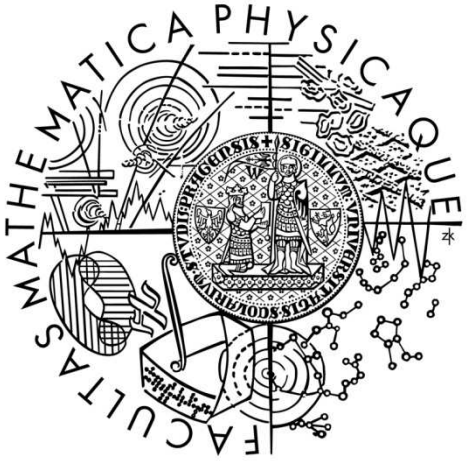
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁRSKA PRÁCA



Martin Boroš

Implementace hry Quoridor

Katedra aplikované matematiky

Vedúci bakalárskej práce: RNDr. Ondřej Pangrác, Ph.D.

Študijný program: Informatika, Obecná informatika

2010

Chcem sa poďakovať vedúcemu tejto práce RNDr. Ondřejovi Pangrácovi, Ph.D. za odbornú pomoc a čas, ktorý mi venoval.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce a jej zverejňovaním.

V Prahe dňa 30. 7. 2010 Martin Boroš

Obsah

Názov práce: Implementace hry Quoridor
Autor: Martin Boroš
Katedra: Katedra aplikované matematiky
Vedúci bakalárskej práce: RNDr. Ondřej Pangrác, Ph.D.
e-mail vedúceho: pangrac@kam.mff.cuni.cz

Abstrakt: Cieľom tejto práce je analýza hry Quoridor a následný návrh a implementácia rozhodovacích algoritmov pre počítačových hráčov. Quoridor je relatívne nová dosková hra pre dvoch alebo štyroch hráčov. Abstraktne je ju možné chápať ako problém hľadania najkratšej cesty v grafe s tým, že je možné odoberať hrany v grafe. V úvode práce si predstavíme Quoridor, jeho históriu a pravidlá. Následne navrhneme vhodnú notáciu a reprezentáciu hry. Ďalej pojednávame o rozhodovacích algoritmoch pri hre dvoch hráčov a potrebných zmenách a vylepšeniach rozhodovania pri hre štyroch hráčov. Na záver si predstavíme aplikáciu, ktorá bola vyvinutá vrámci tejto práce.

Kľúčové slová: hra Quoridor, najkratšia cesta, graf

Title: Quoridor implementation
Author: Martin Boroš
Department: Department of applied mathematics
Supervisor: RNDr. Ondřej Pangrác, Ph.D.
Supervisor’s e-mail address: pangrac@kam.mff.cuni.cz

Abstract: The aim of this thesis is to analyze the game Quoridor. We propose and implement decision-making algorithms for computer players. Quoridor is a relatively new board game for two or four players. It can be viewed as the problem of finding the shortest path in a graph. In addition, edges are allowed to be removed. First, we introduce Quoridor, its history and rules. Then, we propose a suitable notation and a representation of the game. Next, we discuss decision-making algorithms for a two-player game and their necessary changes and improvements for a four-player game. Finally, we introduce the application that was developed to implement and evaluate the proposed ideas.

Keywords: game Quoridor, shortest path, graph

Kapitola 1

Úvod

„Ak sa chystáte vôbec hrať, chystáte sa vyhrať. Baseball, doskové hry, hrať Jeopardy. Nerád prehrávam.“ *Derek Jeter*

Od vzniku prvých hier, zhruba 2600 rokov pred Kristom sa hry stali neoddeliteľnou súčasťou ľudských životov. Všeobecne zahŕňajú duševnú alebo fyzickú stimuláciu, často oboje. Mnoho hier pomáha rozvíjať praktické zručnosti, slúži ako forma cvičenia, prípadne vzdelávania [6].

Často hráme hry len pre zábavu, ale ako povedal známy baseballový hráč Derek Jeter, hráme aby sme vyhrali. Sme súťaživí a chceme byť najlepší v rôznych hrách. S príchodom prvých počítačov sa však v súťažení o najlepších hráčov čosi zmenilo. Ľudia začali hry analyzovať s pomocou počítača a vytvárať agentov, ktorí by boli schopní poraziť najlepších ľudských hráčov v danej hre [3]. Niektoré hry sa podarilo rozriešiť a dokonca nájsť aj neprehrávajúcu stratégiu. Pre také hry sa stalo bezvýznamné usporadúvať súťaže[1].

Hra ktorou sa zaoberá táto práca je Quoridor. Je to pomerne nová a sotva skúmaná hra. Hrá sa na šachovnici s rozmermi 9x9 políčok. Každý hráč dostane na začiatku hry figúrku, s ktorou musí prejsť z jednej strany šachovnice na druhú. Háčik v tomto jednoduchom cieli je v tom, že hráči majú aj určitý počet stien, ktoré môžu na šachovnicu položiť a vytvoriť tak prekážku pre súperov.

Hru môžu hrať dvaja, prípadne štyria hráči. Práce ktoré sa zaoberajú tvorbou programov schopných hrať Quoridor, riešia výlučne hru dvoch hráčov. My sa budeme zaoberať aj analýzou hry štyroch hráčov a tvorbou jej hrajúcich agentov.

V nasledujúcej kapitole si predstavíme hru Quoridor a jej pravidlá. Ďalej navrhneme vhodnú notáciu a reprezentáciu hry. Rozoberieme si zložitosť hry a možnosti použitia minimaxu pri tvorbe počítačového hráča. Napokon, s využitím týchto znalostí budeme schopní zostaviť rozhodovacie algoritmy pre počítačových hráčov.

K naimplementovaným počítačom ovládaným hráčom bolo potrebné zostaviť jednoduché užívateľské rozhranie, ktorého ovládanie je popísané v Kapitole 6 – Užívateľská dokumentácia. Programátorská dokumentácia k aplikácii sa nachádza na priloženom médiu (viď Dodatok A – Obsah priloženého média).

[1] Napríklad známa hra „Dáma“ (Checkers) bola rozriešená a bola pre ňu objavená stratégia ktorá garantuje remízu pri perfektnej hre. Dáma je s počtom prehľadávaných pozícií $5x10^{20}$ najväčšia hra ktorá bola dodnes vyriešená [11].

Kapitola 2

Hra Quoridor

Quoridor je dosková hra pre dvoch alebo štyroch hráčov, hraná na šachovnici o rozmeroch 9 x 9 políčok. Každý hráč má figúrku s ktorou začína na strednom políčku jednej strany šachovnice. Cieľ hry je prejsť s figúrkou na ľubovoľné políčko druhej strany šachovnice. Hráči dostanú na začiatku hry aj určitý počet stien, ktoré môžu položiť na šachovnicu a vytvárať tak bludisko pre protihráčov.

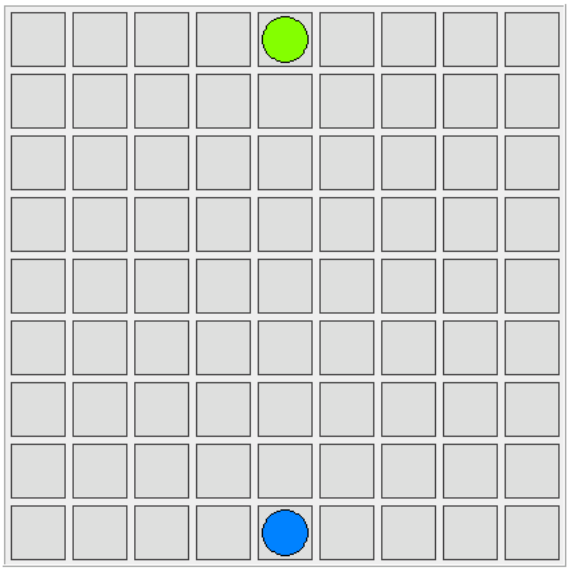
2.1 História

Hru Quoridor navrhol Mirko Marchesi a publikovala ju v roku 1997 spoločnosť Gigamic Games [10]. Quoridor vznikol na základe staršej hry Mirka Marchesiho Blockade, publikovanej v roku 1975. Blockade bola hra výlučne dvoch hráčov, na väčšej šachovnici, s väščím počtom stien rozdelených na horizontálne a vertikálne.

Quoridor získal v roku 1997 ocenenie „Mensa Mind Game“ a bol vyhlásený hrou roka v USA, Belgicku, Kanade a Francúzsku. Časopis Games magazine vyhlásil Quoridor hrou roka 1998.

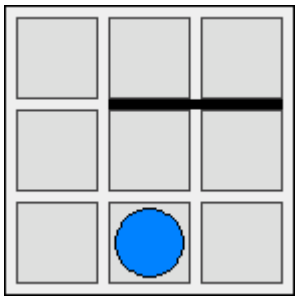
2.2 Pravidlá hry

Hra sa skladá z šachovnice s osemdesiatjedna políčkami, štyroch figúrok a dvadsiatich stien. Hráči dostanú na začiatku hry figúrku a steny. Desať stien pri hre dvoch hráčov a päť stien pri hre štyroch hráčov. V úvode hry sa figúrky umiestnia na prostredné políčko na kraji šachovnice. Pri hre dvoch hráčov musia byť figúrky umiestnené na protiľahlých stranách (viď obrázok 2.1) [10].

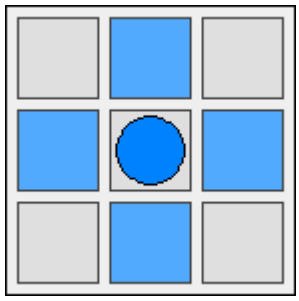


Obrázok 2.1: Úvodná pozícia pri hre dvoch hráčov

Steny majú dĺžku dvoch políčok a môžu byť položené do medzery medzi políčkami. Cez steny nie je možné prejsť figúrkou a akonáhle sú položené, nie je možné ich presunúť ani odobrať. Steny sa nemôžu pretínať a musí existovať cesta na aspoň jedno víťazné políčko pre každého hráča. Vrámci ťahu môže hráč položiť stenu, alebo sa pohnúť s figúrkou.

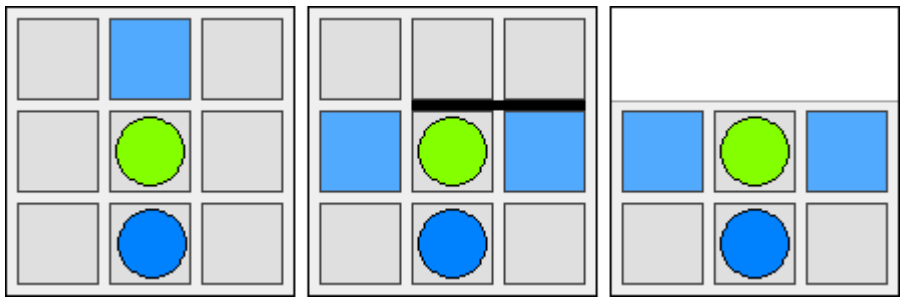


Obrázok 2.2: Regulárne položená stena



Obrázok 2.3: Povolený pohyb figúrky

Figúrka sa môže hýbať vždy na susedné políčko. V prípade že je na susednom políčku figúrka protihráča, môže ju preskočiť. Ak je za figúrkou protihráča tretia figúrka, stena alebo kraj šachovnice, môže sa hráč pohnúť na akékoľvek neblokované políčko, susedné od preskakovanej figúrky.



Obrázok 2.4: Možnosti preskoku figúrky protihráča

2.3 Základné stratégie

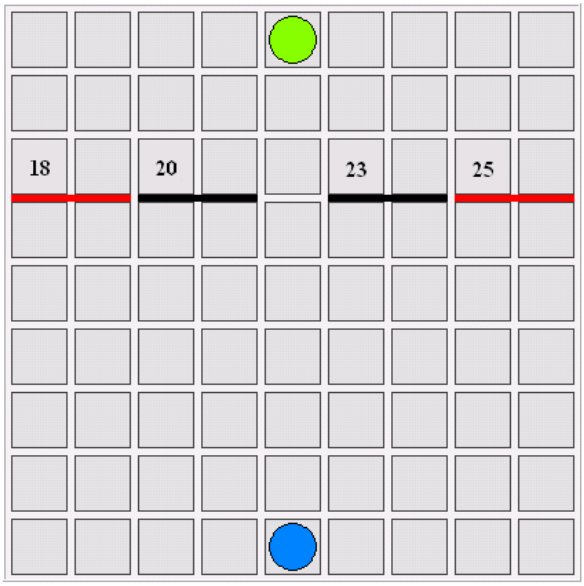
Napriek tomu, že Quoridor ešte nebol hĺbkovo analyzovaný, základné stratégie sa zdajú byť jasné: Ísť do cieľa čo najkratšou cestou a tú protivníkovu čo najviac predĺžiť. Samotné predlžovanie cesty však na víťazstvo proti skúsenému hráčovi nestačí. Najväčšiu strategickú výhodu je možné získať, ak prinútime protihráča na ústup a spätné trasovanie cesty. Aby sa nám podarilo vytvoriť takúto „obchádzku“, potrebujeme nechať otvorených viac ciest do cieľa a tú, ktorou sa vybral protivník uzavrieme, keď sa bude blížiť k jej koncu. Táto taktika však zlyháva, keď sa protivník rozhodne uzavrieť cestu za sebou a tým si zabezpečiť kratšiu cestu do cieľa. Z toho plynie, že najviac z tejto taktiky môžme vyťažiť, keď protivník už nemá žiadne steny.

Iná taktika navrhuje výstavbu cesty do cieľa a zahrnutie protivníka do tejto cesty. Čiže zabezpečením vlastnej cesty do cieľa vytvoríme obchádzku pre protivníka, ktorý sa bude musieť vrátiť na kraj šachovnice z ktorého začínal.

Okrem stratégií ktoré sa uplatňujú v rozohranej partii, je známych aj niekoľko otvorení. Otvorenie je sekvencia úvodných ťahov, ktorých cieľom je získať lepšie postavenie. Za každým otvorením stojí plán, kam chceme smerovať v strednej hre. V nasledujúcom texte si predstavíme dve najznámejšie zahájenia pre hru Quoridor [2].

2.3.1 Reedovo zahájenie

Dr. Scott Reed je známy vojenský stratég a jeho zahájenie pozostáva z položenia stien tri políčka pred protihráčom (políčka 20 a 23[3]) s koridorom širokým jedno políčko v prvých dvoch ťahoch (viď obrázok 2.5) [10]. Obrana proti Reedovmu zahájeniu je doložiť zvyšné dve steny na rovnakú úroveň (políčka 18 a 25).

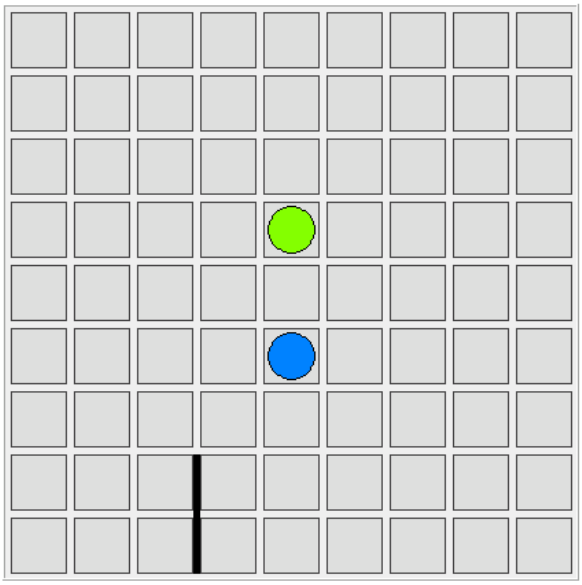


Obrázok 2.5: Reedovo zahájenie
(čierne steny sú súčasťou otvorenia, červené sú odporúčaná obrana)

[2] Zahájenia neboli implementované v rozhodovacích algoritmoch agentov.
[3] Viď kapitola 3.1: Notácia.

2.3.2 Shillerovo zahájenie

V Shillerovom zahájení postúpia hráči o tri políčka dopredu a potom hráč ktorý je na ťahu položí vertikálnu stenu na svoju základnú stranu – minimálne tri políčka od ľavého a pravého kraja šachovnice. Táto stena rozdelí cieľové políčka súpera na dve skupiny. Toto otvorenie spočíva na maximalizácii protivníkovej cesty a minimalizácii vlastnej [10].



Obrázok 2.6: Shillerovo zahájenie

Kapitola 3

Prístup počítača k hraniu hry

Hranie hier bola jedna z prvých úloh vykonávaných umelou inteligenciou. Vznik teórie hier sa datuje okolo roku 1950, teda do čias, keď sa počítače stali programovateľné. Z pohľadu matematickej teórie hier je konkurenčné multi-agentné prostredie hra, v ktorej má každý agent dôležitý vplyv na ostatných agentov. Umelá inteligencia sa spravidla obmedzuje na hry dvoch hráčov, ktorí sa striedajú a majú nulový súčet. Dôvod, prečo sa umelá inteligencia často zaoberá tvorbou agentov schopných hrať hry, je prostý. Hry sú ťažké na riešenie, dobre reprezentovateľné, agenti majú málo akcií a sú zábavné [1].

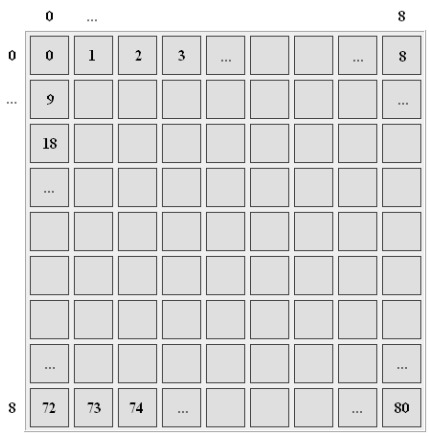
Quoridor patrí v teórii hier k *deterministickým*, *sekvenčným* hrám dvoch hráčov s *nulovým súčtom*, *perfektnou informáciou* a *obmedzeným výsledkom*. Deterministické hry sú také, v ktorých nehrá element štastia žiadnu úlohu. V sekvenčných hrách vykonávajú hráči ťah sekvenčne, čiže nezasahujú do hry všetci naraz. Nulový súčet popisuje situáciu, v ktorej zisk jedného hráča je presne vyvážený stratou druhého (výhra jedného hráča automaticky znamená prehru druhého hráča). V hre s perfektnou informáciou majú obaja hráči plný prístup k aktuálnemu stavu hry. Obmedzený výsledok v prípade hry Quoridor znamená, že po ukončení hry hráč buď vyhral, alebo prehral.

Najčastejšie používaný algoritmus na riešenie tejto skupiny hier je minimax. V tejto kapitole si najprv zadefinujeme vhodnú notáciu a reprezentáciu súčastí hry Quoridor. Následne sa budeme venovať zložitosti hry a na záver popíšeme algoritmus minimax a vhodnosť jeho použitia pri implementácii agenta.

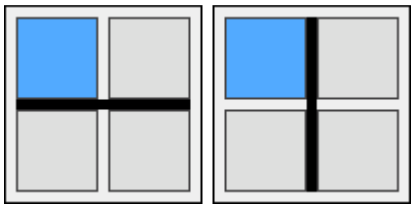
3.1 Notácia

Za účelom popisu hry Quoridor si zavedieme nasledovnú notáciu. Každé políčko šachovnice bude identifikované číslom z množiny {0,1,2,…,80}. Riadky a stĺpce šachovnice budú označené číslami 0-8 s tým, že ľavé horné políčko bude na nultom stĺpci a riadku (viď Obrázok 3.1).

Pozícia hráča bude teda určená číslom políčka. Steny položené na šachovnici sa nachádzajú medzi štyrmi políčkami a na identifikáciu steny bude slúžiť ľavé horné políčko a orientácia steny (viď obrázok 3.2).



Obrázok 3.1: Zápis šachovnice



Obrázok 3.2: Políčko a orientácia (horizontálna / vertikálna) určujú polohu steny

3.2 Reprezentácia šachovnice, stien a hráčov

Zadefinujeme si graf *G=(V,E)*, ktorý bude reprezentovať šachovnicu. Pričom množina vrcholov *V* bude obsahovať všetky políčka šachovnice a hrana $e = (v1, v2), (v1, v2 \in V),\ e \in E \leftrightarrow$ existuje cesta dĺžky 1 medzi *v1* a *v2* (čiže *v1* a *v2* sú susedné políčka ktoré neoddeľuje stena).

Pre potreby našej aplikácie je najvhodnejšia reprezentácia grafu *G* ako zoznamu susedov. Zoznam susedov obsahuje pre každý vrchol množinu vrcholov s ním susediacich.

Steny budeme reprezentovať ako množinu pozícií s horizontálnymi stenami a množinu pozícií s vertikálnymi stenami. Po pridaní steny odoberieme im náležiace hrany z grafu šachovnice.

Pre potreby reprezentácie hráča si vytvoríme triedu ktorá bude obsahovať pozíciu hráča, počet stien ktorými disponuje, číslo hráča a množinu pozícií ktoré musí dosiahnuť aby zvíťazil.

Všetky tieto položky zabalíme do triedy *GameBoard* [4] , ktorá bude reprezentovať pozíciu v hre, bude poznať pravidlá hry, vedieť kto je na ťahu, prípadne kto zvíťazil. Okrem horeuvedených položiek spravuje *GameBoard* aj pridávanie a odoberanie dočasných hrán na šachovnici[5] , dokáže kontrolovať platnosť ťahov a spočítať dĺžku najkratšej cesty pre všetkých hráčov.

3.3 Zložitosť hry

Než sa pustíme do stavby stromu hry a jeho prehľadávania, je potrebné si ilustrovať, s hrou akej zložitosti pracujeme[6]. Pre tieto potreby si uveďme pojmy *veľkosť stavového priestoru* a *veľkosť stromu hry*. *Veľkosť stavového priestoru*, je počet legálnych pozícií ktoré sú dosiahnuteľné zo základnej pozície hry. *Veľkosť stromu hry* je počet hier, ktoré môžu byť odohrané. U väčšiny hier tieto veličiny nevieme určiť presne, ale môžeme ich celkom dobre odhadnúť.

Podľa výpočtov, ktoré vykonal P.J.C. Mertens [3], je hrubý odhad *veľkosti stavového priestoru* hry Quoridor 3.9905×10^{42}. Pri výpočte *veľkosti stromu hry* sa

[4] Pre viac informácií o triede GameBoard viď Programátorskú dokumentáciu na priloženom médiu.
[5] V prípade že sa hráč nachádza vedľa protivníka, môže ho preskočiť. Čiže do grafu reprezentujúceho šachovnicu musíme pre tieto prípady pridávať dočasné hrany.
[6] V nasledujúcom texte uvažujeme hru dvoch hráčov.

Mertens opieral o poznatky z práce *Lisy Glendenning* [2] a dospel k číslu $1.7884x10^{162}$. Pre porovnanie si uveďme zložitosti známych hier [7].

Hra	Veľkosť stavového priestoru (ako log so základom 10)	Veľkosť stromu hry (ako log so základom 10)
Tic-Tac-Toe	3	5
Dáma	20	31
Backgammon	20	144
Quoridor	42	162
Šach	47	123
Go	171	360

Obrázok 3.3: Zložitosti známych hier

Čiže hra Quoridor má porovnateľnú veľkosť stavového priestoru ako šach a prekvapujúco je v nej možné odohrať väčšie množstvo hier ako v šachu. *Lisa Glendenning* odhadla *priemerný vetviaci faktor* Quoridoru na 60.4, *maximálny vetviaci faktor* si môžeme spočítať ako počet možných pohybov figúrky (5) a počet možných položení steny (128). Na obrázku 3.4 vidíme, ako sa nám môže rozrásť strom hry v priemernom a v najhoršom prípade.

Počet ťahov	Počet možných pozícií v priemernom prípade	Počet možných pozícií v najhoršom prípade
1	60	133
2	3,648	17,689
3	220,348	2,352,637
4	13,309,071	312,900,721
5	803,867,911	41,615,795,893

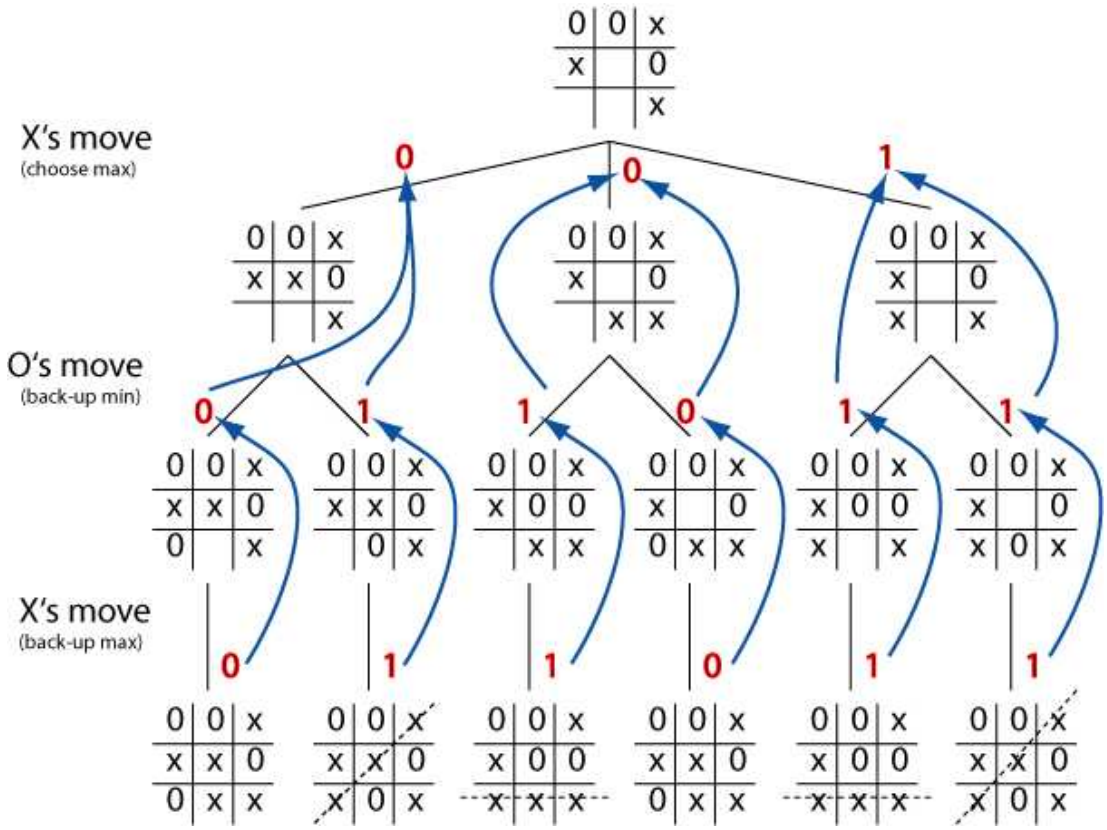
Obrázok 3.4: Tabuľka ilustrujúca šírku stromu hry Quoridor v danej hĺbke

3.4 Strom hry a Minimax

Strom hry je orientovaný graf, ktorého vrcholy sú pozície v hre a hrany sú ťahy. Kompletný strom hry je strom začínajúci v základnom postavení a obsahujúci všetky možné ťahy z každej pozície [8].

Minimax je rozhodovací algoritmus, ktorý prehľadáva herný strom a volí optimálnu stratégiu. Pracuje na princípe minimalizácie možnej straty a súčasne maximalizácii potenciálneho zisku [9]. V prípade hier s malou zložitosťou ako Tic-Tac-Toe (piškvorky na hracom pláne o rozmeroch 3x3), je možné postaviť kompletný strom hry a celý ho minimaxom prehľadať. Čiže je možné garantovať víťaznú, prípadne neprehrávajúcu stratégiu.

V nasledujúcom texte si popíšeme fungovanie minimaxu na analýze pozície z hry Tic-Tac-Toe. Nazvime hráča ktorý je na ťahu MAX (X) a jeho protivníka MIN (O). Tic-Tac-Toe má nízky vetviaci faktor a malý stavový priestor, preto nepotrebujeme obmedzovať hĺbku prehľadávania.



Obrázok 3.5: Príklad práce minimaxu na hre Tic-Tac-Toe (zdroj [10])

Minimax pre hru Tic-Tac-Toe funguje nasledovne (pseudokód):

funkcia MINIMAX(*pozícia*) **vracia** *ťah*

> *hodnota* = MAX(*pozícia*)
> **vráť** *ťah* **z** NÁSTUPCOV(*pozícia*) **s hodnotou** *hodnota*

funkcia MAX(*pozícia*) **vracia** *hodnota*

> **ak** JE-KONCOVÁ(*pozicia*) **tak vráť** -1 *//O VYHRAL*
> *v* = -∞
> **pre každú** *novú_pozíciu* **z** NÁSTUPCOV(*pozícia*) **vykonaj**
> *v* = maximum(*v*, MIN(*novú_pozíciu*))
> **vráť** *v*

funkcia MIN(*pozícia*) **vracia** *hodnota*

> **ak** JE-KONCOVÁ(*pozicia*) **tak vráť** 1 *//X VYHRAL*
> *v* = ∞
> **pre každú** *novú_pozíciu* **z** NÁSTUPCOV(*pozícia*) **vykonaj**
> *v* = minimum(*v*, MAX(*novú_pozíciu*))
> **vráť** *v*

Kompletný strom hry Tic-Tac-Toe má 26,830 listov. Keď to porovnáme s hrou Quoridor, vidíme že je značne jednoduchšie vyriešiť hru Tic-Tac-Toe, ako prehľadať strom z pozície hry Quoridor do tretieho ťahu. U hier ako Quoridor nie je možné postaviť, ani prehľadať kompletný strom hry. V takom prípade musíme obmedziť hĺbku prehľadávaného stromu a zaviesť ohodnocovaciu funkciu, ktorá pridelí listom prehľadávaného stromu hodnotu z oboru celých (prípadne reálnych) čísel. Taktiež sa používa zlepšenie minimaxu zvané α-β orezávanie, ktoré eliminuje podstromy, do ktorých sa počas hrania nedostaneme, pretože tam nevedie optimálna stratégia.

Minimax s α-β orezávaním pre hru Quoridor by vyzeral nasledovne (pseudokód):

funkcia MINIMAX(*pozícia*) **vracia** *ťah*

    hodnota = MAX(*pozícia,1,* -∞, +∞)
    vráť *ťah* **z** NÁSTUPCOV(*pozícia*) **s hodnotou** *hodnota*

funkcia MAX(*pozícia, hĺbka, α, β*) **vracia** *hodnota*

    ak JE-KONCOVÁ(*pozicia*) **tak vráť** -∞
    ak *hĺbka* == MAXIMÁLNA_HĹBKA **tak vráť** OHODNOŤ(*pozícia*)
    v = -∞
    pre každú *novú_pozíciu* **z** NÁSTUPCOV(*pozícia*) **vykonaj**
        v = maximum(*v*, MIN(*novú_pozíciu, hĺbka+1, α, β*))
        ak *v* ≥ *β* **tak vráť** *v*
        α = maximum(*α*,*v*)
    vráť *v*

funkcia MIN(*pozícia, hĺbka, α, β*) **vracia** *hodnota*

    ak JE-KONCOVÁ(*pozicia*) **tak vráť** ∞
    ak *hĺbka* == MAXIMÁLNA_HĹBKA **tak vráť** OHODNOŤ(*pozícia*)
    v = ∞
    pre každú *novú_pozíciu* **z** NÁSTUPCOV(*pozícia*) **vykonaj**
        v = minimum(*v*, MAX(*novú_pozíciu, hĺbka+1, α, β*))
        ak *v* ≤ *α* **tak vráť** *v*
        β = minimum(*β*,*v*)
    vráť *v*

V nasledujúcej kapitole sa budeme venovať samotným návrhom algoritmov pre hru dvoch hráčov.

Kapitola 4

Algoritmy pre hru dvoch hráčov

Na základe skúmania zložitosti hry Quoridor by sme použitím samotného minimaxu s α-β orezávaním ako jediného rozhodovacieho algoritmu nedosiahli žiadaných výsledkov, pretože v súčasnej dobe nie je známa dostatočne dobrá ohodnocovacia funkcia. Aj po použití relatívne obmedzujúcej heuristiky nie je možné preskúmať v (pre hranie hry) reálnom čase väčší herný strom, ako je strom hĺbky 3.

Aby sme zabezpečili dostatočnú náhodnosť počítačového hráča, budeme pridávať nízku náhodnú hodnotu do ohodnocovacích funkcií a rozhodovacích algoritmov. Čiastočne je náhodnosť riešená aj používaním množín. Minimax vráti množinu niekoľko rovnako ohodnotených ťahov. Vzhľadom na to, že množiny nie sú usporiadané, prebehne výber ťahu zo všetkých minimaxom ponúknutých náhodne.

V tejto kapitole si predstavíme pomocné funkcie ktoré agenti pri rozhodovaní využívajú a samotné rozhodovacie algoritmy troch naimplementovaných agentov.

4.1 Pomocné funkcie

funkcia *moveTheShortestPath()*

Funkcia *moveTheShortestPath()* vykoná ťah figúrkou po najkratšej ceste smerom k cieľovým políčkam. Funkcia využíva Dijkstrov algoritmus na hľadanie najkratšej cesty v grafe [5].

funkcia (Boolean) *placePlayerBlockingWall()*

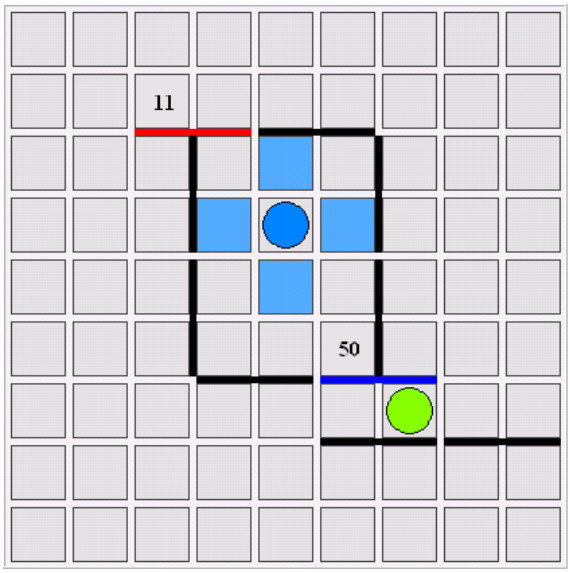
Funkcia *placePlayerBlockingWall()* analyzuje všetky možné položenia steny vrámci súčasnej pozície. Hľadá stenu, ktorá maximalizuje rozdiel medzi dĺžkou najkratšej cesty oponenta a dĺžkou hráčovej najkratšej cesty. Ak nie je možné položiť stenu ktorá by zväčšila rozdiel v dĺžkach ciest medzi oponentom a hráčom, funkcia vráti *false* a nevykoná žiaden ťah. Ak taká stena existuje, funkcia vykoná ťah položením steny a vráti *true*.

funkcia (Boolean) *protectPath()*

Funkcia *protectPath()* minimalizuje škody , ktoré by mohol protivník napáchať položením steny. Vyskúša, či môže vykonať ťah, ktorý minimalizuje maximálnu obchádzku ktorú môže oponent položením steny spôsobiť. Ak existuje taký ťah, ktorý mu môže ušetriť 5 a viac políčok cesty, tak ťah vykoná a vráti *true*. V prípade že neexistuje, vráti *false*.

Napríklad:

Na obrázku 4.1 vidíme pozíciu z hry Quoridor, na ktorej si ilustrujeme využitie funkcie *protectPath()*. Modrý hráč má dĺžku najkratšej cesty do cieľa 4. Ak však zelený v nasledujúcom ťahu položí horizontálnu stenu na políčko 11 (viď červená stena na obrázku 4.1), modrý hráč sa bude musieť vrátiť a jeho najkratšia cesta bude mať dĺžku 11. Čiže existuje stena, ktorá môže modrému predĺžiť najkratšiu cestu o 7 políčok. Modrý ale preskúša všetky ťahy a zistí, že keď položí stenu na políčko 50 (viď modrá stena na obrázku 4.1), tak mu zelený môže predĺžiť cestu maximálne o 2 políčka. Teda modrý môže ušetriť 7-2=5 políčok položením steny na políčko 50. Preto sa rozhodne stenu na políčko 50 položiť .



Obrázok 4.1: Príklad využitia funkcie protectPath()

4.2 Hráč „Dumb Wall Builder“

Algoritmus tohto hráča sa nachádza v triede *AI2Easy*. Jeho rozhodovanie je okamžité a jednoduché. Má veľkú tendenciu minúť v úvode hry všetky steny.

Postup uvažovania hráča:

Ak nemám žiadne steny, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak oponent nemá žiadne steny, tak si pozriem kto je bližšie k cieľu:

- Ak som bližšie, idem do cieľa. (*moveTheShortestPath*)
- Ak je oponent bližšie, obmedzím ho. (*placePlayerBlockingWall*)
  Ak sa mi ho nepodarí obmedziť, idem do cieľa. (*moveTheShortestPath*)

Ak máme obaja steny, spustím minimax do hĺbky 1 – čiže ohodnotím všetky možné ťahy a náhodne vyberiem z množiny najlepších ťah, ktorý vykonám.

Ohodnocovacia funkcia ktorú využíva tento algoritmus počíta rozdiel dĺžky najkratšej cesty oponenta a dĺžky najkratšej cesty hráča. Maximalizáciou vzniknutého rozdielu vyberáme ťah, ktorý predĺži protivníkovi najkratšiu cestu pri snahe o zachovanie minimálnej dĺžky vlastnej cesty.

4.3 Hráč „Smart Wall Builder“

Algoritmus tohto hráča sa nachádza v triede *AI2Medium*. Jeho rozhodovanie je okamžité. Má tendenciu viac hýbať figúrkou ako „Dumb Wall Builder“ a šetriť steny. Jeho rozhodovanie používa rovnaký postup ako „Dumb Wall Builder“, s tým rozdielom, že ak pohyb figúrkou prinesie hráčovi rovnaký osoh ako postavenie steny, uprednostní vo väčšine prípadov pohyb figúrkou.

4.4 Hráč „Smart Player“

Algoritmus tohto hráča sa nachádza v triede *AI2Hard*. Smart player má zložitejší rozhodovací algoritmus a všeobecne mu trvá niekoľko sekúnd kým vykoná ťah. Smart player využíva dynamickú hĺbku minimaxom prehľadávaného stromu[7]. Tento algoritmus taktiež využíva silne reštriktívnu heuristiku, ktorá z prehľadávania vyraďuje pozície v ktorých si hráč na ťahu pohorší – čo do rozdielu dĺžky najkratšej cesty súpera a dĺžky vlastnej najkratšej cesty.

Postup uvažovania hráča:

Ak nemám žiadne steny, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak oponent nemá žiadne steny, tak si pozriem kto je bližšie k cieľu:
- Ak som bližšie, idem do cieľa. (*moveTheShortestPath*)
- Ak je oponent bližšie, obmedzím ho. (*placePlayerBlockingWall*)
  Ak sa mi ho nepodarí obmedziť, idem do cieľa. (*moveTheShortestPath*)

Ak som bližšie k cieľu ako oponent o 1+RAND(3) políčok:
- Skúsim si zabezpečiť cestu. (*protectPath*)
  Ak to nie je potrebné, idem do cieľa. (*moveTheShortestPath*)

Skúsim zabezpečiť cestu. (*protectPath*)
Ak to nepotrebujem, spustím minimax a vykonám ťah ktorý mi ponúkne.

Ohodnocovacia funkcia ktorú využíva tento algoritmus počíta rozdiel dĺžky najkratšej cesty protivníka a dvojnásobku dĺžky najkratšej cesty hráča. Váhu dĺžke najkratšej cesty hráča sme zväčšili na dvojnásobok, aby preferoval pohyb figúrkou. Obmedzovanie protihráčov stenami bude prebiehať rovnako ako u ostatných hráčov, ale nie tak často. V zahájení kladie „Smart Player“ minimálne množstvo stien.

[7] V úvode hry je hĺbka stromu 1, teda minimax funguje ako analyzátor pozícií. Po otvorení hry sa nastaví minimaxu hĺbka stromu 3.

Kapitola 5

Algoritmy pre hru štyroch hráčov

Informácie ktoré dokážeme pri hre štyroch hráčov získať z prehľadávania herného stromu do hĺbky tri pomocou minimaxu nemajú dostatočnú hodnotu, aby sa nám oplatilo počítať možné ťahy dvoch hráčov po nás nasledujúcich. Výpočtovú kapacitu ktorú by sme na to spotrebovali nasmerujeme radšej do hĺbkovej analýzy pozície.

Najprv si uvedieme modifikácie pomocných funkcií z hry dvoch hráčov a následne s ich pomocou zostavíme algoritmy agentov.

5.1 Pomocné funkcie

funkcia *moveTheShortestPath()*

Táto funkcia je rovnaká ako pri hre dvoch hráčov. Vykoná ťah figúrkou po najkratšej ceste smerom k cieľovým políčkam.

funkcia (Boolean) *placeWinnerBlockingWall()*

Táto funkcia je modifikáciou funkcie *placePlayerBlockingWall()* pre hru dvoch hráčov. S tým rozdielom že maximalizuje rozdiel medzi dĺžkou najkratšej cesty oponenta ktorý je najbližšie k cieľu a dĺžkou svojej najkratšej cesty. Z množiny stien ktoré majú tento rozdiel najvyšší vyberie stenu ktorá čo najviac poškodzuje aj ostatných oponentov. Ak neexistuje stena ktorá by tento rozdiel zväčšila, funkcia vráti *false*, v opačnom prípade funkcia vykoná ťah položením steny a vráti *true*.

funkcia (Boolean) *protectPath()*

Funkcia *protectPath()* je rovnaká ako pri hre dvoch hráčov.

5.2 Hráč „Dumb Wall Builder“

Algoritmus tohto hráča sa nachádza v triede *AI4Easy*. Rozhodovanie je okamžié a jednoduché.

Postup uvažovania hráča:

Ak nemám žiadne steny, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak RAND(2) < 1 idem do cieľa. (*moveTheShortestPath*)
V opačnom prípade spustím minimax do hĺbky 1 – čiže ohodnotím všetky možné ťahy a vykonám ťah ktorý mi ponúkne.

Ohodnocovacia funkcia využívaná týmto algoritmom počíta rozdiel súčtu dĺžok najkratších ciest všetkých oponentov a dĺžky najkratšej cesty hráča.

5.3 Hráč „Smart Wall Builder“

Algoritmus tohto hráča sa nachádza v triede *AI4Medium*.

Postup uvažovania hráča:

Ak nemám žiadne steny, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak som najbližšie k cieľu, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak je víťaziaci hráč bližšie k cieľu ako na vzdialenosť 2+RAND(3)
pokúsim sa ho zablokovať. (*placeWinnerBlockingWall*)
ak sa to nepodarí, idem do cieľa. (*moveTheShortestPath*)

Ak mám dve a menej stien, idem do cieľa. (*moveTheShortestPath*)

Ak je RAND(3) < 2, idem do cieľa. (*moveTheShortestPath*)

Ak nebola splnená ani jedna akcia, spustím minimax do hĺbky 1 – čiže ohodnotím všetky možné ťahy a vykonám ťah ktorý mi ponúkne.

Funkcia ktorú používa tento algoritmus na ohodnocovanie prideľuje pozícii hodnotu ktorou je rozdiel súčtu dĺžok najkratších ciest protivníkov, s tým že najkratšia cesta víťaziaceho protivníka je zarátaná dvakrát, a dĺžky najkratšej cesty hráča.

5.4 Hráč „Smart Player“

Algoritmus tohto hráča sa nachádza v triede *AI4Hard*.

Postup uvažovania hráča:

Ak nemám žiadne steny, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak som najbližšie k cieľu, pokúsim sa zabezpečiť si najkratšiu cestu. (*protectPath*)
Ak to nepotrebujem, idem najkratšou cestou do cieľa. (*moveTheShortestPath*)

Ak je víťaziaci hráč bližšie k cieľu ako na vzdialenosť 2+RAND(3)
pokúsim sa ho zablokovať. (*placeWinnerBlockingWall*)
ak sa to nepodarí, idem do cieľa. (*moveTheShortestPath*)

Vyskúšam zabezpečiť cestu (*protectPath*)

Ak mám dve a menej stien, idem do cieľa. (*moveTheShortestPath*)

Ak je RAND(3) < 2, idem do cieľa. (*moveTheShortestPath*)

Ak nebola vykonaná ani jedna z akcií, spustím minimax do hĺbky 1 – čiže ohodnotím všetky možné ťahy a vykonám ťah ktorý mi ponúkne.

Ohodnocovacia funkcia tohto hráča je rovnaká ako hráča „Smart Wall Builder“.

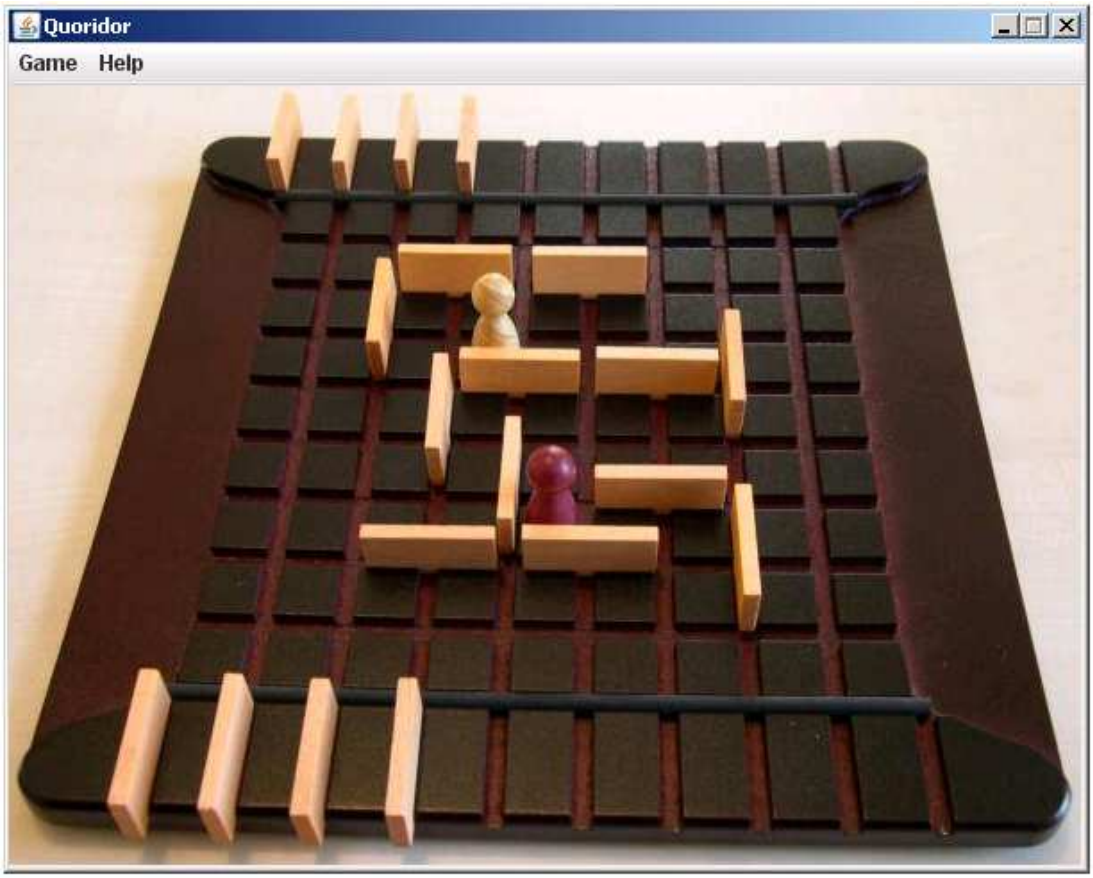
Kapitola 6

Užívateľská dokumentácia

6.1 Spustenie programu

Aplikácia je distribuovaná ako spustiteľný JAR súbor. K svojmu behu vyžaduje Java Runtime Environment[8]. Java technológia zabezpečuje, že program nie je závislý na operačnom systéme. Aplikácia bola testovaná na systémoch Windows a Unix.

Po otvorení spustiteľného JAR súboru sa užívateľovi naskytne pohľad na úvodné okno, ktoré obsahuje menu a obrázok pozície z doskovej hry Quoridor.



Obrázok 6.1: Úvodné okno

6.2 Menu „Game“

Menu „Game“ obsahuje tri položky: „New Game“, submenu „Settings“ a položku „Exit“.

[8] Java Runtime Environment je možné získať na stránke http://java.com/en/download/ .

Obrázok 6.2: Menu „Game“

Po kliknutí na položku menu „New Game“ sa zobrazí dialóg „New Game“ v ktorom si užívateľ vyberie medzi hrou dvoch a štyroch hráčov a nastaví, kto má jednotlivých hráčov ovládať. Následne spustí novú hru. Klávesová skratka pre túto položku je „F2“.

Po kliknutí na „Settings“ sa nám otvorí menu s nastaveniami ktorých význam si popíšeme v podkapitole 6.2.2 Submenu Settings.

Po kliknutí na položku menu „Exit“ sa aplikácia Quoridor ukončí.

6.2.1 Dialóg „New Game“

Po kliknutí na položku v menu „New Game“ sa zobrazí tento dialóg.

Obrázok 6.3: Dialóg „New Game“

Nastavenie „Select game mode“ dovoľuje vybrať mód hry. „2 player game“ čiže hru dvoch hráčov, alebo „4 player game“ čiže hru štyroch hráčov. Dialóg ďalej obsahuje okienka s nastaveniami jednotlivých hráčov. V prípade že je zvolený mód hry 2 hráčov, okienka s hráčmi číslo tri a štyri nie sú editovateľné.

V nastaveniach jednotlivých hráčov môže užívateľ zvoliť meno hráča a typ hráča. Typ môže byť: „Human player“, „Dumb Wall Builder“, „Smart Wall Builder“ a „Smart Player“. „Human player“ je hráč ovládaný užívateľom. Ostatní hráči sú ovládaní počítačom.

Kliknutím na tlačítko **„PLAY“** sa spustí hra s danými nastaveniami.

6.2.2 Submenu „Settings“

Submenu „Settings“ obsahuje nastavenia aplikácie ktoré sú grafického charakteru.

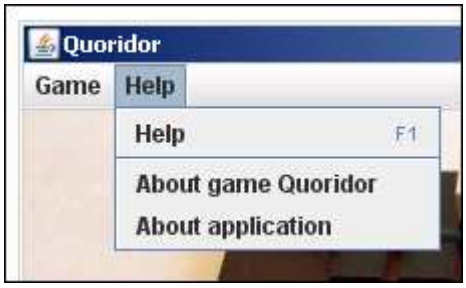
Položka nastavení „Skins“ obsahuje sady farieb pre pozadie, šachovnicu, hráčov a steny. Tieto sady dovoľujú užívateľovi zmeniť vzhľad hry.

Nastavenie „Highlight possible pawn moves“ zvýrazňuje hráčovi políčka na šachovnici, na ktoré sa môže hráč na ťahu pohnúť figúrkou. Štandardne je táto funkcia zapnutá. Klávesová skratka pre túto položku je „M“.

Nastavenie „Highlight final positions“ zvýrazňuje hráčovi na ťahu políčka, na ktoré sa musí so svojou figúrkou dostať, aby zvíťazil. Štandardne je táto funkcia vypnutá. Klávesová skratka pre túto položku je „F“.

6.3 Menu „Help“

Menu „Help“ obsahuje tri položky: „Help“, „About game Quoridor“ a „About application“.



Obrázok 6.4: Menu „Help“

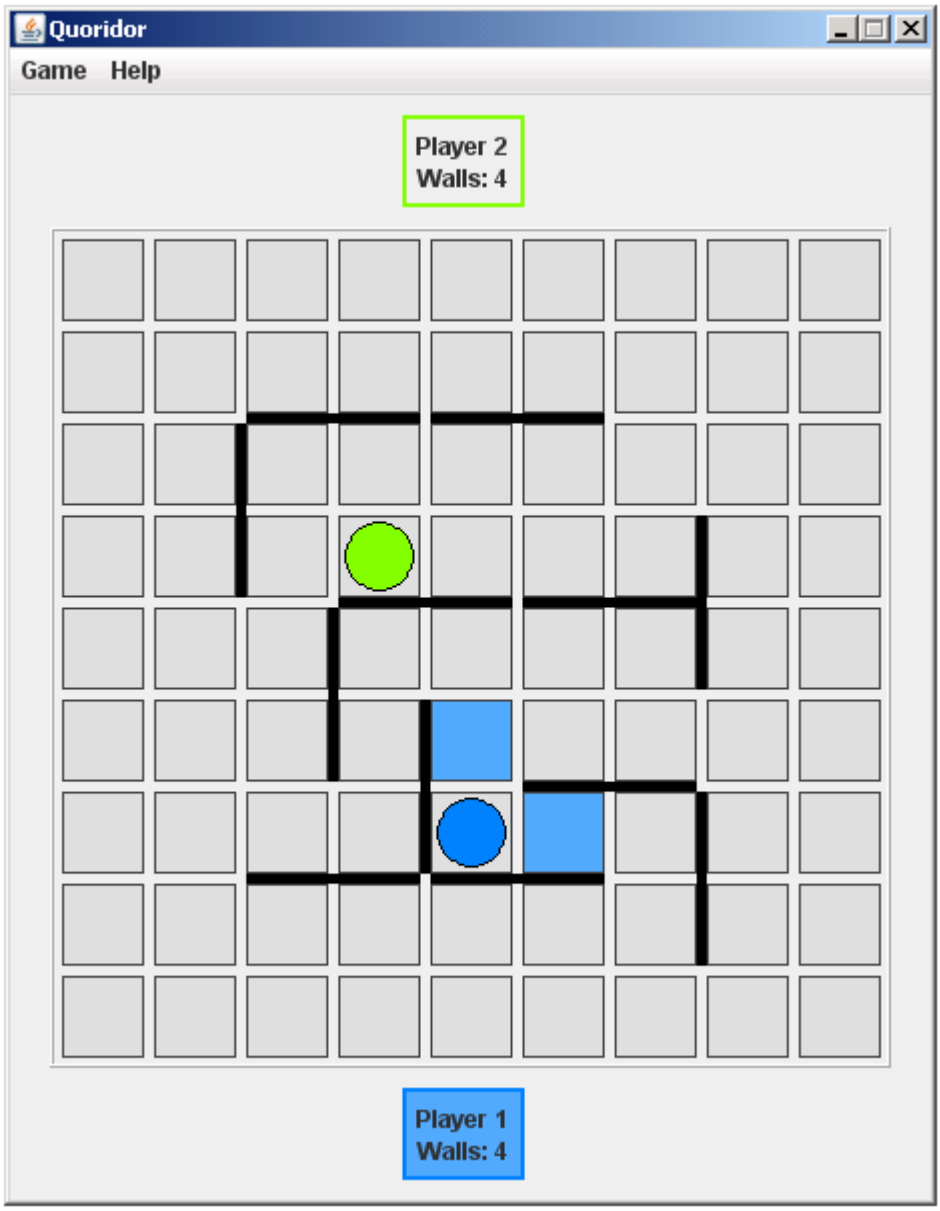
Po klinutí na položku „Help“ sa zobrazí okno s pomocnými informáciami o ovládaní aplikácie. Klávesová skratka pre túto položku je „F1“.

Po kliknutí na položku „About game Quoridor“ sa zobrazí okno s informáciami o hre Quoridor. Informácie sa týkajú vzniku hry a jej pravidiel.

Po kliknutí na položku „About application“ sa užívateľovi zobrazí okno s informáciami o aplikácii Quoridor.

6.4 Ovládanie hry

Hra je ovládaná myšou, konkrétne kliknutím na políčko v prípade pohybu figúrkou a kliknutím medzi políčka v prípade položenia steny.



Obrázok 6.5: Pozícia z hry Quoridor

Políčka na ktoré je povolený pohyb figúrkou sú štandardne zvýraznené. Políčko na ktoré je možné položiť stenu sa zvýrazní, keď sa kurzor myši dostane nad neho.

Každý hráč má pri strane z ktorej začína okienko so svojím menom a počtom stien ktorými disponuje. Toto okienko sa zvýrazní, ak je hráč na ťahu.

Kapitola 7

Záver

Cieľom tejto práce bolo analyzovať hru Quoridor, navrhnúť a naimplementovať rozhodovacie algoritmy pre počítačových hráčov. Algoritmy ktoré sme navrhli sú schopné hrať proti rôzne skúseným oponentom. Najslabší a najjednoduchší počítačom ovládaný hráč je „Dumb Wall Builder“. Má vysokú tendenciu minúť všetky steny hneď v úvode hry a je vhodný ako oponent pre začínajúcich hráčov. „Smart Wall Builder“ je ideálny protivník pre väčšinu hráčov Quoridoru. Ťah vykonáva prakticky okamžite a oproti „Dumb Wall Builderovi“ sa snaží viac pohybovať figúrkou. Najsilnejší hráč s najzložitejším rozhodovacím algoritmom „Smart Player“ je schopný poraziť priemerných ľudských hráčov a odohrať vyrovnanú partiu proti skúseným ľudským hráčom.

Agent pre hru štyroch hráčov využíva navyše jednoduchú formu „aliancie“, čo znamená, že hráči ktorí prehrávajú sa spoločne snažia bojovať proti víťaziacemu hráčovi. Agenti sú konkurencieschopní, aj napriek tomu, že pri hre štyroch hráčov využívame len analyzátor pozície a neprehľadávame strom hry.

Vzhľadom na to, že aplikácia musí fungovať na bežných počítačoch, nie je v súčasnosti možné zväčšovať hĺbku minimaxového prehľadávania. Zlepšenia by sme však mohli dosiahnuť u ohodnocovacích funkcií. Funkcie ktoré sme navrhli sú založené na dĺžke najkratšej cesty. P.J.C. Mertens vo svojej práci [3] zostavil ohodnocovaciu funkciu zo vzdialensti v riadkoch od základnej pozície, rozdielu týchto vzdialeností [9] a na počte políčoch potrebných na posunutie figúrky do ďalšieho riadku smerom k cieľu. Najväčšiu váhu pridelil Mertens práve poslednej zmienenej vlastnosti, počtu políčok potrebných na presun do ďalšej úrovne. Ohodnocovacia funkcia ktorú vytvoril však nie je efektívna a v mnohých prípadoch vyberá nesprávne ťahy. Pretože najkratšia cesta do ďalšej úrovne nemusí byť najkratšia cesta k cieľu. Dokonca môže byť cestou “späť”. Rozhodovací algoritmus vytvorený na základe tejto ohodnocovacej funkcie je na amatérskej úrovni.

Možnosti pre budúci výzkum:

- Na vylepšenie rozhodovacieho algoritmu by bolo treba naimplementovať funkciu, ktorá v prípade že oponent nemá steny, dokáže zúžitkovať steny ktoré má hráč na maximum a pripraví protihráčovi taký plán, aby aj v najlepšom prípade musel prejsť čo najdlhšiu cestu.
- Na optimalizáciu váh vlastností pri ohodnocovacích funkciách je možné použiť evolučné algoritmy. Lisa Glendenning sa zaoberala touto témou vo svojej práci [2].

[9] Rozdiel má indikovať napredovanie jedného z hráčov.

Literatúra

[1] Roman Barták: *Umělá inteligence I*, http://ktiml.mff.cuni.cz/~bartak

[2] Lisa Glendenning (2002): *Mastering Quoridor*, BSc. thesis, University of New Mexico

[3] P.J.C. Mertens (2006): *A Quoridor-playing agent*, BSc. paper

[4] Chris Thornton: *Game playing with Minimax and Pruning*, http://www.cogs.susx.ac.uk/courses/kr/lec05.html

[5] Wikipedia, The Free Encyclopedia: *Dijkstra's algorithm*, http://en.wikipedia.org/wiki/Dijkstra's_algorithm

[6] Wikipedia, The Free Encyclopedia: *Game*, http://en.wikipedia.org/wiki/Game

[7] Wikipedia, The Free Encyclopedia: *Game complexity*, http://en.wikipedia.org/wiki/Game_complexity

[8] Wikipedia, The Free Encyclopedia: *Game tree*, http://en.wikipedia.org/wiki/Game_tree

[9] Wikipedia, The Free Encyclopedia: *Minimax*, http://en.wikipedia.org/wiki/Minimax

[10] Wikipedia, The Free Encyclopedia: *Quoridor*, http://en.wikipedia.org/wiki/Quoridor

[11] Wikipedia, The Free Encyclopedia: *Solved game*, http://en.wikipedia.org/wiki/Solved_game

Dodatok A

Obsah priloženého média

Na priloženom CD nájdeme:

- **adresár „Bin“**

  Obsahuje aplikáciu Quoridor vo forme spustiteľného JAR súboru.

- **adresár „Doc“**

  Obsahuje vygenerovanú programátorskú dokumentáciu k programu.

- **adresár „Text“**

  Obsahuje text tejto práce vo formáte PDF.

- **adresár „Src“**

  Obsahuje zdrojové kódy k programu.

- **súbor „readme.txt“**

  Súbor s potrebnými poznámkami a kontaktnými informáciami.

Zoznam obrázkov