Steering techniky pro virtuální agenty

Markéta Popelová, Michal Bída

Karlova Univerzita, Matematicko-fyzikální fakulta
Malostranské nám. 2/25, Praha, Česká Republika
marketa.popelova@matfyz.cz, michal.bida@gmail.com

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. Některé z těchto steering technik jsme implementovali na platformě Pogamut v 3D prostředí UnrealEngine2Runtime. Vznikla tak knihovna steeringů, kterou mohou využívat i ostatní části platformy Pogamut. Kromě implementace vybraných steeringů navržených C. Reynoldsem, avšak vhodně upravených pro naše potřeby, jsme přidali některé nové. Zároveň jsme zkoumali možnosti kombinací steeringů jako prostředek řešení složitějších situací. Práce je využitelná pro obecnější simulaci postav ve virtuálním prostředí, počítačové hry a výuku informatiky na středních školách.

1 Úvod

Virtuálním agentem myslíme typ autonomního agenta, který existuje ve virtuálním světě, má vlastní tělo a může se pohybovat. Rozhoduje se reaktivně podle stavu okolního světa (nemá předem dané chování). Speciálně se zajímáme o lidské agenty, kterým se snažíme vytvářet lidské chování.

Chování agenta je velmi komplexní pojem. My jsme se zaměřili na jednu jeho složku – pohyb. Vzhledem k této složce můžeme chování virtuálního agenta rozdělit do tří vrstev [1]: *rozhodovací* (*action selection*), *navigační* (*steering*) a *lokomoční* (*locomotion*). Rozhodovací vrstva řeší obecnější důvody a cíle agentova pohybu. Navigační vrstva dostává jednodušší úkoly, jak se má agent pohybovat. Např. zda se má vyhýbat překážkám či ostatním agentům, následovat jiného agenta, jít k určitému cíli, apod. Samotný fyzický pohyb vykonává vrstva lokomoční.

Rozdělení do tří vrstev velmi hezky ukazuje ilustrační situace (vycházíme z [1]) 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.

Tento článek se zabývá především navigační vrstvou a jejím propojením s okolními vrstvami. *Steering* je algoritmus, který řeší navigaci virtuálního agenta vzhledem k jednomu cíli – např. vyhýbat se překážkám, nenarážet do lidí, směřovat ke krávě, atd. Chceme-li dosáhnout složitějšího chování, musíme zkombinovat více steeringů najednou.

Naše steering techniky vycházejí ze steeringů, které navrhl Craig Reynolds [1, 2]. Vytvářel steeringy původně pro tzv. *boidy* (bird-like-objects), stvoření, pomocí kterých simuloval hejna ptáků, rybích školek, apod. Později rozšířil svůj model na obecnější virtuální agenty (chodící či létající agenty a vozidla) a přidal několik dalších steeringů.

Tyto steeringy jsou založeny na vektorovém principu. Virtuální agent má dvě základní informace o svém stavu: *lokaci* (*location*), tedy souřadnice své pozice v prostoru, a *vektor rychlosti* (*velocity*). Kromě těchto základních údajů může mít agent různé doplňující informace, jako je vektor natočení, maximální rychlost (může být jiná pro chůzi a běh), atd.

Pro implementaci steeringu potřebujeme znát různé informace z okolního světa agenta – o překážkách, ostatních agentech (především lokace, případně vektory rychlosti) 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 některé lokace, např. ostatních agentů, a informace o neživých objektech musí získávat jinak – např. pomocí paprsků vycházejících z jeho těla.

Jak bylo řečeno, steeringy fungují na vektorovém principu. Pohyb agenta na úrovni navigační vrstvy vzniká úpravami vektoru rychlosti. 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 všechny blízké překážky působí na agenta odpudivou silou. Nebo chceme-li, aby agent mířil k nějaké lokaci, pak daná lokace na agenta působí přitažlivou silou. Výsledkem steeringu v daném okamžiku je složení působících sil. Výsledná síla určuje nový vektor rychlosti agenta.

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.

Cílem naší práce bylo vytvořit knihovnu steeringů jako rozšíření platformy Pogamut [3], která využívá 3D prostředí UnrealEngine2Runtime (UE2), což je volně šiřitelná verze enginu hry Unreal Tournament 2004. Knihovna steeringů měla obsahovat některé původní Reynoldsovy steeringy a několik nových. Všechny měly být vhodně upraveny: aby vyhovovaly prostředí UE2 a aby imitovaly lidské chování. Zároveň bylo cílem prozkoumat možnosti kombinování steeringů. Navíc vznikla grafická aplikace, která dovoluje spouštět jednotlivé virtuální agenty v 3D prostředí UE2 a přidělovat jim steeringy včetně parametrů.

Následující části článku popisují implementaci knihovny steeringů, jednotlivé steeringy a možnosti jejich kombinací. Poté následuje přehled výsledků celé práce a závěrečné shrnutí.

2 Implementace steeringů

2.1 Obecné vlastnosti implemetace

Knihovna steeringů je napsána v Javě a rozšiřuje platformu Pogamut. Ostatní části této platformy mohou knihovnu využívat pro vlastní účely (čímž plní funkce rozhodovací vrstvy). Pro vykonávání úkolů lokomoční vrstvy používáme možnosti platformy Pogamut, která komunikuje s UE2 pomocí příkazů v jazyce Unreal Script. Podle nich se agenti (tzv. *boti)* v 3D světě pohybují (buď chůzí, nebo během) a přehrávají pohybové animace. Naše mapa světa má podobu jednoduchého města.

V naší implementaci nemá agent přístup k objektům z celé mapy. Každý agent zná svou aktuální lokaci a vektor rychlosti. Zároveň je pevně daná maximální rychlost chůze a maximální rychlost běhu. Informace o druhém agentovi může agent získat jen tehdy, je-li druhý agent v dohledu. Proto je nutno ošetřit situace, kdy druhý agent není vidět či je příliš daleko. Informace o neživých objektech v blízkém okolí získává agent pomocí paprsků. Agent se může zeptat, zda paprsek koliduje, tedy zda narazil do jiného objektu, a jak hluboko se zanořil. Zároveň si může nechat spočítat normálový vektor k objektu v místě srážky s paprskem.

Jsou zde tedy řešeny zvlášť kolize (a jejich předcházení) s živými a neživými objekty. Překážkami v tomto článku myslíme objekty neživé.

Každý agent, který chce používat steeringy z naší knihovny, si vytvoří svého „steering managera“. Tomu řekne, které steeringy chce používat a jaké mají být hodnoty jejich parametrů. Pak v každém tiku zavolá agent steering managera, ať zařídí pohyb v daném okamžiku. Manager pak postupně volá jednotlivé steeringy. Každý z nich vrátí jeden vektor. Tyto vektory (a aktuální vektor rychlosti) vhodným způsobem zkombinuje (viz 2.9) a výsledek použije jako nový vektor rychlosti. Ten předá tomu, kdo představuje lokomoční vrstvu.

V následujících částech popíšeme jednotlivé steeringy. Ty, které jsou implementované podle pravidel C. Reynoldse [1], popíšeme pouze zhruba, či u nich zdůrazníme a vysvětlíme vlastnosti nově přidané. Jsou to steeringy Obstacle-avoidance, People-avoidance (ten je sice nový, ale obdobný jako Obstacle-avoidance) a Target-approaching. Steeringy Path-following, Wall-following a Leader-following (typ 1) vycházejí ze steeringů C. Reynoldse, avšak umí řešit více situací. Steering Leader-following (typ 2) a Walk-along jsou nové.

2.2 Obstacle-avoidance

Tento steering zajišťuje, aby agent nenarážel do překážek. K detekování překážek využívá paprsky. Jestliže se dostane paprsek do konfliktu s překážkou, začne překážka působit na agenta odpudivou silou, tedy steering vrací vektor směřující od překážky k agentovi (čím je agent překážce blíže, tím je síla větší). Pokud žádný paprsek nekoliduje, vrací steering nulový vektor.

2.3 People-avoidance

Tento steering je velmi podobný tomu předchozímu, jen se stará o „živé překážky“, tedy ostatní agenty. Hlavními parametry jsou *vzdálenost od ostatních agentů* a *velikost odpudivé síly*. Agent (označme ho agent A) nepotřebuje paprsky, neboť má přímý přístup k seznamu viditelných agentů. Tento seznam pokaždé projde a pokud je nějaký

agent B příliš blízko, na agenta A začne působit odpudivá síla od agenta B (čím jsou si blíže, tím je síla větší).

Kromě tohoto základního chování můžeme ošetřit ještě jeden typ situace – zapneme-li přepínač *obcházení*. Nechť agenti A i B jdou stejným směrem za sebou tak, že B je před agentem A. Nechť má agent A větší rychlost, viz Fig. 1. V takovém případě by bylo přirozené, aby agent A agenta B „předešel“, tedy ho nějak obešel a pokračoval v původním směru. Nicméně v základním steeringu se stane jen to, že agent B bude agenta A brzdit, ale neexistuje žádná síla, která by agenta A na chvíli vychýlila do jiného směru a dovolila mu agenta B předejít.

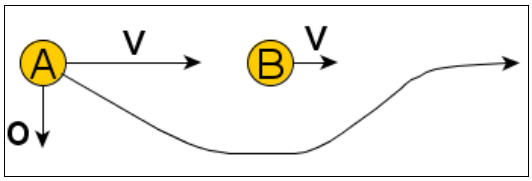

Fig. 1. Agent A obchází agenta B, **v** jsou jejich aktuální vektory rychlosti, **o** je síla, která vychýlí agenta A z původního směru.

Když detekujeme tento typ situace (agent B je přímo před agentem A a jejich vektory rychlosti mají směr buď stejný nebo přesně opačný), k odpudivé síle od agenta B přičteme vektor kolmý na vektor *lokace B – lokace A* (v obrázku Fig. 1. označený jako vektor **o**). Díky tomu se agent A vychýlí ze svého původního směru a získá možnost agenta B obejít.

Pomocí steeringu People-avoidance se dá úspěšně modelovat určitá socialita agentů. Např. jak moc se kdo straní ostatních, či si je nechá pustit k sobě blízko.

2.4 Target-approaching

Jedná se o velmi jednoduchý steering určený pro směřování agenta k cílové lokaci. Má dva parametry: *cílovou lokaci* a *velikost přitažlivé síly* k cílové lokaci.

Steering funguje tak, že cílová lokace agenta konstantně přitahuje (nehledě na jeho vzdálenost).

2.5 Path-following

Tento steering je určený pro následování vytyčené cesty. To může znázorňovat například chůzi po chodbě. Nechceme, aby agent chodil přesně uprostřed chodby, ale aby se pohyboval uvnitř vymezeného koridoru ve směru chodby.

Steering pracuje se sítí navigačních bodů (*navigation graph*), což je graf, jehož vrcholy jsou navigační body (vybrané lokace na mapě světa) a hrany jsou spojnice mezi nimi. Vrcholy a hrany jsou voleny tak, aby nevedly přes překážky a síť se díky tomu dala používat ke snadnému průchodu bez kolizí. V naší mapě města vedou spojnice navigačních bodů často prostředkem ulic.

Steering má dva parametry: *cestu* zadanou seznamem navigačních bodů a *šířku koridoru*, což přesně znamená maximální povolenou vzdálenost od středové osy cesty (tedy od spojnice navigačních bodů).

Průběh výpočtu má tři fáze: správně rozpoznávat, mezi kterými navigačními body na cestě jsme, kontrolovat, zda jde agent po cestě správným směrem (případně ho do správného směru natočit) a v případě, že by měl agent v příštím tiku vybočit z koridoru, na něj zapůsobit přitažlivou silou směrem ke spojnici minulého a příštího navigačního bodu.

Tento steering má dvě typické aplikace. První souvisí s následujícím problémem: chodí-li agenti pouze po spojnicích navigačních bodů, vypadá to nepřirozeně. Použijeme-li Path-following, pohybují se agenti po celé šířce koridoru.

Druhá aplikace se týká situace s více agenty na jedné cestě, kde jedni se pohybují jedním směrem a druzí opačným. Kdyby měli chodit přímo po navigačních bodech cesty, tak budou muset řešit četné kolize a vyhýbat se podobně, jako když se provazochodci chtějí obejít na jednom laně. Budou-li se však pohybovat na základě pravidel steeringu Path-following, budou schopni využít celou plochu koridoru a jejich pohyb bude více připomínat proudící chodce na plné ulici.

2.6 Wall-following

V tomto steeringu chceme, aby agent uměl chodit podél zdi, což je stěna libovolné překážky, nejčastěji budovy.

Steering používá 5 paprsků: 1 dopředný, 2 dlouhé šikmé přední a 2 krátké boční (viz Fig. 2. a 3.). Délky těch postranních jsou nastavené tak, aby přímka procházející jejich konci byla rovnoběžná s dopředným paprskem.

Steering má jediný parametr – *vzdálenost od zdi*. Nejdříve si ukážeme, jak steering řeší chůzi podél rovné zdi. Pak vyřešíme dvě zvláštní situace.

Základní chování steeringu určují paprsky kolidující se zdí. Pokud je paprsek zanořen málo (méně než z jedné třetiny), je agent přitahován ke zdi (čím je dále od zdi, tím je přitahován více). Pokud se však přiblíží natolik, že se paprsek zanoří z více než jedné třetiny, působí na něj zároveň odpudivá síla (čím je agent blíže zdi, tím je síla větší), aby do zdi nenarazil. Existuje vzdálenost, kdy se tyto dvě síly vyruší. Pomocí popsané dvojice sil zařídíme, že agent jde rovně podél zdi – ani do ní nenaráží, ani se od ní nevzdaluje.

První speciální situací je tzv. *silně konvexní roh* zdi (viz Fig. 2.), tedy situace, kdy zeď začne zahýbat tak, že agentovi zahradí cestu v původním směru (úhel rohu je z pohledu agenta nejvýše cca 135°). Je tedy nutno směr pohybu změnit. Nechť má agent zeď po levé ruce a oba jeho levé paprsky kolidují se zdí. Začne-li kolidovat i jeho přední paprsek, znamená to, že agent narazil na silně

konvexní roh. Víme, že se pak musí začít otáčet doprava. Proto na něj začne působit síla, která ho přitahuje doprava. K výslednému vektoru (který již ovlivnila obecná pravidla pro paprsky) přičteme vektor této rotační síly. Situace pro zeď po pravé ruce je symetrická.

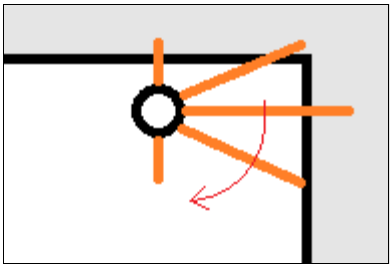

Fig. 2. Silně konvexní roh. Na obrázku jsou znázorněny agentovy paprsky a směr otáčení.

Druhou speciální situací je tzv. *silně konkávní roh* zdi (viz Fig. 3.), tedy situace, kdy zeď zahýbá tak, že agent přestane kolidovat šikmým předním paprskem. Tento roh je nutno obejít a pokračovat podél zdi. Jde-li agent podél zdi zprava, měl by se zdi stále dotýkat oběma levými paprsky. Pokud se však stane, že alespoň jeden z nich nekoliduje, znamená to, že se „zeď někam ztratila“, tedy že agent narazil na silně konkávní roh. Musí tedy roh obejít. Začne na něj tedy působit síla doleva, díky které se dostane za roh a časem narazí na zmizelou zeď. Pro pravou stranu vyřešíme situaci analogicky.

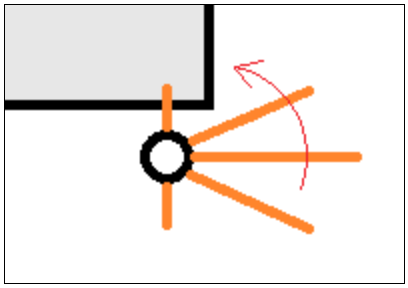

Fig. 3. Silně konkávní roh. Na obrázku jsou znázorněny agentovy paprsky a směr otáčení.

Pro hladké zdi bez ostrých rohů by stačily základní přitažlivé a odpudivé síly zdi. Naše implementace umí řešit nejen hladké zdi, ale i libovolně zakřivené zdi, přičemž agent plynule a přesvědčivě zahýbá a dokáže se vypořádat dokonce i se zdí, která najednou končí. Na jejím konci ji obejde a pokračuje po její opačné straně zpět.

2.7 Leader-following

Tento steering naviguje agenta (*následovníka*) tak, aby nějakým způsobem následoval jiného agenta (*vůdce*).

V naší implementaci jsou dva typy steeringu Leader-following. První typ je tzv. *základní*: následovník je přitahován přímo směrem k vůdci, ale jen tak, aby se k němu nedostal příliš blízko. Druhý typ (tzv. *formační*, neboť umožňuje vytvářet formace) dovoluje určit pozici vůči vůdci, kde se má následovník vyskytovat. Může chodit za ním, ale i před ním, vedle něj apod.

Steering má tedy tyto parametry: *jméno vůdce*, ideální *vzdálenost od vůdce* a *typ*. V případě, že se jedná o druhý typ, pak následují ještě vlastnosti *úhel* a přepínač pro *paměť* vektorů rychlosti vůdce, kde může být navíc nastavena její *velikost*.

První typ vychází přesně ze steeringu C. Reynoldse. Druhý, nově vytvořený typ steeringu Leader-following umožňuje určit pozici vůči vůdci. K výpočtu její lokace se využije aktuální vektor rychlosti vůdce, úhel a vzdálenost. Na agenta pak působí přitažlivá síla k této lokaci.

Paměť je speciální vylepšení, které zajišťuje plynulejší chování. Základní steering bez paměti se potýká s následujícím problémem. 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 za krátko se zase vrací, vypadá to, že zmateně pobíhá. Je-li následovníků dokonce více, situace působí ještě komičtěji.

Tento problém se nám podařilo vyřešit (nebo alespoň zmírnit) tím, že si následovník pamatuje posledních *n* vektorů, kde *n* je parametr velikost paměti. Místo aby pak cílovou pozici počítal z jednoho aktuálního vektoru rychlosti vůdce, použije průměr *n* posledních vektorů rychlosti vůdce. Jeho chování je pak mnohem hladší a plynulejší, neboť krátkodobé výchylky vůdcova pohybu příliš neovlivňují pohyb následovníka.

2.7.1 Využití steeringu Leader-following

První typ nedovoluje složité formace, nicméně i s ním se dá jednoduchých formací docílit. Například tzv. „husí pochod“. Stačí aby nebyl jediný vůdce, ale každý v řadě kromě posledního byl vůdcem agenta za ním.

Druhý typ dává mnohem větší možnosti v rozestavení agentů. Téměř libovolná formace lze popsat pomocí vzdáleností a úhlů. Je však třeba dát pozor na to, že s rostoucí vzdáleností se malé odchylky ve směru rychlosti stávají velkými, tuto vlastnost budeme nazývat *zveličování chyb*. Zveličování chyb se dá zmírnit použitím paměti.

Tento typ steeringu může třeba hezky představovat situaci s magnátem uprostřed a ochrankou v podobě tří bodyguardů okolo něj. Kam jde magnát, tam jdou jeho ochránci. Stačí tedy aby magnát byl vůdce a strážci si hlídali své pozice.

Obecně může být steering Leader-following výhodný i pro zjednodušení výpočtů. Často stačí, když určitou kombinaci steeringů má sám vůdce – a jeho následovníci po něm kopírují i ty steeringy, které sami nemají.

2.8 Walk-along

Steering Walk-along potřebuje dva agenty, tzv. *partnery*. Partneři mají jít vedle sebe ke společnému cíli. Parametry jsou tedy *jméno partnera*, *cílová lokace*, *vzdálenost od partnera* a *velikost přitažlivé síly* (k partnerovi i k cíli).

Na agenta působí dvě síly. První je přitažlivá síla k místu X, které je uprostřed spojnice lokace partnera a cílové lokace. Její velikost se přeškáluje podle toho, kdo a o kolik je vzdálenější cíli.

Nejdříve se vypočítá pomocné *D* jako rozdíl agentovy vzdálenosti od cíle a partnerovy vzdálenosti od cíle, vhodně podělené. Následně se oříznou příliš velké hodnoty *D*. Podle toho se vypočítá velikost vektoru přitahující agenta k bodu X jako *přitažlivá síla* * 2^D.

Pokud to má agent k cíli dál než partner, tato síla je velká a agent se snaží partnera dohnat. Pokud je dál partner, pak je *D* záporné a síla je zanedbatelná, neboť je třeba na partnera spíše počkat.

Druhá síla působí směrem k partnerovi. Čím je partner dál, tím je tato síla větší. Naopak pokud je partner moc blízko, je tato síla záporná, neboli odpudivá.

Díky těmto silám partneři vypadají, že jdou opravdu spolu. Jeden na druhého počká, když se druhému něco stane (narazí do stromu), atd. Když jsou jejich počáteční lokace daleko, místo aby vyrazili k cíli každý sám a setkali se až později na cestě k cíli, tak jakmile se spatří, seběhnou se k sobě a pak teprve spolu vyrazí.

2.9 Kombinace steeringů

Jak bylo řečeno v úvodu, steeringy jsou ještě užitečnějším nástrojem řízení, můžeme-li je kombinovat dohromady. Výhodné je, že vektorový princip, na kterém jsou založeny, je pro skládání ideální. Do skládání budou zahrnuty tyto vektory: aktuální vektor rychlosti a výsledky jednotlivých steeringů (každý steering vrací jeden vektor). Ukážeme si možnosti, jak se dají tyto vektory kombinovat dohromady.

1. Pro mnoho situací stačí, aby se všechny vektory obyčejně sečetly. Např. Obstacle-avoidance + Target-approaching. Výsledek bude tvořit součet 3 vektorů, přičemž jeden z nich bude často nulový a druhý stále stejně veliký.
2. První přístup má jedno úskalí. Mohou takto vznikat příliš velké vektory (pokud vektory v součtu mají podobný směr). Na agenta by pak působila příliš velká síla, což by mohlo způsobit nepřirozené zrychlení. (Jak se pracuje s výslednou velikostí rychlosti, je popsáno na konci této sekce.) Proto druhý přístup velikost výsledného vektoru podělí číslem *N*, které zde odpovídá počtu vektorů.
3. Druhý přístup má také své nevýhody. Některé steeringy (např. Obstacle-avoidance) často vrací nulový vektor. Tento vektor bychom neměli započítávat do hodnoty *N*. Jinak by agent příliš často nepřirozeně zpomaloval. Třetí přístup tedy do hodnoty *N* započítává jen ty vektory, které nevracejí nulové (a ani velmi malé) hodnoty.
4. Čtvrtý přístup je opět vylepšením předchozího. Dovoluje dávat jednotlivým steeringům a aktuálnímu vektoru váhy. *N* je pak součet vah steeringů, které vrací dostatečně velké vektory. Tímto způsobem se dají některé steeringy upřednostňovat před jinými.
5. Dále lze využívat různé heuristiky a jiná vylepšení. Je možno přiřazovat steeringům priority. Výsledný vektor pak závisí na potřebách steeringů s nejvyššími prioritami. Jiným vylepšením je dát steeringům možnost vyjádřit, jak moc je jejich výsledek důležitý, atp.

Náš algoritmus kombinování steeringů využívá čtvrtý přístup a jedno vylepšení, díky kterému mohou steeringy zdůraznit, že je důležité, aby agent v danou chvíli zpomalil. Než se výsledný vektor (označme ho *V*) použije jako nový vektor rychlosti, je potřeba upravit jeho velikost. Tato úprava má dvě fáze:

Zaprvé vyjde-li vektor *V* kratší než je maximální rychlost chůze, je možno ho na ni prodloužit. Proč by se měl prodlužovat? Agenta může na chvíli něco zpomalit, např. vyhýbání se překážce. Jakmile se překážce vyhne, může jít klidně zase maximální rychlostí dál. Nicméně v jiných situacích je naopak smysluplné, aby šel agent pomalu (např. dohání-li ho jiný agent). Proto má každý steering možnost říci, že nechce, aby se rychlost navyšovala. Pokud alespoň jeden ze steeringů takto „alarmuje“ a pokud vyjde vektor *V* malý, je ponechán beze změny; jinak je navýšen na maximální rychlost chůze.

Zadruhé se podle velikosti vektoru *V* určí, zda se bude agent pohybovat chůzí (základní pohyb), nebo během. K tomuto používáme vlastni přechodovou funkci, která zajišťuje plynuly přechod mezi chůzi a během agenta.

3 Výsledky: kombinace steeringů, jejich srovnání a využití

3.1 Typické kombinace

Ať už je záměr řízení pohybu jakýkoliv, většinou chceme, aby agent nenarážel do okolních objektů. Většina kombinací tedy obsahuje Obstacle-avoidance či případně Wall-following, který srážkám také předchází. Pokud se v okolí mají vyskytovat i jiní agenti, většinou se použije i People-avoidance.

3.2 Wall-following + Target-approaching

Původně jsme předpokládali, že steering Wall-following nebude mít smysl kombinovat téměř s ničím, že by to nepředstavovalo žádnou reálnou situaci. Nicméně opak se ukázal býti pravdou.

Zkombinujeme-li vhodným způsobem Wall-following a Target-approaching, získáme chování, které vypadá, jako když agent k cíli dojde pouze po chodnících.

Samozřejmě tato kombinace nefunguje vždy. Nicméně pro město s jednoduchou sítí širokých ulic a nepříliš vzdálený cíl to může vypadat velmi realisticky.

Největší problém činí agentovi fáze, kdy zeď uhýbá přesně na opačnou stranu, než je cílová lokace. Právě toto je příležitost pro využití vah steeringů. Kdyby měl Target-approaching o něco větší váhu, mohlo by se agentovi podařit odbočit od zdi bez většího „váhání".

3.4 Srovnání steeringů Leader-following a Walk-along

Základním smyslem steeringu Walk-along je, aby dva agenti šli vedle sebe k nějakému cíli. To by se dalo zařídit i tím, že jeden z nich by byl vůdce a druhý by šel vedle něj (tedy s úhlem ±90°). Druhá možnost však má několik nevýhod, což byly důvody, že jsme vytvořili nový speciální steering Walk-along.

Zásadní problém steeringu Leader-following druhého typu v této situaci je ten, že vůdce se vůbec nestará o to, zda ho někdo následuje. On o tom vlastně ani neví. Pokud se tedy následovníkovi něco stane (zdrží ho nějaká překážka), vůdce mu klidně uteče. Problém nastává, pokud se vůdce ztratí z dohledu následovníka (např. zajde za roh). V takovém případě následovník nemá šanci vůdce najít, neboť když ho nevidí, nemůže určit jeho lokaci ani vektor rychlosti.

Oproti tomu Walk-along zařídí, že oba agenti na sebe čekají a vykazují četné známky toho, že se vnímají oboustranně. Proto chceme-li, aby byl mezi agenty rovnocenný vztah, použijeme Walk-along. Má-li být zřejmé, že jeden z agentů má ve vztahu vůdčí pozici, je vhodnější varianta se steeringem Leader-following.

5 Závěr

Používání výše navržených steeringů může vytvářet přirozené chování agentů v různých typech situací. Z těch nejzajímavějších zmiňme např. plynulé obcházení zdí budov, průchod městem převážně po chodnících, společnou cestu dvou kamarádů, apod. Pomocí steeringů můžeme vytvářet různé vztahy mezi agenty: jak moc se kdo straní či bojí ostatních, zda je někdo vyloženě neoblíbený, kdo má ve skupině vůdčí pozici, apod. Tyto steeringy taktéž dovolují určit rozestavení agentů ve skupině (tedy vytvářet formace).

V průběhu práce jsme se setkali i s určitými problémy. Některé z nich jsou spíše technického rázu a nezávisí jen na knihovně steeringů. Mezi ně patří např. to, že intervaly mezi voláním steering mangera (tedy délky tiků) jsou příliš dlouhé. Steeringy tedy musí být dostatečně robustní, aby agent vždy včas zareagoval, i když se pohybuje rychle.

Častým problémem jsou dva protichůdné požadavky: plynulé chování, avšak včasné zareagování na danou situaci. Obecně plynulejší chování je možno zařídit, přiřadíme-li vyšší váhu aktuálnímu vektoru rychlosti při vytváření nového vektoru rychlosti. To však logicky vede ke zpomaleným reakcím.

Hlavní výhodou našich steeringů je, že jsou výpočetně nenáročné. Proto by mohly být využitelné v některých počítačových hrách, např. je-li potřeba, aby na pozadí chodili lidé po chodnících, vyhýbali se sobě a jiným překážkám – a přitom aby to nestálo příliš mnoho výpočetních sil.

Další z možných využití je ve výuce matematiky, fyziky a informatiky na středních školách, neboť steeringy ukazují využitelnost vektorů a jejich skládání.

Poděkování

Tvorba počítačové grafiky, kterou náš program využívá, a práce na tomto textu byly podpořeny grantovým projektem CZ.2.17/3.1.00/31162, který je v rámci OPPA financován Evropským Sociálním Fondem a rozpočtem hl. m. Prahy. Výzkum steering pravidel byl podpořen výzkumným záměrem MSM0021620838 (MŠMT ČR) a projektem GA UK č. 0449/2010/A-INF/MFF. Část grafických textur je převzata z knihovny Mayang's Free Textures library: http://mayang.com/textures/.

Literatura

[1] C. W. Reynolds: Steering Behaviors For Autonomous Characters. In: *Proceedings of Game Developers Conference*, Miller Freeman Game Group, San Francisco, California, 1999: 763-782.

[2] C. W. Reynolds: Flocks, Herds, and Schools: A Distributed Behavioral Model. In: *Proceedings of Computer Graphics*, SIGGRAPH, 1987: 25-34.

[3] 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.