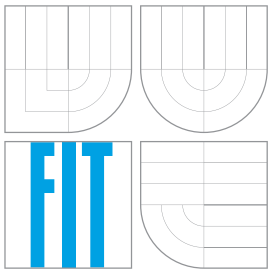
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

STRATEGICKÁ HRA V MULTI-AGENTNÍM PROSTŘEDÍ JASON

STRATEGIC GAME IN MULTI-AGENT SYSTEM JASON

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ROMAN VAIS
AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ KRÁL
SUPERVISOR

BRNO 2015

Abstrakt

Práce se zabývá umělou inteligencí využívanou v oblasti počítačových her, zejména pak tzv. strategií v reálném čase, a implementuje rozšíření pro jednu z těchto her. Analyzuje možnosti využití přístupu multi-agentních systémů právě pro účely umělé inteligence v počítačových hrách. Zabývá se konceptem swarm inteligece (inteligence roje), jako vhodné, ale nevyužívané, varianty umělé inteligence nejen pro strategické hry. Mimo jiné se tato práce pokouší o volbu vhodné reprezentace vjemů jednotlivých smyslů pro softwarové agenty a poukazuje na náročnost tohoto problému.

Abstract

This thesis describes artificial inteligence used in developement of computer games, particularly discusses with theory behing artificial inteligence used in real-time strategy games. It deals with implementation of extensions for one such a game. It analyzes posibylities of use multi-agent systems architecture for purposess of artificial inteligence in games. It describes concept of swarm inteligence as suitable but not used tool for developing not only videogame artificial inteligence. Moreover it attempts to find suitable representation of sensation for software agents and shows the difficulties of this task.

Klíčová slova

Multi-agentní systémy, umělá inteligence, swarm inteligence, inteligence roje, RTS, Java, Jason, agent, strategická hra, BDI agent, reaktivní systémy, rozhodování, R-tree, R*-tree, refactoring, optimalizace mravenčích kolonií, mravenci.

Keywords

Multi-agent systems, artificial intelligence, swarm inteligence, RTS, Java, Jason, agent, strategy game, BDI agent, reactive systems, reasoning, R-tree, R*-tree, refactoring, Ant Colony Optimization.

Citace

Roman Vais: Strategická hra v multi-agentním prostředí Jason, bakalářská práce, Brno, FIT VUT v Brně, 2015

Strategická hra v multi-agentním prostředí Jason

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jiřího Krále. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Vais
27. května 2015

Poděkování

Tímto bych chtěl poděkovat vedoucímu své práce, panu Ing. Jiřímu Královi, za jeho přístup a ochotu. Dále také všem svým přátelům, za jejich pomoc při korekcích tohoto textu a jejich podporu.

Obsah

Kapitola 1

Úvod

1.1 Motivace

Umělá inteligence je velice zajímavá a rozsáhlá problematika. Lidé se jí zabývají snad již od dob co jsou prvních počítačů. V roce 1950 publikoval Alan Turing článek Computing Machinery and Intelligence, ve kterém formuloval principy strojového učení (Machine learning), genetických algoritmů (Genetic algorithms), posilovaného učení (Reinforcement learning) a především testu kvality umělé inteligence (Turing test).[28] Od těchto prvopočátků se začala oblast umělé inteligence rozrůstat. Rozlišujeme různé druhy umělé inteligence. Lze ji rozlišovat podle typů úloh, které jsou jí předkládány, podle způsobů jakými tyto úlohy řeší, nebo i na základě jiných kritérií. V současné době je umělá inteligence součástí mnoha běžných zařízení. Nejedná se však o jednoduchou problematiku.

Tato práce se věnuje teorii umělé inteligence ve spojení s počítačovými hrami a nejen s nimi. Počítačové hry i umělá inteligence si vzájemně poskytují živnou půdu. Počítačové hry se stávají stále komplikovanější a i nároky hráčů na použitou umělou inteligenci vzrůstají. To tvoří výzvu pro odborníky v této oblasti. My se zde zaměříme na druh umělé inteligence, který se v počítačových hrách obvykle nevyužívá a to konkrétně inteligenci roje (swarm inteligence)[10] .

1.2 Struktura

Čtenáře nejprve seznámíme s pojmy, které jsou pro problematiku umělé inteligence klíčové. Začneme samozřejmě tím co je to přirozená a umělá inteligence. Protože se tato práce týká spojení umělé inteligence a počítačových her, ukážeme si jak se obvykle v počítačových hrách implementuje. Zde se trošičku zaměříme na realtimové strategie. Ne zřídka se ve hrách vyskytuje více než jeden charakter. Tyto charaktery lze označit pojmem agenti. Hra jako taková je komplexní systém a pokud je v něm více charakterů - agentů, máme multiagentní systém. Hráči u charakterů ve svých hrách jistě ocení trochu lidské chování. Pokud naši agenti využívají BDI model (viz. dále), jsme schopni zajistit, aby se agnet rozhodoval podobným způsobem jako člověk.

Zmiňovali jsme swarm inteligenci. Ta souvisí s multi-agentními systémy. Jedná se o velmi zajímavý a robustní koncept, který je využíván pro specializované úlohy. My však diskutujeme o potenciálu pro využití swarm inteligence v počítačových strategiích. Kde hledat inspiraci pro multiagentní systém, nebo počítačovou hru ? Co třeba u mravenců ? Co se týká mravenců, řekneme si něco i o těch reálných. Když už je řeč o hrách, imple-

mentaci jedné z nich si rozebereme. Naším úkolem jsou rozšíření této hry[24], která umožní aplikovat koncept swarm inteligence. Rozebereme si navrhovaná rozšíření, podíváme se na implementaci a v závěru zhodnotíme celou práci.

Kapitola 2

Umělá inteligence

V této kapitole bude čtenář seznámen s některými definicemi co je umělá inteligence, Jak je vnímána obecně v počítačových hrách a čím je specifická umělá inteligence ve strategických hrách. Zde lze najít stručné seznámení s tím, jak se obvykle umělá inteligence počítačových her implementuje, a co řeší. Dále také co vedlo k předpokladu, že počítačová hra je vhodným modelem pro testování umělé inteligence.

2.1 Umělá inteligence obecně

Umělá inteligence má mnoho různých definic. Encyklopedie Diderot jako definici umělé inteligence uvádí následující: Modelování intelektuální činnosti člověka počítačem při řešení složitých úloh, kde postup vyžaduje schopnost výběru z mnoha nebo z nezřetelně popsaných variant. Též schopnost samočinného rozpoznávání tvarů nebo předmětů, usuzování z jednoho výroku na jiný, vytváření analogií mezi jednotlivými úsudky, generování a ověřování hypotéz, tvorba a uplatnění znalostí na základě vstupních dat a informací, schopnost eliminovat nepříznivé reakce na podněty z okolí, usměrňovat činnost systému v probíhajících procesech s ohledem na měnící se a často nezřetelné vnější podmínky. Obecně lze říci, že se umělá inteligence zabývá řešením úloh, učením a rozpoznáváním.[28]

Umělá inteligence (Artificial Inteligence - AI) je značně rozsáhlá a členíme ji do menších podoblastí s vlastní problematikou a vlastním přístupem k řešení úloh. Mezi tyto podoblasti patří klasická umělá inteligence, distribuovaná umělá inteligence a tzv. soft computing. Klasická umělá inteligence řeší zadané úlohy prohledáváním tzv. stavového prostorou. Stavový prostor obsahuje množinu všech stavů v jakých se úloha může nacházet. Obvykle je reprezentován grafem. V tomto grafu se pak hledá nejoptimálnější cesta z počátečního do cílového stavu. Pod distribuovanou umělou inteligenci spadá swarm inteligence a multiagentní systémy, které budou podrobněji popsány v dalších kapitolách. Soft computing se pak zabývá např. neuronovými sítěmi, pravděpodobnostním usuzováním, fuzzy logikou a dalšími.[28]

2.2 Umělá inteligence v počítačových hrách

V současné době jsou počítačové hry stále komplexnější. Modeluje a simuluje se stále propracovanější prostředí, které se čím dál více blíží tomu reálnému. S tím narůstá i komplexita pravidel pro chování jednotlivých postaviček a charakterů v těchto hrách. Prostředí získává více interaktivních prvků, kterých může využít hráč i samotná AI. Na AI jsou kladeny vyšší

nároky. Hráči od ní očekávají výkon srovnatelný s jiným lidským hráčem, ne-li lepší. Každý žánr počítačových her staví před AI odlišné problémy. Obecně chceme aby byla AI rychlejší a chytřejší. Důsledkem tohoto trendu je, že AI musí zvládat nejen přibývající počet úloh, ale také jejich zvyšující se komplexitu. To samozřejmě vede i ke složitější AI jako takové. AI je pak rozdělena do mnoha různých částí, kde každá má za úkol řešit jinou část úlohy kterou AI plní.[3]

Počítačové hry jsou simulací prostředí, do něhož jsou samy zasazené, a charakterů, které se v daném prostředí nachází. Především u her běžících tzv. v reálném čase je proto důležitá rychlost zpracovaní dat, vykreslování animací, odezva ovládacích prvků a mnoha dalších podsystémů takové hry. Umělá inteligence není výjimkou a lze ji zařadit mezi systémy náročnější na výkon. Díky pokroku v informačních technologiích dnes lze pro AI i ve hrách běžících v reálném čase s výhodou využít náročných algoritmů. Je ovšem stále nutné kvůli rychlosti zpracování zjednodušovat vnitřní reprezentace prostředí a charakterů.[3]

2.2.1 AI pro hry typu FPS a RPG

Ve hrách tohoto typu AI kontroluje samostatně jednotlivé charaktery a určuje jejich akce samostatně. Je to možné hlavně díky malému počtu charakterů. FPS (First person shooter, střílečka z pohledu první osoby) mají systém umělé inteligence rozvrstvený. Nejnižší vrstva obstarává základní úkony jako tzv. path-finding, neboli hledání cesty pro pohyb charakteru (kudy postavička půjde), nebo synchronizaci animace s vlastním pohybem. Vyšší vrstvy se starají o složitější úlohy jako je vnímání podnětů z okolí nebo v některých případech i učení. Úplně na vrcholu je pak část která se stará o nejsložitější abstrakce a celkové komplexní chování charakteru. Zodpovídá za adekvátní chování charakteru vzhledem k dané situaci a svým znalostem.[3]

Path-finding je obvykle založený na grafech popisujících prostředí. Každý vertex v grafu reprezentuje logickou pozici ve světě jako např. místnost v domě, nebo konkrétní oblasti na bojišti. Pokud AI získá zadání cíle, kam se má dostat, z patřičných grafů si vytáhne množinu navigačních bodů, přes které se má pohybovat, aby se charakter dostal na kýžené místo.[16]

Pro vyšší úroveň rozhodování je mnohdy využíváno jak v FPS tak RPG hrách tzv. naskriptované chování. Jde o vývojáři předem připravenou posloupnost kroků a akcí, které AI vykoná, nastane li nějaká událost. Příkladem může být následující: jestliže hráč vstoupí do určité oblasti, pak do této oblasti přijde nepřítel a na hráče zaútočí včetně způsobu jakým to má provést, nezávisle na tom, zda tento nepřítel mohl hráče zpozorovat či nikoliv, a na tom, zda by toto chování AI zvolila sama v případě jiného způsobu implementace. Ve chvíli kdy se AI rozhodne pro dosažení nějakého cíle nižší vrstvy zajistí optimální cestu k cíli a pokud to není naskriptováno, tak i jakým způsobem.[22]

2.2.2 AI pro závodní a sportovní hry

Ve hrách jako jsou automobilové závody se setkáváme velmi často s rozsáhlým podváděním ze strany AI. Pro potřeby AI je mapa (závodní dráha) rozdělena do polygonů. Tyto polygony patří pouze trati, po níž se má auto pohybovat. K jednotlivým polygonům, které tvoří krátké úseky trati jsou připojené i informace o vlastnostech trati, jako je povrch, sklon a další, v daném úseku. AI pak z těchto informací vytvoří graf zachycující veškeré potřebné informace o dráze. Díky tomu má svým způsobem kompletní přehled o dráze. Ví v jakém okamžiku a jak hodně musí zpomalit, nebo že se blíží ke křižovatce a může to vzít zkratkou.

Ve sportovních hrách se lze setkat s podobným principem podvádění. AI má naskriptované (dopředu předdefinované) veškeré chování, takže výsledek je daný ještě před tím, než se AI dostane ke svému tahu. Toto chování je pak následně provedeno pohybovým systém, zobrazujícím vhodné animace. Některé hodnoty v rámci skriptovaného chování mohu být "jemně doladěné"pomocí generátoru pseudonáhodných čísel, aby nebyl výsledek vždy úplně přesně stejný a tento drobný podvod velkého rozsahu pro hráče zřetelný.

2.2.3 AI pro strategické hry

AI pro strategické hry je z těch komplexnějších. Podobně jako AI sportovních her potřebuje vnitřní reprezentace mapy a podobně jako u FPS nebo RPG her potřebují jednotky strategických her trochu komplexnější chování. Na rozdíl od charakterů, i těch generických, v RPG a FPS hrách jsou jednotky ze strategií trochu méně inteligentní a nepotřebují až tak komplexní chování. Na druhou stranu je těchto jednotek často mnohem více a je pro ně mnohem důležitější spolupráce. Málo kdy nastane situace taková, že pouze jedna jednotka by rozhodla o výsledku sama o sobě a pouze svým individuálním chováním. Jednotlivé druhy jednotek se od sebe vzájemně mnohem více odlišují než charaktery RPG a FPS her. Do rozhodovacího procesu navíc vstupují kromě řízení jednotek a jejich činnosti i další faktory jako je ekonomika, stavba základy, získávání a zpracování informací o protihráči.[3]

2.3 Specifické vlastnosti AI ve strategických hrách

Jednou ze specifických částí u strategických her je path-finding. AI kontroluje své jednotky skupinově, i když individuální kontrola je samozřejmě také možná, v určitých situacích nutná. Jednotky mohou mít své individuální schopnosti, které mohou ve větší či menší míře přispět k dosažení cíle. Jednotky musí být řízeny tak, aby se vzájemně nezablokovaly. AI musí vyhodnotit situaci a najít vhodnou cestu ve zlomku vteřiny a to i pro stovky jednotek najednou. Pro zefektivnění a zjednodušení tohoto úkolu AI používá seskupování jednotek do formací a jejich skupinový pohyb. Zde se využívá toho, že jedna jednotka slouží jako vůdce skupiny a pro ni se vyhledá optimální cesta. Ostatní jednotky ve skupině tuto jednotu následují podle toho v jaké formaci se skupina nachází.

Jednotky ve strategiích také potřebují specifickou roli, pro svoji spolupráci ve skupině. Role je obvykle odvozena od druhu dané jednotky. Ať už se jedná o léčbu, palebnou podporu, průzkum, role silně ovlivňuje činnost dané jednotky. Složení skupiny podle počtu jednotek v různých rolích ovlivňuje skupinovou taktiku. Např. skupina, která nemá medika, může být "opatrnější"a vyhýbat se otevřenému konfliktu. Skupina která se skládá převážně z pomalu se pohybujících jednotek může mít tendenci jít přímo k cíli a ignorovat lehké útoky po cestě, nebo se "zakopat"na výhodné pozici, od níž se nebude vzdalovat a bude ji bránit. Rychle pohybující se jednotky naopak mohou volit taktiku "udeř a uteč"(hit and run). Pokud skupina nemá možnost útočit na letecké jednotky a přijde do kontaktu s leteckými jednotkami, může je buď ignorovat, nebo se stáhnout k protiletecké obraně. Jak je z příkladů patrné skupinová taktika musí zvážit mnoho faktorů a jedná se o poměrně komplexní problém, který AI musí být schopna řešit opět ve zlomcích sekundy.

Dalo by se uvést i mnoho dalších příkladů mnoha dalších druhů chování. Jedno je však i z výše uvedených příkladů patrné. AI si musí velmi rychle dokázat poradit s velkým množstvím komplexních problémů v krátkém čase. na pomoc si proto bere různé druhy map napomáhajících v rozhodovacím procesu.[16]

Rozhodovací mapy (Decision making maps)

Obvykle se jedná a dvourozměrná pole naplněná strukturami popisujícími určité oblasti herní mapy. Tyto struktury obsahují nejrůznější informace týkající se dané oblasti, které mají vliv na rozhodnutí AI. Rozhodovací mapy pomáhají AI dělat smysluplná rozhodnutí.

Mapy zdrojů (Resource maps)

Mapy zdrojů obsahují informace o tom kde a v jakém množství se na herní mapě nachází dílčí zdroje. Tato znalost ovlivňuje rozhodnutí kde postavit další základu (tzv. expansion base) a zároveň poskytuje cenné informace o tom, kde se může nacházet protivník a kde bude pravděpodobně tento protivník stavět své další základny. Rychlý přístup ke zdrojům má pak obrovský dopad na rychlost produkce armády.

Mapy cílů (Objective maps)

Tyto mapy jsou svým účelem podobné rozhodovacím mapám. Obsahují informace o jednotlivých cílech, kterých by měla AI dosáhnout v průběhu hry. Například znič nepřátelskou základnu, postav novou základnu, hlídej a ochraňuj tenhle prostor atp. Zároveň obsahují informace o klíčových jednotkách a budovách. Informace jsou využívány k managementu jednotek. Klíčovým místům a budovám je přiřazena obrana. U míst chráněných protivníkem se zase sledují informace jaký typ obrany používá, aby nedocházelo ke zbytečným ztrátám jednotek atp.

Mapy konfliktů (Conflict maps)

Tyto mapy jsou obvykle aktualizovány a využívány mnohem častěji než ty předcházející. Kdykoliv se jednotky pustí do konfliktu s jednotkami protivníka, aktualizují mapu konfliktu klíčovými informacemi o konfliktu samotném i o soupeřově armádě. Patří sem informace o tom, kde se konflikt odehrává, jaké typy jednotek protivník nasadil (Je v daném místě nějaká statická obrana, produkční budovy?), jaké jsou schopnosti těchto jednotek, počty, jaká je jejich síla. Analýzou těchto informací AI může určit zda mohou nasazené jednotky dostačující, zda mohou bojovat efektivně, kde má protivník mezery ve své obraně, nebo jaká by měla AI udělat opatření - zda je nutné ustoupit, jaké jednotky budou nutné na proražení obrany, zda je nutné poslat podporu a další.

Kapitola 3

Agenti a multi-agentní systém

Tato kapitola popisuje pojmy jako 'agent' a multi-agentní systém[8] tak jak jsou chápány v rámci programovacího jazyka Jason. Jason je jednou z implementací standardu AgentSpeak, který je určen pro vytváření multi-agentních systémů. Zároveň čtenáře krátce seznámí s BDI modelem, na němž je Jason založený. V literatuře[8] týkající se Jasonu je odkazováno na jazyk AgentSpeak, ze kterého Jason vychází. Vzhledem k tomu, že práce se nezabývá ani AgenSpeakem ani Jasonem jako takovým, budou vysvětleny pouze pojmy používané v dalších částech této práce a pojmy použité pro jejich vysvětlení, nebo úzce související. Také bude odkazováno pouze na Jason a literaturu, ze které bylo pro tuto práci čerpáno. Kapitola je z velké části překlad a reformulace výše zmíněné literatury.

3.1 Reaktivní systém

Aby byly pojmy jako agent a multi-agentní systém snáze pochopitelné, kniha věnovaná jazyku Jason nejprve popisuje pojem Reaktivní systém. Dle definice se jedná o takový systém, který nelze vhodným způsobem popsat jako funkcionální nebo relační. Funkcionální systém lze popsat jako matematickou funkci a relační jako matematickou relaci. Takový systém typicky převezme vstupní data, nějakým způsobem je zpracuje, předá výsledek a následně se ukončí. Reaktivní systémy si zachovává interakci se svým prostředím. Z tohoto důvodu musí být popisován a specifikován s ohledem na své chování. Jako příklad reakčního systému se dají uvést webové servery, operační systémy, online bankovní systémy a mnoho dalších. Jedním ze specifických příkladů takového systému je i agent.

3.2 Agent

Pod pojmem agent lze chápat komplexnější podtřídu reaktivních systémů. Tomuto systému může programátor delegovat nějaký úkol (případně množinu úkolů) a tento systém úkoly samostatně vyplní. Systém sám zvolí, jaká posloupnost akcí je nejvhodnější pro splnění zadaného cíle a tyto akce aktivně, v rámci svého prostředí, postupně vykoná. Agenti jsou umístěni do určitého prostření, které mohou ovlivňovat a zároveň vnímat, jak se prostředí mění, ať už samo o sobě, nebo na základě jejich akcí. Agent pak na tyto vnímané změny reaguje a upravuje své chování. Za chování považujeme posloupnost po sobě následujících akcí tak, aby bylo možné dosáhnout zvoleného nebo daného cíle. Agent musí být schopen vyhodnotit získané informace a své chování přizpůsobit. Pro agenta platí následující:

- je autonomní

- je proaktivní
- je reaktivní
- je sociabilní

Autonomie

Autonomií je obecně myšleno samostatné rozhodování, iniciativa při realizaci daného rozhodnutí a vlastní cíle, k jejichž dosažení mají tato rozhodnutí a tyto akce přispět. Není třeba žádného vnějšího impulzu k provádění rozhodnutí, nebo akce z tohoto rozhodnutí vyplývající. Zjednodušeně řečeno, autonomie znamená schopnost dosahovat svých cílů nezávisle na jakýchkoliv příkazech.

Proaktivnost

Proaktivnost je chování zaměřené na dosažení cíle. Tato vlastnost zaručuje, že pokud si agent zvolil nějaký cíl, nebo mu byl nějaký cíl delegován, bude se tento agent aktivně snažit tohoto cíle dosáhnout. Vyloučí se tím existence pasivních agentů, kteří se nikdy zadaného cíle dosáhnout ani nepokusí. Pro porovnání lze použít příklad webového serveru. Webový server je pasivní dokud mu nepřijde požadavek (jinými slovy příkaz) na nějakou konkrétní stránku. Server sám nemá žádný cíl, kterého by se snažil dosáhnout, pouze reaguje na požadavky z venčí.

Reaktivnost

Být reaktivní znamená reagovat na změny v prostředí. Běžná je situace, kdy se naše plány setkají s nějakou překážkou a něco se nepodaří. Někdy jsou tyto plány zmařeny omylem, jindy úmyslně. Když nějaký plán nevyjde, je nutné na tuto situaci reagovat například opakovaným pokusem o dosažení daného cíle, nebo volbou náhradního způsobu řešení. Jestliže proaktivnost může být vysvětlena jako volba akce (či posloupnosti akcí) v závislosti na daném cíli, reaktivnost lze vysvětlit jako volba akce v závislosti na aktuálním stavu prostředí.

Sociabilita

Sociabilitou chápeme schopnost komunikovat a spolupracovat při dosahování společných cílů, nebo cílů, kterých nemůže agent dosáhnout samostatně (v daném okamžiku nemá potřebné znalosti, případně není schopen vykonat nějakou specifickou akci potřebnou k dosažení úkolu). Agent má schopnost sdělovat ostatním agentům konkrétní znalosti, přesvědčení, nebo informovat ostatní agenty o svých záměrech, případně se na tyto typy informací ostatních agentů dotazovat.

3.3 Multi-agentní systém

Jak už název napovídá, multi-agentní systém je systém kde se vyskytuje více (potenciálně různých) autonomních agentů. každý takovýto agent má vlastní sféru vlivu. Jedná se o část sdíleného prostředí, ve kterém agent působí a tato část prostředí je činností agenta ovlivněna. Sféry vlivu zahrnují i ostatní agenty, s nimiž může agent, jemuž tato sféra náleží, interagovat. Mohou, ale nemusí, se vzájemně protínat. I v multi-agentním prostředí tedy

může existovat osamocený agent. Pokud se sféry vlivu vzájemně překrývají, je pro agenty rozhodování složitější, protože do rozhodovacího procesu musí zahrnout i vliv ostatních agentů.

3.4 BDI model

BDI neboli belief-desire-intention model, což by se dalo volně přeložit jako "domněnka-cíl-záměr", byl inspirován lidským chováním a vytvořený filosofy. Základem je běžné lidské uvažování. Jedná se o programovou analogii ke znalostem či předpokladům, tužbám nebo cílům a záměrům lidí. Snahou tohoto modelu je získat v programu něco jako 'stav mysli'. Agenti jsou programy, u kterých lze s výhodou využít této analogie. Je vhodné nastínit rozdíly mezi domněnka, cílem a záměrem, tak jak je to chápáno v rámci jazyka Jason.

Domněnka (belief)

Jedná se o informaci, kterou má agent o svém prostředí. Tato informace může nebo nemusí být aktuální, může a nemusí být pravdivá či přesná. V běžném programu bychom měli proměnnou která udržuje hodnotu této informace. Touto informací může být aktuální teplota v místnosti. V Jasonu je použit princip velice podobný, ovšem tuto informaci považujeme za domněnku agenta - to v co agent v daném okamžiku věří.

Cíl / tužba (desire)

Jedná se o všechny možné cíle, kterých by mohl agent v danou chvíli chtít dosáhnout. To ovšem nutně neznamená, že na základě těchto tužeb bude agent jednat. Tyto tužby ovšem mohou mít vliv na to, jakým způsobem a pro co se agent bude nadále rozhodovat. Je zřejmé, že mnohé z těchto cílů mohou, a pravděpodobně budou, ve vzájemném rozporu. Pokud je tedy dosaženo jednoho cíle, druhý se může stát nedosažitelným. Tyto touhy lze také přirovnat k možnostem, které agent v danou chvíli má.

Záměr (intention)

Podobně jako v případě tužeb se jedná o nějaké cíle. Záměr je na rozdíl od tužby takový cíl, pro jehož dosažení se agent rozhodl aktivně pracovat. Záměr může být cíl, který byl na agenta delegován a je žádoucí (nutné) tohoto cíle dosáhnout, nebo se jedná o jednu z možností (tužeb), pro kterou se agent sám rozhodl. Lze si představit agenta, který má zadaný nějaký úkol a agent volí z aktuálních možností jak tohoto úkolu dosáhnout. Zvolená možnost se pak také stane záměrem a agent se tohoto záměru bude aktivně snažit dosáhnout. Tento proces může být (a často je) rekurzivní. Nově nabytý záměr nabízí další možnosti, ze kterých je nutné vybrat správnou akci, která povede k vyplnění tohoto záměru.

3.5 Rozhodovací proces

Ve výše uvedeném textu bylo několikrát zmíněno rozhodování a rozhodnutí. Agent který má nějaké cíle, znalosti a záměry se musí nějakým způsobem, na základě těchto, rozhodnout a začít podle daného rozhodnutí jednat. Základ na němž BDI model staví je nazýván pragmatické uvažování (practical reasoning). Tento proces je zaměřený směrem k akcím, tj. přijít na to co v dané situaci dělat. Praktické rozhodování je zvažování kladů a záporů

jednotlivých možností, které jsou relevantní k aktuálním hodnotám/tužbám/cílům, které jsou agentovi v danou chvíli vlastní.

3.6 Závislost mezi agenty a prostředím

Zde popisované informace jsou důležité z hlediska významu některých navrhovaných a implementovaných rozšíření. Agent má ve svém prostředí sféru vlivu, tj. prostor ve kterém působí a který ovlivňuje svým chováním. Agent je reaktivní systém, tedy reaguje na změny ve svém okolí nebo akce dalších agentů, kteří se v tomto okolí také nachází. Tyto změny agent vnímá pomocí nějakých senzorů či smyslů. Reakcí agenta bývá zpravidla nějaká akce, která změní stav jeho okolí. Smyslů nebo senzorů může mít agent několik a tyto smysly mohou být odlišné. Obdobně pomocí iniciátorů (manipulátorů) agent provádí změny v prostředí.

Protože agent a jeho rozhodnutí jsou ovlivňovány tím, jaké má o svém okolí informace, jsou smyly pro agenta velice důležité. Právě smysly tyto informace agentovi poskytují. Jestli je získaná informace pravdivá a aktuální, či nikoliv, má vliv na úspěch jednotlivých akcí agenta nebo úspěšné dosažení daného cíle skrze tyto akce. V tomto ohledu je agent závislý na svých smyslech a na současném stavu svého okolí.

Informace získané jednotlivými smysly se mohou doplňovat, potvrzovat, podávat informace, které mezi sebou nemají žádnou souvislost, nebo mohou být v rozporu. Lze uvést několik příkladů. V případě sluchu a zraku můžeme z nějakého směru slyšet přicházet zvuk. Pokud zaměříme tímto směrem zrak, můžeme lokalizovat nebo identifikovat původce tohoto zvuku (smyslové vjemy se vzájemně doplňují). Pokud vidíme míč, pomocí zraku získáme informaci o tom, že se jedná o kulatý předmět. Pokud míč uchopíme, hmatem si můžeme tuto informaci potvrdit (smysly se vzájemně potvrzují). Pokud ovšem budeme sledovat holografickou projekci, náš zrak uvidí v prostoru míč, ale nebudeme schopni získat žádný hmatový vjem (smysly jsou ve vzájemném rozporu). Na základě těchto informací lze usoudit, že více rozdílných smyslů poskytujících odlišné informace ovlivní rozhodovací proces agentů.

Kapitola 4

Swarm inteligence

Kapitola nabízí čtenáři seznámení s pojmem swarm inteligence, čím je specifická. Budou zde uvedeny některé vybrané algoritmy, které jsou v informatice používané a jsou řazeny pod swarm inteligenci. Nakonec bude diskutován potenciál použití swarm inteligence v souvislosti s umělou inteligencí v počítačových hrách používanou.

4.1 Co je to swarm inteligence

Pokud se vrátíme k otázce co je to inteligence, již víme, že se jedná o schopnost řešit nějaký problém. Jsou ovšem situace kdy jedinec / jednotlivec nemá dostatečnou úroveň inteligence a kapacitu na to, aby vyřešil nějaký komplexnější problém. Jako příklad můžeme uvést hmyz - konkrétně (nejen) mravence. Na první pohled se mravenci jeví jako dobře organizované společenství s velmi pozoruhodnou schopností se rychle přizpůsobovat a řešit situace jako je oprava poškozeného mraveniště, hromadné stěhování nebo zásobování velkého počtu jedinců tvořících populaci mraveniště potravou. Mravenci jako jednotlivci postrádají dostatečně vysokou úroveň inteligence na to, aby zvládaly tak komplexní úlohy.[**?**]

Mravenci zároveň postrádají jakoukoliv hierarchii.[21][9] V mravenčí kolonii není žádný inteligentní arbitr, který by určoval co je nutné udělat a vydával rozkazy. Přesto však kolonie "ví"nebo je schopna spočítat kolik dělníků je potřeba vyslat na hledání potravy, zda je nutné opravit např. deštěm poničenou vnější část mraveniště. Mravenčí kolonie je schopna pružně reagovat na měnící se podmínky a zajistit dostatečný počet dělníků pro jednotlivé úkoly. Z toho vyplývá otázka, díky čemu je kolonie schopna bez jakékoliv hierarchie tak komplexního chování. Odpovědí je swarm inteligence (inteligence roje).[17]

Pokud na pojem swarm inteligence nahlédneme z hlediska fungování u hmyzu jako jsou mravenci, včely nebo kobylky, lze dojít k následujícímu zjištění. Jednotlivci nejsou organizováni pomocí žádných příkazů. [20] Veškerá koordinace všech činností vyplývá z mnoha vzájemných interakcí a komunikace velkého množství jedinců kolonie mezi sebou. Komunikace a interakce či jakékoliv jiné předávání informací probíhá pouze lokálně. Ať už se jedná o přímé předávání informace nebo nepřímé v podobě zanechání feromonové cestičky (o této cestičce ví pouze mravenec, který přes ni prochází). Tyto interakce jsou v zásadě jednoduché a poněkud primitivní. Dohromady však tyto interakce tvoří výsledné komplexní chování. Vhodnou analogií může být běžný počítačový program. Jednotlivé příkazy a funkce požité v rámci zdrojového kódu počítačového programu mohou reprezentovat nějakou velmi primitivní činnost (skok na jinou část kódu, přiřazení hodnoty do proměnné, součet či porovnání). Dohromady poskládané za sebe a při vzájemném předávání výsledků činnosti však mohou

utvořit i velmi komplexní počítačový program.

Z pohledu informačních technologií se jedná o abstraktní koncept inspirovaný hmyzem nebo zvířaty žijícími ve stádech. Principielně je swarm inteligence sada několika jednoduchých pravidel, která jsou následována všemi členy skupiny agentů, kterých daný algoritmus využívá. Tento koncept se využívá k decentralizovanému a paralelizovanému řešení složitých problémů. Jedním z problémů, na který je vhodné aplikovat algoritmy založené na swarm inteligenci, je prohledávání stavového prostoru pro nalezení optimální cesty. Především ve velmi složitých grafech, kde k optimálnosti cesty závisí na mnoha různých vlivech a ne jen např. na vzdálenosti. [25][11]

4.2 Principy a použití swarm inteligence

Architektura systému založeném na swarm inteligenci sestává z určitého (potenciálně velkého) počtu autonomních jednotek - agentů. Agenti mají určitou úroveň schopností pro řešení dané úlohy, vzájemnou spolupráci a jednoduchou komunikaci. Komunikace je pro roj (swarm) agentů velice důležitá, protože vzájemná interakce jim umožňuje vzájemnou koordinaci a silně tak ovlivňuje chování roje jako celku. Agenti mají přehled pouze o malé části prostředí, ve kterém se zrovna nachází a obvykle pohybují. Tento přehled ovšem mohou předávat dalším agentům.

Na rozdíl od běžných přístupů k řešení problému jako je dekompozice či abstrakce má swarm inteligence několik výhod. Mezi klíčové vlastnosti swarm inteligence (SI) můžeme zařadit robustnost a adaptabilitu. Pokud bychom se zaměřili na rozhodovací proces, je silně ovlivněn možnostmi které jsou k dispozici. Příliš mnoho těchto možností může vést k tzv. kognitivnímu, jinými slovy informačnímu, přetížení. K tomuto jevy dochází pokud počet možností převyšuje kapacity jednotlivce. V důsledku dochází k volbě nesprávné akce či jiným špatným rozhodnutím. Systém SI je decentralizovaný, na jednotlivých agentech a jejich stochastickém procesu rozhodování nezávislý. Chyby v úsudku jednotlivců, ztráta těchto jednotlivců, nebo chyby při vzájemné komunikaci jsou pouze lokální a systém s nimi dokáže vypořádat. Tyto vlastnosti zajišťují robustnost a odolnost vůči chybám pro systém jako celek. Díky množství agentů je možné snadno pokrýt velkou část stavového prostoru (prostředí) a pružně reagovat na změny. Jakákoliv změna je propagována skrz systém a není tak problém rychle najít alternativní cestu, nebo nahradit ztracené agenty - adaptabilita.[25]

Příkladem problematiky, kde se dá s výhodou využít konceptu swarm inteligence je již zmíněné hledání optimální cesty. Pokud se jedná o složitý graf, kde více různých cest, které se mohou v některých uzlech také křížit, vede do cíle ale optimálnost a cena cesty nejsou pouze pod vlivem vzdálenosti, ale také mnoha dalších dynamicky měnících se podmínek, konvenční algoritmy by musely provádět velké množství složitých výpočtů což by zabralo velké množství zdrojů a času. Mezi reálné problémy odpovídající tomuto popisu můžeme zařadit například směrování paketů páteřních linek počítačových sítí, simulace a řízení silniční dopravy. Pro řešení zde zmíněných problémů je velmi často využíván algoritmus nazvaný Ant Colony Optimization.

4.3 Ant Colony Optimization algoritmus

Tento iterativní algoritmus byl, jak už z názvu vyplývá, inspirován mravenci. Specificky jejich chováním při hledání potravy. Nejedná se přímo o konkrétní algoritmus, ale spíše

o koncept, který je implementován různými algoritmy sdílejícími základní princip s implementačními obměnami nutnými pro aplikaci algoritmu na specifický problém.[10]

Základní princip spočívá ve vypuštění určitého množství agentů do stavového prostoru představujícího zadaný problém. Mravenci se následně snaží dostat k řešení (do cílového bodu). Po cestě za sebou nechávají chemické cestičky. Mravenci se pak pohybují následujícím způsobem. Pokud neexistuje žádná cestička v místě kam se snaží dostat, tak tuto cestičku za sebou vytváří. Pokud tato cestička existuje a mravenec se po ní rozhodne jít, zároveň ji posílí. Ovšem zda po této cestičce půjde se mravenec rozhoduje na základě nějaké pravděpodobnosti. Všichni mravenci preferují kratší cestu, pokud o ní ví. V případě, že se feromonová cestička větví, mravenec je více přitahován k větvi s větší koncentrací feromonů a je tak větší pravděpodobnost že ji zvolí. Další podrobnosti jsou uvedené v kapitole Reální mravenci a jejich strategie.

ACO není jediným konceptem či algoritmem, uplatňujícím se v rámci swarm inteligence. Existují další koncepty a algoritmy uplatňující se u jiných typů úloh, každý s množstvím odlišných variant. Vzhledem k tomu, že se tato práce zabývá rozšířeními pro hru s tématikou mravenčí kolonie, uplatnění konceptu ACO v umělé inteligenci se zdá být zajímavé pro porovnání s původní umělou inteligencí. Více v kapitolách Rozbor původní práce a Navrhovaná rozšíření a strategie.

4.4 Potenciál pro swarm inteligenci ve strategických hrách

Počítačové hry simulují prostředí, které je v nich modelováno. Model prostředí je v různých hrách odlišný. Je pouze na autorech konkrétní hry, zda bude modelováno prostředí reálné nebo fiktivní. Z hlediska umělé inteligence nezáleží ani na tom jaké prostředí je modelováno. Důležitý je model samotný. Pro AI je důležitý především senzorový vstup a nástroje, jimiž může modelem prostředí manipulovat. Při zachování formátu vstupních a výstupních dat AI nezaznamená žádný rozdíl. V rámci hry lze modelovat reálné prostředí, kde vyzkoušíme a otestujeme schopnosti dané AI. Po té lze tuto AI zkopírovat do těla nějakého robota. AI bude vykonávat stejnou činnost a stejně efektivně jako v rámci počítačové hry. Je možné také využít schopnost učení, kde se v rámci simulovaného prostředí AI může učit od jiné AI nebo od lidských protivníků. Virtuální prostředí je vhodné i z hlediska testování. V případě chyby AI může dojít k poškození robota. Toto riziko je virtuálním prostředím eliminováno.

Vzhledem k vzájemně doplňujícím se vlastnostem je swarm inteligence vhodná pro použití převážně ve strategických hrách. Strategické hry i swarm inteligence obvykle využívají většího množství jednotek či agentů k dosahování konkrétních cílů. Na rozdíl od jiných typů her (především RPG a FPS) jednotky a agenti nepotřebují tak komplexní chování. Stejně tak jako jednotky ve strategických hrách agenti se ne zřídka pohybují ve skupinách. V obou případech členové těchto skupin mezi sebou potřebují vzájemně komunikovat a spolupracovat. AI implementovaná ve strategických hrách je obvykle hierarchická. Rozhodovací proces probíhá ve specializovaných podsystémech AI, určených k řešení konkrétních typů úloh. AI jako celek je však řízena v nejvyšší vrstvě centralizovaným způsobem.

Decentralizovaný přístup uplatňující se v rámci swarm inteligence by mohl poskytovat odlišné a taktéž zajímavé výsledky. Pokud by byla swarm inteligence aplikována v některé z propracovanějších a náročnějších strategií, jako jsou série Total War nebo Command and Conquer, případně her, které jsou svými herními mechanikami silně specifické, jako je Starcraft II a Supreme Commander, mohlo by to přinést zajímavé porovnání intelektuálních schopností silně inteligentního jednotlivce (lidského hráče) vůči skupinové inteligenci softwarových agentů.

Kapitola 5

Reální mravenci a jejich strategie

Reální mravenci žijí ve společenstvích čítajících velké množství dělníků. Individuální intelektuální schopnosti jednotlivých mravenců jsou značně omezené.[23] Přes to mají pozoruhodné schopnosti kolektivního rozhodování. Díky tomu se staly jednou z předloh pro vznik swarm inteligence. Pro implementaci algoritmů využívajících vlastností mravenců je vhodné se s těmito vlastnostmi podrobněji seznámit. Uvedeme informace o individuálním chování. Nemohou však chybět dva příklady v nichž se uplatňují principy jejich kolektivního rozhodování. Zaměříme se orientaci v prostoru, porovnání individuálního hodnocení možností vůči tomu kolektivnímu a alokaci zdrojů, tj. způsob jakým je určeno množství dělníků, kteří se věnují konkrétnímu úkolu.

5.1 Mravenčí smysly a komunikace

Vlastnosti mravenčích smyslů jsou závislé na prostředí, v němž daný druh mravenců žije.[1] Některé druhy mravenců se orientují zrakem, jiné druhy mravenců jsou úplně slepé. Nejvíce využívaným smyslem mravenců je čich. Komunikace mravenců je založená především na látkách zvaných feromony. Ty jsou rozlišovány a detekovány právě pomocí čichu. Na základě feromonů se mravenci rozeznávají mezi sebou a předávají si informace. Čich je pro mravence nejcitlivějším smyslem. Přes to že mravenci nemají přímo sluch, dokáží zaznamenávat i zvuk. Jsou schopní detekovat vibrace z okolního prostředí, nebo tyto vibrace úmyslně vytvářet.

5.2 Orientace v prostoru a trailing

Při plnění úkolů se mravenci vzdalují od své kolonie i na větší vzdálenosti. Ovšem ne vždy za sebou nechávají cestičku,[6] která by jim pomáhala v orientaci pro zpětný návrat. Z toho vyhází potřeba znalosti polohy kolonie nebo cíle, kam se snaží mravenec dostat. Jednotliví mravenci však nemají dostatečnou kapacitu na pochopení složitější navigace. Nejsou schopni používat specifický souřadnicový systém, tak jak ho využívají lidé. S určováním polohy, ať už sebe sama či cíle, se pojí orientace v prostoru. Bez ohledu na konkrétní smysl využívaný pro orientaci, mravenci potřebují mít představu o přibližném směru a vzdálenosti cíle. Pro určování směru mravenci využívají různé způsoby. Jsou schopní vnímat magnetické pole země[5], orientovat se podle polohy slunce nebo rozdíly teploty[27] v různých místech svého prostředí. Vizuálně orientující se druhy dokáží využít významných orientačních bodů jako jsou velké kameny nebo stromy.

Mravenci však nejsou schopní svou představu o vzdálenosti použít k vytváření zkratek[26] nebo určování přesné polohy. Jejich orientace je pouze jednosměrná. Když je mravenec přesunut mimo svoji trasu, stále směřuje směrem k cílí tak, jako by pokračoval z bodu, odkud byl přesunut. Svůj cíl tímto směrem nenajde. Změnou výchozí pozice se mění i směr k jeho cíli. Mravenec pak začne bloudit a hledat. Zároveň při svém bloudění ignoruje veškeré cesty, které náleží trasám vedoucím k jiným cílům. Zorientuje se pouze v případě pokud při svém bloudění narazí na původní trasu, z níž byl přesunut jinam, a to nezávisle na konkrétním bodu této trasy.

Mravenci za sebou zanechávají v určitých situacích chemické cestičky / pachové stopy. Toto chování se označuje jako trailing. Cestičky jsou tvořené feromony (odtud také anglický výraz pheromone path). Chemická cestička postupem času vyprchává. Aby nezanikla, je nutné cestičku posilovat a obnovovat zvyšováním koncentrace feromonů v daném místě. Feromonová signalizace slouží také ke komunikaci. Stopy mohou nést specifickou informaci a ostatní mravence tak přitahovat nebo odpuzovat. Mohou být využité za účelem hodnocení lokality pro nové mraveniště, kvality potravy, podporu nebo základ prostorové orientace, i jako nástroj alokace pracovních sil. Princip zanechávání těchto stop je závislý na druhu mravenců i na individuálním chování. Mravenci mohou za sebou nechávat cestičku od opuštění kolonie a získávat tak další mravence pro prohledávání stejné oblasti, nebo naopak při úspěšném návratu od konkrétního objevu a směrovat tak další mravence k tomuto místu.

5.3 Alokace zdrojů

Zajištění dostatečného množství dělníků vykonávajících potřebný úkol je čistě otázkou komunikace mezi dělníky samotnými. Mravenci scouti jednoduše získávají podporu a pomoc rekrutováním.[19] K rekrutování dochází např. v okamžiku, kdy nějaký mravenec objeví zdroj potravy. Může probíhat různým způsobem a to v závislosti na druhu mravenců i na vykonávaném úkolu. Lze rozlišit několik základních variant rekrutování. Mravenci obvykle využívají vzájemnou kombinaci těchto způsobů včetně druhově či situačně závislých variant. Jako základní varianty rekrutování rozlišujeme rekrutování jednotlivců, skupin nebo rekrutování mas dalších mravenců. Rekrutování může probíhat aktivní i pasivní formou propagace. Za aktivní formu považujeme rekrutování, kde mravenec získávající podporu po návratu do mraveniště aktivně propaguje svůj úkol. Za pasivní variantu považujeme zanechávání cestiček na kterou můžou následovat další mravenci. Obě varianty probíhají v mnoha případech současně. Způsob rekrutování se mění i v závislosti na velikosti kolonie.[6]

Základní princip rekrutování je u všech variant stejný.[17] Rekrutující scout vysílá po návratu do mraveniště informaci o svém úkolu. Např. může ostatním sdělovat našel jsem potravu.[18] Mezi ostatními jedinci tento úkol propaguje. Scout si vymění s neaktivními mravenci informace. Pokud scout donesl nějakou potravu, dát ochutnat ostatním část z donesené kořisti. Tím pozitivně posílí zájem. Scout následně odvede mravence, jejichž zájem si získal, k cílovému místu. Tento postup je typický pro aktivní formu individuálního a skupinového rekrutování. Mezi individuálním a skupinovým rekrutováním není příliš velký rozdíl. U skupinového rekrutování se scout snaží místo pouhého jednotlivce propagovat svůj cíl mezi skupinkou. Vůdce skupiny vždy po krátkých úsecích vyčkává, než k němu dorazí alespoň část jeho následovníků.

Při skupinovém či individuálním rekrutování zanechává (posiluje) cestičku obvykle pouze vůdce skupiny nebo mravenci cestičku vůbec nevyužívají. Při masivním rekrutování posilují danou cestičku všichni mravenci kteří ji následují. Největší množství mravenců je však získáno pro daný úkol pasivní formou. Nektivní mravenci jsou při tomto chování silně při-

tahováni k feromonové cestičce a obvykle se ji rozhodnou následovat. Následně ji ještě více posílí. Takto se pro daný úkol rychle alokuje velké množství dělníků. Po splnění úkolu cestička postupně zanikne. Mravenci mající informaci o tom, že úkol byl splněn, ji již neposilují nevracejí se na ni. Mravenci v tomto případě neprohledávají prostor individuálně, ale spíše následují pachy a cestičky.[4] Masové rekrutování je typické především u kolonií nabývajících zvláště vysokého počtu jedinců. Pro velké kolonie je toto chování výhodné. Vysoký počet jedinců je schopen rychle prohledat velkou oblast.

Množství dělníků věnujících se úloze je závislé i na rychlosti, s jakou je tato úloha propagována. Mravenci nehledají optimální cesty systematicky. Nalezení optimální cesty je vedlejším důsledkem mechanismů rekrutování a zanechávání cestiček.[6] Po kratších cestičkách se mravenci stíhají rychleji vracet zpět do mraveniště. Důsledkem toho jsou schopní častěji propagovat svoji úlohu a více posílit koncentraci feromonů na využívané cestičce. Větší množství mravenců tak bude přitahováno k dané cestičce nebo přímo dovedeno do cílového prostoru. Více mravenců pak získá informace o dané úloze a jsou schopni ji dále propagovat. Individuální rekrutování má mechanismus urychlující tento proces. Když se scout domnívá, že rychlost je klíčová, neobtěžuje se s propagací cíle. Místo toho uchopí jednoho z vyčkávajících mravenců a k cíli ho maximální možnou rychlostí odnese.[13] Jakmile se stejným způsobem zachová i unesený mravenec, počty rekrutovaných mravenců se rychle násobí.

5.4 Kolektivní rozhodování

Nadpis Kolektivní rozhodování může vyvolávat představu nějakého shromáždění a následné hlasování. Mravenci ovšem žádný hlasovací systém nemají. Přesto si kolektivně vybírají mezi možnostmi které mají k dispozici. Klíčem ke kolektivnímu rozhodování je podobně jako u výše uvedené alokace zdrojů mechanismus rekrutování. Jak přesně se tento mechanismus uplatňuje v rámci kolektivního rozhodování lze ukázat na příkladech hledání potravy nebo výběru vhodné lokality pro nové sídlo kolonie.[12]

Při rekrutování pro sběr potravy scouti obvykle nesou malou část kořisti, z níž dává ochutnávat případným zájemcům. Ochutnávající mravenec je schopen subjektivně ohodnotit kvalitu této potravy a následně se rozhodnou zda se nechá přesvědčit. Tento mechanismus zajistí že scout který objevil zdroj kvalitnější potravy bude schopen snáze propagovat svůj cíl mezi více mravenců. Podobný systém hodnocení se uplatňuje i při výběru vhodné lokality pro nové sídlo kolonie. Mravenec, který místo objevil, zhodnotí subjektivním způsobem danou lokalitu.[14] Pokud se mu lokalita zamlouvá, zanechá zde feromonovou stopu přitahující další mravence. Vrátí se do kolonie, kde se pokusí lokalitu propagovat a dovést k ní další mravence. Se skupinami dalších[] rekrutovaných mravenců se pravidelně vrací na objevené místo. Při každém návratu místo zkouší hodnotit znovu a svoji reakci vyjádří zanecháním další feromonové stopy. Pokud se mu daná lokalita nezdá vhodná, zanechá zde odpuzující feromony a nebude se ho pokoušet propagovat. Není vyloučeno, že se nemůže později vrátit. Větší koncentrace pozitivně hodnotících feromonů způsobuje u mravenců vyšší úroveň nadšení. Vhodná místa jsou takto častěji a silněji propagována. Ve výsledku se všichni mravenci kolonie přesunou na nejvíc preferované místo.

Kapitola 6

Rozbor původní práce

Účelem této práce je především návrh nové varianty umělé inteligence a rozšíření pro hru Mravenci. Tvorbou této hry, jejích mechanik a vlastnostmi původní umělé inteligence se zabývá práce, na jejíchž základech tato práce staví. Aby bylo možné výše zmíněná rozšíření implementovat a navrhnout vhodným způsobem novou variantu umělé inteligence, je důležité nejdříve zjistit vlastnosti původní implementace. Některé vlastnosti by mohly znesnadňovat, v krajních případech znemožňovat, implementaci rozšíření tak jak byla zamýšlena. V takovém případě je nutné zjistit, co je nutné udělat pro to, aby bylo možné zamýšlená rozšíření implementovat. Je zároveň vhodné zvážit zda je nutné některá rozšíření implementovat. Kapitola shrnuje poznatky získané z textu a zdrojových kódů původní práce[24]. Zdrojové kódy, které se nevztahují k implementaci AI jako takové (ta je implementována pomocí jazyka Jason), jsou psané v programovacím jazyce Java. V následujícím textu se mohou objevit pojmy a výrazy odkazující se na znalosti o tomto jazyce. Podobně jako Jason a AgentSpeak, Java není předmětem této práce a pojmy s ní spojené nebudou vysvětlovány. Pro potřeby této práce je považujeme za všeobecně známé.

6.1 Překlad a spuštění hry

Pro úspěšné přeložení hry bylo nutné ošetřit ve zdrojových kódech několik výjimek v interních akcích. Konkrétně se jedná o neodchycené výjimky v několika třídách odvozených od abstraktní třídy Jasonu nazvané DefaultInternalAction. Po tomto drobném zásahu do původního kódu šlo zdrojový kód přeložit. Po spuštění hry přeložené ze získaných zdrojových kódů se objeví menu pro výběr volitelné hry (custom game), experimentů a nastavení. Nastavení jsou prázdná. Spuštění experimentů většinou končí chybou, takže pro všechna porovnání bylo nutné vycházet z výsledků uvedených v technické zprávě původní práce. Po kliknutí na volitelnou hru je možné nastavit některé parametry hry, jako je rychlost, výchozí počet mravenců a úroveň umělé inteligence pro jednotlivé týmy, herní mapu z několika málo předdefinovaných variant. Není možnost změnit počet týmů ve hře. Vždy jsou na mapě napevno umístěné čtyři týmy.

Jakmile naběhne herní prostředí hráč má k dispozici několik málo mravenců kterým přímo předává příkazy co mají udělat. V případě získávání zdrojů je mravenci sami odnášejí do mraveniště a vracejí se zpět pro další suroviny. Část zdrojů je na mapě předgenerovaná. Další zdroje se na mapě postupem času objevují. Hráč se pak stará o ekonomiku svého mraveniště, tzn. nákup vylepšení (upgradů) pro své mravence a nové přírůstky do svých řad. Cílem hry je pak zničit soupeře tím, že hráč (AI) přijde v boji o všechny své mravence,

následkem čehož je pro něho nemožné pokračovat ve hře.

6.2 Zdrojové kódy

Mít přehled o zdrojových kódech je pro implementaci rozšíření klíčové. Rozšíření by měla pouze implementovat novou funkcionalitu. Do původní implementace by nemělo být nutné nijak výrazně zasahovat. Předpokladem je, že se doimplementuje nová funkcionalita. Ta se následně přidá k té staré. Za předpokladu uplatnění funkcionálního přístupu se přidají další funkce, které se následně použijí na potřebných místech. V případě objektově orientovaného přístupu se odvodí nové třídy čímž zůstane zachována veškerá původní funkcionalita včetně svého rozhraní a není tak nutné měnit okolní kód s výjimkami míst, kde je potřeba použít i nově přidané metody. V případě objektového návrhu využívajícího rozhraní je možné snadno přidat zcela nové typy objektů implementující toto rozhraní a původní funkčnost také nebude narušena. Pokud je původní projekt implementován podle špatného objektového návrhu a architektury, není možné výhod objektově orientovaného programování a tzv. dědičnosti využít

Zdrojové kódy jsou rozdělené do dvou balíčků (packages). Balíček s názvem graphics obsahuje veškerou implementaci týkající se programové reprezentace hry. Balíček actions obsahuje třídy implementující některé interní akce pro interpret jazyka Jason. Implementace grafického rozhraní i vnitřní implementace funkcionality je tak smíchána do jednoho balíčku, což samo o sobě znesnadňuje rozlišení těchto dvou částí. Ty by měli být implementované samostatně a být na sobě vzájemně nezávislé. Samozřejmě by bylo vhodné i jemnější členění na sobě nezávislých bloků v rámci těchto dvou základních částí. Implementace nevyužívá žádných vlastních rozhraní nebo dědičnosti. Některé třídy implementují více různých funkcionalit. Např. třída nazvaná AgentInfo reprezentující jednotlivé agenty se zabývá nejen schopnostmi agentů, ale také tím jak agenta vykreslit, nebo vytvářením a vyhledáváním cest. Z názvu by se mohlo zdát že třída má sloužit především pro ukládání informací o stavu agenta. Ve výsledku však zajišťuje mnohem širší funkcionalitu. Tuto funkcionalitu by bylo vhodné oddělit do samostatných tříd svázaných pomocí vzájemných referencí nebo zapouzdřujícího objektu. Pro daný účel by se dalo využít i výše zmiňovaných rozhraní. Usnadnilo by to čitelnost a udržitelnost kódu.

Vnitřní reprezentace hry také není příliš příznivá pro rozšíření. Vyskytují se v ní další designové chyby. Pro reprezentaci zda mravenec něco nese jsou ve třídě AgentInfo dvě proměnné typu boolean reprezentující suroviny objevují ve hře, tedy jídlo a vodu. Protože mravenec může v jeden okamžik nést pouze jednu z nich, je nutné kontrolovat 2 proměnné pokaždé, když se agent pokouší vzít další surovinu. Pro získání, nebo odhození, samotné suroviny pak má agent pro každý zdroj implementované samostatné metody. Díky tomuto způsobu implementace je v případě potřeby obtížné implementovat další druh zdrojů. Bylo by nutné přidat další proměnnou, další metody, úprava okolních zdrojových kódů, které s těmito zdroji pracují, potenciálně na mnoha místech. Bylo zmíněno, že jsou ve hře přítomné vždy napevno 4 týmy a tento počet nelze měnit. Jedná se o důsledek toho, jakým způsobem jsou tyto týmy ve zdrojových kódech reprezentovány. Jde o 4 samostatné proměnné, mezi nimiž je rozlišováno pomocí podmínek. Dochází tak k částečné duplicitě kódu a tedy i k většímu potenciálu k chybám při úpravách této implementace.

Reprezentace map na tom není o moc lépe. Nějaká reprezentace mapy jako takové ve své podstatě v implementaci chybí. Mapa je tvořena několika seznamy obsahujícími herní objekty, které se na mapě mají vyskytovat (tj. překážky, suroviny a samotní agenti), a hodnotami výšky a šířky, v jejichž rozmezí lze herní objekty umisťovat. Ve chvíli, kdy je nutné

z nějakého důvodu nad herními objekty vyhledávat, jsou seznamy lineárně prohledávány, což je časově náročné (potenciálně se musí projít celý seznam). Zároveň je tímto výrazně omezen počet objektů i velikost mapy. Čím větší nároky bychom chtěli na variabilitu mapy a počtu překážek klást, tím vyšší nároky by byly kladeny na hardware nad kterým by hra běžela.

6.3 Některá řešení použitá v původní práci

Hlavní problematika, kterou původní práce oslovuje, je prohledávání prostoru a nalezení nejkratší možné cesty k danému cíli. S tím se pojí také vyhýbání se případným překážkám. Původní práce tuto problematiku spíše, dalo by se říci, "obchází". Je zde využito dopředné znalosti všech aspektů, které tuto problematiku ve dvourozměrném prostředí ovlivňují, a také toho, že prostředí je neměnné. Vertexy reprezentující všechny překážky jsou vzájemně pospojovány (který vertex patří ke které překážce). A* algoritmus pak vyhledá mezi vertexy překážek nejkratší možnou cestu a to mezi vertexy libovolných dvou překážek. Jak je i v původní práci uvedeno, tento proces je výpočetně náročný. Proto jsou všechny předem známé informace předzpracované. Ve hře jsou pomocí tzv. ray-castingu nejdříve nalezené všechny překážky mezi agentem a jeho cílem. Pomocí A* algoritmu je nalezena nejkratší možná cesta mezi těmito překážkami. Pozice agenta a jeho cíle jsou pak k této cestě připojené. Cesta je následně předána agentovi, který už pouze vykoná potřebný pohyb. Aby byl tento přístup možný, je zapotřebí znalost přesných souřadnic cíle. Výhodou je samozřejmě rychlost zpracování, kterou tento přístup poskytuje.

Jednotlivý agenti mezi sebou potřebují spolupracovat. K efektivní spolupráci je zapotřebí komunikace. Ta je realizována skrze posílání zpráv mezi agenty v rámci týmu. Posílání a zpracování těchto zpráv zajišťuje Jason. V rámci implementace se iteruje mezi jednotlivými agenty a členům týmu je zaslána zpráva skrze Jasonem předpřipravené rozhraní založené na standardu AgentSpeaku. Všichni živí agenti ze stejného týmu tak dostanou zprávu bez ohledu na to kde se momentálně nachází.

6.4 Refactoring

Výše uvedené poznatky týkající se zdrojových kódů naznačují chyby v objektovém návrhu pro implementaci hry. Byly nalezeny i některé další chyby. Je ovšem bezvýznamné všechny uvádět a rozebírat. Pro čtenáře podstatnou informací je důsledek těchto chyb. Navrhovaná rozšíření by bylo velmi obtížné, ne-li nemožné, implementovat. Určitá variabilita, jako změna počtu týmů ve hře, a rozšíření vlastností objektů či jakákoliv obdobná, i menší, změna by se neobešla bez značného zásahu do zdrojových kódů. Kód je tak prakticky neudržovatelný. Aby bylo možné předejít spoustě problémů, bylo nutné provést tzv. Refactoring.

Refactoring Refactoring je proces změny aplikačního kódu takovým způsobem, aby nedošlo ke změně jeho vnějšího chování. Účelem refactoringu je vylepšení některých na funkčnosti nezávislých vlastností kódu, jako je čitelnost, komplexita, udržovatelnost, a rozšiřitelnost. Refactoring tak usnadňuje budoucí úpravy zdrojového kódu. Zdrojem definice je http://www.techopedia.com/[2]

Refactoring je časově náročný proces. Jedná se v podstatě o reimplementaci veškerého chování aplikace, nebo její části, která je refactorována. Diky tomu si refactoring může

v důsledku vyžádat investici stejného množství zdrojů, které bylo investováno do původního vývoje. Rozšíření se do aplikace implementují, aby nebylo nutné vyvíjet aplikaci novou pouze s rozšířením chování. Pokud však nejsou rozšíření do původní aplikace implementovatelná a je nutný refactoring, je výsledný proces mnohem náročnější, než implementace samotných rozšíření nebo původní aplikace jako takové.

Kapitola 7

Navrhovaná rozšíření

Kapitola seznámí čtenáře s rozšířeními navrženými pro tuto práci. Uvedeme zde informace o tom, s jakým záměrem, byla rozšíření implementována. Pro čtenáře zajímavou informací zde budou především předpokládané změny ve výsledcích způsobené rozšířeními. Návrh byl tvořen ještě před rozborem původní práce z hlediska její implementace. Kreativita a potenciál k zajímavým výsledkům se tak staly hlavními faktory při návrhu. Refaktoring ovšem zkonzumoval značné množství z času na vypracování projektu. Z původního návrhu tak zůstala pouze 2 klíčová rozšíření a koncept swarm inteligence. Prvním a nejzajímavějším je rozšíření utvářející smyslové vnímání, které mělo značný dopad na implementaci. Druhým, skromnějším je rozšíření implementující vyčerpání agentů. Pro mravence by tak byl nutný odpočinek nebo konzumace potravy.

7.1 Dynamické prostředí

V předchozí kapitole o původní práci jsme zmiňovaly některé její nedostatky. Jedním z takových, co se vlivu na výsledek týče, je statická a diskretizovaná podopa prostředí. Původní aplikace využívá pouze několika málo předpřipravených statických map. Zdroje jsou v rámci mapy náhodně generovány, ale pro minimální počet agentů, který se ve hře obvykle vyskytuje, je těchto zdrojů velké množství rovnoměrně po celé mapě. Mravenci se tak pouze zřídka po většinu hry musí výrazně vzdálit od mraveniště a přijít tak do kontaktu se soupeři. Jako diskretizované se dají popsat některé úkony samotných agentů, zejména pak získávání zdrojů. Mravenec příjde na souřadnice, kde je umístěn blok zdrojů a bez jakéhokoliv přechodu si zdroje odnáší. Prodobně při boji mravenci musí být přítomní na stejných souřadnicích. Jedná se o vlastnosti napomáhající zjednodušení implementace. Nevyžadují ovšem detailněji rozpracovávat umělou inteligenci. Výsledky získané tímto způsobem se budou pří různých opakováních od sebe lišit minimálně.

Pro vynucení rozpracování AI nebo pro její hlubší prověření by bylo vhodné ji umístit do dynamického prostředí. První možností, jak takové prostředí získat, je toto prostředí náhodně generovat v rámci stanovených parametrů. Parametrů lze, i v této jednoduché reprezentaci, najít mnoho. Všechny jsou malého rozsahu, ale značných důsledků. Změna tvaru, počtu, či hustoty překážek spolehlivě otestuje vyhledávání trasy k cíli. Další z možností je použití hodnotových intervalů a náhodných událostí. Pokud se na mapě začnou překážky objevovat i mizet, opět to silně ovlivní hledání cest. Pokud mravenci boudou mít možnost na sebe útočit v rámci určité vzdálenosti, nemusí tak stát na stejných souřadnicích, ale mohou se při tom pohybovat. Pro získání zdrojů může být nutné vyvinout nějaké úsilí.

Soupeř tak může vynutit přerušení činnosti.

7.2 Vyčerpání agentů

Ideou tohoto rozšíření je vynucení spolupráce zvýšení nároků na rozhodovací proces agentů v oblasti podmínek splnitelnosti. Agenti by byly činností mírně vyčerpáváni. Pro doplnění energie by byl nutný odpočinek, nebo konzumace zdrojů. V případě vyčerpání nebo nízkého zdraví by byly mravenci imobilizovaní. Pokud by v takovém stavu setrvaly delší dobu, zemřeli by. Vyhnout se tomuto osudu by mohli několika způsoby. spřátelený mravenec by vyčerpanému či zraněnému donesl zdroje ke konzumaci, nebo by ho odnesl zpět do mraveniště. Ideální je samozřejmě prevence a zahrnutí hodnoty energie do rozhodovacího procesu. V souvislosti s předcházejícím rozšířením by se dali vynutit zajímavé situace například malým množstvím zdrojů na velké mapě a efektivně tak dohnat některé agenty k vyhladovění. Přidáním fuzzy logiky se tak dá nasimulovat rozhodování v krizových situacích.

7.3 Smyslové vnímání

Nejzajímavější a nejsložitější rozšíření zde uvedené. Člověk běžně využívá své smysly, intuici a přirozený intelekt pro řešení nejrůznějších situací, které se mohou zdát na první pohled velice jednoduché. Když je to tak jednoduché, naučíme to náš program. Jak těžké to může být ? Značně složité, protože zde nás naše intuice a intelekt bohužel klamou. Smysli obvykle zpracovávají velké množství vjemů, spoustu z nichž si obvykle ani neuvědomujeme.Mozek vše zpracuje a naše vědomí pouze dostane výsledek, pro snadné a rychlé použití. V případě tohoto multiagentního systému tvoří mozek našich agentů architektura Jasonu. Jaké vjemy, jejich množství a veškerou jejich informační hodnotu, kterou mohou tomuto mozku poskytnout, je plně v rukou člověka implementujícího prostředí a tyto smysly. Veškeré vjemy je nutné převést do textové podoby vhodné pro Jason. Zároveň čím více vjemů je v každém okamžiku předávat, tím náročnější zpracování těchto vjemů bude.

Příkladem náročnosti pro tuto formu interpretace je zrak. Běžně vidíme velkou spoustu předmětů a u těchto předmětů vnímáme jen za pomoci zraku ještě větší množství detailů - barvy, tvary, vzájemnou polohu atp. Pro prostředí 2D hry samozřejmě není až tak velké množství detailů nutné. I tak je ale nutné množství vjemů a detailů dále redukovat. Ztrácí se tím i schopnost vnímat souvislosti mezi různými předměty. Jako by nebyl úkol už tak náročný, je nutné tyto vjemy používat ve vzájemných souvislostech. Bez souvislostí by nebylo možné v rámci rozhodovacího procesu rekonstruovat požadované informace.

Pokud je řeč o rekonstrukci informací z jednotlivých vjemů, nesmíme opomenout smysly samotné. Pro co nejvěrnější vyjádření reality musíme dodržet i to, že jednotlivé smysly poskytují různé a na sobě nezávislé informace. Je vhodné implementovat naše agenty tak aby byly schopní identifikovat stejný pomocí různých smyslů. Naopak jedním smyslem musí být naši agenti schopní vzájemného rozlišení dvou a více objektů.

7.4 Koncept Swarm inteligence

Co je to swarm inteligence, Ant Colony Optimization algoritmus a jak se chovají skuteční mravenci bylo již popsáno v předcházejících kapitolách. Důvodem k uvedení swarm inteligence do rozšíření jsou následující. V běžných strategických hrách hráč (ať už lidský, nebo AI) ovládá své jednotky z hierarchicky vyšší pozice a to obvykle po skupinách. Swarm

inteligence založená na ACO algoritmu by jednotkám dodala na individualitě. Princip rekrutování převzatý z ACO by mohl změnit způsob jakým se (alespoň vnitřně) tvoří skupiny jednotek a za jakým účelem. ACO se obvykle využívá pro pohyb po grafech na již existující cesty mezi uzly daného grafu. V otevřeném prostředí lze nechat mravence ať si své cesty zbudují sami, podobně jako tomu je v reálném prostředí. Může se jedna o nadstavbu nad běžným path findingem, nebo o alternativní řešení tohoto problému.

Kapitola 8

Implementace

V kapitole Rozbor původní práce jsme uvedly co je to refactoring, a jaké důvody vedly k použití tohoto postupu. Výsledkem refactoringu je kompletní změna vnitřní reprezentace a přístupu aplikovaných na danou problematiku. Během procesu implementace s ovšem čerpalo i z původních idejí a tak je možné tyto ideje najít v pozměněné podobě i zde. Některé vlastnosti bylo ovšem nutné kvůli vzájemné nekompatibilitě vyloučit. Změnilo se uživatelské grafické rozhraní. To však není předmětem této práce a proto bude jeho implementace vynechána. Kapitola čtenáře informuje o specifických vlastnostech některých rozšíření a algoritmech zvolených pro vnitřní reprezentaci částí, které významně ovlivňují výsledné chování a rychlost zpracování umělé inteligence.

Ve fázi návrhu a přípravy kostry programu byl kladen velký důraz na využití vlastností a principů objektově orientovaného programování. Tam kde je to v hodné se ve velké míře uplatňují rozhraní. Rozhraní zajišťují možnost vytváření nových na předcházející implementaci zcela nezávislých tříd, jejichž vnitřní implementace se může lišit, ale jejich metody navenek vykonávají stejnou činnost. Instance těchto tříd pak lze připojit do aplikace velmi malou změnou ve zdrojovém kódu, konkrétně v místě, kde se instance vytvoří a použije. Vzniklý objekt se např. může vložit do mapy bez nutnosti měnit jakoukoliv část funkčního kódu mapy. Funkční kód je potřeba pouze rozšiřovat pro použití nové funkcionality. Součástí zdrojových kódů jsou také prototypové třídy. Jedná se třídy implementující některá rozhraní, ale nejsou samy o sobě použité pro implementaci funkcionality hry. Místo toho jsou z nich odvozené další třídy, které tuto základní funkcionalitu rozšiřují nebo pozměňují pro své specifické potřeby. Brání se tak duplicitám kódu a tím i potenciálním chybám. Výše uvedený přístup zároveň umožňuje snadné rozšíření či modifikaci funkcionality bez nutnosti zásahů do funkcionality původní.

Nová implementace byla navržena tak, aby poskytovala v maximální možné míře podporu pro zamýšlená rozšířeni. Od toho se odvíjí vnitřní implementace klíčových částí systému jako je mapa, třída implementující činnost agentů a třídy podílející se na smyslovém vnímání. Důležitá je rychlost zpracování dat i přesnost těchto klíčových částí systému. Chyby nebo neoptimálnost ve zmíněných částech mohou mít značný dopad na výkon aplikace a výsledky.

8.1 Interní reprezentace prostředí

Reprezentaci prostředí ve své podstatě tvoří herní mapa. Součástí tohoto prostředí jsou všechny objekty které se ve hře vyskytují a to včetně samotných agentů. Agenti potřebují

mít přehled o svém nejbližším okolí. Toto okolí potřebují agenti pravidelně kontrolovat v co možná nejkratších intervalech, aby byly schopni v co nejkratším čase reagovat na změny v tomto okolí. Předpokládáme větší množství agentů pohybujících se na mapě především kvůli swarm inteligenci, která většího množství agentů využívá. Z mapy se tak stává velice frekventovaný bottleneck (úzké hrdlo) celé aplikace. Z tohoto důvodu je třeba mapu výkonově optimalizovat. Do mapy jsou vkládány všechny objekty, které se ve hře vyskytují tj. agenti, překážky, zdroje, feromonové cestičky případně další.

Každý objekt má svůj referenční bod a tvar. Pro většinu objektů je referenčním bodem střed objektu, u některých překážek to ovšem nemusí platit. Tvar objekt je reprezentován vertexy. Protože by nešlo snadno určit z pouhých vertexů, zda bod spadá dovnitř nebo vně objektu (bylo by to výpočetně náročné), každý objekt tak má tzv. bounding box a hrany mezi jednotlivými vertexy. Bounding box je nejmenší možný obdélník, který je schopen zapouzdřit objekt. Hrany se využívají pro detekci kolizí. Vizuální emitor využívá jako jediný tento originální bounding box. Ostatní emitory mají přiřazený vlastní obdélník vyhrazující oblast, v jejímž rámci je objekt, jemuž je emitor přiřazen, detekovatelný pomocí spárovaného smyslu.

8.2 Mapa

Na mapu jsou kladeny následující požadavky. Mapa bude pro statické objekty sloužit především ke čtení, jinými slovy, většina objektů se vygeneruje při vzniku mapy a počet operací vyhledávání v mapě bude značně převyšovat počet operací přidávání, či odebírání objektů. Pro dynamické objekty budou počty zápisů i čtení srovnatelné, vzhledem k nutnosti neustálého pohybu těchto objektů a tím i jejich přesunu v mapě. Zároveň musí být u dynamických objektů v každém kroku zjistitelné, zda se nacházejí v dosahu smyslů agentů. Je důležité, aby nedocházelo ke vzájemnému blokování různých částí (vláken) aplikace, které potřebují k mapě asynchronně přistupovat, a zároveň, aby nedocházelo k datovým nekonzistencím.

Pro splnění výše uvedených požadavků s výhodou využíváme následujících faktorů. Pro vykreslování i pro potřeby agentů je nutné zjistit jaké objekty se nachází v dané oblasti. Objekty jsou detekovatelné na různé vzdálenosti a to v závislosti na konkrétním páru smysl-emitor, čímž se na straně smyslu, mění i prohledávaná oblast. pohybující se objekty neustále mění svoji pozici a tím pádem je nelze podle této pozice hledat. Lze ovšem určit zda se objekt nachází v rámci vymezené oblasti. Bylo by velice obtížné všem těmto faktorům přizpůsobovat mapu. Využíváme tak několika různých map, které jsou optimalizované pro konkrétní účel. Mapa je reprezentována stromovou strukturou pomocí R-tree algoritmu (viz. dále). Pro statické objekty a každý pár smysl-emitor je vytvořena samostatná mapa. Do této mapy se vkládá na základě svého bounding boxu emitor objektu. Detekční oblast smyslu se pak využívá pro vyhledávání. Pro pohyblivé objekty, tj. agenty, existuje samostatná mapa. Do této mapy se ovšem nevkládají přímo agenti, ale malé oblasti nazvané chunky, pokrývající celou mapu. Smysl chanků je zabránit neustálému přeskládávání stromové struktury při pohybu agentů, protože se jedná o výpočetně velmi náročnou operaci. Agenti jsou přiřazeni do konkrétního chunku, kde se mohou volně pohybovat bez nutnosti provádět jiné operace než změnu jejich souřadnic. Při vyhledávání se pak předávají všechny objekty pohybující se v rámci chunků, které se překrývají s prohledávanou oblastí. Filtrování objektů pak provádí implementace smyslového vnímáni.

8.3 R-tree a R*-tree algoritmy

R-tree[15] je abstraktní stromová datová struktura vhodná pro uchovávání a indexování prostorových dat, např. geografické údaje. Zároveň se takto označuje algoritmus pro vyhledávání nad touto datovou strukturou. Algoritmus je implementovatelný pro n-rozměrný prostor. S přibývajícím počtem dimenzí narůstá jeho výpočetní náročnost. Pro potřeby této práce využíváme dvourozměrnou implementaci - agenti se pohybují ve dvourozměrném prostoru. Obvykle se R-tee využívá se v implementacích databází, optimalizovaných pro čtení. Jeho výhodou je totiž vysoká rychlost. R*-tree[7] je optimalizovaná varianta R-tree algorimtu, která měla být pro implementaci mapy původně využita. Její implementace se však projevila jako náročná. Dodatečné optimalizace by také zapříčinily snížení výkonu pro operace zápisu a odstraňování záznamů. Pro potřeby této práce však není navýšení přesnosti pro vyhledávání příliš přínosné a nevyvažuje snížení výkonu pro zápis.

Principem R-tree algoritmu je seskupování prostorových objektů na základě jejich umístění. R-tree algoritmus využívá vyvažovaného stromu. To znamená, že všechny listové uzly jsou na stejné úrovni. Při vkládání nového prvku se ve stromu vyhledá listový uzel pokrývající co největší část z oblasti, kterou zabírá vkládaný prvek. Vyhledávaní pak probíhá na základě překrývajících se oblastí. Pokud se bounding box zapouzdřujícího uzlu neprotíná s bounding boxem prohledávané oblasti, nemůže protínat ani s bounding boxem některého ze zapouzdřených objektů.

8.4 Smysly a emitory

Smysly a emitory jsou implementovány párově. Každý emitor má disperzní oblast, reprezentovanou bounding boxem, a sílu s jakou je propagován. Bounding box slouží pro účely vyhledávání. Síla pak určuje zda bude emitor detekovatelný, případně na jakou vzdálenost, v závislosti na implementaci konkrétního smyslu. Obdobným způsobem jsou implementovány samotné smysly. Každý smysl má boundig box, reprezentující detekční oblast a sílu detekce. U smyslů je velikost detekční oblasti odvozena od síly. Při ověřování detekce jsou pak porovnávány vzdálenosti mezi vnímajícím a vnímaným objektem spolu se silou detektoru a emitoru. Konkrétní porovnání je implementačně závislé. Pro snížení nároků na výkon si smysli udržují cache objektů, které byly detekovány ze statické mapy.

Při prohledávání okolí se z objektu agenta získá seznam smyslů, kterými agent disponuje. Na základě detekční oblasti smyslů se vyhledávají emitory v příslušných statický mapách. Využívá se překrytí bounding boxu emitoru a detekční oblasti. Smysly nad získanými emitory provádí filtrování, zda je objekt skutečně detekován. Box emitorů na dynamické mapě může přesahovat chunk, ale mimo svůj chunk není emitor znám. V takovém případě nemůže být detekován, pokud se jeho chunk neobjeví uvnitř detekční oblasti, ačkoliv by detekován být měl. Řešením tohoto problému je vybavit smysly dynamickým boundig boxem, jehož velikost je větší než velikost běžné detekční oblasti. Vyhledají se tak i chunky přesahující detekční oblast a objekt vlastnící hledaný emitor může být nalezen. Následně probíhá filtrování podobně jako u statických map.

Protože po vzoru swarm inteligence je předávání informací pouze lokální, nevyužíváme pro většinu případů Jasonovské rozhraní pro předávání zpráv, ale opět páru smysl-emitor. Agenti tak nemají k dispozici vysílačku s nastaveným soukromím kanálem a libovolným dosahem. Odpadá nutnost zpracování zpráv interpretem Jasonu. Některé činnosti více odpovídají reálnému prostředí - dané informaci si mohou všimnout všichni v daném okolí.

8.5 Strategie pro swarm inteligenci

Strategie pro swarm inteligenci je založena na konceptu, jak je nastíněn u reálných mravenců. Základem je využití různých rolí podle činnosti kterou agent v danou chvíli provádí. Tyto role se mění podle potřeby využitím principu rekrutování. Ze swarm inteligence jako takové je převzatý princip na podmínkách založeného chování. Agent se snaží mít současně splněných několik jednoduchých základních podmínek. Bohužel v případě swarm inteligence jako takové zůstalo u pouhého konceptu. Více o tomto problému v závěrečné kapitole.

Kapitola 9

Diskuse a závěr

Původním účelem vypracování tohoto projektu měla byt jednoduchá implementace rozšíření a nové varianty umělé inteligence pro hru Mravenci vytvořené v rámci jiné bakalářské práce. Po detailnějším přezkoumání zdrojových kódů však bylo zjištěno, jejich libovolná úprava by byla velmi náročná a navrhovaná rozšíření by ve většině případů nebylo možné implementovat vůbec. Došlo tak ke kompletní reimplementaci celého projektu, což nebylo zpočátku předpokládáno. V rámci vypracování projetu tak bylo nutné řešit i problémy s prací nesouvisející a kvalita práce je tím výrazně ovlivněna.

9.1 Známé nedostatky

Práce prošla náročným refactoringem. Interní reprezentace všech součástí je díky tomu detailně propracovaná a snadno rozšířitelná bez nutnosti zásahů do existujícího kódu. Po stránce prezentovatelnosti však projekt značně utrpěl. Všechny části nutné pro správné fungování aplikace jsou implementovány, avšak na některých místech je propojující logika neúplná, či zcela chybí. Pokud jde o chybějící části, jedná se vždy jen o pár řádků kódu, ale díky tomu je pro spojení funkcionality nutné zasahovat do tohoto propojovacího kódu.

Kvůli technickým problémům se nezdařilo plně funkční propojení interpretu Jasonu a prostředím, které tento projekt tvoří. Přes veškeré úsilí se nepodařilo odhalit chybu, kvůli které jsou agenti neaktivní a neprobíhá u nich rozhodovací proces. I přes volání metody, která by měla podle dokumentace tento proces spouštět, zůstává agent nečinný. Pokud se agenti přidají do systému zvenčí, např. pomocí doplňku Jason do editoru jEdit, který je distribuovaný společně s interpretem, u agentů probíhá volání enviromentálních akcí v pořádku. Na druhou stranu jim chybí jejich protějšek umístěný do prostředí, který by tyto akce vykonával.

Bez možnosti vytvořit umělou inteligenci pro implementované prostředí a tím i jejího testování zůstala swarm inteligence na úrovni konceptu. Z tohoto důvodu není možné provést ani experimenty. Lze testovat pouze samotné prostředí. Kvůli častým změnám a úpravám je kód nedostatečně zdokumentován.

9.2 Diskuse

Navzdory snaze plně dokončit projekt v pořádku a včas se nepodařilo naplnit všechna očekávání stanovená na jeho počátku. Během vypracování bylo nutné zabývat se problematikou i z oblastí, ne které nebyla tato práce zaměřená. Výsledné vypracování trpí značnými a

nepříjemnými nedostatky. Nepodařilo se dosáhnout stanoveného cíle implementovat plně funkční hru, ale i tak má tato práce svoji hodnotu. Zdrojové kódy projektu jsou bohužel špatně dokumentované kvůli neustálým změnám některých řešení. Přesto jsou použitelné a lze je považovat za framework pro další práci. Tento projekt řeší implementačně velmi náročnou problematiku. Navíc v důsledku refaktoringu bylo nutné řešit a reimplementovat věci jako grafické rozhraní. Sloučilo se tak vypracování dvou obtížných projektů, čímž výsledná náročnost značně narostla. Přes neúspěch projekt dotáhnou do úplného konce, po implementační stránce, může poskytnou vhodný odrazový můstek po stránce teoretické.

Literatura

[1] Ant.
URL http://animals.howstuffworks.com/insects/ant-info3.htm

[2] Refactoring.
URL http://www.techopedia.com/definition/3865/refactoring

[3] Artificial Intelligence in Games. 2013.
URL http://www.codeproject.com/Articles/14840/Artificial-Intelligence-in-Games

[4] Aron, S.; Beckers, R.; Deneubourg, J.-L.; aj.: Memory and chemical communication in the orientation of two mass-recruiting ant species. *Insectes Sociaux*, ročník 40, č. 4, 1993: s. 369–380.

[5] Banks, A. N.; Srygley, R. B.: Orientation by magnetic field in leaf-cutter ants, Atta colombica (Hymenoptera: Formicidae). *Ethology*, ročník 109, č. 10, 2003: s. 835–846.

[6] Beckers, R.; Goss, S.; Deneubourg, J.-L.; aj.: Colony size, communication and ant foraging strategy. *Psyche: A Journal of Entomology*, ročník 96, č. 3-4, 1989: s. 239–256.

[7] Beckmann, N.; Kriegel, H.-P.; Schneider, R.; aj.: *The R*-tree: an efficient and robust access method for points and rectangles*, ročník 19. ACM, 1990.

[8] Bordini, R. H.; Hübner, J. F.; Wooldridge, M.: *Programming multi-agent systems in AgentSpeak using Jason*, ročník 8. John Wiley & Sons, 2007.

[9] Buhl, J.; Sumpter, D. J.; Couzin, I. D.; aj.: From disorder to order in marching locusts. *Science*, ročník 312, č. 5778, 2006: s. 1402–1406.

[10] Dorigo, M.; Birattari, M.; Stutzle, T.: Ant colony optimization. *Computational Intelligence Magazine, IEEE*, ročník 1, č. 4, 2006: s. 28–39.

[11] Ducatelle, F.; Di Caro, G. A.; Gambardella, L. M.: Principles and applications of swarm intelligence for adaptive routing in telecommunications networks. *Swarm Intelligence*, ročník 4, č. 3, 2010: s. 173–198.

[12] Edelstein-Keshet, L.; Watmough, J.; Ermentrout, G. B.: Trail following in ants: individual properties determine population behaviour. *Behavioral Ecology and Sociobiology*, ročník 36, č. 2, 1995: s. 119–133.

[13] Franks, N. R.; Hardcastle, K. A.; Collins, S.; aj.: Can ant colonies choose a far-and-away better nest over an in-the-way poor one? *Animal Behaviour*, ročník 76, č. 2, 2008: s. 323–334.

[14] Franks, N. R.; Mallon, E. B.; Bray, H. E.; aj.: Strategies for choosing between alternatives with different attributes: exemplified by house-hunting ants. *Animal behaviour*, ročník 65, č. 1, 2003: s. 215–223.

[15] Guttman, A.: *R-trees: a dynamic index structure for spatial searching*, ročník 14. ACM, 1984.

[16] Kehoe, D.: Designing Artificial Intelligence for Games. 2009.
URL https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1

[17] Nicolis, S. C.; Deneubourg, J.-L.: Emerging patterns and food recruitment in ants: an analytical study. *Journal of Theoretical Biology*, ročník 198, č. 4, 1999: s. 575–592.

[18] Prabhakar, B.; Dektar, K. N.; Gordon, D. M.: The regulation of ant colony foraging activity without spatial information. *PLoS computational biology*, ročník 8, č. 8, 2012: str. e1002670.

[19] Pratt, S. C.; Mallon, E. B.; Sumpter, D. J.; aj.: Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant Leptothorax albipennis. *Behavioral Ecology and Sociobiology*, ročník 52, č. 2, 2002: s. 117–127.

[20] Robinson, E. J.; Feinerman, O.; Franks, N. R.: How collective comparisons emerge without individual comparisons of the options. *Proceedings of the Royal Society of London B: Biological Sciences*, ročník 281, č. 1787, 2014: str. 20140737.

[21] Robinson, E. J.; Jackson, D. E.; Holcombe, M.; aj.: Insect communication: 'no entry' signal in ant foraging. *Nature*, ročník 438, č. 7067, 2005: s. 442–442.

[22] Sachan, J.: A Combination of Video Games and Artificial Intelligence. *complexity*, ročník 1, č. 1, 2013.

[23] Sasaki, T.; Granovskiy, B.; Mann, R. P.; aj.: Ant colonies outperform individuals when a sensory discrimination task is difficult but not when it is easy. *Proceedings of the National Academy of Sciences*, ročník 110, č. 34, 2013: s. 13769–13773.

[24] Šimetka, V.: MULTI-AGENT STRATEGY GAME OVER ANTS.

[25] Von Mammen, S.; Phillips, D.; Davison, T.; aj.: Swarm-based computational development. In *Morphogenetic Engineering*, Springer, 2012, s. 473–499.

[26] Wehner, R.; Boyer, M.; Loertscher, F.; aj.: Ant navigation: one-way routes rather than maps. *Current Biology*, ročník 16, č. 1, 2006: s. 75–79.

[27] Wolf, H.; Wehner, R.: Pinpointing food sources: olfactory and anemotactic orientation in desert ants, Cataglyphis fortis. *Journal of Experimental Biology*, ročník 203, č. 5, 2000: s. 857–868.

[28] Zbořil, F.: Základy umělé inteligence IZU. 2012.

Kapitola 10

Obsah CD

Součástí práce je CD, které obsahuje následující položky:

- Zdrojové kódy původní práce
- Zdrojové kódy této práce
- Elektronická verze této práce ve formátu pdf