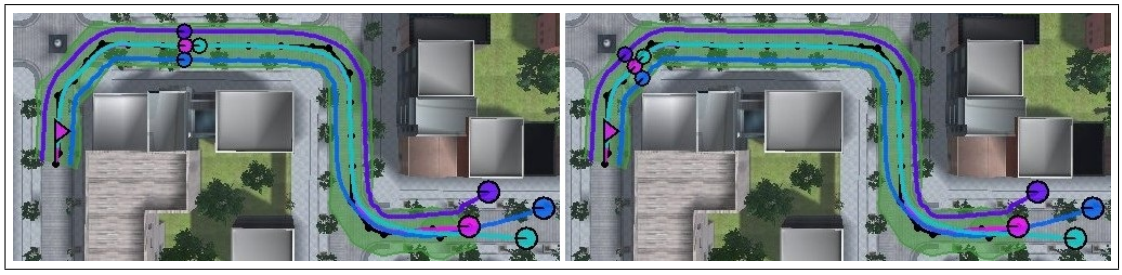
Na Obrázku 6.40 má první agent trojnásobnou rychlost, přičemž ostatní mají rychlost základní, tedy vůdce od začátku běží, ale ostatní se rozeběhnou až tehdy, když jsou od něj daleko. Na Obrázku 6.40 vpravo mají všichni trojnásobnou rychlost, tedy všichni od počátku běží. Vidíme, že tentokrát již nejdou přesně v dráze prvního agenta, avšak stále se jim daří formaci udržovat poměrně dobře.



Obrázek 6.40: Ukázka řešení A za použití vyšší rychlosti pro prvního agenta (vlevo), či pro všechny agenty (vpravo).

Řešení B

Formace „kříž“: Agent uprostřed je vůdcem všem ostatních a má opět libovolnou kombinaci steeringů, díky které projde zadanou dráhu k cíli. Ostatní agenti ho následují s formačním typem v rozšířeném chování tak, že mají po řadě úhly -90°, 90°, 180°.



Obrázek 6.41: Průběh řešení B.

Zhodnocení B

Nastavením parametrů `Leader Force` a `Force Distance` lze ovlivňovat, zda budou agenti spíše schopní udržet formaci, či zda spíše půjdou plynule. Pro schopnost udržování formace je vhodné použít hodnoty vyšší `Leader Force` (např. 240 N_{UT}) a nižší `Force Distance` (např. 50 cm_{UT}). Pro plynulejší chování jsou vhodné nižší hodnoty těchto parametrů (např. 200 N_{UT} a 150 cm_{UT}).

6.3.4 Tajné sledování

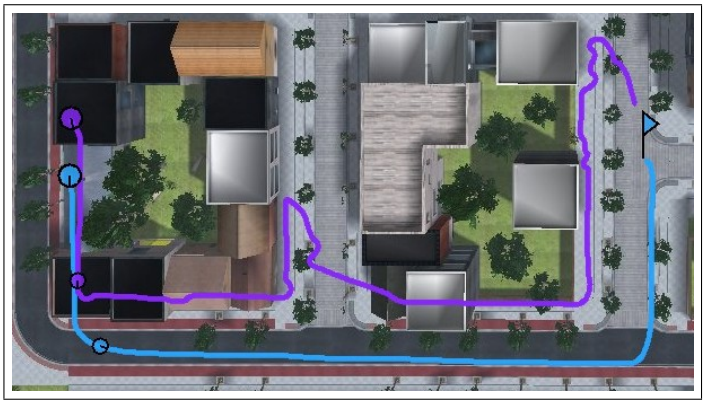
Zadání

Poslední zadání je poměrně „kuriózní“. Že lze řešit (do určité míry) pomocí navržených steeringů, bylo zjištěno spíše náhodou, jedná se však o další pěknou ukázku „sociálního“ využití steeringů.

Jeden agent prochází městem a druhý ho tajně sleduje.

Řešení

První agent: libovolná kombinace steeringů, která ho vede k nějakému cíli, druhý agent: Leader Following základní chování (tedy základní typ) s velkou vzdáleností od vůdce, malou silou k vůdci a vysokým parametrem `Force Distance` (taktéž způsobuje malou sílu k vůdci) + Wall Following v plném rozšířeném chování.



Obrázek 6.42: Ukázka tajného sledování vůdce.

Zhodnocení

Tato kombinace dopadne vhodným způsobem pouze v některých případech. Pak ale může překvapivě jednoduchým způsobem splňovat zadání. Druhý agent následuje prvního, avšak ve velké vzdálenosti a navíc chodí u zdi, kde je méně nápadný. Pokud zeď zahýbá, může zahnout s ní a za chvíli se opět stočit zpátky za prvním agentem, to však může vypadat, že dobře skrývá svůj úmysl následovat prvního agenta. Sporná je cílová fáze, kdy první agent dojde k cíli. Zde by rozhodovací vrstva pravděpodobně změnila chování podle toho, zda by měl druhý agent postávat opodál, prozradit se a dojít k prvnímu (třeba ho zatknout), atd. V ukázkovém případu druhý agent přejde kolem prvního (jakoby nic), ale za chvíli se zezadu vrátí a zastaví se u něj. To je jedno z možných řešení zadání. Zajímavé je především to, že celé toto chování bylo vytvořeno pouhým vhodným nastavením implementovaných steeringů.

Pro dotvoření dojmu této scény by bylo dobré použít vhodné animace pro oba jedince. To však není úkolem stávající navigační vrstvy.

6.3.5 Shrnutí

Nejdůležitější poznatky z evaluace implementovantované navigační vrstvy lze rozdělit podle účelu použití. Obecně platí, že hodnoty sil jsou vhodné okolo 200 N_{UT}, přičemž pro Obstacle Avoidance jsou lepší lehce vyšší hodnoty (např. 240 N_{UT}) a pro Target Approaching naopak hodnoty nižší (např. 100 N_{UT}). Pro rozumné výsledky stačí nastavit všem steeringům váhu 1.

1. Průchod prostředím bez kolizí.

   (a) Má-li rozhodovací vrstva možnost předpočítat cestu prostředím, která nevede přes žádné překážky, je ideální použít steering Path Following v plném rozšířeném chování. Jeho parametry budou: `Path`: naplánovaná cesta, `Distance`: záleží na prostředí, pro naše město se širokými ulicemi např. 400 cm_{UT}, `Projection` např. 15, `Regulation` např. 50 N_{UT}. Je-li riziko, že se v oblasti koridoru budou vyskytovat překážky, je vhodné použít navíc steering Obstacle Avoidance v plném rozšířeném chování. Je-li navíc riziko kolizí s ostatními agenty, je vhodné použít i steering People Avoidance, opět typicky v plném rozšířeném chování.

   (b) Pokud není možnost podobnou cestu předpočítat, nebo je nebezpečí, že by cesta vedla zbytečnou oklikou (například kvůli tomu, že nejsou kvalitně zpracovaná data o terénu prostředí), je nutno použít steering Target Approaching s přitažlivou silou k zadané lokaci a buď steering Obstacle Avoidance, či steering Wall Following. Obstacle Avoidance je pro jednodušší prostředí dostatečný. Pokud se mezi agentem a cílem vyskytne větší překážka či výklenek, agentovi se nemusí podařit dojít do cíle. Oproti tomu steering Wall Following má dvě výhody. Za prvé v mnoha případech dojde do cíle a obejde i větší či složitější překážky. Za druhé, dostane-li se agent po cestě k cíli do blízkosti zdi, bude pokračovat k cíli po chodníku vedoucí podél této zdi, což působí pro lidské agenty přirozeně. Ve chvíli, kdy zeď povede

směrem od cíle, je určité riziko, že bude agent pokračovat podél zdi místo k cíli. Pro snížení tohoto rizika je vhodné nastavit nižší hodnoty parametrů `Attractive Weight` a `Concave Weight`, např. 0.8. Jednou z nevýhod je, že se steeringem Wall Following se cesta k cíli může prodloužit.

2. Předcházení kolizím s ostatními agenty.

   (a) Nechceme-li vytvořit určitý dojem speciálního charakteru agenta, je vhodné použít steering People Avoidance v plném rozšířeném chování. Vhodné hodnoty číselných parametrů jsou: `Distance`: 300 cm_{UT} (víme-li, že se jedná o prostředí s vyšší hustotou agentů na jednom místě, například nějaké náměstí, můžeme hodnotu snížit např. na 200 N_{UT}), `Projection`: 16. Možnosti, jak vytvořit speciální dojem charakteru agenta, jsou shrnuty dále.

   (b) Chceme-li, aby více agentů tvořilo kompaktní skupinu, je lepší pro ně použít pouze základní chování steeringu People Avoidance.

3. Jiné sociální steeringy

   (a) Nastavení parametrů steeringu People Avoidance může podpořit utváření charakteru postavy. Zde je několik příkladů:[2]

      i. Agresivní, asertivní člověk: pouze základní chování. Nebude nikomu uhýbat, pouštět ostatní před sebe, ani se snažit ostatní neomezovat. Pokud budou mít ostatní agenti nastavené plné rozšířené chování, budou se vyhýbat oni jemu.

      ii. Člověk, co se straní ostatních: vyšší hodnoty `Distance` (např. 600 cm_{UT}) a `Repulsive Force` (např. 250 N_{UT}).

      iii. Člověk, který je často zamyšlený a nevnímá okolí: nižší hodnoty `Distance` (např. 150 cm_{UT}), `Projection` (např. 3) zajistí, že si srážky všimne až na poslední chvíli, případně vyšší hodnota `Repulsive Force` (např. 250 N_{UT}), aby byl schopný se srážce na poslední chvíli vyhnout.

      iv. Starší či nemocný člověk: plné rozšířené chování, avšak bez parametru `Acceleration`.

      v. Uspěchaný člověk: patrně bez parametru `Deceleration`. Budou-li mít ostatní agenti plné rozšířené chování, budou to oni, kdo zpomalí, a on proběhne vždy před nimi (aby běžel, musí mít vyšší hodnotu `velocity multiplier` či velkou sílu za jiný steering).

      vi. Vtíravý neoblíbený člověk: nízká hodnota `Distance` (např. 100 cm_{UT}). Budou-li mít ostatní normální hodnoty parametru `Distance`, budou se od něho odvracet.

   (b) Pro dvojici agentů, kteří jdou ke společnému cíli si můžeme zvolit mezi steeringy Leader Following a Walk Along. První možnost zdůrazní vůdčí roli jednoho z nich, druhá možnost vytvoří dojem rovnocennějšího vztahu.

[2] Tato část je poměrně subjektivní a dává uživateli, resp. programátorovi prostor k vlastní fantazii.

7. Závěr

Tato práce se zabývala implementací steering technik inspirovaných Craigem W. Reynoldsem v 3D prostředí hry Unreal Tournament 2004. Hlavním výsledkem je navigační vrstva implementovaná jako knihovna steeringů rozšiřující platformu Pogamut. Speciální důraz byl kladen na uvěřitelnost chování lidských agentů. Steeringy obsahují rozšíření, díky kterým se agenti pohybují plynuleji, přirozeněji se vyhýbají překážkám, pro předcházení kolizím s ostatními agenty zpomalují, zrychlují, obchází se, apod. Některé steeringy dokonce umožňují spoluvytvářet charaktery jednotlivých agentů a vztahů mezi agenty, což podporuje dojem, že se jedná o skutečné virtuální lidi.

Mezi další výsledky patří evaluace provedená pomocí speciálního nástroje grafické aplikace pro vizualizace drah pohybů. Práce obsahuje rozbor některých typických situací a nastavení parametrů navigační vrstvy pro tyto situace.

Podstatným výsledkem je i teoretický návrh celé navigační vrstvy. Existuje mnoho implementací a reimplementací steeringů pro boidy a jim podobné virtuální agenty, viz [7, 37, 19]. Dále existují práce, které navrhují a důkladně zpracovávají jeden ze steeringů pro lidské virtuální agenty, nejčastěji steering pro předcházení kolizím s ostatními agenty, v této práci označovaný jako People Avoidance. Mezi takové práce patří například [46, 36, 45].

Nejsou mi však známy práce, které by navrhovaly pro lidské virtuální agenty ucelenou navigační vrstvu se steeringy, které řeší nejen „kolizní“ úkoly (předcházení kolizím), ale i „sociální“ úkoly (interakce mezi agenty), nabízely vhodné řešení kombinování těchto steeringů a taktéž otestovanou implementaci. Tato vlastnost tedy představuje další přínos a výsledek této práce.

Posledním z cílů této práce bylo zhodnotit, zda jsou steeringy vhodným prostředkem pro řízení pohybů lidských virtuálních agentů. Jak bylo řečeno, práce obsahuje mnoho rozšíření, která slouží právě k tomu, aby výsledné chování vypadalo lidsky. Pro mnoho situací byly tyto snahy úspěšné. Je však nutno podotknout, že sama navigační vrstva nestačí pro vytvoření dojmu lidského agenta. Není to ani jejím cílem. Principy jednotlivých steeringů jsou záměrně poměrně jednoduché, aby byly dobře předvídatelné a aby nedocházelo k nevysvětlitelným nepřirozeným situacím. Výsledné chování je pak tudíž také jednoduché. S tím ale třívrstvá hierarchie vytváření pohybu počítá a řešení těchto problémů je určeno vrstvě rozhodovací. Navigační vrstva jí má nabídnout pouze sadu jednoduchých, ale schopných nástrojů pro vykonávání základních úkolů. Tento účel navržená navigační vrstva splňuje.

Existují názory, že steeringy nejsou z různých důvodů pro navigaci lidských virtuálních agentů vhodné. Jeden z těchto názorů zastává Alex J. Champandard [4, 5]. Podstatné však je, že tím myslí čistě reaktivní steeringy – a to především steering Obstacle Avoidance. Je pravda (a evaluační část této práce to potvrzuje), že tento steering nestačí pro ideální průchod prostředím k určitému cíli. Agent se totiž může o nějakou překážku zaseknout (např. ve výklenku či slepé uličce). To samozřejmě nevypadá přirozeně, neboť reálného člověka by patrně napadlo danou překážku obejít větším obloukem. Tento typ problémů řeší lépe druhý přístup k navigaci – přístup, který je založen na plánování cesty z předzpracovaných dat o prostředí. Zde je podstatné si uvědomit dvě důležité

skutečnosti:

1. Navigační vrstva navržená v této práci obsahuje steering PATH FOLLOWING, který jí poskytuje výhody plánovacího přístupu k navigaci. Navíc je použití tohoto steeringu velmi flexibilní. Buď může rozhodovací vrstva pouze spočítat cestu k cíli a dále se o průběh pohybu agenta nestarat. Pak je veškeré další chování reaktivní a časové nároky na výpočet jsou velmi malé. Nebo může rozhodovací vrstva sledovat průběh pohybu a v případě potřeby trasu přeplánovat a novou cestu předat steeringu PATH FOLLOWING. Díky rozšířenému chování steeringu PATH FOLLOWING půjde agent plynule po cestě, bez ostrých změn směru pohybu, aniž by bylo cestu třeba nějak dopředu upravovat (např. interpolovat).

2. Navigační vrstva v této práci plní i docela jiné úkoly než jen průchod prostředím bez kolizí. Některé z těchto úkolů jsou navíc sociálního charakteru, někdy dokonce velmi specifického pro lidské chování (např. společná cesta dvou kamarádů), což podporuje výsledek, že lze steeringy použít k řízení pohybů lidských virtuálních agentů.

Tyto dvě skutečnosti do určité míry vyvracejí názor, že steeringy nejsou obecně vhodné pro navigaci lidských virtuálních agentů.

Evaluace navigační vrstvy ukázala (či potvrdila) nějaké problémy a vedla k návrhům možných rozšíření. Nejčastější nedostatek výsledného chování bývá nedostatečně plynulý pohyb. To je většinou způsobeno jedním ze tří důvodů:

1. Prvním důvodem jsou situace, kdy musí navigační vrstva skloubit dva protichůdné požadavky a střídá se, zda převáží v daném tiku jeden, či druhý požadavek. To může způsobit časté změny natočení agenta. Evaluace ukázala, že k tomuto jevu nedochází příliš často, avšak v některých typech situací častěji: např. jde-li agent podél budovy, přičemž jedna síla ho táhne za budovu (přitažlivá síla k cíli) a druhá ho táhne od budovy (odpudivá síla od překážky). Tento typ problémů by bylo možné řešit na úrovni rozhodovací vrstvy. Je zde však i prostor pro rozšíření kombinační části navigační vrstvy, která by sama podobné situace detekovala a řešila. Má totiž přístup k požadavkům jednotlivých steeringů a navíc si může pamatovat několik posledních výsledných vektorů rychlosti a detekovat případy, kdy se často mění.

2. Frekvence tiků simulace je poměrně nízká. Kvůli tomu musí steeringy reagovat dostatečně výrazně, neboť by to v příštím tiku už nemusely stihnout (pokud bude příští tik za poměrně dlouho). Tento důvod je úzce spojený s použitým virtuálním prostředím a není to tedy problém navigační vrstvy. Navržená navigační vrstva se ovšem snaží negativní vliv tohoto problému minimalizovat.

3. Třetí důvod je dán také virtuálním prostředím. Jsou jím často zmiňované paprsky – prostředek pro detekování překážek v okolí agenta. Vzhledem k tomu, že nepokryjí celou oblast kolem agenta a nemohou být tak dlouhé, jako je dosah zraku reálných lidí, mohou být překážky detekovány pozdě či vůbec. Proto když se je podaří detekovat, musí na ně být reakce výrazné, jinak by se nemuselo podařit zabránit srážce.

Možná rozšíření této práce mohou být rozdělena do čtyř skupin:

1. Rozšíření a vylepšení steeringů.

   (a) Některá možná vylepšení jednotlivých steeringů jsou zmíněna vždy v závěru popisu daného steeringu. Především by bylo možné přidat další parametry. Bohužel systémy s příliš mnoha parametry bývají složité pro používání i testování. Proto byla snaha počet parametrů udržet v rozumné míře.

   (b) Navigační vrstva byla navržena tak, aby do ní šly snadno přidávat další steeringy. Velmi zajímavé by mohlo být rozšířit skupinu tzv. „sociálních“ steeringů, tedy těch, které se zabývají interakcemi mezi agenty. Mohou se starat například o vhodné natáčení se na sebe během rozhovorů, přitažlivé síly mezi agenty, různé vzdálenosti pro konkrétní dvojice agentů apod. Inspiraci k podobným rozšířením lze najít v pracích autorů Claudio Pedica a Hannes Vilhjalmsson [20, 21, 22].

   (c) Steering Target Approaching dovoluje přesné stanovení síly v závislosti na vzdálenosti od cíle. Možnost takto určit průběh síly by šlo použít i u ostatních steeringů, např. People Avoidance. Například k dalekým agentům by mohla působit síla přitažlivá, k příliš blízkým odpudivá.

2. Objektivnější evaluace steeringů. V rámci stávající evaluace byly použity scénáře navržené v práci SteerBench [38]. SteerBench nabízí i možnosti objektivnějšího hodnocení steeringů: z naměřených trajektorií a zaznamenaných pozic překážek v prostředí počítá na základě různých metrik úspěšnost zvládnutí dané situace z různých hledisek. Tyto metriky zkoumají plynulost pohybu, počet kolizí apod.

3. Vylepšení kombinování steeringů. Především lepší možnosti pro ovlivnění výsledné rychlosti agenta. V rámci kombinování by bylo taktéž možno se zaměřit na ještě lepší plynulost pohybu.

4. Vylepšení mimo navigační vrstvu. Výslednému dojmu by hodně pomohly další animace: nejen chůze a běh, ale např. i ukročení a couvání. Pak by bylo potřeba vymyslet mechanismus, kdy se mají použít: například zda má agent couvat, nebo se otočit.

Seznam použité literatury

[1] AMOR, Heni Ben; MURRAY, Jan; OBST, Oliver. *AI Game Programming Wisdom 3*. First Edition. United States of America : Charles River Media, 2006. Fast, Neat, and Under Control: Arbitrating Between Steering Behaviors, s. 221-232. ISBN 1-58450-457-9.

[2] BAYRAK, Ali; POLAT, Faruk. Formation preserving path finding in 3-D terrains. Springer Netherlands. *Applied Intelligence*. 2010, s. 1-21. ISSN 0924-669X.

[3] BOULIC, Ronan. Relaxed Steering towards Oriented Region Goals. In *Motion in Games : First International Workshop, MIG 2008 Utrecht, The Netherlands, June 14-17, 2008 Revised Papers*. Germany : Springer-Verlag, 2008. s. 176-187. ISSN 0302-9743.

[4] CHAMPANDARD, Alex J. *AI Game Development : Synthetic Creatures with Learning and Reactive Behaviors*. First printing. United States of America : New Riders Publishing, 2003. 721 s. ISBN 1-5927-3004-3.

[5] CHAMPANDARD, Alex J. *AI Game Programming Wisdom 2*. First Edition. United States of America : Charles River Media, 2004. An Overview of Navigation System, s. 131-139. ISBN 1-58450-289-4.

[6] CHOON, Ho, et al. *Motion in Games : Lecture Notes in Computer Science*. Berlin Heidelberg : Springer-Verlag, 2010. Autonomous Multi-agents in Flexible Flock Formation, s. 375-385.

[7] DAVISON, Andrew. *Killer Game Programming in Java*. [s.l.] : O'Reilly Media, 2005. Flocking Boids, s. 592-613. ISBN 978-0-596-00730-0.

[8] GERAERTS, Roland, et al. Using the Corridor Map Method for Path Planning for a Large Number of Characters. In *Motion in Games : First International Workshop, MIG 2008 Utrecht, The Netherlands, June 14-17, 2008 Revised Papers*. Germany : Springer-Verlag, 2008. s. 11-22. ISSN 0302-9743.

[9] GUY, Stephen J.; LIN, Ming C.; MANOCHA, Dinesh. Modeling collision avoidance behavior for virtual humans. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 2 - Volume 2*. Richland : International Foundation for Autonomous Agents and Multiagent Systems, 2010. s. 575-582. ISBN 0-98265-712-9.

[10] HARRISON, Joseph; VO, Christopher; LIEN, Jyh-Ming. *Motion in Games : Lecture Notes in Computer Science*. Berlin Heidelberg : Springer-Verlag, 2010. Scalable and Robust Shepherding via Deformable Shapes, s. 218-229.

[11] JOHNSON, Geraint. *AI Game Programming Wisdom 2*. First Edition. United States of America : Charles River Media, 2004. Avoiding Dynamic Obstacles and Hazards, s. 161-170. ISBN 1-58450-289-4.

[12] JOHNSON, Geraint. *AI Game Programming Wisdom 3*. First Edition. United States of America : Charles River Media, 2006. Smoothing a Navigation Mesh Path, s. 129-139. ISBN 1-58450-457-9.

[13] KARAMOUZAS, Ioannis, et al. A Predictive Collision Avoidance Model for Pedestrian Simulation. In *Motion in Games : Second International Workshop, MIG 2009 Zeist, The Netherlands, November 21-24, 2009 Proceedings*. Germany : Springer-Verlag, 2009. s. 41-52. ISSN 1867-8211.

[14] KARAMOUZAS, Ioannis; OVERMARS, Mark. A Velocity-Based Approach for Simulating Human Collision Avoidance. In *Intelligent Virtual Agents : 10th International Conference, IVA 2010 Philadelphia, PA, USA, September 20-22, 2010 Proceedings*. Germany : Springer-Verlag, 2010. s. 180-186. ISSN 0302-9743.

[15] KRUSZEWSKI, Paul. *AI Game Programming Wisdom 3*. First Edition. United States of America : Charles River Media, 2006. Real-Time Crowd Simulation Using AI.implant, s. 233-248. ISBN 1-58450-457-9.

[16] MATTHEWS, James. *AI Game Programming Wisdom*. I. Title. United States of America : Charles River Media, 2002. Basic A* Pathfinding Made Simple, s. 105-113. ISBN 1-58450-077-8.

[17] MORI, Masahiro. The Uncanny Valley. *Energy* 7(4) pp. 3335 (1970), translation by Karl F. MacDorman and Takashi Minato.

[18] ONDŘEJ, Jan, et al. A Synthetic-Vision Based Steering Approach for Crowd Simulation. In *ACM SIGGRAPH 2010 papers*. New York, USA : ACM New York, 2010. ISBN 978-1-4503-0210-4.

[19] PARKER, Conrad. *Boids* [online]. 1995, 2010 [cit. 2011-05-23]. Boids. Dostupné z WWW: <http://www.vergenet.net/ conrad/boids/>.

[20] PEDICA, Claudio; VILHJÁLMSSON, Hannes Högni. Spontaneous Avatar Behavior for Human Territoriality. In *Intelligent Virtual Agents : Proceedings of the 9th International Conference on Intelligent Virtual Agents*. Berlin Heidelberg : Springer-Verlag, 2009. s. 344 - 357. ISBN 978-3-642-04379-6.

[21] PEDICA, Claudio; VILHJÁLMSSON, Hannes Högni. Social Perception and Steering for Online Avatars. In *Intelligent Virtual Agents : Proceedings of the 8th International Conference on Intelligent Virtual Agents*. Berlin Heidelberg : Springer-Verlag, 2008. s. 104 - 116. ISBN 978-3-540-85482-1.

[22] PEDICA, Claudio; VILHJÁLMSSON, Hannes Högni; LÁRUSDÓTTIR, Marta. Avatars in Conversation: The Importance of Simulating Territorial Behavior. In . *Intelligent Virtual Agents : 10th International Conference, IVA 2010 Philadelphia, PA, USA, September 20-22, 2010 Proceedings*. Germany : Springer-Verlag, 2010. s. 336-342. ISSN 0302-9743.

[23] GEMROT, J., et al.: Pogamut 3 Can Assist Developers in Building AI (Not Only) for Their Videogame Agents. In: *Agents for Games and Simulations*, LNCS 5920, Springer, 2009: 1-15.

[24] POPELOVÁ, Markéta; BÍDA, Michal. Steering techinky pro virtuální agenty. In KELEMEN, Jozef; KVASNIČKA, Vladimír; POSPÍCHAL, Jiří. *Kognice a umělý život XI.* Opava : Slezská univerzita v Opavě, 2011. s. 207-212. ISBN 978-80-7248-644-1.

[25] PORCINO, Nick. *AI Game Programming Wisdom 2.* First Edition. United States of America : Charles River Media, 2004. An Architecture for A-Life, s. 339-349. ISBN 1-58450-289-4.

[26] PORCINO, Nick. *AI Game Programming Wisdom 3.* First Edition. United States of America : Charles River Media, 2006. Insect AI 2: Implementation Strategies, s. 189-204. ISBN 1-58450-457-9.

[27] RABIN, Steve. *Game Programming Gems.* First Edition. United States of America : Charles River Media, 2000. A* Speed Optimizations.

[28] RABIN, Steve. *Game Programming Gems.* First Edition. United States of America : Charles River Media, 2000. A* Aesthetic Optimizations.

[29] REESE, Bjorn; STOUT, Bryan. Finding Pathfinder. In *SYMPOSIUM ON ARTIFICIAL INTELLIGENCE AND COMPUTER GAMES.* [s.l.] : [s.n.], 1999.

[30] REYNOLDS, Craig W. Flocks, Herds, and Schools: A Distributed Behavioral Model. In *Proceedings of Computer Graphics.* Anaheim, California : ACM SIGGRAPH, 1987. s. 25-34. Dostupné z WWW: <http://www.red3d.com/cwr/papers/1987/SIGGRAPH87.pdf>.

[31] REYNOLDS, Craig W. An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion. In *From Animals to Animats 2 : Proceedings of the Second International Conference on Simulation of Adaptive Behavior.* Cambridge : MIT Press, 1993. s. 384-392. Dostupné z WWW: <http://www.red3d.com/cwr/papers/1993/sab92.pdf>. ISBN 0-262-63149-0.

[32] REYNOLDS, Craig W. An Evolved, Vision-Based Model of Obstacle Avoidance Behavior. In LANGTON, C. *Artificial Life III : Santa Fe Institute Studies in the Sciences of Complexity Proceedings Volume XVI.* California : [s.n.], 1993. s. 327-346. Dostupné z WWW: <http://www.red3d.com/cwr/papers/1993/alife3.pdf>. ISBN 0-201-62494-X.

[33] REYNOLDS, Craig W. Evolution of Corridor Following Behavior in a Noisy World. In *From Animals to Animats 3 : Proceedings of the Third International Conference on Simulation of Adaptive Behavior.* Cambridge : MIT Press, 1994. s. 402-410. Dostupné z WWW: <MIT Press>. ISBN 0-262-53122-4.

[34] REYNOLDS, Craig W. Steering Behaviors For Autonomous Characters. In *Proceedings of Game Developers Conference.* San Francisco, California : Miller Freeman Game Group, 1999. s. 763-782. Dostupné z WWW: <http://www.red3d.com/cwr/papers/1999/gdc99steer.pdf>.

[35] REYNOLDS, Craig W. Steering Behaviors For Autonomous Characters [online]. September 5, 1997 , June 6, 2004 [cit. 2011-05-19]. Steering Behaviors For Autonomous Characters . Dostupné z WWW: <http://www.red3d.com/cwr/steer>.

[36] ROSSMANN, Jürgen; HEMPE, Nico; TIETJEN, Philipp. In *Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics.* USA : IEEE Press Piscataway, 2009. ISBN 978-1-4244-2793-2.

[37] SCHNELLHAMMER, Christian; FEILKAS, Thomas. *The Steering Behaviors Webpage* [online]. 2001, 2010 [cit. 2011-05-23]. The Steering Behaviors Webpage. Dostupné z WWW: <http://www.steeringbehaviors.de/>.

[38] SINGH, Shawn, et al. Watch Out! A Framework for Evaluating Steering Behaviors. In *Motion in Games : First International Workshop, MIG 2008 Utrecht, The Netherlands, June 14-17, 2008 Revised Papers.* Germany : Springer-Verlag, 2008. s. 200-209. ISSN 0302-9743.

[39] *The Internet Movie Database (IMDb)* [online]. 1990, 2011 [cit. 2011-05-23]. Batman returns (1992). Dostupné z WWW: <http://www.imdb.com/title/tt0103776/>.

[40] TOZOUR, Paul. *AI Game Programming Wisdom 2.* First Edition. United States of America : Charles River Media, 2004. Search Space Representations, s. 85-102. ISBN 1-58450-289-4.

[41] *UDN - Two : UnrealEngine2Runtime22262002* [online]. 2001, 2010 [cit. 2011-05-23]. UnrealEngine2 Runtime 2226.20.02 . Dostupné z WWW: <http://apacudn.epicgames.com/Two/UnrealEngine2Runtime22262002.html>.

[42] VAN BASTEN, Ben J.H.; JANSEN, Sander E.M.; KARAMAZOUS, Ioannis. Exploiting Motion Capture to Enhance Avoidance Behaviour in Games. In *Motion in Games : Second International Workshop, MIG 2009 Zeist, The Netherlands, November 21-24, 2009 Proceedings.* Germany : Springer-Verlag, 2009. s. 29-40. ISSN 1867-8211.

[43] VAN DEN BERG, René; REJEN, Juan Manuel; BIDARRA, Rafael. Collision Avoidance between Avatars of Real and Virtual Individuals. In *Motion in Games : Second International Workshop, MIG 2009 Zeist, The Netherlands, November 21-24, 2009 Proceedings.* Germany : Springer-Verlag, 2009. s. 1-12. ISSN 1867-8211.

[44] *Wall Following : RoboWiki* [online]. 14 July 2010 [cit. 2011-05-23]. Wall Following. Dostupné z WWW: <http://psurobotics.org/wiki/index.php?title=Wall_Following>.

[45] YEH, Hengchin, et al. Composite agents. In *Proceedings of the 2008 IEEE international conference on Systems, Man and Cybernetics.* Switzerland : Eurographics Association Aire-la-Ville, 2008. s. 39-47.

[46] ZAHARIA, Mihai Horia, et al. Agent-Based Simulation of Crowd Evacuation Behavior. In *Proceedings of the 11th WSEAS International Conference on Automatic Control, Modelling and Simulation.* [s.l.] : 2009. s. 529-533.