VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SIMULÁTOR UMĚLÉHO ŽIVOTA

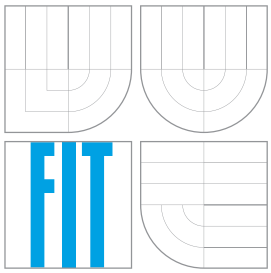
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MILOŠ POPEK

BRNO 2007

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

SIMULÁTOR UMĚLÉHO ŽIVOTA
ARTIFICIAL LIFE SIMULATOR

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MILOŠ POPEK
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID MARTINEK
SUPERVISOR

BRNO 2007

Abstrakt

Oblast umělého života je velmi rozsáhlá. Tato práce je zaměřena na simulace s reaktivními agenty. Je popsán návrh simulátoru umělého života a jazyka pro definování chování agenta. Jsou popsány návrhy a výsledky provedených simulací.

Klíčová slova

simulace, umělý život, multiagentní systém, celulární automat, emergence, samoorganizace

Abstract

Area of artificial life is very extensive. Simulations with reactive agents are the target of this work. Simulator of artificial life and language for definition agent's behavior are described in this work. Porposals a results of simulation made are described in this work.

Keywords

simulation, artificial life, multi-agent system, cellular automaton, emergence, self-organization

Citace

Miloš Popek: Simulátor umělého života, bakalářská práce, Brno, FIT VUT v Brně, 2007

Simulátor umělého života

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Davida Martinka.

.....
Miloš Popek
15.5.2007

Poděkování

Děkuji Ing. Davidovi Martinkovi za odborné vedení bakalářské práce, za poskytování rad a materiálů.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá návrhem a vytvořením simulátoru umělého života. Simulátor by měl obsahovat prostředí, do kterého lze vkládat mravence s definovaným chováním a další objekty (zdroj potravy, zeď). Je vytvořen jazyk pro implementaci chování mravenců.

Motivací této práce je vytvoření simulačního prostředí vhodného pro simulace s jednoduchými modely umělého života. Simulační prostředí včetně jazyka pro popis chování mravenců může sloužit k výukovým účelům. Cílem je vytvoření chování mravenců, které umožňuje pozorovat při simulacích projevy emergentního chování nebo samoorganizace.

Simulace umělého života zahrnují širokou problematiku. V kapitole umělý život jsou popsány principy, prostředky a cíle simulací umělého života. Jsou zde zmíněny základní pojmy vztahující se k umělému životu. V textu je kladena vetší pozornost na simulace umělého života s nižšími formami života jako je hmyz. Kapitola návrh řešení obsahuje návrh simulátoru umělého života. Je zde popsán návrh prostředí, agenta a jazyka, kterým lze definovat chování agenta. Při navrhování těchto částí se vychází z chování skutečných mravenců. Cílem návrhu je vytvořit simulátor mraveniště, který bude umožňovat simulaci hlavních rysů skutečných mravenců. Je ale třeba zachování jisté obecnosti i pro další druhy simulací. V kapitole implementace je popsána implementace jednotlivých částí simulátoru. V této části se nachází popis syntaxe jazyka pro implementaci chování agenta a seznam příkazů s poznámkami o jejich funkčnosti. Závěrečná kapitola obsahuje popis, parametry a zhodnocení simulací, které jsem v simulátoru provedl. Dále jsou zde zmíněny návrhy na možná rozšíření a další vývoj tohoto projektu.

Kapitola 2

Umělý život

Umělým životem rozumíme počítačem simulované jedince či celé populace životních forem v umělém prostředí. Cílem vytváření modelů inspirovaných skutečnými živočichy včetně jejich životního prostředí je snaha o pochopení a získání nových znalostí o zkoumaných systémech. Zkoumány jsou přitom nejrůznější formy života od bakterií počínaje, přes hmyz v podobě mravenců či včel, až po lidskou společnost.

V souvislosti s pojmem umělý život se setkáváme s pojmem umělá inteligence. Cílem obou oborů je snaha o vytvoření inteligentního chování strojů. Rozdílný je přístup k dosažení takového chování. Cílem umělé inteligence je vytvořit komplexní chování podobné lidskému rozhodování. Naopak snahou při simulaci umělého života je vytvořit relativně primitivní chování jednotlivců a jejich vzájemnými interakcemi dosáhnout vynoření inteligence - princip emergence.

Obecně platí, že se při realizaci inteligence umělého života necháváme inspirovat hlavně přírodními fenomény a vycházíme z poznatků psychologie, biologie, sociologie a etologie (srovnávací výzkum chování živočichů).Klasická umělá inteligence je oproti tomu převážně založená na logice, lingvistice a racionalitě[13].

Pro modely živočichů používáme termín agent, který zapouzdřuje sadu vlastností a dovedností relevantních pro simulaci. Stejně tak model prostředí, který je v mé práci reprezentován pomocí celulárního automatu, obsahuje pouze potřebné části a spolu s agenty tvoří ucelený agentní systém.

Existuje mnoho druhů agentů. V této práci se zaměřuji na nejjednodušší z nich a to na reaktivní agenty. Při tvorbě těchto agentů se necháváme inspirovat nižšími formami života, jako je například hmyz. Při simulacích s těmito agenty se snažíme navodit efekt rojové inteligence a pozorujeme procesy jako jsou samoorganizace a emergence globálního chování.

2.1 Umělá inteligence (UI)

Pojem umělá inteligence poprvé použil John McCarthy[4]. Nazval jím vědu zabývající se tvorbou inteligentních strojů. Snahou umělé inteligence je vytvořit stroje vykazující chování na určitém stupni inteligence (ve smyslu lidské inteligence). Podle míry inteligence můžeme dělit na:

- slabou UI - projde Turingovým testem a vykazuje známky inteligentního jednání
- silnou UI - nejen vykazuje inteligenci, ale obstojí i argumentu čínského pokoje, čili skutečně rozumí tomu, co dělá

Inteligenci můžeme definovat jako schopnost uchovávat si model skutečného světa, schopnost plánování a předvídání budoucích kroků. Mírou inteligence jsou parametry komplexnosti a přesnosti daného rozhodnutí v čase. Výsledný algoritmus evokující inteligentní chování je posloupností logických kroků a při jeho tvorbě se výhradně používá metodologie návrhu shora dolů.

Jedním z nejznámějších případů použiti UI jsou šachové simulátory. Nejslavnější z nich, Deep Blue[6], dokázal v květnu 1997 jako první počítačový systém porazit úřadujícího mistra světa v šachu Garryho Kasparova. Deep Blue je ukázkovým příkladem slabé umělé inteligence. Jeho algoritmus výpočtu dalšího tahu je založen na tzv. "brutální síle", kdy prohledává všechny možnosti do hloubky několika tahů (ve většině případů mezi 6 a 12, výjimečně 40). S každým krokem do hloubky narůstá hráčská síla vyjádřená hodnotou Elo průměrně o 80 bodů. Pouze čtyři lidé v historii dosáhli hodnoty Ela přes 2 800 (Deep Blue přibližně až 3 200).

Některé argumentace tvrdí, že programy se slabou UI nemohou být nazývány inteligentními, protože ve skutečnosti nepřemýšlí. Touto problematikou se zabývá ve svém článku Drew McDermott, který v něm píše: "Saying Deep Blue doesn't really think about chess is like saying an airplane doesn't really fly because it doesn't flap its wings."[8]

2.2 Emergence

Na rozdíl od umělé inteligence se při tvorbě modelů umělého života používá metodologie návrhu zdola nahoru. Vytvářejí se jednodušší entity s elementárním chováním. Definují se jejich vzájemné interakce, ale i interakce s okolím. Právě tyto interakce, které jsou v lokálním měřítku zanedbatelné, v globálním měřítku evokují určitý stupeň inteligentního chovaní - princip emergence, kdy z jednoduchého chování jednotlivce a vzájemných interakcí s ostatními jedinci a s okolím vznikají dovednosti a poznatky vyššího řádu.

Příkladem může být mravenec nesoucí potravu do mraveniště, který za sebou zanechává feromonovou stopu. Ostatní mravenci následují feromonovou stopu při hledání potravy. Mravenec, který se vracel do mraveniště nejkratší cestou, se vrátil jako první. Jeho stopa je nejintenzivnější a ostatní ji následují. Tímto způsobem dokáží najít nejkratší cestu od mraveniště k potravě, aniž by to kterýkoliv jedinec tušil. Mezi základní charakteristické znaky emergence pak patří[7]:

- inovace - v systému se objevují nové skutečnosti
- soudržnost a soulad - celek funguje na principu samoorganizace
- globální úroveň - některé znaky jsou charakteristické pouze pro celek jako takový
- dynamičnost - systém se vyvíjí
- pozorovatelnost - dá se sledovat

Emergenci lze kategorizovat na:

- slabou - efektu emergence lze dosáhnout i pomocí jedince (například Langtonův mravenec)
- silnou - efektu emergence lze dosáhnout pouze spoluprací v celku. Celek je více než všechny jeho součásti (odpovídá výše zmíněnému příkladu)

2.3 Samoorganizace

Samoorganizace je proces, při kterém jsou jednotlivé části systému spojovány do komplexnějšího celku bez jakéhokoliv vedení. Nejvíce příkladů systémů založených na samoorganizaci pochází z vědních oborů jako jsou fyzika nebo chemie (struktura a složení látek).

Pojem samoorganizace je velmi úzce spjat s emergencí a má s ní některé rysy podobné. Přesto může existovat systém se samoorganizací, který nevykazuje známky emergence a naopak. Mezi základní principy patří[11]:

- pozitivní zpětná vazba - má na systém explozivní účinek a způsobuje kumulování příčiny
- negativní zpětná vazba - působí proti změně, která ji vyvolala, a tím reguluje stav systému
- neustávání fluktuací - náhodné jevy pomáhají systému neustat v hledání globálního nejlepšího řešení
- mnohonásobné vzájemné interakce

2.4 Rojová inteligence

Rojová inteligence je pojem, se kterým se setkáváme při simulacích kolektivního jednání v decentralizovaném samoorganizačním systému[12]. Takový systém je složen z populace agentů s jednoduchým deterministickým chováním, kteří interagují se svým okolím, ať už to jsou ostatní agenti nebo prostředí. Přestože se jedná o systém bez centrálního řízení, tak na základě lokálních interakcí vzniká efekt domnělého centrálního řízení a dochází k emergenci globálního chování. K dosažení těchto rysů je třeba, aby systém obsahoval určitý počet jedinců, a došlo tak k vytvoření tzv. roje. V souvislosti s projevy inteligentního chování takového systému hovoříme o inteligenci roje.

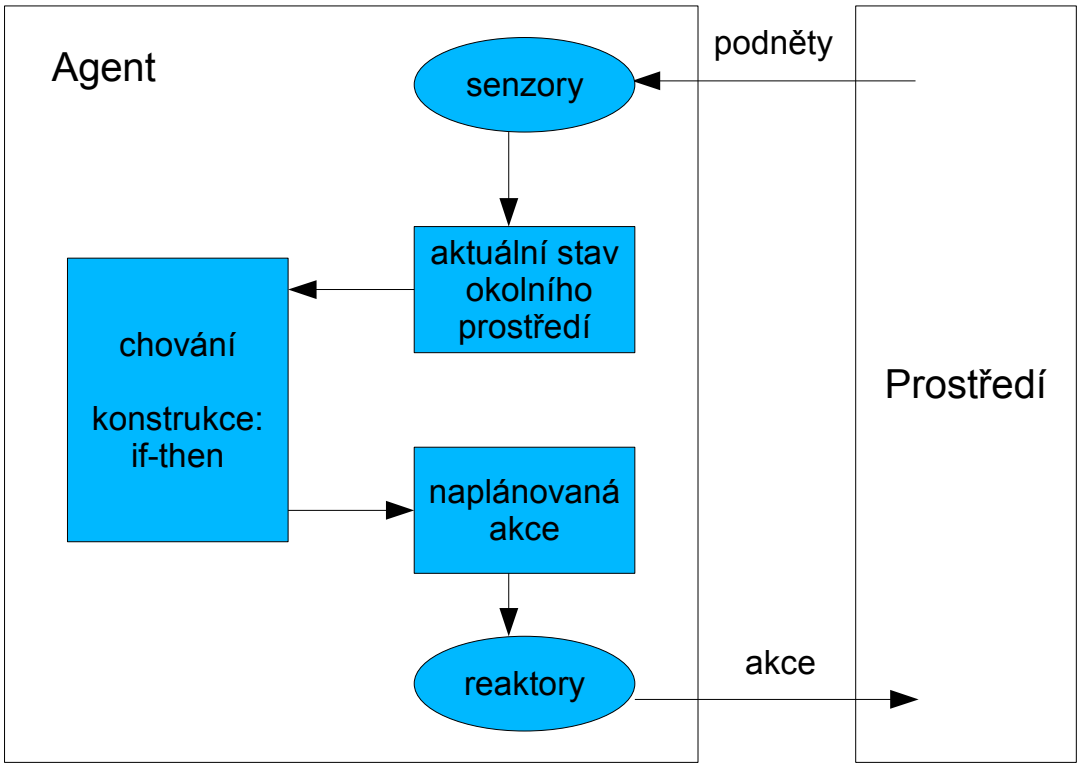
2.5 Agent

Umělý agent je autonomní entita situovaná v určitém prostředí[15]. Na základě podnětů z prostředí, které vnímá svými senzory, rozhoduje o svém budoucím jednání. Na druhé straně dokáže své okolí ovlivňovat svými efektory, a tím prosazovat uspokojení svých potřeb. Mezi základní vlastnosti agenta patří:

- autonomnost - má plnou kontrolu nad svým jednáním v prostředí
- reaktivnost - je schopen reagovat na stav okolí
- proaktivnost - ovlivňuje okolní prostředí za účelem uspokojení jeho potřeb
- sociální schopnost - schopnost spolupráce s ostatními agenty
- persistence - jeho kód není prováděn na požadavek, ale běží neustále, a sám se rozhoduje, zda vykoná nějakou akci

2.6 Reaktivní agent

Jedná se o model agenta, který se používá při simulacích umělého života inspirovaných nižšími formami života. Takový agent nepotřebuje uchovávat model prostředí, ve kterém se nachází, ale reaguje na aktuální stav světa kolem sebe. Okolí vnímá svými senzory, které bývají úzce spojeny s jeho reaktory, pomocí nichž ovlivňuje své okolí za účelem dosažení svých cílů (obr. 2.1).



Obrázek 2.1: reaktivní agent

2.7 Značkování

Komunikace mezi agenty je dvojího druhu:

- přímá - fyzickým nebo vizuálním kontaktem
- nepřímá - jeden agent přemění své okolí a druhý pozoruje dané změny

Příkladem nepřímé komunikace je feromonová stopa u mravenců. Mravenci dokáží rozlišovat nejen jednotlivé druhy stop (až 20 druhů: jídlo, nebezpečí atd.), ale i intenzitu jednotlivých stop.

2.8 Simulační prostředí

Simulační prostředí, ve kterém se nacházejí agenti, je odrazem skutečného světa, přičemž si z něj bere pouze relevantní rysy pro simulaci. Definuje pravidla pro veškeré děje v něm: fyzikální zákony, interakce agentů s okolím i mezi agenty navzájem. Prostředí lze klasifikovat na[15]:

- spojité / diskrétní - stavy prostředí tvoří spojitou / diskrétní množinu. Počítače jsou diskrétní systém - umělí agenti vnímají prostředí jako diskrétní
- přístupné / nepřístupné - agent svými senzory vnímá celé / část prostředí, ve kterém se nachází
- statické / dynamické - ve statickém prostředí, na rozdíl od dynamického, se jeho stav nemění, pokud agent nejedná
- deterministické / nedeterministické - v deterministickém prostředí je jeho následující stav dán aktuálním stavem nebo případnou akcí prováděnou agentem
- strategické (MAS) - více agentů jedná v prostředí současně

2.9 Multiagentní systém

Multiagentní systém se skládá z prostředí a určitého počtu agentů, kteří se vzájemnými interakcemi a spoluprací snaží dosáhnout cílů, kterých by jako jednotlivci nebyli schopni dosáhnout. Jednání agentů v systému nikdo neřídí, jedná se tedy o samoorganizační systém. Multiagentní systémy můžeme dělit podle vědních disciplín, ze kterých vycházejí[15]:

- sociologie - zkoumání zákonitostí lidských společenstev. Z této oblasti vychází systémy na modelu AGR (Agent-Group-Role)
- psychologie - zkoumání lidského vědomí. Systém SOAR, který je založen na výsledcích úsilí jednoho ze zakladatelů oboru umělé inteligence Allena Newella (UTC, Unified Theory of Cognition).
- biologie - zkoumání chování u nižších forem života (např. hmyz). Realizace reaktivních agentů, jejichž chování je založeno na podmětech z okolí. Zkoumání rojové inteligence.
- filosofie - zkoumání záměrů počínání. Systém s agenty, kteří mají své chování řízeno svými záměry (BDI systém, Belief-Desire-Intention)

2.10 Celulární automaty

Celulární automat je dynamický systém, který je diskrétní v prostoru i čase[5]. Skládá se z nekonečného počtu buněk uspořádaných v pravidelné N-rozměrné (nejčastěji N=2) mřížce. Každá buňka je v jednom z konečného počtu stavů. Hodnoty stavů buněk se vypočítávají paralelně pomocí funkce, která pro svůj výpočet bere v úvahu nejbližší okolí dané buňky. Okolí buňky může být:

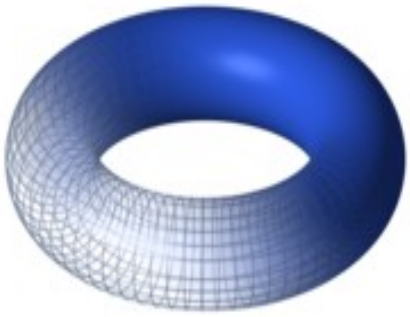
- Neumannovské okolí - čtyři přilehlé buňky

- úplné okolí - čtyři přilehlé buňky a další čtyři buňky, které se vyšetřované buňky dotýkají v rozích

Často se místo nekonečného počtu buněk používá prstencový model (obr. 2.2) s konečným počtem buněk. Mapa má konečné okraje, a když při pohybu nahoru narazí na horní okraj mapy, pak pohyb pokračuje na odpovídající buňce na dolním okraji mapy. Obdobný je i pohyb do stran. Nejznámějším příkladem celulárního automatu je "Game of Life"[14] z roku 1970, jehož autorem je matematik John Conway. Každá buňka v tomto automatu má úplné okolí a může být v jednom ze dvou stavů - mrtvá / živá. Platí tři jednoduchá pravidla:

- jestliže má živá buňka méně než dvě sousední buňky živé, pak umírá na opuštěnost
- jestliže má živá buňka více než tři sousední buňky živé, pak umírá na přeplnění
- jestliže má mrtvá buňka právě tři sousední buňky živé, pak ožije

Zajímavost tohoto automatu spočívá v možnosti pozorování samoorganizace a emergentního chování. Jedná se o jednoduchá pravidla, na jejichž základě vznikají různé druhy objektů a vzorců chování.



Obrázek 2.2: prstencový model

2.11 Použití simulací umělého života v praxi

Důvody pro nasazování umělého života na řešení úloh reálného světa jsou stejné jako u ostatních simulací[2]:

- Finance jsou důležitým faktorem při tvorbě simulací a to z několika důvodů. Pokusy s reálným systémem mohou být velmi drahé. Další důvod je využití výsledků simulace ke zhodnocení investic - příkladem je model burzovního trhu, kde se jednotliví agenti snaží dosáhnout nejlepších výdělků. Vzájemným obchodováním dochází k simulaci vývoje cen jako na reálném trhu.
- Rychlost experimentování je další výhodou simulací před reálnými systémy. Rozdíl v čase mezi získanými výsledky ze simulace a ze skutečnosti je asi nejvíce citelný u genetických algoritmů. Během okamžiku můžeme získat na základě simulace genofond N-té generace, na který bychom jinak museli čekat v řádech desítek let.

- V některých případech je simulace jediná možnost jak zkoumat daný problém. Například neexistuje matematický vzorec, který lze na problém aplikovat.

Jedním z prvních algoritmů umělého života je tzv. Langtonův mravenec[9]. Tento algoritmus má dvě jednoduchá pravidla pro pohyb mravence na mapě:

- Je-li na černém poli: otočí se doprava, přebarví pole na bílé a pohne se o jedno pole dopředu
- Je-li na bílém poli: otočí se doleva, přebarví pole na černé a pohne se o jedno pole dopředu

Jedná se o velmi jednoduché chování, které po několika chaotických krocích začne vykazovat v pohybu mravence určitý složitější vzorec. Tento algoritmus tedy demonstruje základní princip umělého života - z jednoduchého chování se simulací v čase objeví složitější vzorec chování.

Velmi známý příklad nasazení umělého života je spojen s úlohou obchodního cestujícího. Obchodní cestující má za úkol projet určitý počet měst, každé navštívit právě jednou a na konci cesty se vrátit do počátečního města, aby při tom cesta, kterou urazí, byla co nejkratší.

V roce 1999 přišel Marco Dorigo spolu se svými kolegy na řešení této úlohy, které bylo časově efektivnější než tradiční způsoby řešení[1]. Jeho řešení, pomocí algoritmu ACO (Ant Colony Optimization), spočívalo v nasazení umělých mravenců. Tito mravenci prochází všemi možnými cestami a zanechávají za sebou virtuální feromonovou stopu, podél delších cest méně a podél kratších více. Po prvním kole objevování se další procházející mravenci orientují podle intenzity zanechané stopy a volí cesty s větší intenzitou, tedy ty kratší. Tímto způsobem se vytvoří na nejkratších cestách vrstva feromonu a na méně efektivnějších trasách stopa vyprchá. Po oznámení výsledků s tímto řešením byl tento postup použit telekomunikačními společnostmi ve Francii a Velké Británii pro optimalizaci směrování v jejich sítích. Pro potřeby směrování v sítích byl vytvořen algoritmus AntNet, který je založen na algoritmu ACO. Použití umělých mravenců je pro vyhledávání cest v síti velmi vhodné díky těmto jejich vlastnostem:

- optimálnost - dokáží nalézt nejlepší cestu
- adaptivnost - nepřestávají hledat nové lepší cesty
- robustnost - při chybě ve spojení systém nespadne, ale hledají se jiná řešení

Algoritmus AntNet v porovnání s algoritmem OSPF (Open Shortest Path First), který je oficiálním směrovacím protokolem v síti internet, dosahuje lepších výsledků viz. [3]:

Zatížení (%)	Ztrátovost paketů u OSPF (%)	Ztrátovost paketů u AntNet (%)
110	8,95	2,19
150	33,2	12,86
200	49,9	26,65

Oblast autonomních agentů a jejich spolupráce v decentralizovaném systému je v dnešní době používána například u robotů, kteří jsou použiti na záchranné akce při katastrofách. NASA využívá autonomně řízené systémy pro zkoumání vesmíru. Ze všech těchto příkladů je vidět, že se jedná o velmi perspektivní a zajímavou oblast vhodnou pro zkoumání.

2.12 Umělí mravenci

Při tvorbě umělých mravenců se necháváme inspirovat skutečnými mravenci a jejich chováním v přírodě:

- shánění potravy
- stavba mraveniště
- rozdělování prací

Jedním z hlavních rysů převzatých od živých mravenců je nepřímá komunikace pomocí značkování svého okolí feromonovou stopou. Feromon je chemická látka, kterou za sebou mravenci v přírodě zanechávají. Existuje několik druhů stop, které mravenci vytvářejí. Lze je dělit podle jejich účelu:

- signál pro rozpoznání činnosti - hledám potravu
- signál pro zatraktivnění cesty - tady je potrava
- poplašný signál - pozor, nebezpečí

Kromě druhu umí mravenci také rozpoznat intenzitu dané stopy. Při hledání vhodné cesty se mravenci řídí intenzitou stopy (čím intenzivnější stopa, tím větší pravděpodobnost, že se po ní mravenec vydá). Feromon se může v prostředí kumulovat a s časem se z prostředí vypařuje.

Příklad: Existují dvě cesty. První je dvakrát delší než druhá. Jestliže mravenec jde první cestou tam a zpět a projde ji za čas t, tak druhý mravenec projde za stejný čas druhou cestu tam a zpět dvakrát. Na druhé cestě bude pak dvakrát větší koncentrace feromonu a tím i dvakrát větší šance na zvolení druhé cesty před první ostatními mravenci. Čím větší počet mravenců projde danou cestou, tím větší na ní bude koncentrace feromonu a tím větší bude šance, že ji zvolí ostatní mravenci.

Toto je princip kladné zpětné vazby, která je jedním ze znaků samoorganizace. Výsledkem je výběr kratších cest před delšími, což můžeme označit za emergentní chování (necílené, dosažené v čase vzájemnou spoluprací). I když mravenci mají svou královnu, jejich činnost není nijak centrálně řízena, společenstvo funguje jako decentralizovaný systém. Princip řízení takového systému spočívá v samoorganizaci. Příkladem je způsob rozdělování prací:

1. Mravenec z okolního prostředí získá díky feromonovým stopám přehled o tom, jaké práce vykonávají ostatní.
2. Na základě zjištěných informací se rozhodne o tom, jakou činnost bude vykonávat.

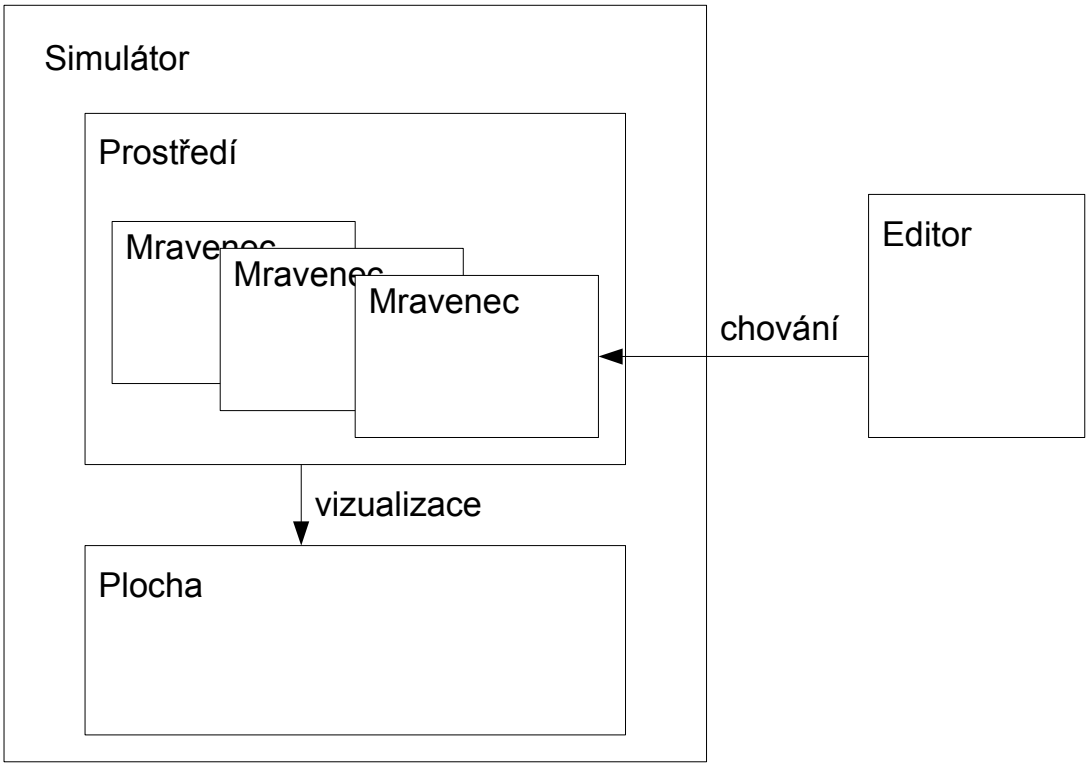
Tento přístup může v lokálním měřítku selhávat, ale v celkových součtech bude rozdělení prací v požadovaném poměru. Mravenci tedy dokáží najít nejkratší cestu k potravě nebo si potřebně rozdělí práci, což lze považovat za určitou inteligenci. Žádný z mravenců ale nezná způsob nalezení nejkratší cesty. Kdybychom vypustili deset mravenců do prostoru běžného pokoje, tak by měli problém, aby se potkali, a natož si správně rozdělili práci. Klíčovým parametrem pro jejich spolupráci je proto jejich počet. Existuje určitý kritický počet jedinců pro vytvoření tzv. roje a pro vznik rojové inteligence, která umožňuje výše zmíněné spolupráce. Pro funkčnost a přežití roje jako decentralizovaného systému jsou tedy nezbytně potřebné následující vlastnosti[1]:

- množství - počet je velmi důležitý faktor pro vzájemnou spolupráci v rámci roje. Několik jedinců nedokáže spolupracovat jako několik set jedinců.
- jednoduchost - relativní jednoduchost způsobu dorozumívání mravenců umožňuje růst systému, aniž by nepřiměřeně vzrůstala složitost systému a hrozilo jeho uváznutí. Naopak jednoduché části spolu komunikují, spolupracují a vytvoří znalosti a dovednosti na vyšší úrovni.
- pozorování ostatních - pro mravence, kteří jednají na základě aktuálního stavu svého okolí a nemají žádné postupy pro plánování, je nezbytností sledovat, co dělají ostatní mravenci. Pozorování může probíhat přímým setkáním nebo prostřednictvím feromonových stop. Tím dosáhnou nejen lepší vzájemné spolupráce, ale i lepší znalosti svého okolí.
- náhodné experimenty a náhodný průzkum okolí - pro každý decentralizovaný systém je velmi důležité získávání nových informací, proto je třeba prozkoumávat okolí. Bez nových informací by nemusela kolonie najít nové zdroje potravy nebo se adaptovat na změny v prostředí.

Kapitola 3

Návrh řešení

Na základě teoretických poznatků je třeba navrhnout řešení jednotlivých částí simulátoru mraveniště. Celou práci lze rozdělit na dvě základní části, kterými jsou editor chování mravenců a vlastní simulátor (obr. 3.1).



Obrázek 3.1: Návrh řešení

3.1 Simulátor

Simulátor zahrnuje většinu z celé práce. Při návrhu je třeba vytvořit model prostředí a agentů (mravenců). Je třeba řešit otázku interakcí agentů s prostředím a s ostatními

agenty. Systém musí umožňovat vložení agentů do prostředí, dále definování jejich chování a dalších parametrů. Do prostředí je možno vložit i ostatní objekty (zdroj potravy, překážka). Je třeba vytvořit funkce pro grafický výstup, které budou zobrazovat průběh simulací.

3.1.1 Prostředí

Cílem při tvorbě simulačního prostředí je vytvoření takového prostředí, které se nejen inspiruje skutečným mraveništěm, ale zůstává i obecným modelem prostředí pro jiné druhy simulace než je simulace mraveniště. Z toho důvodu je třeba při návrhu ponechat všem entitám jistou obecnost, která se dá docílit větším počtem parametrů nastavení prostředí. Při tvorbě simulačního prostředí pro simulátor mraveniště je třeba převzít některé rysy a objekty z reálného světa skutečných mravenců:

- potrava - energetický zdroj
- jehličí - stavební materiál
- feromonové stopy - způsob nepřímé komunikace mravenců
- zdi - překážky, přes které mravenci neprojdou

Prostředí bude muset spravovat fyzikální model pro tyto objekty, který bude zajišťovat jejich správné chování v prostředí:

- vyprchávání feromonových stop
- rozpad jehličí položeného na hromadě po okolí
- neprůchodnost zdí

Návrh prostředí dále vychází z teorie celulárních automatů:

- diskrétní prostor
- diskrétní čas

3.1.2 Agent

Při tvorbě agenta (mravence) je opět snaha převzít některé rysy skutečných mravenců:

- jednoduché chování
- nepřímá komunikace pomocí feromonových stop

Návrh agenta vychází z koncepce reaktivních agentů:

- neuchovává si model okolního světa - pouze reaguje na aktuální stav prostředí
- jednoduché rozhodování - konstrukce typu: if-then

Je třeba definovat příkazy, pomocí nichž lze implementovat chování agenta. Agent bude příkazy postupně číst a plnit je. Příkazy musí být dvojího druhu:

- příkazy představující činnost senzorů - agent pomocí nich může zjišťovat stav svého okolí
- příkazy umožňující agentovi v prostředí jednat (simulující činnost efektorů) - příkazy pro pohyb, zvedání a pokládání předmětů atd.

3.1.3 Jazyk pro popis chování agenta

Při návrhu jazyka vycházím z požadavků kladených na chování agenta:

- jednoduchost - pokud možno, co nejelementárnější operace pro příkazy pohybu a ostatních interakcí s okolím
- neuchovává si model okolního světa - není potřeba proměnných
- jednání na základě aktuálního stavu okolí - konstrukce typu if-then, návrh příkazů na zjišťování aktuálního stavu prostředí vrací hodnoty pravda / nepravda

3.2 Editor chování

Editor umožňuje vytváření algoritmů reprezentujících chování agentů. Protože navrhovaný jazyk je velmi jednoduchý a nepříliš rozsáhlý, lze vytvořit editor, který bude obsahovat sadu tlačítek pro zadávání příkazů. Editor by ale také měl podporovat režim ručního psaní kódu.

Kapitola 4

Implementace

Impementačním jazykem je Java (JDK 1.5). Díky zvolení Javy je výsledný program přenositelný na různé platformy.

4.1 Simulátor

Třída Simulator je stěžejní třídou v simulátoru jako celku. Tato třída řídí běh simulací. Obsahuje metody pro ukládání a načítání simulací i nastavení. Při otevření nové simulace nebo při načtení uložené simulace se nastavení prostředí změní na implicitní hodnoty a je třeba si prostředí znova nastavit nebo nastavení prostředí načíst. Dále tato třída implementuje paletu nástrojů pro vkládání a editaci objektů v prostředí:

- jídlo - vloží jedno jídlo na vybrané pole, pokud na tomto poli není zeď
- jehličí - vloží jedno jehličí na vybrané pole, pokud na tomto poli není zeď
- mravenec - vloží jednoho mravence na vybrané pole. Pokud na tomto poli je zeď, pak je smazána. Při kliknutím pravého tlačítka myši na ikonu s tímto nástrojem se zobrazí výběrové menu, které umožňuje nastavení vlastností vkládaného mravence
- zeď - vloží zeď na vybrané pole (všechny ostatní objekty na poli se nacházející budou smazány)
- guma - smaže veškeré objekty, které se na poli nacházejí
- detail pole - umožňuje prohlížet a editovat stav zvoleného pole a to včetně atributů agenta, který se na daném poli vyskytuje
- paleta pro pokládání stop - na vybrané pole je vložena taková intenzita, jakou by vložil agent příkazem `put` u příslušné stopy. Tuto hodnotu lze nastavit

4.1.1 Prostředí

Pro implementaci prostředí jsem vytvořil třídu Prostredi. Tato třída spravuje mapu prostředí:

- uchovává aktuální stav prostředí - pozice všech agentů a ostatních objektů v prostředí
- řídí fyzikální zákonitosti - vyprchávání feromonových stop, rozpad předmětů položených na hromádce, energetický model

- volá funkce třídy Plocha pro zobrazení provedených změn

Mapa prostředí má diskrétní charakter (je složena z jednotlivých polí) a je prstencového modelu. Jako relevantní okolí každého pole se považuje Neumannovské okolí. Simulační čas je také diskrétní. V prostředí je definováno pět druhů objektů a pravidla pro jejich výskyt v prostředí:

- agent (mravenec) - na každém poli mapy může být pouze jediný agent
- jídlo - objekt, který se na jediném poli může nacházet ve více exemplářích. Množství jídla na poli závisí na parametru prostředí, kterým je spád prostředí, a na množství jídla na okolních polích. Spád prostředí určuje maximální počet jídla, o který může být více nebo méně jídla na poli než na okolních polích. Jestliže je tento počet překročen, dojde v prostředí ke změně tak, aby toto pravidlo bylo splněno. Výjimku tvoří pole, kde se nacházejí zdi. Taková pole nemohou obsahovat jídlo a při výpočtu rozpadu se neberou v potaz. Zároveň nemůže jídlo přes zeď propadnout, tzn. jestliže bude pole obklopeno zdmi, pak může být počet jídla na poli nezávislý na spádu prostředí.
- jehličí - obdoba jídla
- zeď - jestliže se na poli vyskytuje zeď, nemůže se zde vyskytovat jiný další objekt. Zeď má nekonečnou výšku a nemůže být jakýmkoliv způsobem překonána.
- feromonová stopa - je definováno šest druhů stop (červená, modrá, žlutá, zelená, bílá, černá). Na jednom poli se může vyskytovat libovolný počet stop. Každá stopa má svou intenzitu, která je libovolná, ale nepřesahuje nastavenou maximální velikost. Prostředí se stará o vyprchávání stop v čase.

Dále prostředí zajišťuje veškeré interakce a to jak agentů s prostředím tak agentů mezi sebou. Jestliže chce agent zjistit stav okolního prostředí, pak volá metody této třídy, které mu vracejí hodnoty typu boolean, udávající zda platí / neplatí zjišťovaná skutečnost. Pokud chce agent provést akci, která by mohla vést k porušení fyzikálního modelu, jako jsou: move, take, put, eat, pak musí také volat metody třídy Prostredi. Ostatní akce (turnLeft, turnRight, wait) nemůžou narušit fyzikální model prostředí. Každá provedená akce v prostředí má za následek úbytek odpovídajícího množství energie. Na konci simulačního kroku se navíc každému agentovi odečte množství energie příslušící jeho stavu podle parametru "nesoucí" (nenese nic, nese jídlo, nese jehličí). Celý energetický model lze vypnout, a tím zcela odhlédnout od spotřeby energie.

4.1.2 Agent

Agent je implementován pomocí třídy Mravenec. Tato třída obsahuje atributy reprezentující stav agenta v prostředí:

- chování - algoritmus, který agent vykonává
- energie - hodnota energie agenta
- nesoucí - určuje stav agenta. Může nabývat tří stavů:
  - nenese nic - je prázdný a může cokoliv vzít, položit může pouze stopy
  - nese jídlo - není prázdný a může vzít pouze stopy, položit může stopy a jídlo

  - nese jehličí - není prázdný a může vzít pouze stopy, položit může stopy a jehličí

- směr - určuje otočení v prostředí
- druh - určuje barvu agenta (vhodné pro odlišení agentů s různým chováním)

Agent funguje jako interpret jazyka. Čte jednotlivé příkazy a okamžitě je vykonává.

4.1.3 Jazyk pro popis chování agenta

Výsledný navržený jazyk je velice jednoduchý. Při návrhu jeho konstrukce jsem se inspiroval programovacím jazykem Karel [10]. Jazyk obsahuje příkazy pro konání činností v prostředí. Dále obsahuje příkazy podmíněného a nepodmíněného skoku a návěští, které plní funkci konstrukce if-then(-else):

- příkazy určující činnost agenta v prostředí- move, turnLeft, turnRight, take *předmět*, put *předmět*, eat, wait
- nepodmíněný skok - go *návěští*
- podmínění skok - on *podmínka* go *návěští*
- návěští, cíl podmíněných a nepodmíněných skoků - *návěští*

Jazyk je case sensitive. Každý příkaz musí být na samostatném řádku. Kromě příkazů definuje jazyk další objekty a to *předmět* a *podmínka*. Předměty jsou objekty, se kterými může agent v prostředí zacházet:

- food - jídlo
- needle - jehličí
- red, blue, yellow, green, white, black - stopy: červená, modrá, žlutá, zelená, bílá, černá

Podmínky jsou metody, které vrací hodnoty typu boolean. Jejich volání simuluje činnost senzorů prostředí. Agent si neuchovává model okolního světa, ale reaguje pouze na současný stav prostředí, proto mu stačí informace typu pravda / nepravda. Návěští je složeno z libovolného množství nebílých znaků.

Příkazy

Zde jsou jednotlivé příkazy, které plní funkce efektorů, každý z nich trvá právě jeden simulační krok a po jejich provedení končí tah agenta:

- move - pohyb agenta vpřed. Agent se posune vpřed o jedno pole, jestliže má na pohyb dostatek energie a na daném poli není jiný agent nebo zeď. Při pokusu pohybu agenta na pole, které je obsazeno jiným agentem či zdí, se pohyb nezdaří a agent zůstává na své současné pozici. Energie, která se mu odečte při tomto nezdařeném pokusu o pohyb, je stejná jako v případě úspěšného pokusu.
- turnLeft, turnRight - otočení doleva nebo doprava se provede, jestliže má agent dostatek energie. Po úspěšném pokusu se mu odečte příslušné množství energie.

- take - umožňuje agentovi zvedat v prostředí předměty, které leží o jedno pole před ním. Pokud má agent na zvednutí dostatečnou energii, pak se vykonaná akce liší podle druhu předmětu, který chce zvednout. Zvednutí jiného agenta nebo zdi jsou ilegální operace. Jestliže chce odebrat určitou značku, pak se na daném poli sníží intenzita této značky o určitý (nastavitelný) počet, přičemž intenzita nemůže klesnout pod nulu. Jestliže chce odebrat jídlo nebo jehličí, pokud se na poli před ním vyskytuje a stav agenta je takový, že nic nenese, pak odebere z pole jedno jídlo (resp. jehličí) a jeho stav se změní na "nesoucí jídlo" (resp. jehličí). Ať už je pokus o zvednutí úspěšný nebo ne, odečte se příslušné množství energie.

- put - umožňuje agentovi pokládat v prostředí předměty. Pokud má agent na položení dostatečnou energii, vykonaná akce se liší podle druhu předmětu, který chce položit. Položení jiného agenta nebo zdi jsou ilegální operace. Jestliže chce položit určitou značku, pak se na poli, na kterém se agent nachází, zvýší intenzita této značky o určitý (nastavitelný) počet, přičemž intenzita nemůže přesáhnout své maximum. Jestliže chce položit jídlo (resp. jehličí), pokud se na poli před ním nevyskytuje zeď a stav agenta je takový, že nese jídlo (resp. jehličí), pak přidá na pole před sebou jedno jídlo (resp. jehličí) a jeho stav se změní na "nic nenese". Ať už je pokus o položení úspěšný nebo ne, odečte se příslušné množství energie.

- eat - umožňuje agentovi jíst jídlo a tím získávat energii. Pokud má agent dostatečné množství energie a na poli před ním je jídlo, pak se na poli před ním sníží hodnota jídla o jedna a jeho energie se zvýší o energii obsaženou v jídle. Ať už je pokus o snězení úspěšný nebo ne, odečte se příslušné množství energie.

- wait - agent zůstává neaktivní. Pouze ztratí příslušnou energii. V tomto případě je vhodné nastavit úbytek energie na zápornou hodnotu, tím agent získá danou energii a lze tím tak docílit efektu odpočinku.

Podmínky

Největší oblastí jsou podmínky, kterými agenti získávají poznatky o okolním světě (plní roli senzorů). Agent se může pomocí jednotlivých podmínek dotázat pouze na stav pole, které je před ním. Při tom volá metody třídy Prostredi, které vrací dva možné výsledky dotazu pravda / nepravda:

- isFood - vrací pravdu, jestliže se na poli před agentem vyskytuje jídlo. V opačném případě vrací nepravdu.

- isNeedle - vrací pravdu, jestliže se na poli před agentem vyskytuje jehličí. V opačném případě vrací nepravdu.

- isAnt - vrací pravdu, jestliže se na poli před agentem vyskytuje jiný agent. V opačném případě vrací nepravdu.

- isWall - vrací pravdu, jestliže se na poli před agentem vyskytuje zeď. V opačném případě vrací nepravdu.

- isEmpty - vrací pravdu, jestliže se na poli před agentem nevyskytuje agent ani zeď. V opačném případě vrací nepravdu.

- isRed (isBlue, isYellow, isGreen, isWhite, isBlack) - vrací pravdu, jestliže se na poli před agentem vyskytuje koncentrace červené (modré, žluté, zelené, bílé, černé) feromonové stopy větší než nula. V opačném případě vrací nepravdu.
- isEqualFood (isEqualNeedle) - vrací pravdu, jestliže se na poli před agentem vyskytuje stejný počet jídla (jehličí) jako na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.
- isEqualRed (isEqualBlue, isEqualYellow, isEqualGreen, isEqualWhite, isEqualBlack) - vrací pravdu, jestliže se na poli před agentem vyskytuje stejná intenzita červené (modré, žluté, zelené, bílé, černé) feromonové stopy jako na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.
- isHigherFood (isHigherNeedle) - vrací pravdu, jestliže se na poli před agentem vyskytuje větší počet jídla (jehličí) než na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.
- isHigherRed (isHigherBlue, isHigherYellow, isHigherGreen, isHigherWhite, isHigherBlack) - vrací pravdu, jestliže se na poli před agentem vyskytuje větší intenzita červené (modré, žluté, zelené, bílé, černé) feromonové stopy než na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.
- isLowerFood (isLowerNeedle) - vrací pravdu, jestliže se na poli před agentem vyskytuje menší počet jídla (jehličí) než na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.
- isLowerRed (isLowerBlue, isLowerYellow, isLowerGreen, isLowerWhite, isLowerBlack) - vrací pravdu, jestliže se na poli před agentem vyskytuje menší intenzita červené (modré, žluté, zelené, bílé, černé) feromonové stopy než na poli, na kterém se agent nachází. V opačném případě vrací nepravdu.

Zbývající skupinu tvoří podmínky, kterými agent zjišťuje svůj vlastní stav:

- isHungry - vrací pravdu, pokud má agent nedostatek energie. Při dostatečném množství energie vrací nepravdu. Úroveň dostatku či nedostatku energie lze nastavit.
- isCarry - vrací pravdu, pokud agent nese jídlo nebo jehličí. V případě, že nic nenese, vrací nepravdu.
- isCarryFood - vrací pravdu, pokud agent nese jídlo. V opačném případě vrací nepravdu.
- isCarryNeedle - vrací pravdu, pokud agent nese jehlicí. V opačném případě vrací nepravdu.
- isRandom - vrací pravdu s pravděpodobností 0,5. Pomocí této funkce lze dosáhnout nahodilosti v jednání agenta.

Poznámky k jazyku

Jazyk se skládá ze základních příkazů. Složitější příkazy vznikají jejich posloupností:

```
krok_vzad             // návěští
   turnLeft
   turnLeft
   move
```

Velmi důležitým prvkem v simulacích umělého života je náhodnost v chování agentů. Tuto vlastnost umožňuje podmínka `isRandom`, která vrací true s pravděpodobností 0,5. Jejím kombinováním lze dosáhnout dalších pravděpodobností. Při implementaci chování agentů jsem se výhradně setkával s potřebou pravděpodobnostní volby mezi 2,3 a 4 možnostmi.

Příklad 1: S pravděpodobností 0,5 se otočí doleva, v opačném případě se otočí doprava.

```
   on isRandom go otocit_vpravo
   turnLeft         // možnost A
   go konec
otocit_vpravo       // možnost B
   turnRight
konec
```

Příklad 2: S pravděpodobností 0,33 se otočí doleva, s pravděpodobností 0,33 se otočí doprava, s pravděpodobností 0,33 udělá krok vpřed.

```
start
   on isRandom go otocit
   on isRandom go start    // neprovedl výběr a volí znovu
   move           // možnost C
   go konec
otocit
   on isRandom go otocit_vpravo
   turnLeft       // možnost A
   go konec
otocit_vpravo
   turnRighht     // možnost B
konec
```

Podmínek, respektive podmíněných skoků lze během jednoho simulačního kroku provést libovolné množství. Závisí pouze na programátorovi, do jaké míry chce, aby byl jeho agent inteligentní.

Příklad: Pohyb agenta po mapě.

Nejprve úplně obyčejný. Při dosažení první překážky agent uvázne.

```
start
   move
   go start
```

Nyní vylepšený o rozpoznání překážky a změnění směru.

```
start
   on isEmpty go pohyb
   turnLeft      // překážka
   go start
```

```
pohyb
   move
   go start
```

Poslední vylepšení spočívá ve zjištění druhu překážky. Pokud se jedná o jiného agenta, lze očekávat, že tato překážka časem pomine.

```
start
   on isEmpty go pohyb
   on isAnt go pockat
   turnLeft      // překážka - zed'
   go start
pockat
   wait
   go start
pohyb
   move
   go start
```

4.2 Editor chování

Editor pracuje na principu vkládání jednotlivých příkazů pomocí tlačítek:

- move - vloží příkaz: `move`
- turnLeft - vloží příkaz: `turnLeft`
- turnRight - vloží příkaz: `turnRight`
- wait - vloží příkaz: `wait`
- eat - vloží příkaz: `eat`
- take - vloží příkaz: `take` předmět, je třeba zvolit příslušný předmět
- put - vloží příkaz: `put` předmět, je třeba zvolit příslušný předmět
- label - vloží návěští, je třeba zvolit příslušné návěští
- go - vloží příkaz nepodmíněného skoku: `go` návěští, je třeba zvolit příslušné návěští
- on go - vloží příkaz podmíněného skoku: `on` podmínka `go` návěští, je třeba zvolit příslušné návěští a podmínku

Dále obsahuje nástroje:

- guma - vymaže příkaz na aktuálním řádku
- řádek - vloží na aktuální pozici prázdný řadek pro lepší přehlednost
- vytvořit návěští - před použitím návěští musí být návěští vytvořeno
- zkompilovat chování - zkontroluje, zda jsou veškeré skoky na použitá návěští a zda jsou veškerá použitá návěští použita jen jednou. Poté uloží chování pod zadaným jménem.

Editor podporuje přímý režim psaní kódu prostřednictvím importu zdrojového kódu ze souboru. Soubor lze vytvořit v libovolném textovém editoru a poté ho načíst v editoru. Na druhé straně umožňuje i export příkazů z editoru do souboru.

Kapitola 5

Závěr

5.1 Simulace

Navrhl jsem sadu simulací, které jsem ve svém simulátoru vyzkoušel. Vytvořil jsem jednotlivá chování agentů a provedl simulace. V této části popisuji principy algoritmů chování, jejich důsledky a výsledky simulací.

5.1.1 Langtonův mravenec

Pomocí jazyka lze vytvořit algoritmus vykazující stejné chování jako algoritmus Langtonova mravence. Simulace tohoto chování vykazuje stejné výsledky, pouze na jednotlivé kroky je třeba více simulačních kroků.

Simulaci lze provézt po nastavení prostředí a vložení mravence s příslušným chováním. Algoritmus tohoto chování je stejně jako nastavení prostředí a chování mravence uložen na CD:

- nastavení prostředí: langton
- chování mravence: langton
- algoritmus: langton.txt

Místo použití dvou barev (bílá a černá) a přebarvování jednotlivých polí, má agent k dispozici pouze černou stopu, kterou pokládá nebo odebírá:

- Je-li před ním pole s černou stopou: odebere černou stopu, udělá krok vpřed a otočí se doprava
- Je-li před ním pole bez černé stopy: udělá krok vpřed, položí černou stopu a otočí se doleva

5.1.2 Langtonův mravenec v bludišti

Zkoušel jsem Langtonova mravence použít na prohledávání bludiště. Hlavní problém nastává při pohybu na pole, na kterém je zeď. Vyzkoušel jsem několik variant. Nejlepší varianta spočívá v přidání tří pravidel a v použití další stopy (bílé):

- Je-li před ním na poli zeď: položí bílou stopu a otočí se doprava

- Je-li před ním pole, kde je pouze bílá stopa: odebere bílou stopu, udělá krok vpřed a otočí se doleva
- Je-li před ním pole, kde je bílá a současně černá stopa: odebere bílou stopu, odebere černou stopu a otočí se doleva

Vyzkoušel jsem toto chování na několika bludištích a nepodařilo se mi dosáhnout uvíznutí ve smyčce. Mravenec za sebou nechává permanentní červenou stopu, která má pouze informační charakter (ukazuje, která pole již mravenec navšívil).

Simulaci lze provézt po nastavení prostředí a vložení mravence s příslušným chováním. Algoritmus tohoto chování je stejně jako nastavení prostředí a chování mravence uložen na CD:

- nastavení prostředí: langton
- chování mravence: langtonBludiste
- algoritmus: langtonBludiste.txt

5.1.3 Simulace hejna

Cílem této simulace je navození chování, které můžeme pozorovat u živočichů žijících v hejnech:

- seskupování jedinců do hejna
- vzájemná spolupráce v hejnu - varování před nebezpečím

Chování mravence lze rozdělit do několika dílčích úloh:

1. nalezení dalšího mravence - mravenec prohledává náhodně své okolí a současně zanechává za sebou červenou stopu, aby ho ostatní mravenci mohli lépe najít. Jestliže mravenec najde stopu, vydá se po ní.
2. růst hejna, neustat v lokálním řešení problému (nalezení jednoho mravence)- mravenec se otočí směrem do prostoru a udělá krok vpřed. Poté se otočí čelem vzad a s pravděpodobností 0,5 se rozhodne vrátit k hejnu (přejde do bodu 1). V opačném případě chvíli počká (aby ho ostatní následovali) a opakuje danou akci (přejde na začátek bodu 2).
3. útěk a varování ostatních před nebezpečím - nebezpečí je simulováno pomocí modré stopy. Jestliže se před mravencem objeví modrá stopa, tak mravenec položí modrou stopu (varuje tím ostatní), otočí se čelem vzad a udělá krok vpřed (utíká). Při útěku dále pokládá modrou stopu. Po několika krocích zastaví (už je po nebezpečí) a snaží se přidat do hejna (přejde do bodu 1).

Výsledné chování projevuje požadované rysy. Při simulaci lze pozorovat, že následek nebezpečí (varování ostatních a úprk) přetrvává v systému i po zániku příčiny nebezpečí (prvotního impulsu).

Simulaci lze provézt po nastavení prostředí a vložení mravence s příslušným chováním. Algoritmus tohoto chování je stejně jako nastavení prostředí a chování mravence uložen na CD:

- nastavení prostředí: hejno
- chování mravence: hejno
- algoritmus: hejno.txt

5.1.4 Nalezení nejkratší cesty

Tato simulace je zaměřena na nalezení nejkratší cesty mezi mraveništěm a zdrojem potravy. Je třeba vytvořit takové chování, které pomocí kladení stop umožňuje mravencům spolupracovat a nalézt nejkratší cestu. Jsou použity tři stopy:

- červená - značí cestu k jídlu. Zanechává ji za sebou mravenec, který nese jídlo
- modrá - značí cestu k mraveništi. Zanechává ji za sebou mravenec, který nenese jídlo (jde z mraveniště)
- černá - značí mraveniště, cílový prostor určený pro shromažďování jídla

K navození chování vedoucího ke spolupráci slouží následující pravidla:

- Mravenec, který nenese jídlo, náhodně prohledává okolí a současně pokládá modrou stopu.
- Jestliže mravenec, který nenese jídlo, narazí na červenou stopu, vydá se po ní.
- Jestliže mravenec, který nenese jídlo, najde jídlo, pak ho zvedne.
- Mravenec, který nese jídlo, náhodně prohledává okolí a současně pokládá červenou stopu.
- Jestliže mravenec, který nese jídlo, narazí na modrou stopu, vydá se po ní.
- Jestliže mravenec, který nese jídlo, narazí na černou stopu, položí jídlo.

Zkoušel jsem několik variant tohoto postupu. Žádné řešení nemělo příliš úspěchů.

Simulaci lze provézt po nastavení prostředí a vložení mravence s příslušným chováním. Algoritmus tohoto chování je stejně jako nastavení prostředí a chování mravence uložen na CD:

- nastavení prostředí: nejkratsiCesta
- chování mravence: nejkratsiCesta
- algoritmus: nejkratsiCesta.txt

Neúspěch navození chování spatřuji v nevhodnosti použitého simulačního prostředí pro tento druh chování:

- zdlouhavé zjištění aktuálního stavu prostředí - aby mravenec mohl zjistit stav kolem sebe, musí se třikrát otočit
- malá informovanost o okolí - mravenec vidí pouze jedno následující pole
- fyzikální zákonitosti modelu prostředí - na jednom poli může být nejvýše jeden mravenec, mravenci si navzájem blokují cestu

Pro řešení požadovaného chování by bylo třeba navrhnout vhodnější model simulačního prostředí. Zde jsou některé vlastnosti, které by mělo zahrnovat:

- mravenec vnímá v jednom simulačním kroku celé své okolí (Neumannovské nebo úplné)

- prostředí poskytuje větší viditelnost agentům (do určité vzdálenosti nebo k nejbližší zdi)
- fyzikální zákonitosti modelu prostředí - na jednom poli může být více mravenců současně

5.1.5 Vytvoření mraveniště

Na počátku této simulace je mapa s náhodně rozmístěnými zdroji (jídlo, jehličí) a mravenci (sběrači jídla, sběrači jehličí). Cílem simulace je shromáždit jídlo a jehličí na jednu hromadu - vytvořit mraveniště. Návrh řešení této úlohy rozdělím na dva podproblémy, respektive na dvě chování: sběrače jídla a sběrače jehličí. Algoritmus obou chování je stejný, liší se pouze ve zdrojích, které sbírají, proto budu dále popisovat jen algoritmus sběrače jídla. Pro tento algoritmus je použita modrá stopa, která značí mraveniště. Když mravenec položí jídlo, tak označí modrou stopou místo, na které jídlo položil. Zde jsou základní pravidla, kterými se mravenec řídí:

- Mravenec náhodně prohledává okolí.
- Když nalezne jídlo, současně nenese jídlo a na poli s jídlem není modrá stopa, pak vezme jídlo.
- Když nalezne jídlo a současně nese jídlo, pak se pokusí jít s jídlem "do kopce" (snaží se postupovat vpřed na pole s větším počtem jídla než je na současném poli). Když nemůže jít výše, položí jídlo a místo položení označí modrou stopou. Poté se snaží sejít z mraveniště (pohybuje se do té doby, dokud na poli pod ním je jídlo).

Obě chování (sběrač jídla, sběrač jehličí) lze použít pro samostatné simulace. Výsledkem těchto simulací je tvoření větších hromádek jídla (resp. jehličí) ze zdrojů náhodně rozptýlených po mapě. Růst hromádek spočívá v jejich velikosti. Zpočátku se díky náhodě vytvoří několik menších hromádek. Na větších z nich je položeno více modré stopy, proto z nich nebude delší dobu odebíráno jídlo. Protože je hromádka větší, je větší i pravděpodobnost, že ji mravenec nesoucí jídlo najde a položí na ni jídlo a modrou stopu, tak získá hromádka na velikosti i na ochraně. Modrá stopa a nastavení jejího vyprchávání z prostředí určuje stabilitu nalezeného řešení. Čím je vyprchávání větší, tím je nalezené řešení méně stabilní. Použití obou druhů chování v jedné simulaci má za následek zajímavý efekt. Oba druhy mravenců (s rozdílným chováním) si mohou stavět vlastní hromádky a zpočátku tak doopravdy jednají. Časem ale v simulaci převáží efekt modré stopy a hromádky s jídlem začnou kopírovat polohu hromádek s jehličím - vznikne mraveniště.

Simulaci lze provézt po nastavení prostředí a vložení mravence s příslušným chováním. Algoritmus tohoto chování je stejně jako nastavení prostředí a chování mravence uložen na CD. Zde jsou uloženy i dvě simulace s vloženými mravenci s definovaným chováním a rozmístěnými zdroji:

- nastavení prostředí: mraveniste
- chování mravence: sberJidla, sberJehlici
- algoritmus: sberJidla.txt, sberJehlici.txt
- uložené simulace: mraveniste, mraveniste21767

5.2 Zhodnocení a návrh dalšího vývoje projektu

5.2.1 Zhodnocení projektu

Simulace umělého života představuje rozsáhlou a zajímavou oblast. V mém projektu jsem se zaměřil na reaktivní agenty jako na nejjednodušší formu umělého života. Navržený jazyk a vytvořený simulátor poskytují možnosti pro různé druhy simulací s reaktivními agenty. V simulacích, které jsem provedl, lze pozorovat požadované jevy jako jsou emergentní chování a principy samoorganizace.

5.2.2 Návrh dalšího vývoje projektu

Protože je tento projekt zaměřen na reaktivní agenty, kteří představují nejjednodušší agenty při simulacích umělého života, lze vhodná navázání hledat právě zde:

- vytvoření agentů s rozsáhlejšími schopnostmi - návrh jazyka s proměnnými a funkcemi
- vytvoření agentů orientovaných na cíl
- vytvoření agentů se schopností učit se

Literatura

[1] Steven Johnson. *Emergence*. Simon & Schuster, 2002. ISBN 0-684-86876-8.

[2] Petr Peringer. *Modelování a simulace, studijní opora*. 2006.

[3] WWW stránky. Antnet: Aco routing algorithm in practice. `http://telecom.section.informs.org/conference/16227.pdf`. květen 2007.

[4] WWW stránky. Artificial intelligence. `http://en.wikipedia.org/wiki/Artificial_intelligence`. květen 2007.

[5] WWW stránky. Celulární automat. `http://alife.tuke.sk/index.php?clanok=500`. květen 2007.

[6] WWW stránky. Deep blue. `http://en.wikipedia.org/wiki/Deep_Blue`. květen 2007.

[7] WWW stránky. Emergence. `http://en.wikipedia.org/wiki/Emergence`. květen 2007.

[8] WWW stránky. How intelligent is deep blue? `http://www.psych.utoronto.ca/~reingold/courses/ai/cache/mcdermott.html`. květen 2007.

[9] WWW stránky. Langton's ant. `http://pages.prodigy.net/nylesheise/langton.html`. květen 2007.

[10] WWW stránky. Programovací jazyk karel. `http://karel.webz.cz/`. květen 2007.

[11] WWW stránky. Self-organization. `http://en.wikipedia.org/wiki/Self-organization`. květen 2007.

[12] WWW stránky. Swarm intelligence. `http://en.wikipedia.org/wiki/Swarm_intelligence`. květen 2007.

[13] WWW stránky. Umělá inteligence. `http://www.automatizace.cz/article.php?a=681`. květen 2007.

[14] WWW stránky. What is the game of life? `http://www.math.com/students/wonders/life/life.html`. květen 2007.

[15] František Zbořil. *Agentní a multiagentní systémy, studijní opora*. 2006.