VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

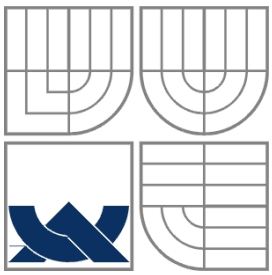
PIŠKVORKY

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

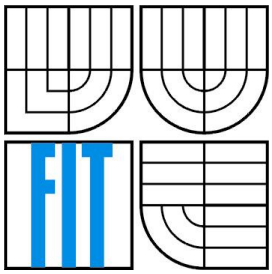
AUTOR PRÁCE
AUTHOR

MICHAL VRTÍLEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

PIŠKVORKY
FIVE IN A ROW

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL VRTÍLEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN

BRNO 2008

Zadání bakalářské práce

Piškvorky

Five in a Row

Vedoucí:

Rozman Jaroslav, Ing., UITS FIT VUT

Přihlášen:

Vrtílek Michal

Zadání:

1. Seznamte se s hrou piškvorky
2. Seznamte se s algoritmy pro programování umělé inteligence
3. Navrhněte program pro hraní hry včetně rozhraní pro řízení Manažerem pro deskové hry
4. Navržený program implementujte
5. Zhodnoťte řešení, diskutujte možná vylepšení do budoucna

Kategorie:

Umělá inteligence

Literatura:

internet

Licenční smlouva

Licenční smlouva je uložena v archivu Fakulty informačních technologií Vysokého učení technického v Brně.

Abstrakt

Tato bakalářská práce se zabývá návrhem a implementací umělé inteligence pro stolní deskovou hru Piškvorky. Hlavním cílem je vytvořit aplikaci schopnou komunikovat s Manažerem pro stolní deskové hry s pomocí síťové komunikace se schopností hrát piškvorky na kvalitní úrovni. V práci lze nalézt teoretickou část o historii a pravidlech piškvorek, o možnostech využití umělé inteligence pro tuto hru a o struktuře aplikací hrajících tuto hru. Dále se zabývá návrhem a implementací samotné aplikace, doplněné o její ovládání. V závěru jsou zhodnoceny dosažené výsledky a načrtnuty další možné směry vývoje aplikace.

Klíčová slova

manažer, brain, deskové hry, piškvorky, C#, .NET, umělá inteligence

Abstract

This bachelor's thesis describes the desing and implementation of artificial intelligence for the board game Five in a row. The main goal of this thesis is to create an application able to communicate with the Manager For Board Games with the use of network communication and able to play the board game Five in a row on a suitable level. This work contains theoretical parts about the history and rules of Five in a row and it's variations, about the possibilities of use of artificial intelligence in this game and about the structure of aplications used for playing this game. It's concers is about the actual desing and implementation of said application, including it's controls. In the end I evaluate achieved results and propose additional directions for the program development.

Keywords

Manager, Brain, Board Games, Gomoku, Five In A Row, C#, .NET, Artificial Intelligence

Citace

Vrtílek Michal: Piškvorky. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Piškvorky

Prohlášení

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

……
Michal Vrtílek
31.7.2008

Poděkování

Rád bych poděkoval především vedoucímu práce za možnost zabývat se tématem umělé inteligence a za shovívavost. Rád bych také poděkoval Jiřímu Kusákovi za vytvoření Manažeru pro stolní deskové hry, který byl základem pro tuto práci.

Obsah

Úvod

Hry jsou nedílnou součástí dnešního světa. Ty nejběžnější jsou o souboji dvou lidí, s určitými nástroji či pravidly, kde jde o nalezení schopnějšího jedince. Lidé se takto poměřovali od pradávna, s nástupem informačních technologií se však možnosti her značně rozrostly a vyvinuly. Stále je populární soupeření dvou jedinců, objevila se však i nová možnost souboje proti počítačovému protivníkovi s tzv. „umělou“ inteligencí. A protože zájem o umělou inteligenci a její využití je v posledních letech značný, v některých případech se hry změnily až do té míry, že spolu soupeří jen umělé inteligence, přeneseně jejich programátoři. Především poslední zmiňovaná možnost je předmětem mé práce, ve které se zaměřím na jednu z jednodušších her (z pohledu umělé inteligence) jménem Piškvorky (anglicky nejčastěji známá jako Five in a Row či Gomoku). Účelem mé práce tedy bude vytvořit program obsahující umělou inteligenci hrající stolní hru Piškvorky.

V první kapitole se krátce zaměřím na teorii Piškvorek, seznámení se s pravidly, dostupnými informacemi o hře a různými zajímavými detaily.

V kapitole druhé se budu zabývat možnostmi umělé inteligence pro hru Piškvorky – různými postupy, úskalími a mnou zvoleným řešením.

V kapitole třetí a čtvrté popíši, jak jednotlivé programy pro hru mohou vypadat. Jedná se především o zažité postupy, terminologii, komunikaci, záznamy her, hodnocení programů a samotné souboje či turnaje. Také se budu věnovat konkrétně Manažeru stolních deskových her, se kterým moje bakalářská práce úzce souvisí.

Kapitola pátá pojednává o vlastním návrhu mého řešení v podobě konkrétní aplikace. Kapitola šestá o jeho implementaci a kapitola sedm seznamuje s jeho ovládáním, nastavením a dalšími doplňujícími informacemi.

V závěru pak zhodnotím dosažené výsledky, nastíním vývoj do budoucna, co by se dalo vylepšit či změnit a zhodnotím také přínos této práce.

1 Teorie piškvorek

1.1 Historie

Piškvorky jsou velice stará hra, první známky o jedné z variant této hry jsou staré přes 4000 let, pocházejí z Číny, přičemž o něco pozdější údaje o této hře naznačují, že byla zhruba ve stejné době jako v Číně, známá též v antickém Řecku či předkolumbovské Americe.

Do dnešní doby se hra téměř nezměnila, nicméně existuje mnoho variant této hry, či různá doplňující pravidla.

1.2 Pravidla

Základem hry je střídání se dvou hráčů v pokládání značek na čtverečkovanou hrací plochu (vždy jen na pole, které neobsahuje žádnou značku), přičemž vyhrává ten, který jako první spojí alespoň pět svých značek za sebou v libovolném směru a to jak vertikálně tak horizontálně či diagonálně.

Variant této hry je nespočet. V České republice je tato hra známá jako Piškvorky, hraje se na libovolně velké ploše bez jakýchkoliv omezení. Jako značky se používají kolečka a křížky, a protože značky po umístění již svou polohu po celou dobu hry nemění, hrává se často na čtverečkovaném papíře. Existuje i minimalistická varianta hry na ploše 3x3, kdy stačí spojit tři své značky. V této variantě hry je velice výrazně vidět převaha začínajícího hráče, který při tzv. „perfektní hře“ nemůže prohrát, avšak ani vyhrát.

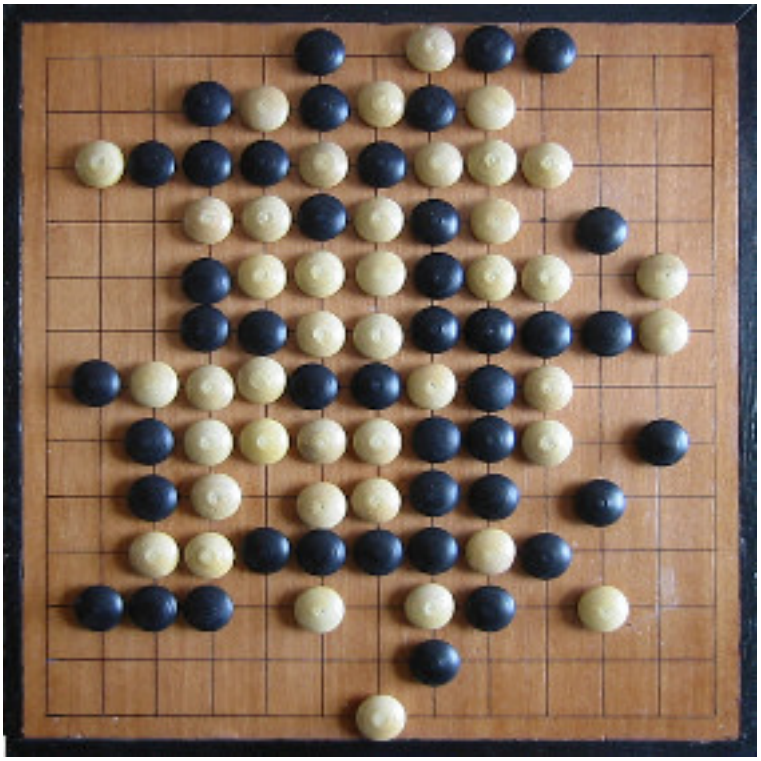
Výzkum doktora L. Victora Allise prokázal, že při perfektní hře na ploše 15x15 polí vždy vyhraje začínající hráč a při jinak velké omezené ploše či v neomezené variantě hry, začínající hráč nikdy neprohraje. Zkráceně: druhý hráč nemůže nikdy vyhrát. Pro vyvážení této výhody jsou v mnohých variantách doplňující pravidla, například:

- vítězná řada musí mít přesně pět kamenů
- vítězná řada musí mít šest a více kamenů
- hra začíná na přednastavené hrací ploše, která eliminuje výhodu začínajícího hráče
- je zakázán tah, který vytvoří současně dvě trojice s alespoň jedním prázdným políčkem na každé straně trojic,
- je zakázán tah, který vytvoří dvě čtveřice, obě s alespoň jedním prázdným políčkem na jedné ze stran každé čtveřice.
- vítězná řada pěti značek nesmí být z žádné strany ohraničena nepřátelským kamenem

Podobných pravidel je pro každou variantu mnoho. Také se používá různých typů značek a velikostí hrací plochy. Například v zemích, ve kterých je populární hra Go (jedna z nejstarších známých stolních deskových her), se používá stejná hrací plocha (19x19) a kameny bílé a černé barvy.

1.3 Informace o hře

Piškvorky jsou hra s kompletní informací, vždy jsou nám známy všechny informace ovlivňující hru a známe také všechny naše i soupeřovi přípustné tahy, neexistuje zde žádný prvek náhody. Díky tomu jsou velice vhodné pro programování umělé inteligence – vše je možné se stoprocentní jistotou propočítat.



Obrázek 1: Piškvorky hrané na 13x13 Go desce

2 Umělá inteligence pro piškvorky

Způsobů řešení umělé inteligence piškvorek není mnoho. Jediné informace, které při rozhodování o následujícím tahu máme, jsou podoba hrací plochy a indikátor vítězství (pět značek stejné barvy či tvaru v řadě). Dokážeme tedy poznat přípustné tahy (není na nich žádná značka) a kdy jedna ze stran vyhrála. Z těchto informací je potřeba zjistit, který z přípustných tahů má nejvyšší pravděpodobnost dosáhnout vítězství.

2.1 Ohodnocení tahu/hry a výběr nejlepší možnosti

Ohodnocování určité části hry je základem většiny piškvorkových programů a tyto jsou jen tak dobré, jako jejich ohodnocovací funkce. Indikace stavu hry výhra/prohra je nedostačující a program se rozhoduje na základě toho, jak si myslí, že je situace pro něj výhodná. Vybírá tah na pole, které je podle něj v aktuální situaci nejlepší, či který posune hru do takové situace, která je podle ohodnocovací funkce nejvýhodnější a má nejvyšší šanci dosáhnout vítězné kombinace.

Ohodnocovací funkce je většinou založena na rozpoznávání určitých seskupení značek na hrací ploše, počítá sousední značky stejného druhu, jejich ohraničení (je-li možné řadu dále rozšiřovat jedním,či dvěma směry, či je-li ohraničena nepřátelským kamenem nebo okrajem hrací plochy) a informace ukládá ve formátu, z kterého je pak možné vybrat nejlepší možnost.

Funkce může projít celou hrací plochu a hledá na ní známá seskupení (dvojice, trojice, čtveřice...) a podle počtu a typu nalezených seskupení určí, který hráč a v jaké míře vyhrává (má větší šanci dosáhnout výhry). Výsledkem je jedna hodnota určující situaci na hracím poli, kladná čísla znamenají výhodu právě hrajícího hráče, záporná jeho oponenta. V druhém případě funkce prochází jednotlivá pole, v jejich okolí hledá stejná známá seskupení a u každého pole zaznamenává, jakou má toto pole šanci dosáhnout vítězného seskupení a jak je proto vhodné pro následující tah (některého konkrétního hráče). Po ohodnocení vznikne hrací plocha, jejíž každé pole je ohodnoceno podle toho, jak výhodné je na něj táhnout buď pro právě hrajícího hráče nebo pro jeho soupeře.

Výhodou těchto metod je to, že program umí rozpoznat možnost výhry a dosáhnout jí a také rozpozná kdy se o výhru pokouší nepřítel a je schopen se efektivně bránit. V případě kvalitní ohodnocovací funkce již lze proti slabému protivníkovi vyhrát, avšak nejlepší tah v kole nebývá vždy nejlepší cestou k vítězství a v závislosti na ohodnocovací funkci také není garance, že tah je opravdu nejlepší (například: je lepší vytvořit dvě dvojice, či jednu trojici?).

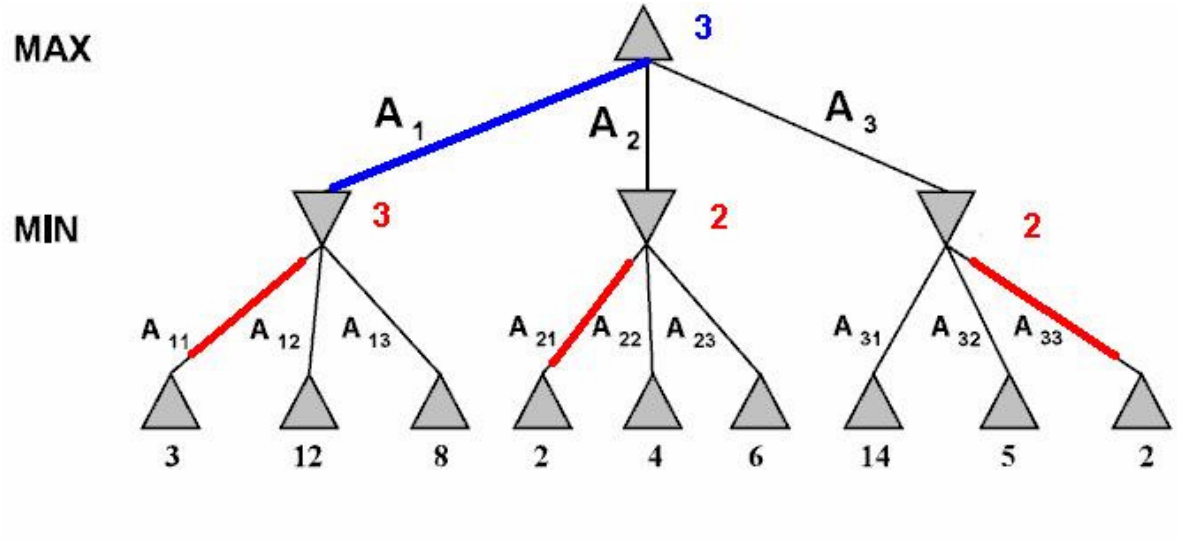
2.2 Prohledávání stavového prostoru

Předchozí systém se dá drobně upravit tak, že program neprozkoumává pouze aktuální situaci, ale posune se o jeden půl-tah dál zahráním vybraného nejlepšího tahu nanečisto a pokusí se vyhodnotit situaci z pohledu soupeře, aby zjistil, jak bude muset v příštím tahu reagovat. Při omezené hrací ploše lze vytvořit strom všech možných tahů až do konce partie a v případě piškvorek vybrat větev, která jednoznačně vede k vítězství. Je dokázáno, že takováto cesta při velikosti hracího pole 15x15 existuje, nicméně její nalezení by i nejvýkonnějším počítačům trvalo několik desítek či stovek let, proto se tato metoda vylepšuje, omezuje či kombinuje s jinými.

2.2.1 MIN-MAX

Toto je metoda prohledávání stavového prostoru s omezením na určitý počet tahů (půltahů). Vytvoří se strom na daný počet tahů dopředu (stejně jako v předchozím případě, avšak po určitém počtu rekurzí se strom dále nerozrůstá) a situace na spodu stromu se ohodnotí ohodnocovací funkcí. Hráč, který byl v prvním tahu na řadě je označen MAX, oponent MIN. MAX se snaží vybírat větve s nejlepším ohodnocením, MIN s nejhorším. Výběr se provádí od nejspodnější části stromu v jednotlivých uzlech a postupuje se nahoru až do prvního tahu. MIN-MAX tedy vybírá nejhorší možnou cestu k výhře. Ukázka funkce MIN-MAX na obrázku 2.

Náročnost této metody na výpočetní výkon již značně narůstá, při b možných tazích v každém uzlu a hloubce prohledávání je časová náročnost $O(b^m)$. Není proto možné prohledávat strom do příliš velké hloubky, je potřeba hloubku rozumně omezit (což ale není nejlepší řešení, program by měl vidět co možná nejvíce dopředu), popřípadě omezit množství možných tahů v jednom uzlu.



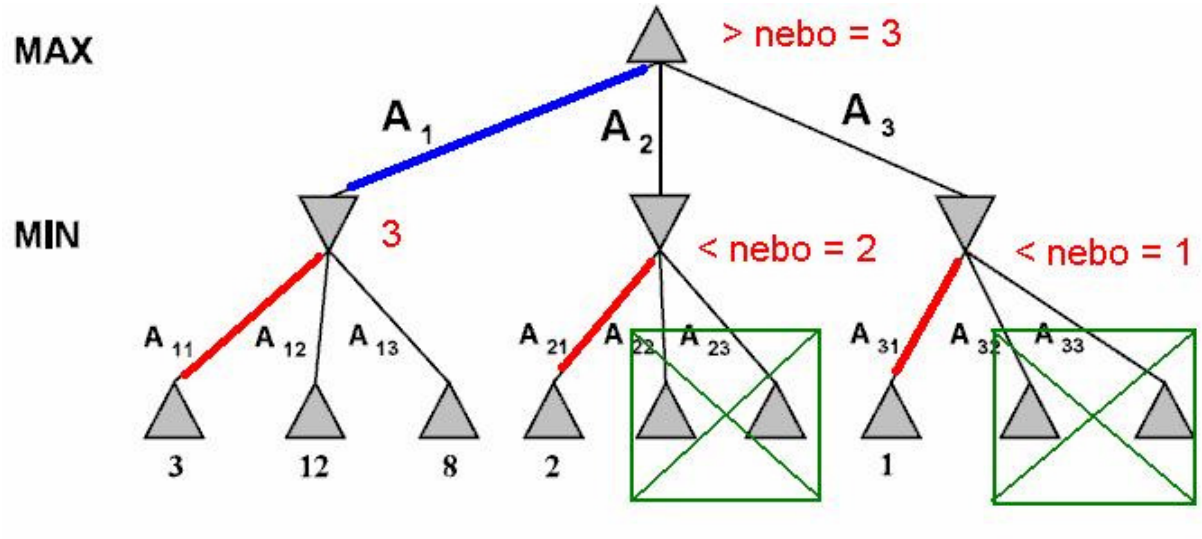
Obrázek 2: Ukázka ohodnoceného MIN-MAX stromu s výběrem větví

2.2.2 ALFA-BETA

ALFA-BETA je rozšířením MIN-MAXu, princip spočívá v tom, že program nepočítá variantu, která, ať už dopadne další propočet jakkoliv, stejně neovlivní výběr nejlepšího tahu, či víme-li již, že nějaký tah je špatný (soupeřova odpověď vede k výhře), nemá smysl dále propočítávat tuto část stromu. Tímto způsobem jsou „zahazovány“ vždy jen varianty, které jsou pro nás prokazatelně nepoužitelné, je proto využití tohoto rozšíření velice vhodné, šetří se tím výpočetní náročnost. Názorný příklad na obrázku 3.

Princip metody: víme-li, že v nejlevější části stromu vybere oponent tah s ohodnocením 3, nemá smysl dále dopočítávat tahy, kde může vybrat ohodnocení ještě nižší, tyto části stromu naším tahem určitě nevybereme.

Účinnost tohoto vylepšení je závislá na tom, v jakém pořadí větve prohledáváme – pokud bychom začali v místě, které vrací nejlepší tahy, bylo by prohledávání velmi rychlé.



Obrázek 3: ALFA-BETA algoritmus

2.2.3 Vyhledávání ohrožení

Tato metoda vychází z předpokladu, že pro vítězství je nutné dosáhnout situace, kdy je oponent nucen bránit současně na dvou místech. Z praxe vyplývá, že dosažení této situace předchází řada jiných situací, kdy je oponent nucen v každém tahu bránit. Algoritmus tedy uvažuje pouze ty své tahy, které nutí oponenta k obraně a snaží se najít takovou sekvenci, která vede k vítězství. V průběhu vytváření stromu těchto sekvencí uvažuje pouze obranu právě vytvořeného ohrožení a teprve po nalezení vítězné sekvence kontroluje, nevznikla-li v průběhu sekvence možnost pro protiútok oponenta a je-li tedy tato sekvence opravdu vítězná.

Algoritmus je velice útočný, jeho základ tkví v přesvědčení, že hráč, který je právě na tahu a nemusí bránit, je hráčem útočícím a existuje alespoň jedna vítězná sekvence. Má základ ve hře profesionálních hráčů, efektivní je v částečně rozehrané partii. Většinou je nutné jej kombinovat s jinými metodami, ovšem existuje-li sekvence vedoucí k vítězství, je schopen ji ve velice krátkém čase najít.

2.2.4 Databáze tahů

Vhodným rozšířením je, zvláště pro začátek partie, databáze tahů. V určité situaci nám databáze řekne, že nejlepší tah v dané situaci je na jedno konkrétní pole – například v piškvorkách je začátek partie velice důležitý a není mnoho možností, jak táhnout na nejlepší pole.

Je nutné mít databázi kvalitní, neboť doporučuje-li špatné tahy, na které program vždy táhne, je šance na výhru nízká. Při databázi nejisté kvality je proto vhodné porovnávat doporučený tah s ohodnocovací funkcí, je-li zvolený tah jedním z těch lepších (ale ani to nám nezaručí, že je tento tah opravdu dobrý - potřeba opravdu kvalitní databáze je žádoucí, je lepší mít menší množství prověřených tahů než spoustu těch nejistých).

V případě kvalitní prověřené databáze začínajících tahů však lze počátek partie velice urychlit a souboj s podřadným oponentem tak velice rychle skončit. Základem je dát hráči pozici vhodnou k rozšiřování útočných formací či vhodně reagovat na počáteční ohrožení. Předpoklad začínajícího hráče je pozice útočníka proti bránícímu druhému hráči a chyba na startu partie může tuto situaci snadno otočit.

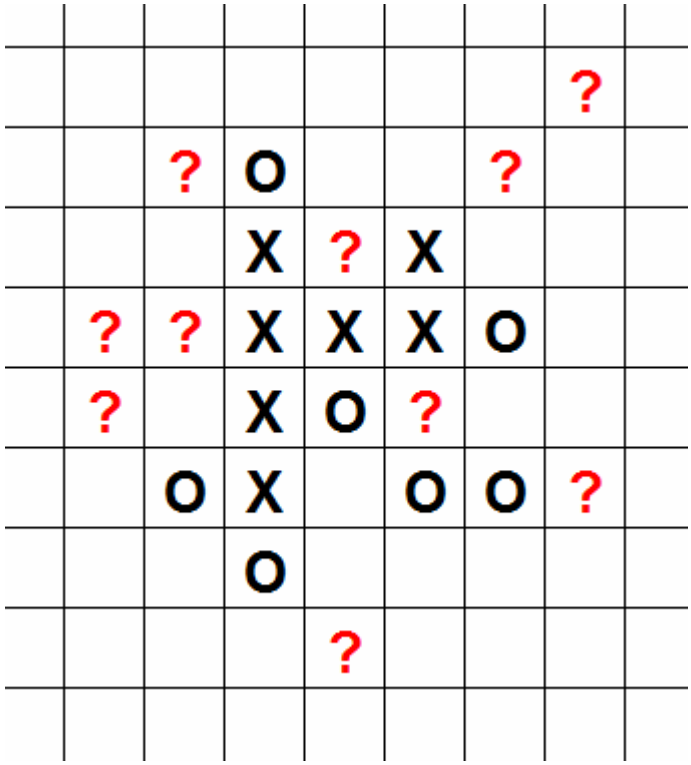
2.3 Další možná rozšíření:

2.3.1 Selektivní metoda

Při uvažování možných tahů v jednom rozhodovacím uzlu nemusíme brát v potaz všechny možné tahy - ty, které se na první pohled jeví jako bezpředmětné, vyřadíme. V piškvorkách například nemá příliš význam táhnout na políčko, které je od ostatních značek dále než počet značek potřebných pro výhru – takovéto pole do bezprostředního dění partie nezasáhne. Vyřazujeme proto všechny bezpředmětné tahy, či ještě určitý počet nejhorších tahů, či pro zjednodušení vybereme jen ty nejlepší tahy.

Tímto můžeme velice výrazně zmenšit větvení stromu (a tím i časovou náročnost), neboť nedochází k větvení mnoha desítek možností, ale jen například deseti. Navíc, při postupném vybírání nejlepších tahů, zvyšujeme šanci, že urychlíme metodu ALFA-BETA. Vystavujeme se ale riziku, že zahodíme i tahy, které by mohly vést k lepšímu výslednému ohodnocení. Také musíme pole

ohodnocovat již v průběhu větvení stromu, abychom mohli vybrat ty tahy, které se budou nadále rozvíjet a které budou ignorovány.



Obrázek 4: Selekce nejlépe hodnocených tahů

2.3.2 Transpoziční tabulky

Při prohledávání stavového prostoru můžeme narazit na miliony možných kombinací a jelikož algoritmy MINIMAX ani ALFA-BETA neuchovávají informace o již prošlých částech stromu, je velká šance, že se bude podobná herní situace opakovat a program bude stejné situace muset vyhodnocovat znovu. Jelikož jeden tah nijak nezmění stav hry na zbytku hrací plochy, je možné pouhým prohozením pořadí dvou tahů získat naprosto stejnou situaci.

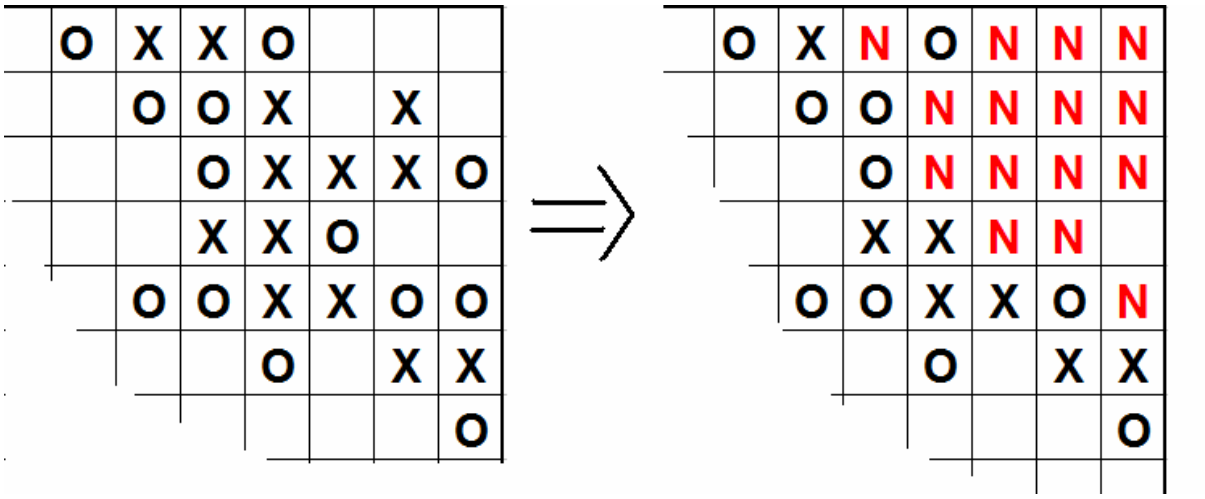
Pro urychlení výpočtu je používána tabulka s rozptýlenými hodnotami, kde jsou uloženy všechny již řešené situace. Narazí-li program na novou situaci, zjistí, zda-li již tuto situaci neřešil nahlédnutím do tabulky transpozic a najde-li ji, zachová se stejně jako v předchozím případě. Pokud ne, je nucen vypočítat nejlepší tah standardní metodou a situaci do tabulky doplnit. Nastane-li v jiné části stromu stejná situace, je již v tabulce obsažena.

Toto rozšíření je časově nenáročné neboť při použití vhodné hashovací funkce je časová složitost čtení či zápisu do tabulky konstantní. Použitím můžeme dosáhnout prohledání složitějšího stromu za menší čas.

2.3.3 Zjednodušení hracího pole

Na hracím poli často vznikají oblasti, kde již díky hranicím pole či seskupení dříve hraných tahů není možné vytvořit seskupení pěti symbolů stejné barvy či tvaru. Tyto oblasti mohou být ve všech metodách ignorovány, neboť je zřejmé, že správný tah do takovéto oblasti nepatří a všechny zde již umístěné značky nemají na hodnotu aktuální partie vliv – jedná se o tzv. mrtvá pole.

Na začátku každého tahu je hrací pole zkontrolováno, algoritmus se snaží najít všechna mrtvá pole a umístit na ně neutrální symbol značící, že na pole nelze táhnout a že jej není třeba brát v potaz pro ohodnocení pole ani pro jednoho hráče. Na obrázku 5 lze vidět názorný příklad, jak takovéto zjednodušení hracího pole vypadá. Jedná se o situaci v rohu hrací plochy, kde velice často dochází k velkým oblastem mrtvých polí, která k vítězství vést nemohou, avšak program je při každém tahu zbytečně prohledává



Obrázek 5: Zjednodušení hracího pole

2.3.4 Učení se z předchozích her

Vylepšovat lze databázi tahů či ohodnocovací funkci. V případě hry piškvorky je však třeba brát v úvahu, že spolehlivými informacemi jsou pouze výhra/prohra, tudíž lze těžko poznat, kde udělala databáze či ohodnocovací funkce chybu. Pouze v případě, kdy dosáhneme výhry nebo prohry, můžeme říct, že několik posledních tahů bylo dobrých, či špatných (a ani tady si nemůžeme být jistí, neboť chyba v tahu 5 se může projevit prohrou až v tahu 20).

Učení se je také závislé na tom, od koho se učíme. Je-li soupeř slabý, budou výsledky učení se také slabé – je proto nutné učit se od pokud možno co největšího množství soupeřů, nejlépe těch, o kterých víme, že mají kvalitní hru.

Výše zmíněné metody se spolu dají určitým způsobem kombinovat, některé z nich dosahují uspokojivých výsledků samostatně, jiné jen v určitých situacích a metody rozšiřující nejsou schopny samostatně fungovat avšak vylepšují slabá místa samostatných metod.

Vždy je třeba uvážit, zda-li vybraná metoda či rozšíření nejsou na možnosti celkového programu příliš náročné, budeme-li mít dostatek výpočetních a paměťových zdrojů a zvládá-li program hru od začátku partie až po její konec. Tyto nároky je nutné porovnat s tím, jak moc se celkový algoritmus ve hraní hry zlepší.

3 Brainy, manažery, komunikace

Jak již bylo v první kapitole vysvětleno, piškvorky jsou hra pro dva hráče, kteří se v tazích střídají. Je tedy potřeba hrát za každou ze stran, tyto strany také musí mít neustále kompletní informaci o hře a je nutné umět reagovat na tahy druhé strany.

Vzhledem k tématickému zaměření této práce budou chtít dva programátoři, zabývající se umělou inteligencí, kteří naprogramují program hrající piškvorky, provést souboj těchto dvou inteligencí. Jeden program by proto měl zaujmout pozici začínající strany, druhý strany odpovídající na první tah. Je třeba také zajistit vzájemnou komunikaci mezi nimi. Obecně uznávaný a používaný je systém „brainů“ a „manažerů“. Nejprve vysvětlení pojmů.

3.1 Brain

Jako „brain“ se označuje program, který má při hře dvou a více hráčů, přidělenu jednu ze stran a který rozhoduje o všech úkonech, vyžadujících účast jemu přiřazené strany. V praktickém příkladu to znamená, že hrají-li nějakou hru hráči (v jejich zástupu brainy) A,B,C a D, každý za jednu stejně označenou stranu, tak v situaci, kdy je potřeba se rozhodnout, jak na určitou situaci zareagují strany B a D, jsou na reakci tázány brainy B a D a žádné jiné a oba reagují pouze za svou stranu. V případě piškvorek hraje první brain, jako reakci na něj druhý brain a v tomto se střídají, dokud není určen vítěz. Jak je zřejmé, brainy fungují vždy jen jako jedna z částí celé hry (v oblasti her dvou a více hráčů) a bez jiných brainů a možnosti s nimi komunikovat jsou nefunkční.

Brain tedy obsahuje umělou inteligenci, schopnou hrát za (alespoň) jednu stranu a nějaký způsob komunikace s jinými brainy či manažerem.

Funkce brainu může být obsažena i v programu, který primárně nabízí jinou funkčnost, jako například hru člověka proti počítači. V takovémto případě je člověk nahrazen jiným programem a pro algoritmy hrající obsaženou hru se tam nic nemění, jen způsob komunikace.

3.2 Manažer

Manažer je program, na který jsou delegovány určité funkce, potřebné k sehrání úspěšného souboje mezi dvěma či více hrajícími stranami. Jeho účelem je usnadnění práce na programování softwaru pro hry, je schopen práci velice zjednodušit. Programátor se tak může více soustředit na umělou inteligenci a uživatelské rozhraní a podstatnou část věcí s tímto nesouvisející ponechat na manažeru. Většinou je umístěn na komunikační vrstvě mezi jinými programy, ve funkci arbitra celé hry. Některé příklady využití manažeru:

Komunikace mezi hrajícími stranami

Každá hrající strana je před zahájením hry připojena pouze k manažeru, který zajišťuje vzájemnou komunikaci. Manažer zpracuje příchozí informace a přeposílá je ostatním stranám, které tuto informaci pro rozhodnutí o dalším tahu potřebují. Manažer má vždy kompletní informace o aktuálním stavu hry.

Toto je nejčastější využití, zjednodušuje komunikaci a řídí tok dat. Zároveň mu to umožňuje provádět v průběhu komunikace některé kontrolní či zobrazovací funkce.

Zobrazení hrací plochy

V programech, kde není zobrazení hrací plochy důležité, či by bylo s ohledem na programovací jazyk příliš složité, může být tato funkce delegována na manažer, který pak zobrazuje aktuální stav hry.

Využívá se především u her, kde má každá strana pouze část informace o celkové hře a zobrazení aktuálního a kompletního stavu je vyžadováno pro potřeby statistik či možnosti sledovat hru diváky. Například hrají-li dva programy nějakou karetní hru, při které o kartách, patřících oponentovi, nemají žádné informace, je možné divákům zobrazit aktuální stav karet obou hráčů. Manažer také může informace o hrací ploše přenášet na internet.

Pořizování záznamu o hře

Pro pozdější referenci je možné uložit záznam, například pro účely učení se z her či vytváření statistik. Pro většinu her však není stanoven žádný jednotný formát pro ukládání těchto informací, záleží tak většinou na manažeru, jak tyto informace uloží. Ze záznamu musí být možné zrekonstruovat přesný průběh zaznamenané hry.

Organizování turnajů

Manažer je schopen být napojen na několik herních programů i ve chvíli, kdy nehrají a proto je vhodný jako pomoc při organizování turnajů, kdy se na začátku turnaje vše nastaví a manažer se již sám postará o všechny vzájemné souboje a zobrazuje průběžné a konečné výsledky.

Nastavování her, jejich pravidel a stavu před začátkem hry

Je možné pomocí manažeru nastavit stav herní plochy před prvním tahem (pro testování určitých situací, či pro vyvážení výhody začínajícího hráče ve hrách, kde je to vhodné), zvolit doplňující pravidla či cokoliv jiného, souvisejícího se hrou. Výhodou je, že manažer se sám postará o distribuci této informace mezi všechny účastnící se strany a uživatel tak nastavuje veškeré parametry pouze na jednom místě.

3.3 Komunikace

Jak bylo v kapitole 3.1 naznačeno, brainy jsou většinou samostatné programy a musí tedy spolu nějak komunikovat. Především se jedná o nastavení konkrétní hry (pravidla, stav hrací plochy před začátkem, atd.) a předávání informací o právě probíhající hře. Důležitý je způsob předávání a toku informací (kdo předává informace komu).

3.3.1 Předávání informací

V průběhu hry potřebujeme předat informace z jednoho programu do druhého, jsou však nutná určitá omezení vzhledem k tomu, že programují různí lidé využívající různých prostředků (různé programovací jazyky, operační systémy, jazykové sady...). Je proto nutné používat systém co možná nejméně svazující, abychom byli schopni komunikaci zrealizovat v téměř jakémkoliv prostředí – platformová nezávislost. Komunikace se musí řídit určitými pravidly, tzv. protokolem, aby obě strany informacím rozuměly.

Je důležité rozlišovat také komunikaci lokální, kdy jsou všechny zúčastněné programy spuštěny na jednom počítači a komunikaci síťovou, kdy jsou jednotlivé součásti hry spojeny pomocí síťových technologií. Nejběžněji používané systémy:

3.3.1.1 Soubory

Všichni účastníci komunikace zapisují informace do definovaných souborů na předem určeném místě. Tento způsob se může zdát jednoduchý na implementaci, skrývá však mnoho postranních problémů, především správné řízení přístupu k souborům (kdy a kdo se soubory pracuje).

Nejčastějším řešením je, že při hře přes manažer tento vytvoří soubory s popisem hry (hrací plocha, pravidla, atd.), nastaví je na současnou situaci a spustí herní program. Ten si přečte informace ze souborů, odehraje jeden tah, přepíše informace o hře do příslušného souboru a ukončí se. Na to manažer spustí druhý herní program a takto se programy střídají do ukončení partie.

Tento způsob je zastaralý a velice nevhodný, při každém tahu se musí brain znovu spustit a načíst všechny potřebné informace, také pro uchovávání dat potřebných pro běh brainu (jsou-li nějaká) je potřeba vytvářet pomocné soubory, hra lidských protivníku je nepraktická díky neustálému spouštění programů.

3.3.1.2 Roury

Roury, neboli v angličtině pipes, jsou spojení vstupu jednoho programu s výstupem jiného programu a obráceně. Většinou se používá standardních vstupů a výstupů, což bez problémů zvládají všechny programovací jazyky na všech platformách.

Oproti souborové komunikaci není třeba složitě uchovávat informace či komplikovaně předávat řízení. Všechny programy, účastnící se partie, jsou v provozu neustále, většinou si i samy

uchovávají informace o stavu hrací plochy. Můžou tedy čas, kdy čekají na odpověď druhé strany, využít nějakým způsobem například k urychlení svého následujícího tahu (což ovšem představuje určitá rizika, mohou se v této době pokusit zahltit CPU či RAM a znemožnit tak jiným programům přístup k výpočetním zdrojům).

3.3.1.3 Síťová komunikace, TCP, UDP

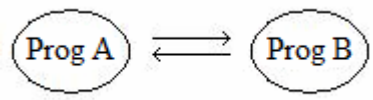
Zřízení komunikace po síti pomocí síťových protokolů TCP či UDP je náročnější na implementaci, nicméně kromě podobných možností jako komunikace skrze roury navíc umožňuje, aby každá část celkové hry (jednotlivé brainy, manažer) běžela na samostatných strojích oddělených například lokální sítí či internetem. – což umožňuje, aby každý program měl veškerý výpočetní výkon jednoho počítače neustále pro sebe (čímž eliminuje riziko zmíněné v případě rour).

3.3.2 Tok informací

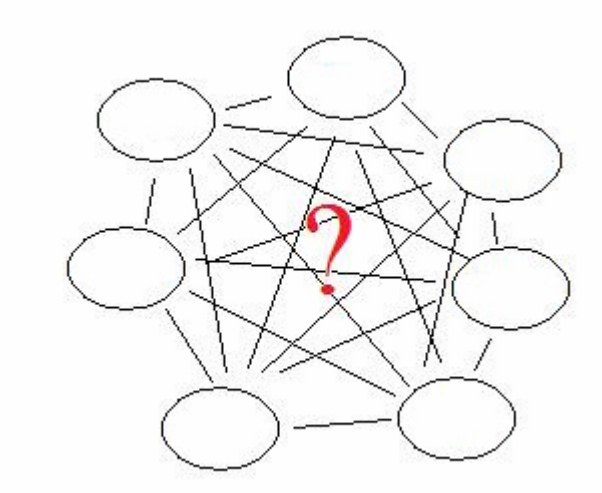
Dalším důležitým aspektem komunikace je to, kdo bude komu různé informace předávat. Komunikovat spolu mohou buď jen samy herní programy, nebo mohou využívat prostředníky – manažery. Tyto metody se dají označovat jako přímá komunikace či komunikace přes prostředníka.

3.3.2.1 Přímá komunikace

Jeden z běžně používaných způsobů komunikace, kdy si herní programy vyměňují informace mezi sebou bez účasti další strany. V případě pouze dvou programů model této komunikace nevypadá příliš složitě (obrázek 1), ovšem s vyšším počtem brainů složitost výrazně narůstá (obrázek 2). Také je třeba říci, že se v tomto případě musejí programy starat o všechny aspekty hry, například kontrola tahů či konce hry a předávání informací všem zůčastněným stranám je poněkud složitější. Drobnou výhodou je, že není potřeba žádný další program, avšak mohou nastat komplikace při komunikaci, neboť jeden z programů musí fungovat jako iniciátor hry a mít tak rozhodující právo v případě sporných situací.



Obrázek 6: Přímá komunikace dvou programů

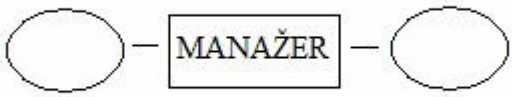


Obrázek 7: Přímá komunikace mnoha programů

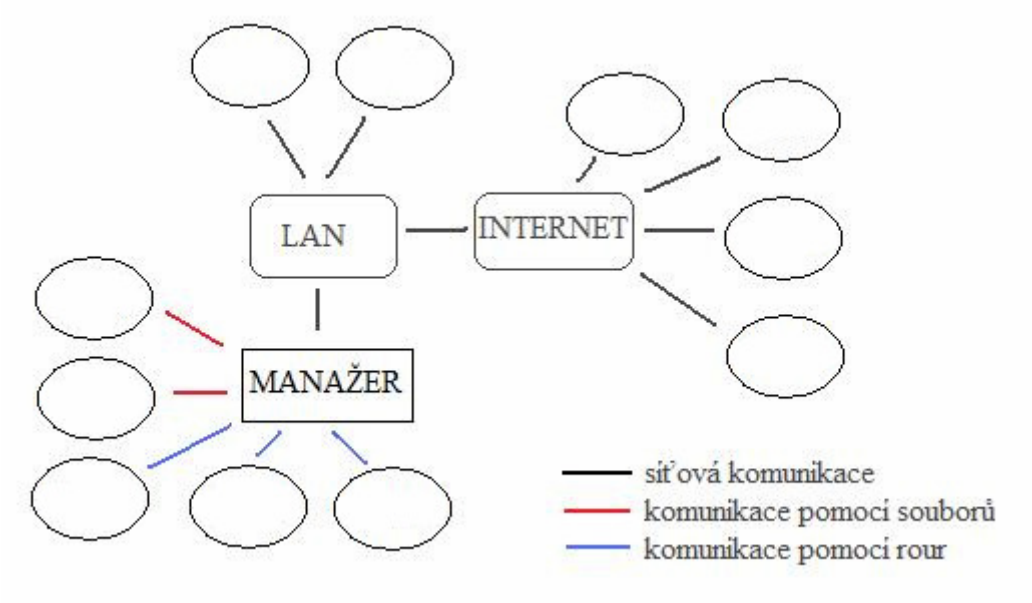
3.3.2.2 Komunikace přes prostředníka

Při tomto způsobu komunikace je herní program spojen pouze s manažerem a o spojení s ostatními účastníky hry se nestará (obrázek 3). Výhodou je, že kvalitně naprogramovaný manažer, kromě možnosti poskytování dalších funkcí (načrtnutých v podkapitole o Manažerech), může podporovat několik různých způsobů komunikace či verzí komunikačního protokolu.

Jako nejlepší řešení je bezpochyby komunikace přes TCP či UDP pomocí manažeru, programátorům her odpadne mnoho zbytečných starostí, zvlášť když se k manažeru dodá kostra programu obsahující veškerou komunikaci, kde stačí doprogramovat funkci reagující na poslední herní situaci. Využití manažerů také napomáhá při organizaci a realizaci různých turnajů, kdy stačí vše nastavit a pak jen oznámit výsledky, odpadá neustálé spouštění všemožných programů či zapisování výsledků. Na obrázku 4 je možné vidět, jak může být kvalitní manažer použit při organizaci turnaje.



Obrázek 8: Komunikace dvou programů přes prostředníka



Obrázek 9: Využití manažeru při organizaci turnaje

3.4 Manažer stolních deskových her

Takto se jmenuje jeden z možných manažerů pro piškvorky. Jedná se o bakalářskou práci Jiřího Kusáka z roku 2007 a právě tento manažer by měl doprovodný program mé bakalářské práce využívat.

3.4.1 Podporované hry

Kromě piškvorek, které jsou zde ve variantě Gomoku (hrací plocha 15x15), podporuje také hru Reversi, Šachy (Chess), Go a Dáma (Checkers). Na manažer je to velice úctyhodný počet podporovaných her, každá hra má svou hrací plochu a zároveň mohou všechny tyto hry běžet nezávisle na sobě.

3.4.2 Komunikace

Využívá síťové komunikace pomocí protokolu TCP, který používá i pro komunikaci na lokálním počítači. Využívá vícevláknového zpracování, což mu umožňuje komunikovat s velkým množstvím programů současně hrajících různé hry.

3.4.2.1 Komunikační protokol

Využívá systém „dotaz, odpověď“, kdy vždy zašle připojenému programu (cíli) zprávu a čeká na odpověď (čekání na odpověď není vždy nutné, v dost případech předpokládá, že informace je vždy pochopena). Uvádím zde seznam příkazů:

- **hello msg** – Dotaz má určit, zda je cíl připraven komunikovat. Jedinou možnou odpovědí je hello ok. Pokud cíl odpoví jinak, považuje se to za selhání komunikace.
- **com msg** – Zjišťuje maximální podporovanou verzi komunikačního protokolu. Cíl odpoví verzí, kterou používá –například com 0,1. Pokud verze podporována není, manažer ukončí komunikaci. (POZOR: oproti dokumentaci manažeru je očekávána odpověď ve formátu „celá_čast čárka desetinná_část“, nikoliv s oddělující tečkou jak je uvedeno.)
- **game gomoku** – Má zjistit, zda cíl hraje piškvorky. Odpověď od cíle může být buď kladná (game ok), nebo záporná (game failed).
- **board [velikost]** – Nastavuje velikost herní desky (posílá se i v případě, že je možná jen jedna velikost). Parametrem je jedno celé číslo, zpráva tedy může vypadat např. takto: board 8. Pokud cíl danou velikost podporuje, pošle odpověď board ok, jakákoliv jiná odpověď je považována za selhání. Manažer podporuje pro piškvorky pouze board 15.
- **player [barva]** – Nastavuje identitu cíle, tedy za jakou barvu bude hrát. Možné varianty: cross, round. Akceptování se provede zasláním odpovědi player ok, jiná odpověď je považována za selhání.
- **name msg** – Požadavek na název cíle. Ten odpovídá ve formátu name [jmeno], tedy třeba name brain_mv_00.1. Toto jméno pak textově identifikuje cíl například v turnajích.
- **timeout [čas]** – Nastavuje maximální možnou dobu pro odpověď. Parametrem je přirozené číslo větší než nula představující délku timeoutu v sekundách. Cíl neodpovídá, bere však timeout na vědomí.
- **move [tah]** – Tato zpráva oznamuje tah oponenta. Tah je ve formátu X0 – za X se dosadí velká písmena A-O, za 0 číslice od 1 do 15. Například *move K2*. Zpráva *move start* má speciální charakter, oznamuje začátek zápasu a tedy to, že cíl má provést první tah. Standardní odpovědí je jiná move zpráva, tentokrát s parametrem svého tahu. Může však odpovědět i zprávou forfeit, která znamená, že se vzdává tahu. Poslání jiných odpovědí,než těchto, má za následek selhání komunikace a konec hry.
- **new game** – Informace o začátku nové hry. Pokud již nějaká hra probíhá, je ukončena. Cíl neodpovídá, ale je povinen resetovat svůj stav (především hrací plochu) do počáteční pozice pro hru.
- **end game** – Ukončuje hru. Cíl neodpovídá, ukončí však hraní .
- **quit msg** – Oznamuje ukončení komunikace. Cíl neodpovídá a ukončí se.
-

Poznámka: je třeba brát ohled na to, že manažer posílá každý dotaz ve formě jednoho řádku a určité informace posílá najednou (jedná se především o informace o nastavení).

3.4.3 Možnosti

Manažer umožňuje tyto funkce:

- pořizování záznamů o hře ve formátu XML a jejich následné přehrávání
- organizování turnajů
- v nastavení her je možné zvolit, která strana začíná a na kolik kol se bude hrát
- zobrazuje počet vyhraných a prohraných partií,
- umožňuje nastavit maximální dobu jednoho tahu
- lze sledovat partii *real-time*

3.4.4 Nevýhody

Jak již bylo zmíněno výše, manažer podporuje velké množství her, což se podepsalo na jeho funkčnosti. Komunikační protokol je autorova vlastní invence, proto neexistují kompatibilní programy pro většinu her a také je vcelku jednoduchý, nepodporuje například přednastavení hrací plochy před prvním tahem. Testování mohla být věnována delší doba – manažer obsahuje mnoho chyb, některé až fatálního charakteru znemožňujícího hru, bylo proto nutno zasáhnout do zdrojových kódů a pokusit se tyto chyby opravit.

4 Turnaje umělých inteligencí

Souboje umělých inteligencí nejsou příliš běžnou záležitostí z několika důvodů. Programátorů zabývajících se vývojem umělé inteligence na amatérské úrovni není mnoho. Většina z nich jsou začátečníci, zabývající se poněkud jednoduššími hrami, které nejsou příliš divácky atraktivní (stolní hry typu piškvorky, šachy, dáma, go, atd.).

Přestože i tyto jednodušší hry jsou stále populární a velice hrané ve variantě člověk versus člověk (existují desítky serverů, kde může každý s připojením na internet, ať už neustálým či alespoň krátkodobým, například jednou za týden přes email, hrát mnoho stolních a deskových her) a programování umělé inteligence pro ně není příliš složité, nemají na poli umělé inteligence příliš co nabídnout. Ve většině případů je nejefektivnější prohledávání stavového prostoru a hledání vítězných, nebo alespoň nejlepších tahů. Ve takovýchto případech umělá inteligence hraje neustále velice podobně, pro člověka zdatného v dané hře je relativně snadné nalézt vítěznou strategii a aplikovat ji.

U mnoha her, zvláště u různých karetních, kde jednotlivé strany nemají o hře úplnou informaci, se při reálně hře člověk řídí instinktem a pozorováním ostatních hráčů, což umělá inteligence nezvládá a tak se v těchto případech snižuje až k podvádění.

4.1 Stolní a deskové hry

Většina her, pro které se umělá inteligence na amatérské soutěžní úrovni vytváří, jsou hry bez prvku náhody či hry s úplnou informací. Piškvorky, šachy, go, dáma, reversi, backgamon... z těchto her jsou nejjednodušší první zmíněné piškvorky a proto jeden z mála turnajů pro umělé inteligence je zaměřen právě na ně. Jedná se o turnaj kolem webu http://gomocup.wz.cz/

Gomocup

Tento turnaj piškvorkových brainů má dlouhou tradici, probíhá každý rok již od léta 2000 (letos v květnu proběhl 9. ročník). Z pěti soutěžících autorů v prvním ročníku se turnaj pomalu rozrostl až na 26 v posledním ročníku – což je sice více než pětinásobek prvotní účasti, ovšem přihlédneme-li k faktu, že se jedná o jediný známější turnaj, tak je vidět, že komunita je velmi malá.

Turnaj začínal s komunikací pomocí souborů, bez nějakých výrazných úprav či limitů. V dnešní době jsou zavedeny časové a paměťové limity, hraje se z předem připravených pozic a komunikace se řeší pomocí rour (pipes). Hry se vždy hrají na sudý počet partií, v nichž se brainy střídají v zahajování hry, systémem každý s každým. Pro velký počet účastníku bylo však třeba v posledních ročnících rozdělit brainy do startovních skupin, jejichž vítězové postupují do finálového klání o nejlepší místo.

Olympiády

Ve světě probíhá další množství různých turnajů ve stolních deskových hrách, na serveru http://www.grappa.univ-lille3.fr/icga/ je možné získat podrobné informace o minulých i plánovaných akcích. Většinou se jedná o „olympijské“ turnaje v programování herních brainů, kde se hrané hry každoročně mění (snad až na šachy, které se hrají nejčastěji). Účast přesto nebývá nijak hromadná, nejvíce účastníků všech her v jednom turnaji bylo v roce 1984 při první olympiádě v Londýně. Na serveru je možné také zjistit výsledky jednotlivých turnajů či záznamy her. Na olympiádě v roce 2000 však byla varianta hry „Gomoku“ (na poli 15x15) prohlášena za vyřešenou a již se dále neobjevuje.

V případě zájmu o záznamy her lidských protivníků je možné navštívit jeden z mnoha serverů, pořádajících online zápasy či turnaje.

O systémech turnajů, vhodných k pořádání brainových zápasů, je možné více se dozvědět na serveru http://deskovehry.cz/index.php/Turnajové_systémy

5 Návrh aplikace

Vytvořená aplikace by měla být především schopna komunikovat s Manažerem pro stolní deskové hry a poskytovat funkce brainu, avšak zároveň by měla mát přijatelné uživatelské rozhraní a umožnit vyzkoušet kvalitu umělé inteligence i bez použití manažeru hrou proti člověku.

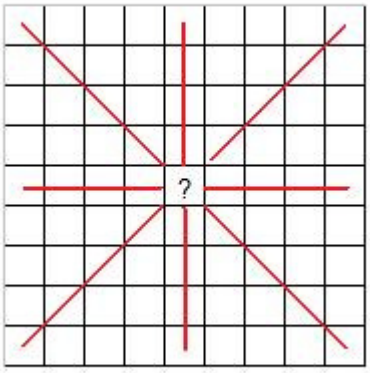
Pro realizaci umělé inteligence jsem zvolil variantu ALFA-BETA prohledávání stavového prostoru se selektivním výběrem nejlepších tahů (je tedy nutná ohodnocovací funkce jak pro jednotlivá pole, tak pro celou hrací desku) s částečným zjednodušením hrací plochy. Požadavkem vedoucího bylo i učení se z jiných, či předchozích her. Algoritmus pro ALFA-BETA a selektivní metodu byl popsán výše, proto zde uvedu pouze doplňující informace.

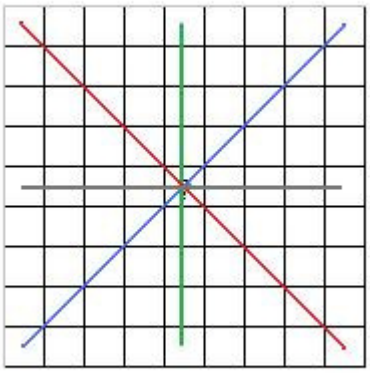
5.1 Prohledávání stavového prostoru

Hloubku prohledávání a počet selektovaných tahů měním v závislosti na délce hry a zvolené obtížnosti. Na počátku partie vybírá program menší množství tahů, ale prohledává je do větší hloubky, neboť možností, jak dobře táhnout, je málo a to, jak první tahy hru ovlivní, je vidět až za relativně mnoho tahů. V průběhu hry zmenšuje hloubku prohledávání, neboť efekt tahu je vidět již dříve, a naopak zvyšuje množství selektovaných nejlepších tahů, neboť v pozdějších fázích hry je mnoho možností, jak táhnout a správná může být téměř kterákoliv z nich. Takto může program dosáhnout podobné časové náročnosti po celou dobu hry a zároveň zlepšit své šance na nalezení správného tahu.

5.2 Ohodnocovací funkce

Díky zvolení implementace selektivní metody je třeba v průběhu prohledávání stavového prostoru v každém tahu ohodnotit všechna relevantní políčka, aby bylo možné vybrat nejlepší tahy a určit situaci hry. Pro ohodnocení políčka se funkce dívá všemi osmi možnými směry (obrázek 10), spočítá počet bezprostředně následujících stejných značek jako právě hrající strana, počet prázdných polí a počet stejných značek, oddělených alespoň jedním prázdným polem. Prohledávání končí nalezením nepřátelské značky či prohledáním 4 polí v daném směru. Ve dvou protilehlých směrech se pak snaží rozpoznat známá seskupení značek, součtem všech 4 dvojsměrů pak získává výsledné ohodnocení pole (obrázek 11).





Obrázek 10: Prohledávání osmi směrů **Obrázek 11: Spojení dvojsměrů**

Na obrázku 12 je názorná ukázka ohodnocení jednoho pole v rozehrané partii. Je nutné ohodnotit pole označené otazníkem z pohledu hráče hrajícího křížky. Funkce se nejprve podívá vlevo a vpravo. Na levé straně jsou samá volná pole, na straně pravé je jeden přátelský symbol, volné pole a ukončující nepřátelský symbol. Spojením je zjištěno, že se jedná o dva symboly v řadě s možností rozšíření až na pět symbolů. Poli je tedy přiřazena příslušná hodnota. Dále se funkce podívá nahorů a dolů. V obou směrech bezprostředně následuje shodný symbol a dále 3 volná pole. Toto seskupení je rozpoznáno jako trojice symbolů s možností rozšíření na pět v obou směrech, oponent je nucen bránit. Příslušné ohodnocení tohoto tahu je přičteno k dosavadnímu ohodnocení pole. Stejně postupuje funkce i v diagonálních směrech.

Dále je nutné spočítat stejným způsobem ohodnocení i pro oponenta, jak moc by získal příštím tahem na toto pole. Toto zajišťuje obranu.

Obě dvě ohodnocení jsou samostatně uložena a například pro výběr aktuálně nejlepšího tahu je vybráno to pole, kde je součet obou hodnot nejvyšší. Jedná se o pole, kde nejvíce získáme a zároveň oponentovi zabráníme v dobrém tahu.

Funkce umí rozpoznat několik různých seskupení ve dvou směrech. Nejčastějšími z nich jsou tyto:

- vedle políčka jsou 4 další stejné značky bez mezer v podobě volných polí => jedná se o tah ukončující hru, nejvyšší priorita

- 3 další značky a na obou stranách alespoň jedno pole volné => oponent již nemůže zabránit vítězství, pokud sám hru neukončí příštím tahem

- 3 další značky a jen na jedné straně volné pole => oponent je nucen bránit jinak přístím tahem prohraje

- v okolí (nemusí všechny tvořit souvislou řadu) jsou 3 stejné symboly a celkem alespoň 3 volná pole => oponent je opět nucen bránit, avšak má více možností pro obranu

- 2 značky u sebe a obě strany 2 či více volných polí => oponent by měl bránit ihned, jinak následuje prohra po dvou tazích. Nutnost bránit ještě další tah

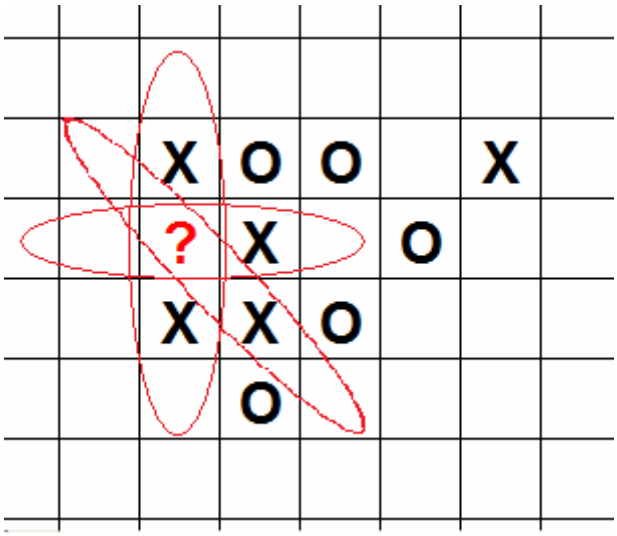
- 2 značky u sebe, jedna strana volná, ale na druhé straně jen 1 volné pole => opět nutnost bránit, jinak následuje prohra po dvou tazích, na zabránění pětice je však potřeba jen jeden protitah

- v okolí jsou někde 2 značky a celkem 4 volné pozice => podobná varianta jako předchozí možnost, oponent má ale na výběr více protitahů

Existují i další situace, většinou se však jedná buď o mrtvé tahy (nelze z nich udělat vítěznou pětici), pak je ohodnocení nulové, nebo tahy, které výše popsaným předcházejí, pak je ohodnocení úměrné části ohodnocení výsledných tahů v poměru s počtem tahů nutných k jejich dosažení.

Pří přiřazování číselných hodnot k seskupením je třeba dbát na to, že tahy, kdy je nutno bránit hned, jsou nejsilnější, neboť zdarma vytvářejí více možností pro útok. Ovšem tah, který vytváří dvě potencionální hrozby, umožňují dosáhnout vítězství a tak by tah tvořící tuto situaci měl být ve výsledku lépe ohodnocen než útočný tah bez možnosti výhry (součet dvou otevřených trojic by měl být vyšší než z části uzavřená čtveřice).

Algoritmus dokáže rozpoznat mnoho druhů seskupení a sdružit je do jedné podobně ohodnocené skupiny, jejíž hodnota se dá drobně upravovat. Nevýhodou pak je to, že při velkém množství vlastních značek v jednom dvojsměru je rozpoznávání seskupení složitější (mnoho možností) a ne vždy přesné.



Obrázek 12: Příklad ohodnocení jednoho pole

5.3 Učení se z her

Program by si měl vytvářet databázi tahů, kterou bude postupně rozšiřovat a upravovat, v závislosti na tom, zda-li tahy vedly k vítězství, či prohře. Při každém ohodnocování projde databázi tahů a pokusí se najít situaci shodnou s tou aktuální, ke které bude v souboru uložena přesnější (časem prověřená a upravená) hodnota výsledného pole.

Pro usnadnění vyhledávání je třeba využít například tabulku s rozptýlenými hodnotami, pro hashovací funkci je možné použít počet vlastních, cizích a prázných polí ve vzorku.

Vzorek bude obsahovat pouze informace o čtyřech polích v každém z osmi směrů, ostatní políčka jsou pro hodnotu aktuálního tahu nepodstatná.

Pro uchování záznamu je nunté uložit informaci o 8*4 polích, každé pole může nabývat jen čtyř hodnot:

- prázdné pole
- vlastní značka
- cizí značka
- konec hrací plochy

Také je potřeba ukládat poopravenou výslednou hodnotu pole, výsledná velikost jednoho záznamu by tedy měla být :

32(pozic)x2(bitů pro všechny možnosti)+32(bitů pro uložení hodnoty situace, jako integer) bitů.

Tedy 96 bitů na záznam, neboli 12 bajtů. Pro představu, do 1MB se vleze něco přes 87000 záznamů. Což by mělo být dostačující a nepříliš prostorově nákladné.

5.4 Zjednodušení hracího pole

Pro zmenšení počtu ohodnocovaných polí byl zaveden systém, který pole příliš vzdálená od již umístěných, označí jako momentálně nepodstatná pro bezprostřední průběh hry. Při umístění nového symbolu dojde ke kontrole blízkého okolí tohoto nového symbolu a volná políčka zbavena označení jako nepodstatná. Jedná se především o urychlení začátku partie, kdy je nepodstatných polí velké množství.

6 Implementace

Nejvyšší prioritou bylo implementovat schopnou umělou inteligenci pro hru piškvorky a schopnost komunikovat s Manažerem pro stolní deskové hry. Nižší prioritu mělo rozšíření umělé inteligence o učení se z her a rozšíření programu z původní konzolové verze ve formě pouhého brainu na aplikaci s uživatelským rozhraním schopnou hrát i lokálně proti člověku.

Jako programovací jazyk byl zvolen jazyk C# .NET, neboť v tomto jazyce je naprogramován i Manažer pro stolní deskové hry a spolu s vývojovým prostředím Microsoft Visal Studio 2005 obsahuje všechny potřebné prostředky pro tvorbu aplikace s uživatelským rozhraním s možností síťové komunikace.

6.1 Komunikace s Manažerem pro stolní deskové hry

Komunikace je řešena pomocí naslouchání na TCP portu (program v režimu server). Po navázání spojení a obdržení úvodních informací, probíhá výměna informací o tazích. Jednotlivé zprávy jsou načítány do bufferu , postupně se vyhodnocují a následně ja zaslána odpověď, je-li vyžadována.

Testováním byly objeveny některé chyby v dokumentaci protokolu manažeru, byly proto třeba drobné úpravy, především v pořadí zasílaných a obdržených zpráv a drobné úpravy jejich obsahu.

Manažer také podporuje automatické spouštění hracích programů na lokálním disku, předává jim jeden číselný parametr (port). Je-li takto program spuštěn, automaticky zahájí naslouchání na zvoleném portu a očekává hru v módu Brain.

6.2 Datové struktury

Pro uchování informací o jednom herním poli byla zvolena struktura z obrázku 13, obsahující x-ové i y-ové souřadnice pole, ohodnocení pro oba hráče a návratová hodnota pro algoritmus ALFA-BETA. Pro uchování celé hrací desky bylo použito pole těchto struktur o rozměrech 15x15.

Pro potřeby ohodnocovací funkce byla vytvořena struktura uchovávající informace o stavu polé v jednom směru, zobrazena na obrázku 14. Obsahuje informaci o bezprostředně následujících stejných symbolech, o jim následujících volných polích a nepropojených shodných symbolech.

```
public struct field
    {
       public int x, y, value, weightP1, weightP2;
    }
```

Obrázek 13: Jedno pole

```
public struct lineWeight
    {
        public int adjecent, empty, away;
    }
```

Obrázek 14: Ohodnocení směru

6.3 Oprava manažeru

6.3.1 Kontrola konce hry

Manažer nepoznal, je-li právě provedený tah tahem vítězným, nikdy tak nemohlo dojít k ukončení právě hrané partie, popřípadě uměl-li připojený program vítězný tah sám rozpoznat, docházelo k chybám v komunikaci, neboť již žádný další nepřátelský tah neočekával. Do programu proto byla doplněna funkce na rozpoznání vítězného tahu, byla použita totožná verze z mého programu pro piškvorky. Po každém tahu zkontroluje ve všech směrech, nenachází-li se v bezprostředním okolí řada pěti či více nepřerušených kamenů stejné barvy.

6.3.2 Kontrola tahu

Kontrola, je-li právě zpracovávaný tah legální, či nenastala-li patová situace (v případě úplného zaplnění hrací desky), úplně chyběla. Bylo nutné doplnit funkce FindMove a ValidateMove v souboru pomocne.cs o kontrolu, zda-li je pole, na které chce jedna ze stran táhnout, volné či je-li po provedení tahu alespoň jedno další neobsazené pole a druhá strana tak může reagovat dalším tahem. V případě patové situace je hra prohlášena za remízu.

6.3.3 Chyba připojení

Pří pokusu o připojení druhého klienta poté, co už byl první připojen, docházelo k chybě oznamující snahu o použití již obsazeného portu, i když podle zadaných parametrů tomu tak nebylo. Část kódu kontrolující znovupoužití stejného portu při zahajování komunikace druhého klienta byla zkopírována z kontroly klienta prvního, avšak nebyly provedeny všechny změny nutné pro správnou funkčnost, program tak porovnával port prvního klienta, který už v používání byl. Jednalo se o funkci btn_brain2_ok_Click v souboru From1.cs.

6.3.4 Zobrazováni plochy

Při vykreslování hrací plochy se opět objevil problém s kopírováním kódu, manažer vykresloval pouze body na souřadnicích [0,0] až [7,7], příčemž hrací plocha pro piškvorky má rozměry 15x15. Bylo třeba upravit funkci AktualizujBoard v souboru Pomocne.cs. K aktualizaci hrací plochy také nedocházelo po každém tahu, bylo proto nutno doplnit překreslení plochy po každém umístěném symbolu.

6.3.5 Závěr oprav

V manažeru nebyly objeveny a opraveny všechny chyby, ovšem již je možné zahájit a odehrát hru piškvorek. Stále zůstávají neotestovány hry šachy, dáma či Go. Pro tyto hry chybí značná část kódu a pro otestování by byl nutný další program hrající zbývající hry schopen komunikovat s tímto manažerem. Zde je nejvíce vidět nevýhoda vytvoření nového protokolu.

6.4 ALFA-BETA, selektivní metoda

Při implementaci algoritmu,vybírajícího nejlepší tah, byla nejprve vytvořena pomocná funkce, která se stará o umístění značek na hlavní hrací plochu a volá další funkci, která je vlastní implementací ALFA-BETA algoritmu se selektivní metodou. Návratovou hodnotou této druhé funkce je nejlepší tah pro aktuální situaci, ten je zahrán na hlavní hrací desku a také je oznámen manažeru jako následující tah.

Při implementaci bylo nutno ošetřit situace, které mohou nastat, například jako objevení výhry/prohry, kdy již není nutné dále nic počítat, a rovnou oznámíme vítězný tah či přiznáme porážku pomocí zprávy forfeit.

6.5 Učení se z her

Při implementaci návrhu jsem narazil na řadu problémů. Původní návrh se nezdál jako příliš efektivní (k vytvoření obstojné databáze tahů by bylo potřeba několik desítek tisíc her, což by zabralo obrovské množství času), byl také příliš časově a paměťově náročný (neustále otvíral/zavíral soubory, načítal spoustu dat, která pak bez využití zahazoval, neboť nenašel vhodnou situaci).

Dalším průzkumem této oblasti jsem zjistil, že implementace nějaké kvalitní formy učení při zachování ostatních navržených metod je téměř nemožná, bylo by třeba provést rozsáhlé úpravy ostatních již implementovaných funkcí a navrhnout kompletně nový systém, či úplně předělat způsob řešení umělé inteligence. Učení se z her bylo proto z implementace vypuštěno.

6.6 Asynchroní procesy

Díky výskytu blokujících a výpočetně náročných operací (komunikace, výběr nejlepšího tahu) bylo třeba tyto implementovat v podobě postranních asynchronních procesů, aby nedocházelo k zamrzání uživatelského rozhranní. K tomuto účelu je využita třída Background Woker, pracující na bázi eventů. Asynchroní proces je odstartován metodou DoWork v novém vlákně, odkud v průběhu či po ukončení hlásí dílčí výsledky vyvoláním eventů ReportProgress či RunWorkerCompleted.

Tyto asynchronní procesy byly využity pro následující perace:

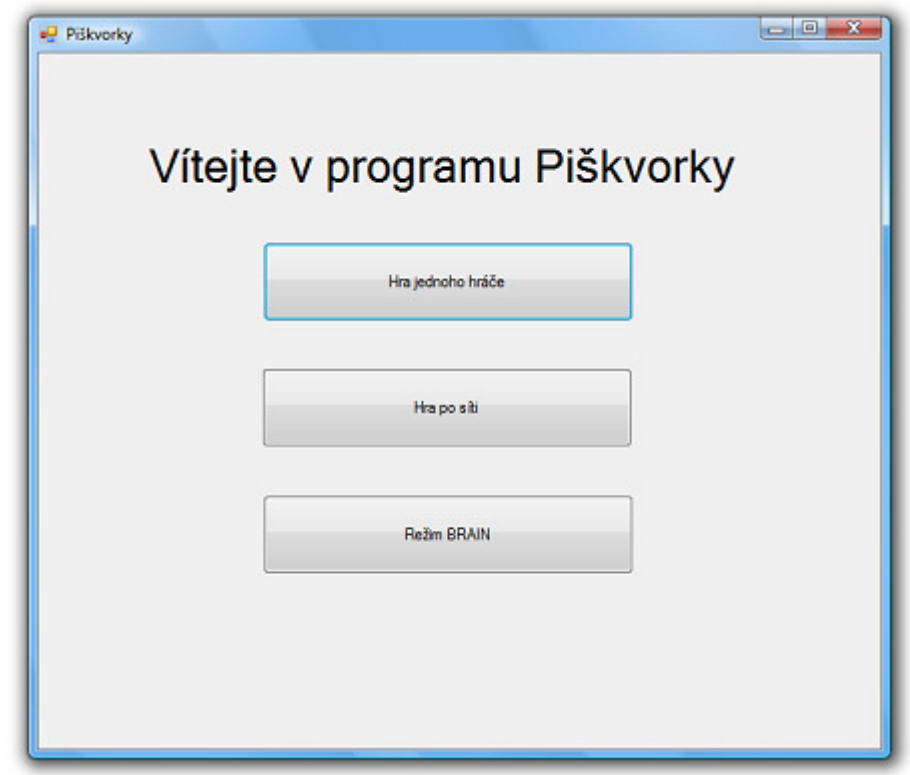
- Ustavení komunikace s manažerem, vytvoření objektu client, který obsahuje informace o komunikačním spojení
- Výměna informací s manažerem, průběžné logování příchozích a odchozích zpráv, volání funkce pro výpočet dalšího tahu, značení tahů obdržených od manažera na hrací plochu
- Funkce výpočtu dalšího tahu pomocí navržených algoritmů, aktualizace herní plochy podle vybraného tahu, oznámení vybraného tahu manažeru

6.7 Uživatelské rozhraní a hra jednoho hráče

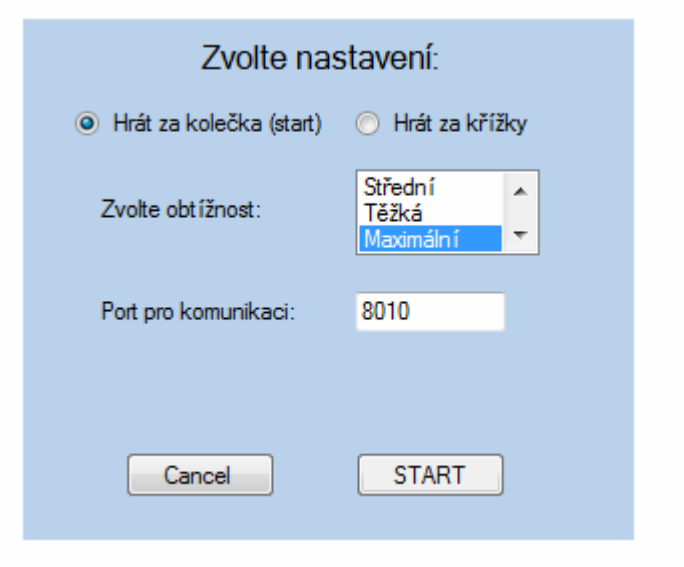
Aplikace byla vytvořena pomocí Application Designeru v MS Visual Studiu 2005. Při tvorbě jsem se volně inspiroval z části brainu bakalářské práce Jiřího Kusáka.

Aplikace obsahuje úvodní menu (obrázek 15), kde je možné zvolit typ hry a následně potřebné nastavení vzhledem ke zvolenému modu hry (obrázek 16).

Po zahájení hry je zobrazena hrací plocha 15x15 polí, okno pro výpis komunikačních zpráv, informační pruh s aktuálním stavem hry a tlačítko pro návrat do hlavního menu, je-li to v aktuálním modu hry možné. (obrázek17, dále).



Obrázek 15: Hlavní menu

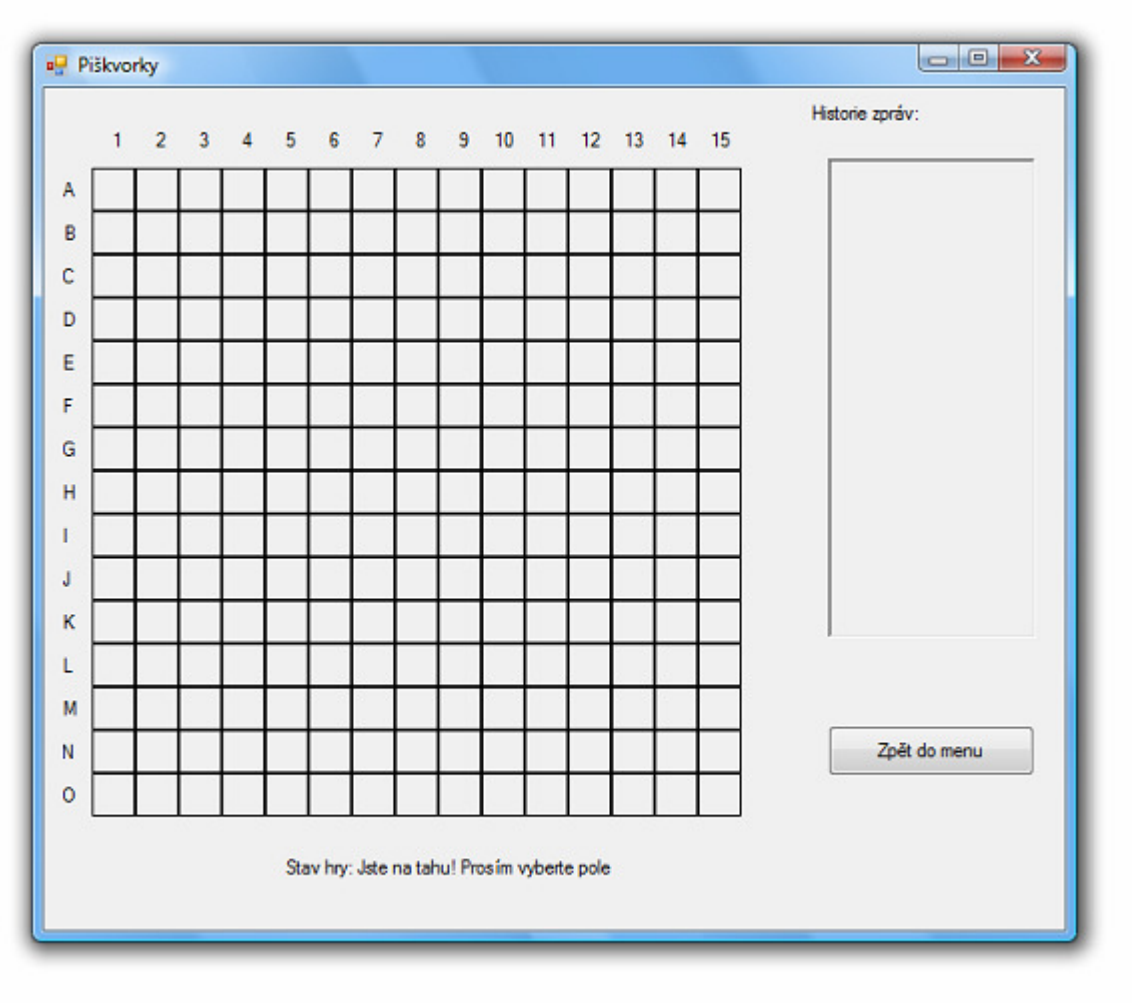


Obrázek 16: Volba nastavení

7 Ovládání

Ovládání aplikace je jednoduché, uživateli postačuje několik kliknutí myši k zahájení hry. Je-li jedním z hrajících člověk, je ke svému tahu vyzván v informačním pruhu a tento tah provede kliknutím na neobsazené pole na hrací desce. Poslední nepřátelský tah je pro lepší identifikaci označen červeným pozadím.

V příloze 1 je popis instalace programu a v příloze 2 citace z bakalářské práce Manažer pro stolní deskové hry o ovládání tohoto manažeru.



Obrázek 17: Hrací plocha, výpis zpráv

8 Závěr

V rámci práce se podařilo vytvořit aplikaci, která je s pomocí Manažeru stolních deskových her schopna hrát hru piškvorky. Jde především o zpracování umělé inteligence piškvorek a síťovou komunikaci s manažerem. Také se podařilo uživatelské rozhraní, díky kterému je umožněna hra lidskému hráči, ať už proti obsažené umělé inteligenci či pomocí manažeru proti jinému protivníkovi či programu.

Do aplikace ve funkci „brain“ se mi nepodařilo implementovat jeden z rozšiřujících požadavků zadání, učení se z her. Jistě by šla vylepšit i umělá inteligence, především v drobných úpravách ohodnocovací funkce, což by však vyžadovalo detailní znalost a zkušenosti s profesionálním hraním piškvorek. Možností by bylo i vyzkoušet úplně jiný systém rozhodování, například neuronové sítě, tato oblast by však vyžadovala rozsáhlejší zkušenosti v této oblasti.

Dále by bylo vhodné doplnit možnosti komunikovat s jinými manažery (kvůli účasti v turnajích), neboť pomocí Manažeru pro stolní deskové hry nelze komunikovat s žádným jiným programem, hrajícím piškvorky. Umožnilo by to, mimo jiné, také lepší otestování obsažené umělé inteligence srovnáním s jinými programy či možnost nastavit určitou konkrétní herní situaci.

Z pohledu autora bylo hlavním přínosem seznámení se s programováním jednoduché umělé inteligence pro počítačové hry, a přestože piškvorky nejsou hra nijak náročná, zkušeností se zde dá získat mnoho, protože i ve složitějších systémech se aplikují některé základní funkce a postupy, které jsem při vypracovávání této bakalářské práce získal. Vývoj aplikace mi též umožnil seznámit se s prostředím C# a .NET a umožnil mi nahlédnout do amatérského prostředí programování brainů a manažerů a následně s turnaji, využívajícími tyto programy.

Celkově jsem s výslednou prací spokojen a doufám, že získané zkušenosti budu moci využít v praxi.

Literatura

[1] http://www.piskvorky.net/ : [online] Hraní piškvorek online

[2] http://deskovehry.cz/ : [online] Server zabývající se deskovými hrami

[3] http://www.piskvorky.cz/ : [online] Server zabývající se hraním piškvorek

[4] http://gomocup.wz.cz/cz/index.htm : [online] Programátorská soutěž Gomocup, piškvorky

[5] http://www.grappa.univ-lille3.fr/icga/ : [online] Seznam turnajů umělých inteligencí

[6] GAME PLAYING COMPUTERS & ARTIFICIAL INTELLIGENCE
Convention presentation - by Zulqernain Akhter

[7] Go-Moku and Threat-Space Search, Ph.D. thesis
L.V. Allis, H.J. van den Herik, M.P.H. Huntjens
University of Limburg, Vrije Universiteit, 1994

[8] Games and Artificial Intelligence, Allis, L.V. (1994), Ph.D. thesis

[9] An Analysis of Alpha-Beta Pruning
Knuth, D.E. and Moore, R.W. (1975) Artificial Intelligence Vol. 6 No. 4

Seznam příloh

Příloha 1:

Instalace

Program je distribuován ve formě aplikace spustitelné v prostředí operačního systému MS Windows XP/Vista, vyžadováno je taktéž prostředí .NET Framework verze 2.0 či vyšší. Stejným způsobem je přiložen i Manažer pro stolní deskové hry. Programy se spouští soubory piskvorky.exe a manager.exe.

Příloha 2:

Ovládání Manažeru pro stolní deskové hry

Ovládání programu

Ovládání programu je do značné míry intuitivní, zde uvedený popis tedy bude stručný. Postupy pro spuštění samostatné hry, turnaje a práce se záznamy jsou uvedeny v samostatných podkapitolách, protože se jedná o stěžejní funkce aplikace.

Kromě výše zmíněného se v programu nachází nápověda a nastavení programu. S nápovědou se pracuje stejným, či podobným způsobem jako s kterýmkoliv jiným prohlížečem. Uživatel se pohybuje mezi stránkami prostřednictvím odkazů, případně pomocí tlačítek Vpřed a Zpět. Nápověda je dostupná pod záložkou Help.

Na záložce Settings se pak nachází možnost nastavit některé vlastnosti programu. Zatím jich není mnoho – zobrazování informačních zpráv, možnost spustit brainy bez případného GUI, výběr složky pro ukládání záznamů turnajových her, volba jazyka nápovědy.

Samostatná hra

K úspěšnému zahájení hry je nutné provést několik kroků:

1. Vybrat brainy. To se provede kliknutím na tlačítko Brain 1, resp. Brain 2. Po kliknutí se zobrazí dialog, kde uživatel vybere, zda se jedná o lokální aplikaci, nebo zda je brain na síti. V prvním zmíněném případě vybere spustitelný soubor, ve druhé zadá IP adresu nebo doménové jméno. V obou případech pak nastaví port, na kterém se bude komunikovat. Následně může stisknout tlačítko OK. Dojde ke kontrole braina, stav se zobrazí na patřičných místech.

2. Nastavit parametry zápasu. Jedná se o: počet her v zápase, volbu střídání se v začínání, zadání timeoutu na tah, volba sledování zápasu v reálném čase a určení konstantní doby tahu.

3. Odstartovat zápas. Provede se snadno kliknutím na Start.

Po provedení těchto úkonů již brainy začnou partii. Na herní ploše můžete sledovat průběh, na pravé straně okna se zobrazují průběžné informace. Hraní můžete pozastavit tlačítkem Pause a analyzovat situaci. Stiskem tlačítka Stop se hra ukončí. V takovém případě se však neukládá záznam.

Turnaj

Postup při zakládání turnaje se do jisté míry podobá tomu předchozímu. Je nutné opakovaně přidávat brainy pomocí tlačítka Add brain. Je nutné si ale dávat pozor při zadávání portu. U samostatné hry jsou porty pro oba brainy předvoleny, takže se o toto uživatel nemusí příliš starat. Při přidávání brainů do turnaje je nutné dbát na to, aby uživatel zadal jiný port pro každý brain. Nejvíce problémů to může způsobovat při vybrání brainu lokálně, protože zadávat komunikační port při spouštění lokálních procesů není zvykem. Pro zjednodušení tam bylo přidáno malé tlačítko +. Odebírání brainu se provádí tlačítkem Delete brain. Při každém přidání či odebrání hráče se aktualizuje tabulka turnaje, statistiky i záložka s hodnocením ELO. Následuje nastavení turnaje (podobně jako u samostatné hry).

Až je uživatel spokojen se vším nastavením, odstartuje turnaj tlačítkem Start.

Záznamy her

Práce s herními záznamy je pak již snadná. Levý box obsahuje seznam záznamů v aktuálně vybrané složce. Název složky je zobrazen v dolní části panelu a lze zvolit jinou složku tlačítkem Change folder. Po označení konkrétního záznamu se vpravo zobrazí vybrané podstatné informace poskytující bližší představu o charakteru záznamu. Pod těmito daty lze nastavit konstantní dobu přehrávání tahu. Pokud tato volba není zaškrtnuta, přehrává se záznam rychlostí, jež byl hrán. Přehrání se odstartuje tlačítkem Watch the game.