České vysoké učení technické v Praze
Fakulta elektrotechnická

Katedra kybernetiky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: Adam Lenger

Studijní program: Otevřená informatika (bakalářský)

Obor: Informatika a počítačové vědy

Název tématu: Vytvoření kooperujícího týmu duchů do hry Ms. Pac-Man

Pokyny pro vypracování:

Hlavním úkolem studenta je implementace kooperujícího týmu duchů do hry Ms. Pac-Man, přičemž by duchové měli využívat algoritmy založené na prohledávání prostoru, konkrétně Monte-Carlo Tree Search.

Student musí implementovat a následně upravit daný algoritmus tak, aby byl použitelný v doméně dané hry. Zároveň je nutno v řešení adresovat problém kooperace mezi duchy, přičemž se předpokládá porovnání možností implicitní a explicitní koordinace.

Seznam odborné literatury:

[1] E. Raboin, D. Nau, U. Kuter, S. K. Gupta, and P. Svec: Strategy generation in multi-agent imperfect-information pursuit games. In International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 947-954, 2010.
[2] S. M. Lucas: Ms. Pac-man Competition. ACM SIGEVOlution. Num 4, Vol 2, pp. 37-38, ACM, 2007.
[3] A. Fitzgerald, P. Kemeraitis and C.B.: Congdon: RAMP: a rule-based agent for Ms. Pac-Man. World Congress on Computational Intelligence. 2008.
[4] S. Samothrakis, D. Robles and S. M. Lucas: Fast Approximate Max-n Monte Carlo Tree Search for Ms Pac-Man. IEEE Transactions on Computational Intelligence and AI in Games, Vol 3, pp. 142-154, 2011.

Vedoucí bakalářské práce: Mgr. Branislav Bošanský

Platnost zadání: do konce zimního semestru 2012/2013

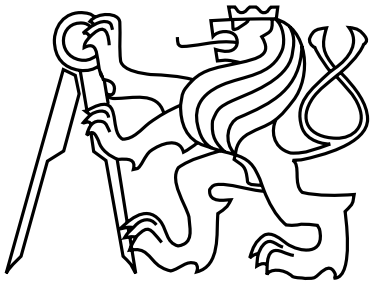
L.S.

prof. Ing. Vladimír Mařík, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 9. 1. 2012

České vysoké učení technické v Praze
Fakulta elektrotechnická
Katedra kybernetiky



Bakalářská práce

Vytvoření kooperujícího týmu duchů do hry Ms. Pac-Man

Adam Lenger

Vedoucí práce: Mgr. Branislav Bošanský

Studijní program: Otevřená informatika, Bakalářský

Obor: Informatika a počítačové vědy

3. ledna 2013

Poděkování

V první řadě bych chtěl poděkovat vedoucímu své práce Mgr. Branislavovi Bošanskému za jeho podporu a cenné rady při vytváření této práce. Dále bych chtěl také poděkovat své rodině a přátelům za neutuchající podporu.

Prohlášení

Prohlašuji, že jsem práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne 4. 1. 2013 .....

Abstract

The goal of this thesis is to use Monte Carlo Tree Search for game Ms. Pac-Man represented as extensive-form game. The purpose of extensive-form is to describe game with sequential move ordering. First goal is to find out how this representation will influence game with simultaneous moves and if this representation won't make worse results. The second goal is to discover, which one from implicit and explicit cooperation is better. We present results based on experimentation.

Abstrakt

Cílem této práce je aplikovat Monte Carlo Tree Search na hru Ms. Pac-Man reprezentovanou extenzivní formou hry. Účelem extenzivní formy hry je formulovat hru se sekvenčním pořadím tahů. Prvním cílem této práce je zjistit jak tato reprezentace bude ovlivňovat hru se simultánními tahy a zda nezhorší výsledky. Druhým cílem je objevit, zda je vhodnější implicitní či explicitní kooperace. Formou experimentů se vyzkouší jaké jsou podávány výsledky.

Obsah

Seznam obrázků

Kapitola 1

Úvod

V počítačových vědách existuje obor teorie her, který se mimo jiné věnuje také tomu, jak formulovat situace z reálného světa jako hru. U většiny situací tuto formulaci lze provést a tím pádem se znalosti z této oblasti se přenášejí i k řešení reálných problémů. Hry totiž slouží jako dobrý nástroj pro vyzkoušení případných možných postupů a algoritmů. Mnoho her dostatečně a vhodně reprezentuje danou situaci i při relativní jednoduchosti, proto se používají jako vhodný nástroj k řešení těchto problémů.

Ve hrách totiž dochází ke střetu hráčů, kteří se snaží maximalizovat svůj zisk. Vezmuli příklad z této bakalářské práce: používá se zde hra Ms. Pac-Man, kde Pac-Man utíká před duchy, kteří ho chytají. Hráči jsou tedy Pac-Man a duchové a snaží se maximalizovat (minimalizovat) dosažené skóre. V reálném světě si pak lze Pac-Man představit jako uprchlíka a duchy jako policisty. Použitý a vyzkoušený algoritmus z uvedené hry se pak dá aplikovat i ve skutečnosti. Tento postup v základě vystihuje většinu her. U každé hry je samozřejmě třeba počítat se specifickými vlastnostmi a je třeba s ní pracovat individuálně.

Pro hraní těchto her se využívá mnoho algoritmů. Ty musí brát v potaz řadu atributů, mezi něž patří reprezentace prostředí, reprezentace stavů hry, správnost řešení, rychlost, jakou lze získat řešení, atd. Zjednodušení prostředí lze docílit například nahrazením kontinuální cesty bodovým grafem, kde jednotlivé body jsou důležitá místa na cestě. Hráči se poté pohybují po jednotlivých uzlech grafu. Zaměření na rychlost spočívá v tom, že případně nejsou zohledněny všechny možnosti, či pomocné atributy, a rozhoduje se na základě hlavních údajů co nejrychleji. Tím pádem nejsou tak přesné výsledky. Existují ale i hry, kdy hráči na čase nezáleží a snažím se najít co nejlepší řešení bez ohledu na to, jak dlouho to bude trvat. Existují algoritmy specifické pro jeden typ hry, ale je zde i řada těch, které se dají používat ve více hrách. Mezi nimi najdeme takové, které jsou vyzkoušené a zaběhlé, ale jsou zde i relativně nové, které nebyly vyzkoušeny všude. Jeden z nich, Monte Carlo Tree Search, se již několik let setkává s velkým úspěchem.

Některé hry jsou pro počítač značně jednoduché, jiné naopak nadmíru složité. Záleží zde na několika faktorech. Jedním z nich je například možný počet tahů v daném stavu (tzv. branching factor). U šachů je jich velmi mnoho (a ještě více u Arimy), a tím pádem je

situace složitější a náročnější na výpočet. Dochází zde totiž ke generaci velkého množství možných stavů a nelze je v reálném čase všechny prohledat. Dalším faktorem může být zda oba hráči táhnou najednou či táhnou sekvenčně. První varianta je samozřejmě pro počítač těžší, protože neví, jak se v posledním tahu protihráč zachoval. V některých hrách záleží také na rychlosti tahu, tím pádem má počítač ztíženou situaci, jelikož nemá dostatek času na výpočet nejlepšího řešení.

Existuje mnoho druhů her a pro každou se používá či hodí jiný algoritmus. Cílem je samozřejmě najít nejlepší řešení, takže se dané algoritmy přizpůsobují, popřípadě se zkouší různé kombinace. Např. pokud má několik hráčů stejný cíl, je nutno řešit, jakým způsobem a zda mají kooperovat. Spolupráce se samozřejmě liší hru od hry. V některých hrách je lepší vytvořit menší týmy, jinde je zase třeba, aby všichni hráči spolupracovali dohromady. Jako příklad lze uvést duchy z hry Ms. Pac-Man.

Z důvodů zmíněných v tomto úvodu jsem se rozhodl implementovat algoritmus Monte Carlo Tree Search pro ovládání duchů ve hře Ms. Pac-Man.

Přehled Bakalářská práce je rozdělena do těchto kapitol:

Kapitola 2 - představuje hru Ms. Pac-Man, její pravidla a historie. Dále jsou zde také rozebrány problémy, které se v této hře vyskytují.

Kapitola 3 - nastiňuje základy herní teorie (např. herní strom, extenzivní formu hry, ...) a základní algoritmy používané pro hry. Nachází se zde i sekce věnovaná algoritmu Monte Carlo Tree Search.

Kapitola 4 - popisuje implementaci herního rozhraní, které je použito v této práci. Také je zde představena veškerá má implementace k této práci.

Kapitola 5 - rozebírá provedené experimenty. Nastíní očekávané výsledky a porovná je s reálnými.

Kapitola 6 - Závěr. Zde jsou popsány jednotlivé výsledky a dosažené cíle.

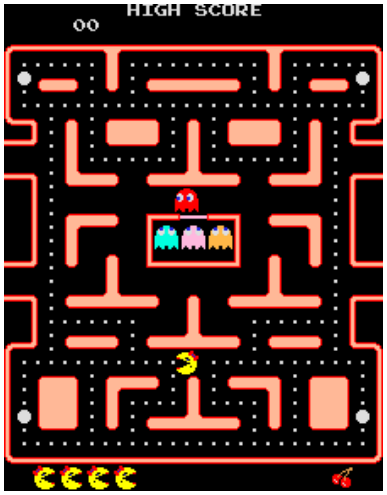
Kapitola 2

Hra Ms. Pac-Man

V této kapitole je popsáno, proč jsem se rozhodl pro hru Ms. Pac-Man a ovládání duchů. Jak už bylo naznačeno v úvodu, hra se hodí pro využívané prostředí, které částečně simuluje reálnou situaci a lze ho vhodně reprezentovat i počítačem. Hra patří do typu Pursuit-evasion her, kde je typickým zástupcem, jelikož duchové pronásledují Pac-Man. Vyvstává tedy problém, jaký typ koordinace mezi duchy zavést, zda implicitní či explicitní. Řeší se zde, který z těchto přístupů se vzhledem k větvení ukáže být nejlepším k určení optimální strategie.

2.1 Popis a pravidla hry

Hra Ms. Pac-Man je variantou hry Pac-Man. V této hře se Ms. Pac-Man nachází v bludišti. Snaží se sníst co nejvíce pilulek a vyhnout se přitom duchům, kteří ji při kontaktu zabijí. V bludišti jsou 4 duchové, kteří jsou do něj postupně vpuštěni z vězení, v němž se nacházejí na začátku hry. Hlavním úkolem pro Ms. Pac-Man je sníst všechny pilulky v daném bludišti, pro duchy je to chytit Ms. Pac-Man. Když se Ms. Pac-Man podaří sníst všechny pilulky, postupuje do dalšího kola (bludiště). Za každou pilulku, kterou sní, se jí přičítají body k celkovému skóre. V každém bludišti jsou také 4 tzv. "power-pill", což jsou speciální pilulky, po nichž může Ms. Pac-Man sníst duchy za dodatečné body. Tento efekt trvá pouze po krátký čas. Cílem celé hry je získat co nejvíce bodů. Ukázku hry můžete vidět na obrázku 2.1.



Obrázek 2.1: Ukázka hry Ms. Pac-Man

2.2 Historie hry

Hra Pac-Man byla vytvořena již v roce 1980, její varianta Ms. Pac-Man, která je použita v této práci, byla vyvinuta o rok později, v roce 1981. Lišila se od původní hry různými drobnostmi a náhodným pohybem duchů, tím pádem nešlo vytvořit pravidla pro překonání daných úrovní. Nejlepší lidský hráč v této hře zatím drží rekord asi 900000 bodů, kdežto počítačový hráč má rekord asi 25000 bodů. Tento velký rozdíl je způsoben např. časovým limitem na tah, simultánními tahy a dalšími omezujícími faktory. Od roku 2008 se pravidelně koná soutěž, kde účastníci mohou nahrát jak programy ovládající Ms. Pac-Man, tak od roku 2011 programy ovládající duchy. Programy, použité v této soutěži, ovládající duchy jsou buď zaměřené na minimalizování dosaženého skóre nebo na hratelnost hry. [3]

2.3 Algoritmy

V této soutěži je použito množství odlišných přístupů jak vypočítat nejlepší tah. Některé jsou založeny na herním stromu (viz odstavec 3.1), jiné používají ohodnocenou mapu nebo pouze jednoduše pracují se vzdálenostmi duchů od pacmana. S herním stromem pracují např. algoritmy jako Alfa-beta, NegaScout, MiniMax či právě Monte Carlo Tree Search, které je využito v této práci. Všechny uvedené algoritmy jsou dále představeny a rozebrány v následující kapitole.

2.4 Pursuit-evasion hry

Hra Ms. Pac-Man patří do kategorie tzv. "Pursuit-evasion"her. V uvedeném typu her se skupina pronásledovatelů snaží chytit či sledovat uprchlíka. Prostor v těchto hrách je většinou stavěný jako graf, tak je tomu i v implementaci hry Ms. Pac-Man, jež je v této práci použita. Daný prostor částečně simuluje opravdový svět, např. městské ulice, a postavy (hráči) v nich

simulují skutečné postavy reálného světa, např. policisty a zloděje. Z této kategorie her jsem si vybral Ms. Pac-Man vzhledem k jeho ideálním vlastnostem jako testovacího prostředí pro "Pursuit-evasion"hry. Prostředí této hry se dá reprezentovat jako graf, v každém tahu se mohou Pac-Man i duchové pohnout pouze o jedno políčko a dá se zde také dobře zkoušet implicitní a explicitní koordinace.

2.5 Hry s úplnou informací a simultánními tahy

Ms. Pac-Man je hra s úplnou informací a simultánními tahy. Úplná informace znamená to, že v okamžiku svého tahu, má hráč k dispozici všechny informace o daném stavu hry. Hráči ale v této hře hrají simultánně, tzn., že Ms. Pac-Man i duchové dělají rozhodnutí ve stejnou chvíli. V jednom okamžiku se tedy pohnou jak duchové, tak Pac-Man. Při počítání tahu může tudíž hráč předpovídat tah soupeře, ale nemůže ho s jistotou určit. Pro zjednodušení jsou v této práci tahy serializovány, takže se Ms. Pac-Man a duchové střídají na tazích. To ale způsobuje například problém s tím, že i když si duch myslí, že v následujícím tahu Pac-Mana chytí, on již může být o políčko dále.

2.6 Implicitní a explicitní koordinace

Implicitní koordinace nebere v potaz všechna možná rozhodnutí ostatních spoluhráčů, zaměřuje se spíše na spočítání co největšího množství výsledků. Spočívá v předávání pouze několika základních informací (jako například další rozhodnutí v blízké budoucnosti) dalším spoluhráčům.
Explicitní koordinace naopak bere vždy v potaz všechna možná rozhodnutí ostatních spoluhráčů a pracuje s nimi při počítání. Na konci má tedy přesnější výsledky, ale trvá zase o to déle.
Vzhledem k tomu, že se u explicitní koordinace berou v potaz všechny možné tahy hráčů, větví se herní strom mnohem více než při implicitní koordinaci. Tím pádem se při jeho stavění nedosáhne takové hloubky jako při implicitní koordinaci a výsledky nejsou tak vypovídající.

V této kapitole je představena hra a ukázáno, proč má smysl se jí zabývat. V následující kapitole jsou popsány různé přístupy a možnosti řešení dané hry a algoritmus, který je v této práci použit.

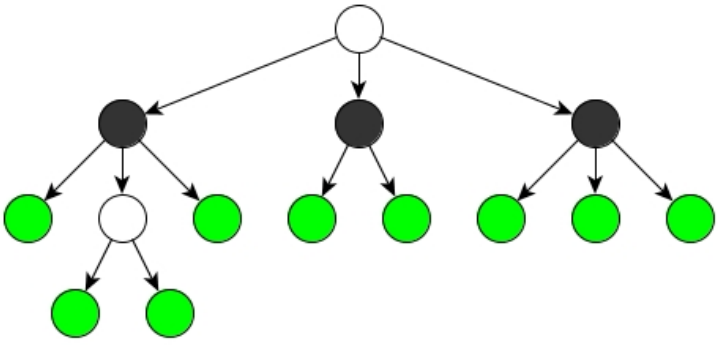
Kapitola 3

Algoritmy a Monte Carlo Tree Search

V předchozí kapitole byla popsána hra, jíž se v této práci zabývám. V této kapitole budou přiblíženy algoritmy, které se používají na řešení dané hry, a také zvolený přístup.

3.1 Herní strom

Jedná se o strom z teorie grafů. To znamená graf, který je souvislý a neobsahuje žádnou kružnici. Ve hrách, kde se hráči střídají v tazích (např. piškvorky), jsou jednotlivé uzly ve stejné úrovni tohoto stromu vždy možné tahy jednoho hráče. Jsou zde ovšem i hry jako např. mikado, kde toto zaručeno není. Kořen tohoto stromu je výchozí situace a jednotliví potomci jsou pak situace, do nichž se dostaneme po jednom tahu. Tah je reprezentován hranou stromu. Jeho listy jsou finální stavy hry. Příklad herního stromu, kde se hráči vždy střídají na tazích, je na obrázku 3.1. Světlou barvou jsou znázorněny stavy prvního hráče, tmavší barvou jsou znázorněny stavy druhého hráče.

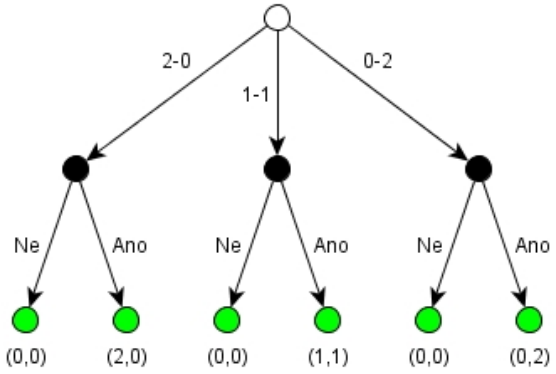


Obrázek 3.1: Herní strom - bíle jsou označeny stavy prvního hráče a tmavě druhého hráče, zeleně jsou označeny listy

3.1.1 Extenzivní forma hry

Extenzivní forma hry je jedna z možností, jak vyjádřit hru jako takovou, tzn. stavy, akce a výsledky. Využívá se k tomu herního stromu z předchozí kapitoly. Rozdíl je pouze v tom, že v listech se nacházejí výsledné zisky pro hráče. Všichni potomci daného uzlu pak tvoří jednotlivé volby hráče.

Příklad Zde uvedu příklad extenzivní formy hry z [5]. Jedná se o tzv. Rozdělovací hru. Představte si bratra a sestru, kteří musí následovat daný postup při rozdělování dvou stejných dárků. Nejdříve bratr nabídne rozdělení dárků, které může být jedno ze tří: nechá si oba dárky, dá sestře oba dárky nebo si každý nechají jeden. Poté se sestra rozhodne, zda chce přijmout rozdělení, či ho odmítnout. Když ho přijme, oba dostanou dárky podle rozdělení, pokud nepřijme, ani jeden nedostane žádný dárek. Za předpokladu, že oba sourozenci si dárky cení stejně, vypadá vyjádření hry podle obrázku 3.2.



Obrázek 3.2: Příklad extenzivního vyjádření hry - Rozdělovací hra

3.2 Algoritmy používané pro hry

Algoritmy používané pro hry převážně pracují s herním stromem, jenž je popsán v kapitole 3.1. Jedním z předpokladů, který je třeba zmínit, je, že algoritmy pracují s racionálně uvažujícím protivníkem. Tedy že protivník bude hrát tah, který je pro něj nejlepší. Popis algoritmů je převzat z [4]

3.2.1 Minimax

Jeden z nejzákladnějších algoritmů je tzv. "Minimax". Tento algoritmus se snaží při tahu hráče najít situaci, která pro něj bude nejlepší a při každém soupeřově tahu situaci, která pro něj bude nejhorší. Hodně algoritmů, jež na "Minimax"navazují, se dále snaží zrychlovat prohledávání herního stromu. Ty popíši v následujících podkapitolách.

3.2.2 Alfa-beta

První z algoritmů navazujících na Minimax je Alfa-beta prořezávání. Jeho hlavním cílem je snížit počet navštívených a ohodnocených uzlů, které se procházejí při použití Minimaxu. Drží si hranice, alfu pro nejlepší známou maximální hodnotu a betu pro nejlepší známou minimální hodnotu (odtud vznikl i název pro tento algoritmus). Pomocí těchto hodnot se vždy zjistí, zda má vůbec cenu prohledávat následníky daného uzlu. Díky daným hodnotám tedy již neprohledává stavy, které mají hodnotu horší než alfa či beta, a "prořezává"tedy herní strom.
V tomto algoritmu velmi záleží na pořadí, v jakém se tahy prohledávají. V ideálním případě má Alfa-beta časovou náročnost dvakrát lepší než Minimax, v nejhorším může dosáhnout stejné časové náročnosti.

3.2.3 Negamax

Upravená varianta Minimaxu. Používá se pro hry s nulovým součtem (zisk jednoho hráče se rovná ztrátě druhého hráče). Využívá toho, že $max(a, b) = -min(-a, -b)$, tudíž zjednodušuje algoritmus.

3.2.4 Negascout

Vychází z algoritmů Negamax a Alfa-beta. Je pro něj velmi důležité správné uspořádání uzlů. Předpokládá, že nejlepší tah je brán jako první, proto ostatní prohledává s nulovým okénkem (kde alfa=beta). Pokud postup selže, tedy nějaký z dalších tahů se ukázal jako lepší než první, prohledávání pokračuje jako normální Alfa-beta. Pokud ovšem byl opravdu první tah nejlepší, ušetří se prohledávání dalších uzlů. Správného pořadí uzlů se dosahuje např. pomocí mělkého prohledávání. To každý uzel jednoduchým prohledáním, např. jenom v potomcích, ohodnotí a pomocí tohoto ohodnocení pak uzly seřadí.

3.2.5 Monte Carlo Tree Search

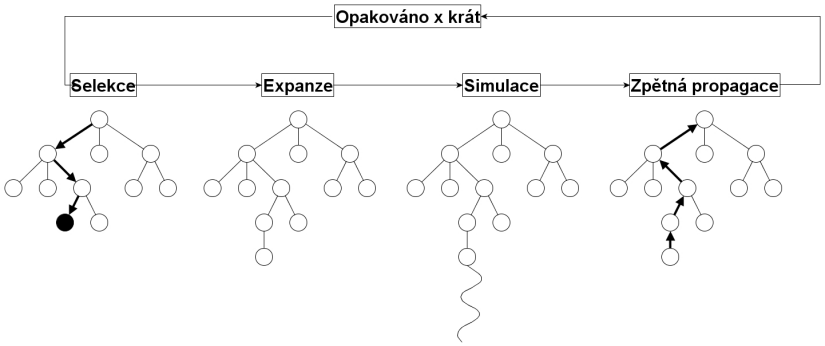
Na trochu jiném principu, než představené algoritmy, funguje algoritmus Monte Carlo Tree Search, který vám představím v této sekci.

3.2.5.1 Vznik a úspěchy algoritmu

Algoritmus, jenž je v této práci použit, se nazývá Monte Carlo Tree Search. Tento algoritmus patří mezi nejnovější algoritmy používané v současné době. Jeho princip je relativně jednoduchý a přesto má úspěchy tam, kde jiné algoritmy selhávají. Daří se mu i v hrách, kde je prozatím počítač horší než člověk, například Go, KriegSpiel, apod. Většina algoritmů se také potýká s problémy při velkém větvení herního stromu, s čímž si Monte Carlo Tree Search dokáže, díky tomu, jak funguje, poradit.

3.2.5.2 Princip a fungování algoritmu

Monte Carlo Tree Search je vlastně "best-first"prohledávání, což znamená, že při prohledávání se jako první bere vždy nejlépe vypadající uzel podle určitého pravidla. Dá se rozdělit do 4 částí, které jsou zobrazeny na následujícím obrázku(3.3):



Obrázek 3.3: Fáze Monte Carlo Tree Search - původní obrázek: [1]

Selekce V selekci se začíná v kořeni stromu. Postupně podle určité heuristiky, která je popsána v následujícím odstavci o UCT, se vybírá vždy nejvhodnější potomek. Je třeba zvolit vhodný poměr mezi prozkoumáváním nových stavů a dalším prohledáváním starých. Toto se opakuje, dokud se nedorazí do listu stromu.

UCT Vybírání uzlu se neprovádí úplně náhodně. Slouží k tomu UCT[2] metoda (Upper Confidence bounds applied to Trees). Tato metoda vybírá více uzly, které se zdají být lepší a tím pádem je potřeba je více prozkoumat, zároveň ale nezanedbává horší uzly. Používá k tomu vzorec, jenž bere v potaz počet průchodů rodiče daného uzlu (parent.passes), počet průchodů daným uzlem (node.passes), vlastní ohodnocení daného uzlu (evaluation) a také konstantu, pomocí níž lze určit, jak se bude prohledávat. Při nízkých hodnotách této konstanty se prohledávají spíše dobře ohodnocené stavy, při vysokých se naopak prohledávají i hůře ohodnocené stavy. Matematické vyjádření vzorce:

$$UCT = evaluation + constant * \sqrt{\ln(\frac{parent.passes}{node.passes})} \tag{3.1}$$

Expanze Když se algoritmus dostane do uzlu, který ještě není uložen ve stromu, dojde k jeho přidání do postupně stavěného stromu. Tím pádem je vytvořen nový uzel při každém cyklu algoritmu.

Simulace Z nově přidaného uzlu je odsimulována hra. Simuluje se buď do finálního stavu hry, nebo do předem určené hloubky. Tahy se při simulaci většinou vybírají úplně náhodně. Dá se použít i jistá heuristika, aby se vybíraly lepší, tedy více pravděpodobné tahy.

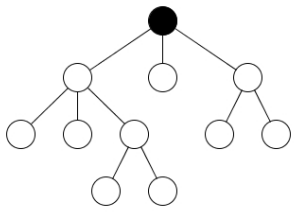
Zpětná propagace Po dokončení simulace probíhá zpětná propagace výsledku. Výsledek, který je dosažen v simulaci, se zpětnou rekurzí propaguje přes všechny rodiče až do kořene stromu.

3.2.5.3 Finální výběr tahu

Většinou se vybírá tah, jímž se nejvíce procházelo. Díky daným pravidlům to totiž většinou bývá i nejlépe ohodnocený tah. Velký vliv na počet průchodů má konstanta použitá ve vzorci UCT metody.

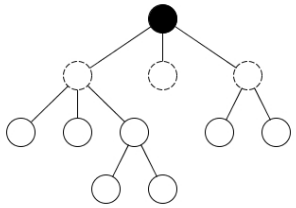
3.2.5.4 Příklad běhu jednoho cyklu algoritmu

Začíná se vždy v kořeni herního stromu.



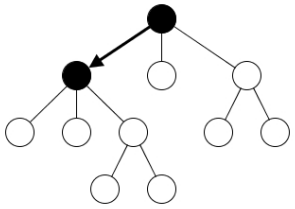
Obrázek 3.4: 1.krok - kořen

Zde se rozhoduje např. hráč 1, má na výběr z několika tahů.



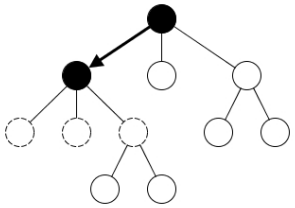
Obrázek 3.5: 2.krok - výběr z možných tahů

Použije UCT vzorec a vybere z nich nejvhodnější tah.



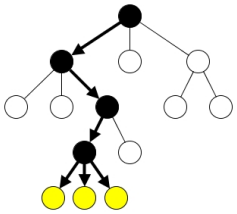
Obrázek 3.6: 3.krok - výběr nejvhodnějšího tahu

Nyní vybírá hráč 2 ze svých možných tahů, opět použije UCT pro vybrání nejvhodnějšího.



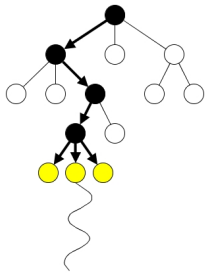
Obrázek 3.7: 4.krok - výběr z možných tahů 2. hráče

Takto se hráči střídají, dokud se nedostanou do uzlu, jehož potomci ještě nejsou uloženi ve stromu. Jeho potomci se tedy přidají.



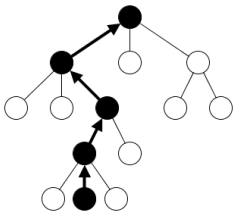
Obrázek 3.8: 5.krok - nový uzel

Z uzlu, který ještě nebyl navštíven a přišla na něj řada, se spustí simulace do konečného stavu hry či do určitého množství tahů. Zpět se propaguje výsledek hry.



Obrázek 3.9: 6.krok - simulace

Po získání výsledku se jeho hodnota vypropaguje zpět přes rodiče až do kořene stromu.



Obrázek 3.10: 7.krok - zpětná propagace

Takto běží smyčka, dokud je čas na výpočet, nebo na předem nastavený počet iterací. Pokud již došel čas či iterace, vybírá se nejvhodnější tah na zahrání, opět podle UCT.

Kapitola 4

Popis implementace

V této kapitole jsou zachyceny nedůležitější body v implementaci. Je zde představena základní implementace algoritmu, rozhraní pro hru a vypsány problémy, se kterými jsem se setkal při implementaci algoritmu pro duchy.

4.1 Algoritmus

Základní algoritmus jsem implementoval podle kroků, jež byly představeny v předchozí kapitole. Tyto 4 kroky tedy běží ve smyčce, pomocí které se kontroluje, kolik iterací proběhne. Jsou zde tedy funkce UCT, expand a doBeamAndPropagateScoreToParents (tato funkce zastává poslední 2 kroky, tedy simulaci a zpětnou propagaci). Uvedené funkce jsou popsány v jednotlivých odstavcích.

UCT Tato funkce slouží ke zvolení nejvhodnějšího uzlu, jímž se má dále ve stromu pokračovat. K tomu je používán vzorec UCT, popsán v 3.2.5.2. Pokud se zde narazí na uzel, který ještě nebyl ohodnocen, je vybrán právě ten. Je volána ve smyčce, která běží, dokud vybraný uzel má potomky. Pokud potomky nemá, znamená to, že jsme dorazili do listu a je třeba provést expanzi.

expand Slouží k posunutí do dalšího stavu hry a přidání potomků vybraného uzlu do stromu. Je vrácen první uzel, který byl přidán, a na něm dále probíhá simulace. Mohl by být vrácen kterýkoli z uzlů, jelikož ještě ani jeden z nich není ohodnocen a nelze určit, který je vhodnější.

doBeamAndPropagateScoreToParents V této funkci probíhá simulace z uzlu vráceného v expanzi. Dále je zde také provedena zpětná propagace získaného výsledku do všech rodičů uzlu, ze kterého proběhla simulace.

4.2 Doména

4.2.1 Implementace rozhraní

Od roku 2011 běží soutěž "The Ms. Pac-Man vs Ghosts competition"[3], která pro Ms. Pac-Man vyvinula rozhraní. To je napsané kompletně v Javě. Jednotliví soutěžící měli za úkol napsat pouze ovládání pro Ms. Pac-Man, případně duchy. Celý zbytek logiky hry a její běh má na starosti již zmíněné rozhraní. Toto rozhraní využívám i já.

4.2.2 Popis implementace rozhraní

Rozhraní zastřešuje celý běh hry, jak prostředí, tak hráče. Hra si drží bludiště jako graf. Jednotlivé uzly jsou místa v bludišti, myšleno přesnou pozicí. Pozice pilulek jsou přiřazeny k jednotlivým uzlům. Všechny tyto informace jsou načteny na začátku hry ze souboru, takže přístup k nim v průběhu hry je velmi rychlý. V tomto rozhraní je řada nastavení pro samotnou hru a dále také obsahuje množství užitečných metod. Např. vzdálenosti mezi jednotlivými uzly jsou již dopředu načteny a stačí se na ně pouze dotázat.

4.3 Moje implementace Monte Carlo Tree Search

4.3.1 Počáteční implementace Monte Carlo Tree Search

V počáteční implementaci jsem napsal úvodní smyčku algoritmu, jak je popsáno v 4.1. Jednotlivé kroky byly posuny o jednu pozici dále v grafu, to znamená stejně jako v připraveném rozhraní. Vytvářelo se ale velké množství uzlů, kdy se duchové nacházeli v uličce a nemohli se rozhodovat. Pokud vznikl stav, kdy se v uličce nacházel Pac-Man, měl na výběr pokaždé 2 akce. Tím pádem byla tato implementace pomalá a vytvářelo se mnoho zbytečných stavů. Na přechod mezi jednotlivými křižovatkami se vytvořilo 20-30 nových stavů pro Pac-Mana i pro ducha.

4.3.2 Předělaná reprezentace bludiště

V předělané reprezentaci bludiště jsem použil reprezentaci používanou v rozhraní a upravil ji. V bludišti jsem ponechal pouze uzly, které reprezentovaly křižovatky. K těmto uzlům jsou dopočítány informace o jejich sousedních křižovatkách. Každá z křižovatek má všechny informace o jejích sousedech potřebné k její identifikaci, např. vzdálenost, počet pilulek, apod. Bludiště jsem si tedy zjednodušil a postavil ho jako graf křižovatek. Tím pádem se duchové rozhodují jenom na křižovatkách a ne na jednotlivých uzlech v uličce mezi křižovatkami. Nevýhodou ovšem je zanedbání či zjednodušení některých situací. Například Ms. Pac-Man může měnit směr také jen na křižovatkách a snědení power pilulky není na přesném místě jako v původní reprezentaci.
Touto úpravou se tedy změnil i vytvářený herní strom, kde nyní na sebe navazují pouze pozice, kdy je jeden z hráčů na křižovatce a musí se rozhodnout. Při použití původní reprezentace docházelo i k výskytu pozic, v nichž nebyla žádná možnost volby.
Aplikací této reprezentace se zmenšil herní strom. Zároveň se pomocí smazání uzlů, kde

se duch rozhodoval, i když neměl možnost volby, eliminovaly i irelevantní strategie. Díky tomu se hra dostane při simulaci mnohem dále. U stavění stromu se při stejném množství vytvořených uzlů dostane též několikanásobně dál, neboť jsou smazány uzly bez možnosti volby. Vzhledem k přesunům pouze mezi křižovatkami a ne jednotlivými uzly se v simulaci při stejném množství opakování dostane mnohem blíže finálnímu stavu hry, než tomu bylo v předchozí variantě.

4.3.3 Reprezentace uzlu

Uzel v herním stromu reprezentuje třída Node. V ní se drží informace potřebné pro výpočet hodnoty z UCT vzorce, tedy průměrné skóre, počet průchodů, odkaz na rodiče, konstanta ze vzorce. Dále se zde ještě drží tahy, které jsou potřeba pro posun hry z kořene do tohoto stavu hry. Dále zda je na tahu duch či Pac-Man, což je pomocný atribut k fungování mé implementace.

4.3.4 Reprezentace herního stromu

Pro ukládání herního stromu využívám Hashmapu. Jako klíč se zde používá Node a vrácená hodnota je list jeho potomků.

4.3.5 Provedené úpravy oproti původní hře

Pro lepší běh a testování jsem upravil počet životů Pac-Mana na 1 život. V původní verzi měl Pac-Man 3 životy.
V původní hře existuje také určitá pravděpodobnost, že všichni duchové změní svůj směr. Tuto pravděpodobnost jsem nastavil na 0, aby nenastávala.

4.4 Kooperace duchů

Vytvořil jsem 4 verze algoritmu, které jsou popsány v následujících podkapitolách.

4.4.1 Verze pro samostatného ducha

V této verzi se vždy při běhu algoritmu vytváří strom pouze z tahů vybraného ducha a tahů Pac-Mana. Tahy ostatních duchů se do herního stromu nevkládají. Pokud se dostane na řadu jiný duch, bere se vždy jeho první možný tah.

4.4.2 Verze pro dvojice duchů

Ve verzi pro dvojice duchů má každý duch přiděleného druhého ducha, na něhož se ohlíží při rozhodování, tedy vytváření herního stromu. Má tedy lepší odhad, jaký tah pravděpodobně druhý duch z dvojice zahraje. V tom je velká výhoda oproti první verzi, kde se bere vždy první možný tah.

4.4.3 Verze pro trojici duchů

Tato verze je prakticky stejná jako předchozí, avšak nyní jeden duch bere v úvahu vždy 2 další duchy.

4.4.4 Verze pro všechny 4 duchy

V této verzi je vytvořen nový uzel při rozhodování každého ducha. Tím pádem se vždy vybírá nejvhodnější možný tah při rozhodování.

4.4.5 Implicitní a explicitní koordinace

Implicitní koordinaci jsem řešil tak, že se v ghostControlleru (třída, která počítá a vrací tahy pro duchy) drží pole s nejlepšími spočtenými tahy duchů. Tyto tahy pak může použít další duch při rozhodování. Uvedu příklad z verze pro samostatného ducha:
Duch 1 se rozhoduje ještě bez akcí ostatních duchů, protože se dostal na řadu jako první. Po skončení výpočtu ale uloží svých několik prvních nejlepších tahů do pole. Poté, když se rozhoduje duch 2 a dostane se do situace, kdy může použít uložený tah, použije ho a tím pádem zlepší svůj odhad.
Implicitní koordinace funguje v prvních 3 verzích, ve 4. verzi pro 4 duchy je logicky explicitní koordinace, protože každý duch zahrnuje všechny ostatní do svého výpočtu.

Kapitola 5

Experimenty

V této kapitole jsou představeny provedené experimenty s mojí implementací.

5.1 Konstanta

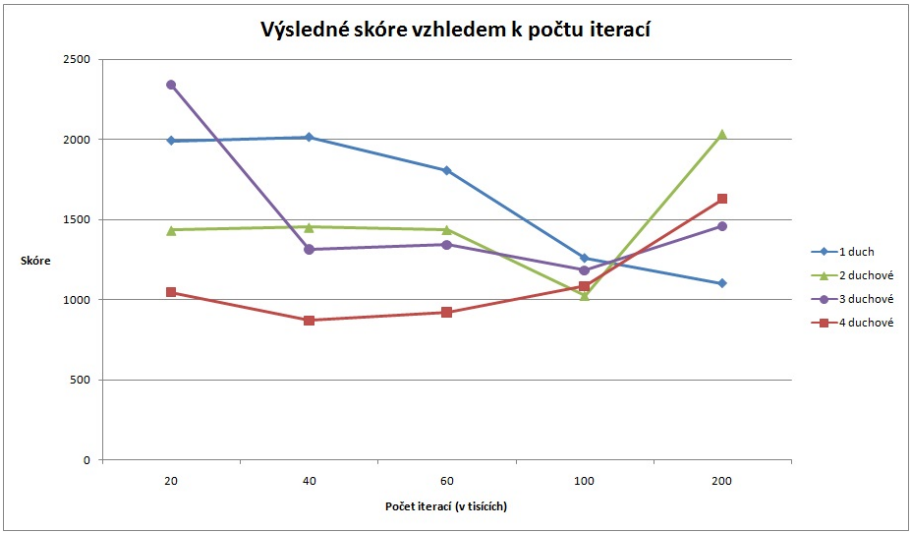
V první řadě jsem potřeboval stanovit, jakou hodnotu by měla mít konstanta používaná ve vzorci UCT. Toho jsem dosáhl opětovným pouštěním algoritmu a zjišťováním, jak se dělí průchody mezi jednotlivé potomky uzlu. Konstantu jsem měnil, dokud jsem nezajistil dělení vhodné pro použitý algoritmus.

5.2 Typy experimentů

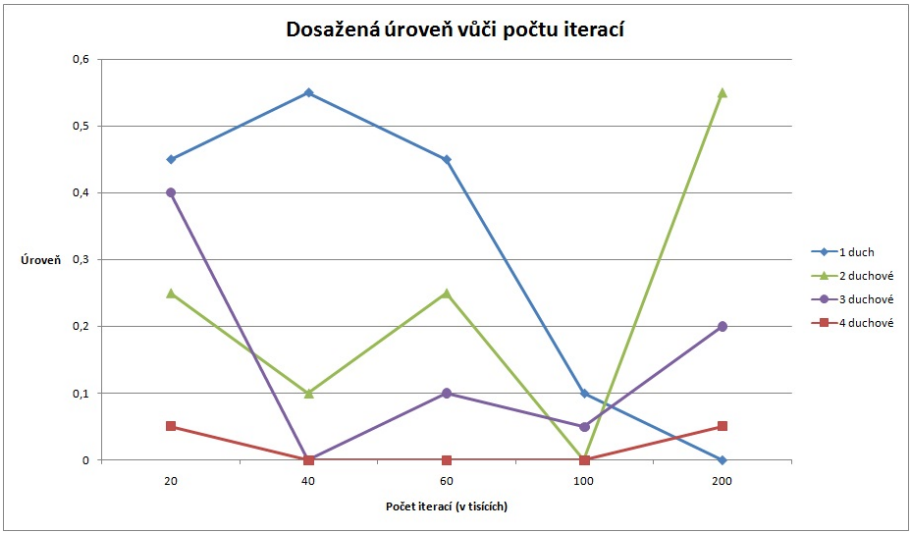
Nejprve jsem se rozhodl zkusit zvyšovat počet iterací algoritmu a sledovat jaký to má vliv na výsledné skóre hry. Logicky by se měly výsledky zlepšovat se zvyšováním iterací, protože se prohledá více stavů hry a tím se zpřesní odhad výhodnosti tahů. Dále jsem zkusil algoritmus neomezovat počtem iterací, ale časem. I zde by měla vyšší časová dotace pomoci k lepším výsledkům. Na závěr jsem se pokusil měnit počet uložených tahů při implicitní kooperaci.

5.3 Změna počtu iterací

Začal jsem na 20 tisících iteracích. Tuto hodnotu jsem postupně zvětšoval na 40, 60, 100 a 200 tisíc. Nejprve jsou zobrazeny grafy a pod nimi se nachází rozbor výsledků. Pro každou možnost proběhlo 20 her.



Obrázek 5.1: Výsledné skóre vzhledem k počtu iterací



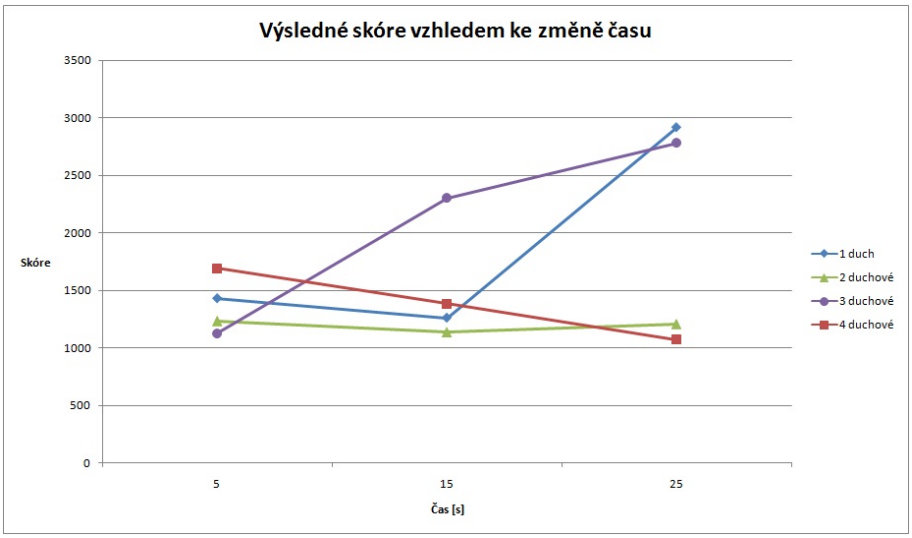
Obrázek 5.2: Dosažená úroveň vůči počtu iterací

V prvním grafu je zřetelné, že do počtu 100 tisíc iterací se algoritmy chovají očekávaně. Výsledné skóre se tedy snižuje, pouze u algoritmu pro 4 duchy se podstatně nemění a drží se na podobné úrovni. Výsledky pro 200 tisíc iterací se rapidně zhoršily. Od této hranice iterací se zdají výsledky zkreslené, které neberu do celkového hodnocení algoritmu. Druhý je graf zobrazující průměrnou dosaženou úroveň. Nultá úroveň znamená, že se Pac-Man nedostal přes první úroveň. Jelikož je graf vytvořen ze stejných her jako předchozí, jsou opět zřetelné očekávané výsledky do počtu 100 tisíc iterací.

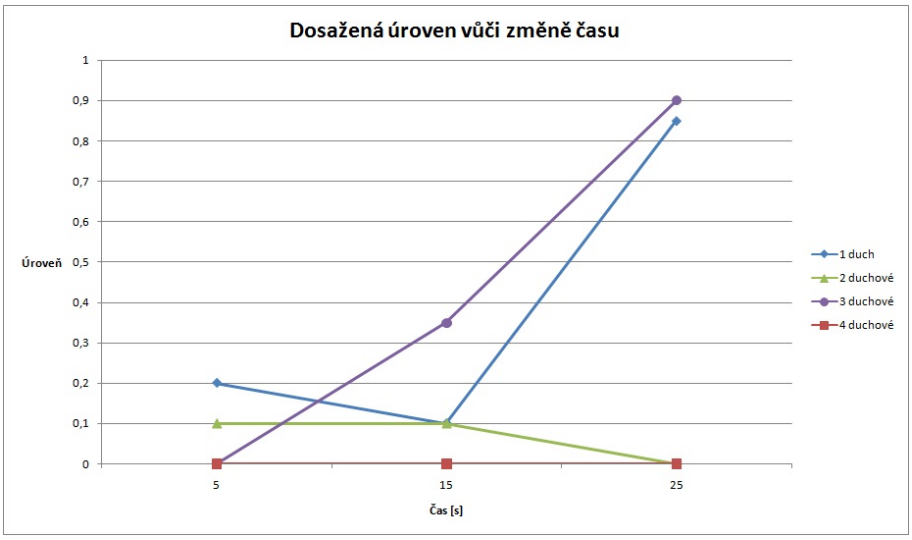
Z těchto grafů je tedy vidět, že již od začátku byl nejlepší algoritmus pro 4 duchy s explicitní koordinací. Ostatní algoritmy se mu se zvyšujícím počtem iterací začínaly v úspěšnosti přibližovat.

5.4 Změna času

V tomto případě algoritmus běžel, dokud měl vyhrazen čas. Začal jsem na čase 5 sekund a postupně zvyšoval na 10 a 15 sekund.



Obrázek 5.3: Výsledné skóre vzhledem ke změně času



Obrázek 5.4: Výsledné skóre vzhledem ke změně času

Bohužel jak je vidět z výsledků, pro 3 duchy se algoritmus chová nestandardně po celou dobu a pro 1 ducha při velkém množství času také. Ostatní výsledky se zdají být odpovídající. Pro 4 duchy se výsledky zlepšují, i když ne rapidně. Pro 2 duchy se výsledky drží na podobné hranici. Již od 5 sekund se tedy algoritmy výrazně nezlepšují, tím pádem se pravděpodobně blíží ke svému maximu.

5.5 Změna počtu uložených tahů

Změna počtu uložených tahů se měla projevit zlepšením výsledků u implicitní koordinace. Zkoušel tedy jsem zvyšovat počet uložených tahů. Bohužel zde byly výsledky nevypovídající. Místo toho, aby se výsledky postupně zlepšovaly přidáváním uložených tahů, naopak se zhoršovaly. Tím pádem nemohu posoudit, jak velký vliv mělo přidávání počtu uložených tahů.

Kapitola 6

Závěr

V této práci jsem se pokusil implementovat kooperující tým duchů do hry Ms. Pac-Man. Mým záměrem bylo rozebrat problémy, jež se vyskytly při jeho tvoření, a také zjistit, jak byl tento algoritmus vhodný pro danou doménu.

Nejprve je zde rozebrána hra samotná, představena její historie a doména. Přiblíženy jsou specifikace a vlastnosti dané pro tuto hru. Postupně jsou vysvětleny prvky z herní teorie a algoritmy používané pro hry. Rozsáhlejší část se týká Monte Carlo Tree Search. V kapitole věnované implementaci jsem se zabýval hlavně problémy, na které jsem narazil při vytváření své implementace.

Z výsledku v experimentech lze vypozorovat, že explicitní koordinace je od začátku lepší než implicitní. Ta se jí ale s přibývajícími iteracemi přibližuje. Od určitého momentu bohužel výsledky nejsou relevantní a nelze je použít, vycházím tedy z těch správných. Pokud se jedná o celkové hodnocení, podle mého názoru se Monte Carlo Tree Search spíše nehodí pro řešení tohoto problému, protože není dostatečně rychlé. Pro tuto doménu bych spíše použil více specializovaný algoritmus.

Tato práce může dále sloužit komukoli kdo se rozhodne aplikovat Monte Carlo Tree Search či jiný algoritmus do podobné hry. Díky ní lze zjistit, kde mohou vyvstat problémy a na co se zaměřit.

Příloha A

Obsah CD

Přiložené CD obsahuje projekt s hrou a spustitelný jar soubor s ukázkou. Níže je popsán návod jak s daným projektem pracovat a soubory pracovat.

- $/mspacman_v2.1.2/game/Exec.java$ - třída sloužící k puštění celého projektu. V ní se určuje, který z daných algoritmů poběží a zda formou normální hry, nebo formou experimentu bez grafického prostředí. Jednotlivé možnosti jsou zakomentovány a po jejich odkomentování je možno spustit hru s daným nastavením.
- Implementace algoritmu a bludiště se nachází v $/mspacman_v2.1.2/game/entries/ghosts/$. Okomentována je třída MazeGhosts, ostatní jsou prakticky stejné, jenom se liší v tom pro kolik duchů se strom staví, případně zda hra probíhá na iterace či na čas.
- Na CD je také připraveno spustitelné jar s algoritmem pro jednoho ducha s 20 tisíci iteracemi. U tohoto jar se musí nacházet složka data.

Literatura

[1] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. 2008.

[2] Levente Kocsis, Csaba Szepesvári, and Jan Willemson. Improved monte-carlo search. 2006.

[3] Philipp Rohlfshagen and Simon M. Lucas. Ms pac-man versus ghost team cec 2011 competition. 2011.

[4] Stuart Russell and Peter Norvig. *Artificial Intelligence A Modern Approach*, pages 165–180. Prentice Hall, third edition, 2010.

[5] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, Game-Theoretic, and Logical Foundations*, pages 118–119. Cambridge University Press, 1.1 edition, 2009.