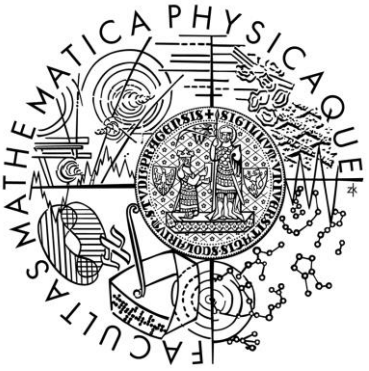
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Lucia Bečvarová

Implementace hry Quoridor ve variantě pro tři hráče

Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.

Studijní program: Informatika

Studijní obor: obecná informatika

Praha 2015

Touto cestou by som sa rada poďakovala svojmu vedúcemu RNDr. Ondřejovi Pangrácovi, Ph.D. za jeho trpezlivosť, cenné rady a usmernenie pri písaní mojej práce.

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona.

V Praze dne..... podpis

Název práce: Implementace hry Quoridor ve variantě pro tři hráče

Autor: Lucia Bečvarová

Katedra / Ústav: Informatický ústav Univerzity Karlovy

Vedoucí bakalářské práce: RNDr. Ondřej Pangrác, Ph.D.

Abstrakt: Quoridor je hra dvou hráčů, cílem které je dostat svou figurku z jedné strany hrací plochy na druhou. Hra je stížená možností prodlužovat cestu soupeře kladením překážek. Tahle práce řeší rozšířenou variantu pro tři hráče. Rozebírá možnosti vytváření umělé inteligence téhle hry. Počítačový protivník je naprogramován pomocí algoritmu Max^n na nalezení optimálního tahu. Vytvořené řešení poskytuje aplikaci, v které může uživatel hrát hru proti třem úrovním protihráčů.

Klíčová slova: quoridor, hry, stratégie

Title: The Quoridor Game variant for three players

Author: Lucia Bečvarová

Department: Computer Science Institute of Charles University

Supervisor: RNDr. Ondřej Pangrác, Ph.D.

Abstract: Quoridor is a 2-player board game. Its objective is to get the player's pawn to the opposite side of the game board from which it begins. The difficulty of this game is increased by the possibility of prolonging the opponent's path by placing walls. This thesis focuses on the variant for three players. It discusses the options of creating an artificial intelligence for this game. The computer opponent is programmed using the Max^n algorithm for finding the optimal move. The solution provides an application which allows the user to play the game against three levels of opponents.

Keywords: quoridor, game, strategy

Obsah

Úvod

Hra Quoridor je stolová hra na vzostupe svojej popularity. Jej silnou stránkou sú až prekvapivo jednoduché pravidlá na to, aké komplikované stratégie si môže vo svojom priebehu vynútiť. V posledných rokoch vzniklo niekoľko prác, ktoré sa venujú práve tejto hre, tak ako aj počítačovým možnostiam programovania protihráča. Takmer žiadna, ak vôbec nejaká, sa však neodchýlila od pôvodných pravidiel a implementácie doposiaľ vznikali väčšinou na verziu pre dvoch hráčov, resp. občas na verziu pre štyroch hráčov.

Táto práca prichádza s rozšírením hry pre troch hráčov. Kladie si za cieľ upraviť hru ako takú, zmeniť hraciu plochu a pravidlá a prispôsobiť ju novému počtu hráčov. Hlavným úmyslom tejto práce je však vytvoriť jednoduchú aplikáciu, ktorá implementuje upravenú verziu hry a navyše bude schopná hrať proti ľudským protihráčom.

Práca je otvorená uvedením do témy, predstavením samotnej hry v originálnej verzii. Následne sú vysvetlené zmeny a rozdiely medzi verziou pre dvoch hráčov a novou verziou pre troch hráčov. Zvyšok práce sa venuje prístupu počítača k riešeniu hry. Najprv sú analyzované hlavné aspekty hry dvoch hráčov, ako je jej zložitosť a algoritmus MiniMax. Podstatná časť práce sa sústreďuje na rozbor počítačového riešenia hry troch hráčov. Spomenuté a vysvetlené sú dva prevzaté algoritmy a ich implementácia. Bližšie sú popísané rozhodovacie algoritmy jednotlivých typov počítačových hráčov.

Ako výstup práce vznikajú rôzne typy hráčov, ktoré sú vzájomným porovnávaním a testovaním usporiadané do troch úrovní počítačových protivníkov. Hotový výsledok, program, v ktorom si užívateľ môže zahrať rozšírenú verziu hry Quoridor sám proti počítaču alebo s inými ľudskými protihráčmi, je súčasťou práce ako jej príloha.

1. Hra Quoridor

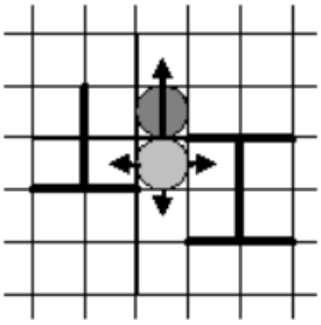
Quoridor je pôvodne stolová abstraktná strategická hra pre dvoch, respektíve štyroch hráčov. Ide o hru, ktorá nezávisí na nijakom kontexte a minimalizuje faktor šťastia hráčov. Jej autorom je Mirko Marchesi a v roku 1997 získala ocenenie Mensa Select. [1] Táto kapitola popisuje pravidlá hry v pôvodnej verzii.

1.1. Pravidlá hry pre dvoch hráčov

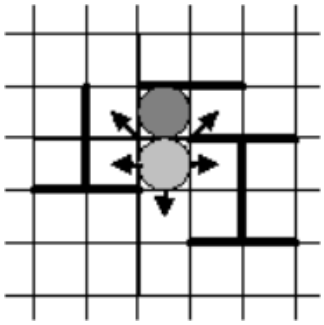
Hra obsahuje figúrky hráčov, steny a hraciu dosku. Doska sa skladá z 81 polí usporiadaných do štvorcovej siete rozmerov 9x9. Každý hráč má k dispozícii svoju vlastnú figúrku a desať stien.

Hra sa začína s prázdnou hracou plochou, bez stien, len s figúrkami. Počiatočné pozície figúrok sú stredy protiľahlých strán hracej plochy. Cieľom hry je dostať svojho panáčika na ľubovoľné políčko steny naproti tej počiatočnej. V hre sa hráči striedajú jednotlivo po odohratých ťahoch. Počas svojho ťahu si hráč môže vybrať jednu z dvoch možných akcií: posunúť svoju figúrku alebo umiestniť stenu.

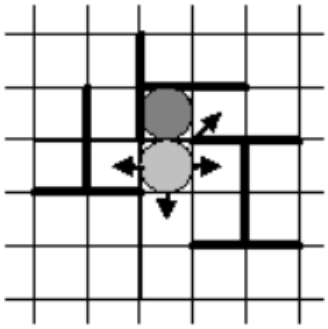
Figúrku môže hráč posunúť o jedno políčko v ľubovoľnom smere (hore, dole, doprava, doľava) s tým, že nemôže prechádzať cez steny. V prípade, že sa na susednom políčku nachádza súperova figúrka, môže ju hráč preskočiť priamo na nasledujúce políčko (obr. 1.1). Ak sa za protihráčovou figúrkou nachádza stena alebo koniec hracej plochy, vtedy ho môže hráč preskočiť nepriamo na jeho susedné políčka. To však môže urobiť len v prípade, že nie sú oddelené stenou (obr. 1.2. a 1.3).



Obrázok 1.1 Priame preskočenie protihráča



Obrázok 1.2 Nepriame preskočenie bez steny



Obrázok 1.3 Nepriame preskočenie so stenou

Steny, resp. prekážky, majú dĺžku dvoch políčok a slúžia na predĺženie súperovej cesty do cieľa, prípadne na zabránenie súperovi v blokovaní. Pokladajú sa priamo na hraciu plochu do medzier medzi políčkami. Stena však nesmie vytŕčať von z hracej plochy a tak isto sa nesmie prekrývať so žiadnou už umiestnenou stenou, a to ani len čiastočne. Hráč tiež nemôže položiť stenu tak, aby sebe alebo protihráčovi úplne znemožnil prístup k cieľu. Teda pre každého hráča musí vždy existovať aspoň jedna cesta do cieľa. Po položení steny sa s ňou už nedá hýbať, ostáva na svojom mieste do konca hry. Hráč nemôže prejsť cez umiestnenú stenu svojou figúrkou.

Hráč vyhráva, keď sa dostane so svojou figúrkou na ľubovoľné zo svojich cieľových políčok. Tým sa hra končí, teda nemôže nastať remíza.

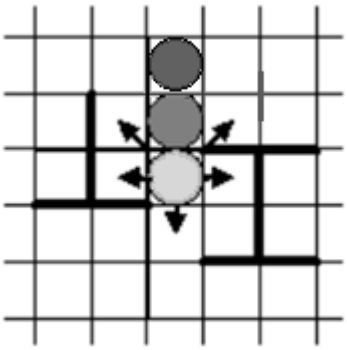
1.2. Pravidlá hry pre štyroch hráčov

Vo verzii pre štyroch hráčov sú pravidlá veľmi podobné, upravené o pár drobností.

Počiatočné pozície figúrok sú stredy každej zo strán hracej plochy. Hráč má k dispozícii len päť stien, namiesto pôvodných desiatich. Cieľom ostáva dostať sa na protiľahlú stenu.

V tomto variante je však zakázané preskočiť cez viac ako jedného protihráča. V prípade, že nastane situácia, kedy by hráč mal preskočiť súpera na políčko

obsadené ďalším súperom, považuje sa toto políčko za nedostupné, rovnako ako keby sa medzi nimi nachádzala stena (obr. 1.4).



Obrázok 1.4 Preskakovanie protihráča vo verzii pre štyroch hráčov

2. Variant hry pre troch hráčov

V nasledujúcej kapitole sú rozobrané zmeny v hre, ktoré sú potrebné na hru pre troch hráčov. Cieľom je čo najmenej sa vzdialiť od pôvodných pravidiel hry pre dvoch hráčov.

2.1. Hracia doska

Aby bola pointa pôvodných pravidiel udržaná, je treba zmeniť hraciu plochu zo štvorca na nejaký n-uholník, kde n je prirodzené číslo deliteľné tromi. V úvahu pripadajú možnosti trojuholník a šesťuholník, keďže možnosť deväť a viac-uholníka je zbytočne komplikovaná a navyše ním nejde dláždiť rovina. Kvôi dodržaniu pravidla, že počiatočná pozícia figúrky je stred strany a nie vrchol hracej plochy, z dvoch spomínaných možností je za výslednú zvolený šesťuholník.

Hracia plocha je teda sieť šesťuholníkov, ďalej nazývaná hexagonálna sieť. Ostáva vyriešiť otázku veľkosti plochy. Keďže hráč začína hru v strede steny hracej plochy, dĺžka steny musí byť nepárne číslo. Pre priblíženie k pôvodnému počtu políčok v hre, čo je 81, je na výber zo strán dĺžky päť alebo sedem. V prvom prípade je spolu na ploche 61 políčok, v druhom prípade 127. Pochopiteľne, k verzii pre dvoch, resp. štyroch hráčov, čo sa počtu políčok týka, je bližšia možnosť so stranou dlhou päť políčok. Avšak pri porovnaní pomeru počtu políčok v hre k počtu hráčov, vyhráva druhá možnosť, teda dĺžka steny sedem. Pomer 127/3 je totiž oveľa bližšie pomeru 81/2 ako pomer 61/3.

2.2. Steny

Doteraz bola za prekážku považovaná rovná stena s dĺžkou dvoch políčok. Dĺžka steny ostáva zachovaná, mení sa len jej tvar. Vo verzii pre troch hráčov sa ako prekážka berie zakrivená stena o uhol 120°, ktorá rozdeľuje dve dvojice susedných políčok, tak ako vo verzii pre dvoch hráčov.

Dôležité je zvoliť počet stien, ktorý budú mať hráči k dispozícii. Ten bude závisieť od veľkosti hracej plochy, pričom snahou je zachovať pomer počtu

vnútorných hrán na hracej ploche (teda miest, na ktoré sa dá stena položiť) k počtu všetkých stien v hre. Tento pomer je v pôvodnej verzii rovný 144/20. V prípade, že hracia plocha má stenu dlhú päť políčok, optimálny je pomer 156/21, ktorý je dosiahnuteľný tým, že každému hráčovi je poskytnutých práve sedem stien. Druhá možnosť, stena dlhá sedem políčok, má optimálny pomer 342/48, ktorý nastane, ak každý hráč bude mať k dispozícii 16 stien.

2.3. Ťah hráča

Hráč má vo svojom ťahu opäť na výber, či posunie svoju figúrku alebo umiestni stenu na hraciu plochu.

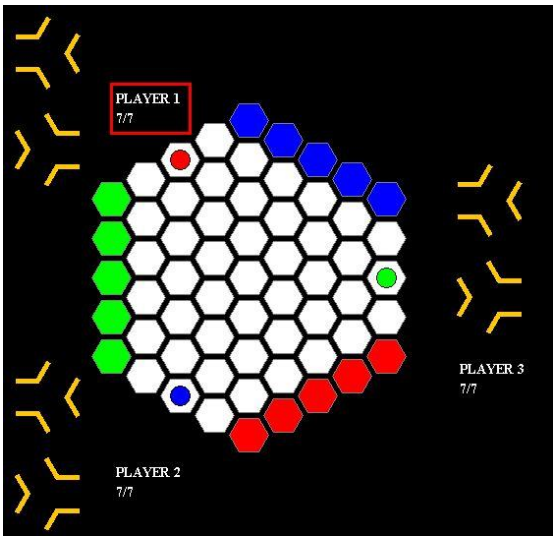
Figúrkou sa dá hýbať vo všetkých smeroch na ľubovoľné susedné políčko, pričom sa nedá prechádzať cez steny. Susedné políčka sú chápané intuitívne ako tie, ktoré spolu zdieľajú hranu. Zobrazené sú na obrázku v zhrnutí kapitoly. Rozdiel oproti pôvodnému variantu je v preskakovaní protihráča. Ak sa na susednom políčku nachádza súperova figúrka, môže ju hráč preskočiť priamo o jedno políčko ďalej. Ak sa nedá preskočiť priamo kvôli stene alebo kvôli umiestneniu tretieho hráča, môže hráč preskočiť na susedné políčka súpera o jednu úroveň ďalej, teda na políčka po bokoch súpera pri 120° uhle. Ak sú aj tieto políčka zablokované, či už stenou alebo tretím hráčom, môže hráč preskočiť na susedné políčka súpera ešte o jednu úroveň ďalej, teda na políčka po bokoch súpera pri 60° uhle. Nové spôsoby preskakovania sú tiež znázornené na konci kapitoly. Treba zdôrazniť, že tak ako vo verzii pre štyroch hráčov, a tu je zakázané preskočiť viac ako jedného protihráča.

Pravidlá pre umiestňovanie stien ostávajú nezmenené. Prekážky sa pokladajú do medzier medzi hracími políčkami. Nemôžu sa žiadnou časťou vzájomne prekrývať ani vytŕčať von z hracej plochy. Stále platí, že stena nemôže byť umiestnená tak, aby nejakému hráčovi znemožnila prístup k cieľu.

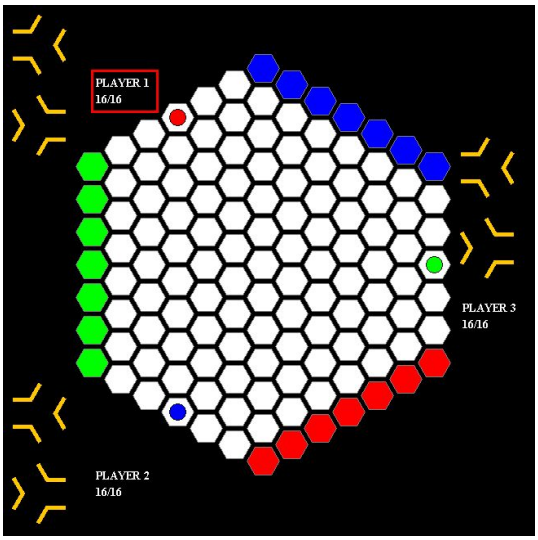
2.4. Zhrnutie

Variant hry Quoridor pre troch hráčov sa hrá na šesťuholníkovej ploche tvorenej hexagonálnou sieťou. Keďže z analýzy uvedenej vyššie sa nedá

jednoznačne rozhodnúť aká veľkosť hracej plochy je optimálna, v implementácii ostanú na výber obidve možnosti, a to hracia plocha tvorená hexagonálnou sieťou so stranami dlhými päť políčok, kde má každý hráč k dispozícii sedem stien (obr. 2.1) alebo stranami dlhými sedem políčok, kde má každý hráč k dispozícii 16 stien (obr. 2.2).

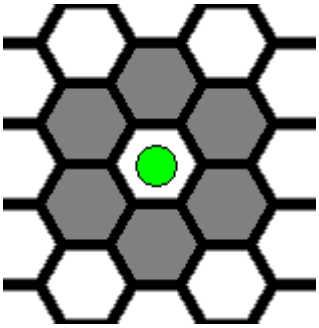


Obrázok 2.1 hracia plocha so stranami dĺžky päť

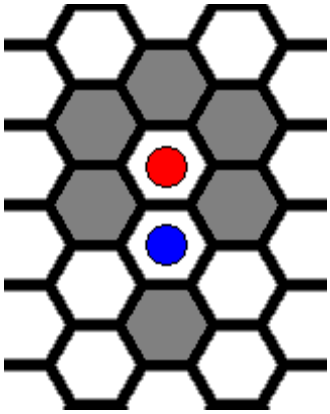


Obrázok 2.2 hracia plocha so stranami dĺžky sedem

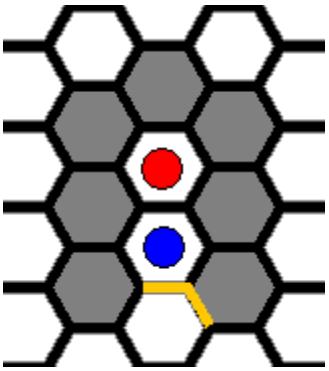
Hráč sa vo svojom ťahu môže posunúť na susedné políčko (obr. 2.3). Ak sa na susednom políčku nachádza protihráčova figúrka, môže ju preskočiť priamo, ak to políčko nie je blokované stenou alebo iným hráčom (obr. 2.4). Ak je blokované, môže ho preskočiť nepriamo na prvej úrovni (obr. 2.5). Ak sa ani to nedá, môže ho preskočiť nepriamo na druhej úrovni (obr. 2.6).



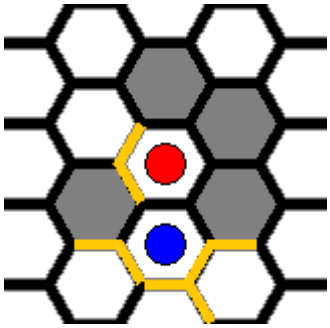
Obrázok 2.3 Susedné políčka



Obrázok 2.4 Priame preskočenie protihráča



Obrázok 2.5 Nepriame preskočenie prvej úrovne



Obrázok 2.6 Nepriame preskočenie druhej úrovne

3. Počítačové riešenie hry pre dvoch hráčov

V tejto kapitole je v stručnosti zhrnutý prístup počítača k riešeniu hry Quoridor vo verzii pre dvoch hráčov.

3.1. Zložitosť

Vzhľadom na to, že Quoridor je ešte stále relatívne nová a nie príliš populárna hra, neboli jej počítačové stratégie až tak dopodrobna preskúmané. Čo sa týka zložitosti hry, dá sa však vypočítať, že Quoridor má podobné hodnoty ako napríklad šach.

Pri zložitosti hier sa skúmajú dve zásadné veličiny. Prvou je veľkosť stavového priestoru, čo je hodnota určujúca počet platných pozícii, ktoré sú dosiahnuteľné z počiatočnej pozície. Ide teda o počet stavov, do ktorých sa hra môže vo svojom priebehu dostať. Tou druhou je veľkosť stromu hry. Táto hodnota určuje počet možných hier, ktoré sa vôbec dajú odohrať. [8]

Hra Quoridor pre dvoch hráčov, podobne ako šach, je hra, ktorá má obidve spomínané hodnoty vysoké. Veľkosť stavového priestoru, ako log so základom 10, v šachu je 46 a v hre Quoridor 42, pričom veľkosť stromu hry, tiež ako log so základom 10, v šachu je 123 a v hre Quoridor 162. Z toho vyplýva, že počet možných odohraných hier v Quoridore je dokonca ešte väčší ako v šachu. [4]

3.2. Algoritmus MiniMax

Najjednoduchší a najčastejší algoritmus, ktorý sa používa pri hrách dvoch hráčov je MiniMax.[6] Je to algoritmus, ktorého cieľom je zvoliť najvýhodnejší ťah hráča z jeho aktuálnej pozície. MiniMax sa môže implementovať iba na hry dvoch hráčov s úplnou informáciou, kde sa hráči jednotlivo po ťahoch striedajú. Hra s úplnou informáciou je hra, v ktorej obidvaja hráči majú všetky informácie o stave hry. V nasledujúcom texte sú hráči rozlíšení ako hráč A a hráč B.

Myšlienka algoritmu je veľmi jednoduchá. Spočíva v tom, že prehľadáva stavový strom, ohodnocuje jeho vrcholy a z nich potom vyberie optimálny ťah. Základom je teda vybudovať stavový strom. Vzhľadom na to, že najmä pri tzv. veľkých hrách ako je napríklad šach, nie je možné skutočne vygenerovať a uložiť do pamäti celý strom naraz, algoritmus je realizovaný rekurziou prehľadávaním do hĺbky. Pre zjednodušenie sa však uvažuje predstava reálneho stromu.

Uzly stromu reprezentujú jednotlivé stavy hry, do ktorých sa hra môže vo svojom priebehu dostať z počiatočného stavu, ktorý je reprezentovaný koreňom stromu. Z každého uzlu vychádza toľko synov, koľko ťahov sa dá z daného stavu vykonať. Hladiny stromu určujú, ktorý hráč je práve na ťahu. Párne hladiny teda predstavujú ťah jedného hráča a nepárne ťah druhého. Nakoniec, listy reprezentujú koncové stavy hry.

Uzly stromu sa ohodnocujú nasledovne. Predpoklad je, že na ťahu je práve hráč A, v opačnom prípade ohodnocovanie prebieha analogicky. Najprv sa ohodnotia listy a to tak, že sa im priradí hodnota 1, ak v koncovom stave vyhral hráč A, -1, ak v koncovom stave vyhral hráč B a 0, ak list reprezentuje remízu. Zvyšné uzly stromu sa ohodnocujú od listov po vrstvách, pričom sa striedavo vyberá minimum a maximum z hodnôt synov na základe toho, ktorý hráč je na ťahu. Ak sa vyberá hodnota uzlu, v ktorom je na ťahu hráč A, tak sa vyberie maximum, pretože sa snaží maximalizovať svoju šancu na výhru, teda dostať sa do koncového stavu ohodnoteného 1. Naopak, hodnota uzlu, v ktorom je na ťahu hráč B je minimum jeho synov, pretože tento hráč sa tiež snaží maximalizovať svoj ťah, teda pre hráča A to bude najhorší možný ťah. Nakoniec, hráč, ktorý je práve na ťahu, si zo svojho stromu vyberie ako výsledný ťah toho syna, ktorý má najvyššie ohodnotenie.

Tento postup je však možný iba v tzv. malých hrách, ako sú napríklad piškvorky na malej obmedzenej ploche. Pri veľkých hrách, ako Quoridor, už nie je možné vypočítať celý strom hry, preto algoritmus počíta iba do určitej hĺbky. Keďže v obmedzenej hĺbke nemusí nutne každý list skutočne reprezentovať koncový stav hry, je treba zmeniť ohodnocovanie listov. Vtedy sa používa tzv. ohodnocovacia funkcia, ktorá každému stavu priradí hodnotu tak, aby stav výhodný pre hráča A mal

kladnú hodnotu a stav výhodný pre hráča B mal zápornú hodnotu. Ohodnocovacia funkcia závisí od konkrétnej hry. Ďalej algoritmus funguje rovnako ako v prípade malých hier.

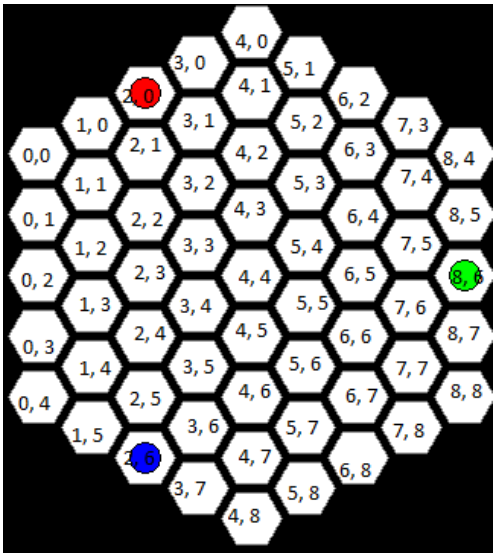
4. Počítačové riešenie hry pre troch hráčov

Hra Quoridor vo svojej verzii tak, ako bola vymyslená, vôbec nepozná variant pre troch hráčov. Z tohto dôvodu tiež neexistuje žiadna analýza počítačových protivníkov ani preskúmané stratégie tohto variantu. Nasledujúca kapitola preto popisuje najdôležitejšie algoritmy, ktoré sú na riešenie problému potrebné a samotnú implementáciu umelej inteligencie hrajúcej hru Quoridor vo variante pre troch hráčov.

4.1. Reprezentácia hry

Pred uvedením použitých algoritmov je nutné vysvetliť reprezentáciu hry v počítači. Najdôležitejšia je samotná hracia plocha. Ako je uvedené v druhej kapitole, ide o hexagonálnu sieť. Tá môže byť v programe reprezentovaná viacerými spôsobmi v závislosti od dôvodu, pre ktorý je reprezentácia potrebná.

V prípade, že je treba vyjadriť vzťah medzi hráčovou figúrkou a hracou plochou, inak povedané, treba zaznamenať hráčovu pozíciu, používa sa súradnicová reprezentácia plochy. Každé políčko je označené dvomi súradnicami x a y. X-ová súradnica označuje stĺpec a y-ová riadok, pričom za riadky v hexagonálnej sieti sa považujú susediace postupnosti šesťuholníkov spojené pravou hornou stranou (obr. 4.1).



Obrázok 4.1 Súradnicová reprezentácia hracej plochy

Dôležitejšie však je, ako je hracia plocha reprezentovaná pri hľadaní cesty hráča do cieľa. V takejto situácii je najvhodnejšie predstaviť si plochu ako graf, aby sa následne dal využiť algoritmus na hľadanie najkratšej cesty v grafe popísaný nižšie.

V nasledujúcom texte je využívaná grafová terminológia tak, ako ju zaviedli Jiří Matoušek a Jaroslav Nešetřil v knihe Kapitoly z diskrétní matematiky.[3] V grafovej reprezentácii hraciu plochu predstavuje neorientovaný graf. Množinu vrcholov tvoria práve hracie políčka. Hrana medzi dvomi vrcholmi vedie práve vtedy, keď políčka reprezentované danými vrcholmi spolu na hracej ploche susedia. Dve políčka na hracej ploche spolu susedia, keď zdieľajú jednu stranu.

4.2. Použité algoritmy

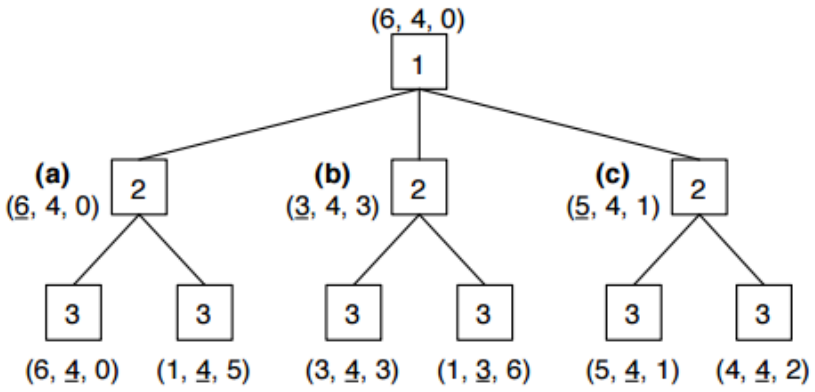
Na konkrétnu implementáciu umelej inteligencie hráčov hry Quoridor vo variante pre troch hráčov bol použitý algoritmus Max^n, ktorý vyberá najvýhodnejší ťah hráča. K tomu ako pomocný algoritmus na hľadanie najkratšej cesty v grafe je využitý Dijkstrov algoritmus. Obidva známe prevzaté algoritmy sú opísane nižšie.

4.2.1. Algoritmus Max^n

Ako je spomenuté v predchádzajúcej podkapitole, MiniMax sa používa len na riešenie hry pre dvoch hráčov, takže sa nedá využiť vo variante pre troch hráčov. Vhodné je však jeho zovšeobecnenie vo forme algoritmu Max^n. [5] Ide o algoritmus, ktorý sa používa na hry s ľubovoľným počtom hráčov a pri použití v hre dvoch hráčov funguje rovnako ako pôvodný algoritmus MiniMax. Hlavným rozdielom však je, že hodnoty uzlov stromu sú n-tice, v ktorých i-tá hodnota reprezentuje skóre i-tého hráča v danom stave. Skóre hráča je vypočítane ohodnocovacou funkciou podobne ako v predošlom prípade.

Ohodnocovanie uzlov opäť začína od listov. Každému listu je priradená n-tica, v ktorej je na i-tej pozícii hodnota ohodnocovacej funkcie pre i-tého hráča v konkrétnom stave. Zvyšné uzly sú ohodnotené zdola po vrstvách tak, že uzlu reprezentujúcemu ťah i-tého hráča je priradená tá n-tica z jeho synov, v ktorej je i-tá

zložka maximálna. Ak je takých možností viac, môže sa vybrať ľubovoľná z nich. Algoritmus nakoniec ako výsledný ťah zvolí ten, ktorý vedie do syna s rovnakým ohodnotením ako v koreni.



Obrázok 4.2 Príklad ohodnotenia uzlov stromu v algoritme Max^n pre troch hráčov

Algoritmus Max^n je tak isto vhodný aj pre veľké hry, avšak znova platí, že stromy veľkých hier sa nedajú vygenerovať celé. V takých situáciách sa opäť obmedzí maximálna hĺbka stromu. Riešenie potom nie je optimálne, teda algoritmus nevyberie ťah, ktorý vedie do víťazného stavu, ale len do stavu s najvyšším skóre.

Priebeh algoritmu je znázornený v nasledujúcom pseudokóde. [2]

```
funkcia maxn (stav, hĺbka)
        ak (hĺbka == 0 alebo stav je koncový)
                vráť hodnota(stav)
        maxHodnoty = maxn(prvýSyn, hĺbka - 1)
        pre každého syna stavu
                hodnoty = maxn(syn, hĺbka - 1)
                ak (hodnoty[hráčNaŤahu] > maxHodnoty[hráčNaŤahu])
                        maxHodnoty = hodnoty
        vráť maxHodnoty
koniec funkcie
```

4.2.2. Dijkstrov algoritmus

Dijkstrov algoritmus je najznámejší algoritmus na určovanie najkratšej cesty v grafe. [3] [7] Vstupom algoritmu je graf $G = (V, E)$ a počiatočný vrchol s. Výstupom sú hodnoty $d(v)$ určujúce vzdialenosť vrcholu v od počiatočného vrcholu s. Pokiaľ ako výsledok nestačí dĺžka cesty, ale je potrebné poznať aj cestu samotnú,

algoritmus je upravený tak, aby vracal aj hodnoty *p(v)* udávajúce predchodcu vrcholu *v* na najkratšej ceste z *s* do *v*. Najkratšia cesta sa získa jednoducho spätným prechodom od cieľového vrcholu k počiatočnému.

Algoritmus hľadá najkratšiu cestu prechodom grafu do šírky, začína v štartovnom vrchole *s*. Výpočet končí po tom, ako sú navštívené všetky vrcholy dostupné z počiatočného vrcholu. Spôsob prechádzania sa riadi priebežne počítanými hodnotami *d(v)*, ktoré sa priraďujú vrcholom. Tieto hodnoty môžu byť buď dočasné alebo trvalé. Na začiatku má každý vrchol dočasnú hodnotu $d(v)=\infty$ a počiatočný vrchol $d(s)=0$. Výpočet prebieha po krokoch, kým nemá každý dostupný vrchol trvalú hodnotu. V jednom kroku sa vykonajú nasledujúce akcie:

1. *Vyberie sa vrchol x s najmenšou dočasnou hodnotou*
2. *Hodnota vrcholu x sa nastaví ako trvalá*
3. *Následníkom vrcholu x, vrcholom i, s dočasnou hodnotou sa prepočíta hodnota:* $\mathbf{d(i) = min(d(i), d(x) + 1)}$*, kde e je hrana medzi vrcholom x a vrcholom i*

Tým sa znížia hodnoty vrcholov, ktoré sa dajú znížiť pomocou cesty vedúcej cez *x*. Vždy keď je takto znížená hodnota vzdialenosti vrcholu *i* pomocou vrcholu *x,* nastaví sa hodnota predchodcu $p(i)=x$, aby bolo možné získať samotnú cestu a nie len jej dĺžku.

5. Implementácia

Táto kapitola sa detailnejšie venuje jednotlivým prvkom a krokom konkrétnej implementácie počítačových protivníkov v hre Quoridor pre troch hráčov, ktorá vznikla ako produkt tejto práce. Sú v nej použité odkazy na konkrétne metódy a triedy, ktoré sú bližšie rozobrané v dokumentácii programu, Prílohe 1.

5.1. Rozhodovanie

Princíp rozmýšľania počítačového hráča je založený na skúmaní možných ťahov, ich vyhodnocovaní a následnom zvolení najvýhodnejšieho ťahu. Hlavná časť logiky tohto rozhodovania je obsiahnutá v triede AutoPlayer. Nachádza sa v nej funkcia makeAMove(), ktorá z daného stavu zvolí z hráčových možných ťahov ten najvýhodnejší.

Rozhodovania počítačového hráča je v zásade priamočiare, pričom detaily sú rozobrané v nasledujúcich podkapitolách:

- *Ak nastane nejaká špeciálna situácia, vykonaj akciu v závislosti od situácie*
- *Ak nenastane žiadna špeciálna situácia, zvoľ najvýhodnejší ťah pomocou algoritmu Max^n alebo pomocou ohodnocovania ťahov*

5.1.1. Voľba najvýhodnejšieho ťahu

V prípade, že sa hráč nedostane do stavu, ktorý treba riešiť zvláštnym spôsobom, sú dve možnosti výberu optimálneho ťahu. Prvou je voľba najlepšieho ťahu pomocou ohodnocovacej funkcie bez počítania do hĺbky a druhou je výpočet najlepšieho ťahu s prehľadávaním do hĺbky pomocou algoritmu Max^n.

5.1.1.1. Ohodnocovanie ťahov

Ohodnocovanie ťahov je postup, pri ktorom sa neprihliada na možný budúci priebeh hry, teda neprehľadáva sa stavový strom. Algoritmus jednoducho prejde všetky možné (a zároveň zmysluplné) ťahy hráča, ktorý je práve na rade, ohodnotí

ich ohodnocovacou funkciou a vyberie ten ťah, ktorý má najvyššie ohodnotenie. V prípade, že je takých ťahov viac, vyberie z nich náhodne jeden ťah s tým, že pred umiestnením steny uprednostňuje posunutie figúrky.

Ťahy sa ohodnocujú ako porovnanie stavu, z ktorého hráč vychádza a stavu, do ktorého sa vykonaním ťahu dostane. Dôležité sú pri ňom štyri veličiny: rozdiel vzdialeností hráča od cieľa (*A*), rozdiel vzdialeností lepšieho protihráča (*B*), rozdiel vzdialeností horšieho protihráča (*C*) a rozdiel počtu stien, ktoré má hráč k dispozícii (*D*). Všetky veličiny sa počítajú ako rozdiel danej hodnoty pred a po vykonaní ťahu. Za lepšieho protihráča je považovaný ten, ktorý je v počiatočnom stave bližšie k cieľu. Ak je vzdialenosť obidvoch súperov od cieľa rovnaká, ako lepší protihráč je označený ten, ktorý nasleduje bezprostredne po hráčovi, ktorý je aktuálne na ťahu.

Porovnanie stavov sa v programe počíta funkciou *valuationFunction* a má nasledujúci predpis:

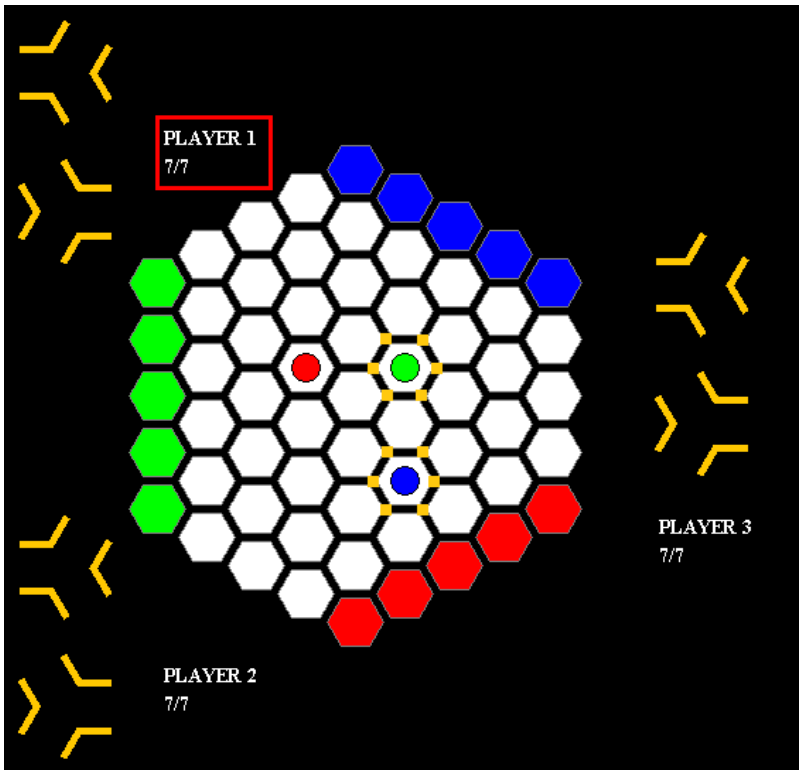
$$f(\text{ťah}) = a * A - b * B - c * C - d * D$$

Za najvýhodnejší ťah je zvolený ten ťah, ktorý ma ohodnotenie najvyššie. Funkcia je nastavená tak, aby kladná hodnota veličiny *A* maximalizovala počet políčok o koľko sa hráč priblíži k cieľu a záporné hodnoty zvyšných veličín maximalizujú počet políčok o koľko sa protihráči od cieľa vzdialia.

Koeficienty *a, b, c, d* sa líšia v závislosti od typu hráča, ako je rozobrané v Kapitole 5.2., prípadne od špeciálnej situácie. Všeobecne povedané, koeficienty vyjadrujú, ktorá z akcií má najvyššiu prioritu.

Potrebné je ešte vysvetliť, aký ťah sa považuje za zmysluplný. V prvom rade je to legálny ťah, ktorý je v súlade s pravidlami. Je to buď posun panáčika na susedné políčko, ktoré nie je oddelené stenou alebo na políčko, kam sa dá preskočiť cez protihráča, alebo je to umiestnenie steny na hraciu plochu tak, aby sa neprekrývala so žiadnou už položenou stenou, nevytŕčala von z hracej plochy a aby po jej položení pre každého hráča existovala aspoň jedna cesta do cieľa. V druhom rade sa za

zmysluplný ťah považuje taký legálnych ťah, ktorý môže ovplyvniť priebeh hry. Čo sa týka posunu figúrky, môže nastať maximálne sedem možností, čo je dostatočne malé množstvo na to, aby ho bolo treba orezávať. Čo sa týka položenia steny, môže nastať až 288 možností v menšej verzii hracej plochy a až 648 vo väčšej verzii. To je príliš veľké množstvo skúmaných ťahov, najmä vzhľadom na to, že väčšina z nich je zbytočná. Z toho dôvodu je ako zmysluplná pozícia umiestnenia steny považovaná len pozícia bezprostredne pri protihráčovi (obr. 5.1). Vzhľadom na to, že blokovať protihráča má význam len na jeho najkratšej ceste do cieľa, tento spôsobom orezania možností určite nevylúči optimálny ťah. Ak totiž na súperovej najkratšej ceste existuje lepšia možnosť ako tá, ktorú algoritmus uvažuje, bude ju uvažovať niekedy počas nasledujúcich ťahov, keď sa k nej protihráč dostane. Tým sa práve účinnosť bloku ešte zvýši, keďže hráč zablokuje súpera neskôr, a teda protihráčovu cestu predĺži o to viac. Táto stratégia sa štandardne nepoužíva v hre pre dvoch hráčov (v tom prípade môže byť výhodné položiť stenu za svoju figúrku, keďže sa predpokladá, že tade by išiel protihráč do svojho cieľa). Vzhľadom na to, že vo verzii pre troch hráčov neplatí, že štartovacia stena jedného hráča je cieľová stena iného hráča a najmä vzhľadom na ohodnocovaciu funkciu uvedenú vyššie, môžeme skúmané ťahy naozaj orezať uvedeným spôsobom. Najdôležitejšie je však zdôrazniť, že medzi zmysluplné ťahy sa radia pokladania stien okolo protihráča len vtedy, keď hráč, ktorý je práve na rade má k dispozícii aspoň jednu stenu a zároveň vzdialenosť protihráča od cieľa je menšia ako vzdialenosť hráča. Čiže hráč sa pokúša blokovať súpera len vtedy, keď je súper bližšie k cieľu.



Obrázok 5.1 Zmysluplné pozície umiestnenia stien červeného hráča číslo 1

5.1.1.2. Použitie algoritmu Max^n

Iným prístupom k voľbe optimálneho ťahu je algoritmus Max^n. Ako funguje a kedy sa používa je spomenuté v Kapitole 4.2.1. V konkrétnej implementácii sa používa do maximálnej hĺbky 3. To je hodnota, v ktorej algoritmus vracia dostatočne dobré výsledky (viac v Kapitole 5.2.) a zároveň nie je príliš pomalý. Aj napriek tomu však výpočet jedného ťahu pomocou tohto algoritmu trvá niekoľko sekúnd.

Princíp ohodnocovacej funkcie v algoritme Maxn je rovnaký ako v predošlom spôsobe ohodnocovania ťahov. Detaily sa však trochu líšia, preto je na výpočet skóre v programe určená zvlášť funkcia, *scoreFunction*. Zásadným rozdielom je ten, že skóre, ktoré sa počas algoritmu počíta je trojica a nie len jedna hodnota. Jednotlivé hodnoty v trojici sa však počítajú takmer rovnakou ohodnocovacou funkciou, pričom sa ale neprihliada na špeciálne situácie. Na druhej strane sa však vo funkcii prihliada tiež na zmenu počtu stien, ktoré majú k dispozícii súperi (E, F). To z toho dôvodu, že v určitých prípadoch môže byť výhodné prinútiť súpera minúť si vlastnú stenu. Ohodnocovacia funkcia v algoritme Max^n teda vyzerá takto:

$$f(stav) = 4 * A - 2 * B - 2 * C - 1 * D + 0{,}5 * E + 0{,}5 * F$$

V tomto prípade už koeficienty sú koeficienty konštanty, ktoré sa počas hry nijak nemenia. To z toho dôvodu, že algoritmus Max^n sa používa len pri jednej z úrovní počítačových hráčov a nezohľadňuje žiadne špeciálne situácie.

5.1.2. Špeciálne situácie a ich riešenie

V priebehu hry môže nastať niekoľko výnimočných situácii, ktoré si vyžadujú iný prístup ako bežné situácie. Tieto situácie a ich postupy riešenia sú rozobrané v bodoch nižšie. V tejto sekcii „hráč A“ označuje hráča, ktorý je práve na ťahu, „hráč B“ nasledujúceho hráča a „hráč C“ posledného hráča.

1. **Vzdialenosť hráča A od cieľa je jedno políčko**

   V prípade, že sa hráč vo svojom ťahu môže posunúť do cieľovej pozície, a tým pádom vyhrať, nepočíta sa ohodnocovacia funkcia pre položenie steny a koeficient *a*, určujúci prioritu hráča znížiť svoju vzdialenosť od cieľa, sa nastaví na maximum. Tým je zaručené, že hráč posunie svoju figúrku do cieľa a vyhrá.

2. **Vzdialenosť hráča B od cieľa je jedno políčko**

   Ak má hráč A k dispozícii aspoň jednu stenu, musí hráča B blokovať, inak by hráč B v nasledujúcom ťahu vyhral. Koeficient *b*, určujúci prioritu blokovania lepšieho hráča (v tomto prípade hráča B) sa nastaví na maximum. Tým je zaručené, že hráč A zablokuje protihráča B, ak je to vzhľadom k pravidlám možné. Ak nie, najlepší ťah sa vypočíta štandardným spôsobom.

3. **Vzdialenosť hráča C od cieľa je jedno políčko**

   V tejto situácii rozlišujeme dva nasledujúce prípady. Ak hráč B má k dispozícii aspoň jednu stenu, hráč sa spoľahne na to, že hráč B vo

svojom záujme zablokuje hráča C, a teda nemusí ho blokovať sám. Žiadne koeficienty sa nemenia, ťah sa vyberie štandardným spôsobom.

Naopak, ak hráč B nemá žiadne steny, ktorými by mohol hráča C blokovať, musí tak vykonať hráč A sám a to rovnakým spôsobom ako je popísaný v 2. bode.

4. **Vzdialenosť hráča B od konca „chodbičky“ je jedno políčko**

Najprv je potrebné zadefinovať si pojem „chodbička“. Ide o postupnosť vzájomne susediacich políčok, ktorá sa začína na nejakom z cieľových políčok protihráča. Postupnosť je tvorená políčkami, ktoré majú práve dve susedné políčka. Končí sa teda na políčku, z ktorého je možné posunúť sa na tri a viac iných políčok. Presne týmto spôsobom je chodbička detekovaná v programe (viď funkcie *existujeChodba*, *koniecChodby*). Je to teda priama cesta (dokonca podľa grafovej terminológie ide naozaj práve o cestu), ktorá sa nemá ako vetviť. Z pohľadu hry je to chodba vytvorená stenami a/alebo hranicou hracej plochy taká, že keď raz do nej hráč vojde, nedá sa ho už nijak blokovať. Z toho dôvodu je dôležité súperovi zabrániť vôbec stúpiť na počiatočné políčko takto definovanej chodbičky. Situácia sa však uvažuje len ak je hráč A od svojho cieľa vzdialenejší ako hráč B a zároveň hráč A má k dispozícii aspoň jednu stenu. Rieši sa opäť tým, že v ohodnocovacej funkcii sa zvýši koeficient blokovania hráča B.

5. **Vzdialenosť hráča B od konca „chodbičky“ je jedno políčko**

Táto situácia je vyhodnocovaná rovnako ako v predošlom bode.

6. Hráč vie zabrániť protihráčovi v blokovaní

V určitých prípadoch sa môže stať, že existuje pozícia pre položenie hrany hráča tak, aby znemožnil súperovi blokovať. Tento princíp je založený na pravidlách o pokladaní stien, a to na pravidle, že steny sa nemôžu ani čiastočne prekrývať a na pravidle, že pre každého hráča musí existovať aspoň jedna cesta do cieľa. Na základe týchto poznatkov môže hráč položiť vlastnú stenu tak, aby už protihráč kvôli uvedeným pravidlám nemohol položiť svoju stenu na blokovanie hráča.

Túto stenu nájdeme postupom popísaným nižšie, v programe realizovaným pomocnými funkciami *opponentsBlock* a *blockMyself*. Je na mieste spomenúť, že takýto druh blokovania je úplne iný ako vo verzii pre dvoch hráčov. Rozdiely plynú najmä z odlišných tvarov stien a hracej plochy.

1. Nájdi pozíciu steny, ktorou vie súper čo najefektívnejšie blokovať hráča (funkcia *opponentsBlock*) a dočasne ju polož do hry.
2. Ak taká existuje, nájdi pozíciu steny, ktorá porušuje nejaké zo zmienených pravidiel na pokladanie steny (funkcia *blockMyself*) práve kvôli súperovej dočasnej stene, čiže buď nájdi stenu, ktorá sa aspoň čiastočne prekrýva so súperovou stenou, alebo nájdi stenu, ktorá len vďaka súperovej stene úplne zablokuje všetky možné cesty hráča do cieľa.
3. Ak taká existuje, odstráň súperovu dočasnú blokovaciu stenu z kroku 1, dočasne polož vlastnú stenu z kroku 2 a znova nájdi súperov najefektívnejší blok.
4. Porovnaj rozdiel súperovho pôvodného bloku z kroku 1 (teda o koľko súper svojou stenou predĺži hráčovi cestu do cieľa) s blokom súpera po položení hráčovej dočasnej steny z kroku 3. Ak je rozdiel týchto dvoch hodnôt väčší ako 1, stena nájdená v kroku 2 je zvolená za hráčov ťah.

Môžeme si všimnúť, že rozdiel, ktorý sa počíta v poslednom kroku vlastne znamená, o koľko políčok vie hráč zachrániť svoju najkratšiu cestu do cieľa. Ak ju vie zachrániť aspoň o dve políčka, považuje sa taký ťah za výhodný a hráč ho vykoná. Opäť platí, že táto situácia sa uvažuje len v prípade, že hráč A má k dispozícii aspoň jednu stenu.

Na koniec je treba definovať, ktorá akcia sa vykoná, ak nastane viac špeciálnych situácii naraz. Prvé tri spomenuté prípady sú najdôležitejšie, ak nastane ľubovoľný z nich, ostatné špeciálne prípady sa neuvažujú. Priority prvých troch prípadov sú vzájomne nastavené tak, ako sú usporiadané v zozname, čiže napríklad ak hráč a zároveň nejaký jeho protihráč sú od cieľa vzdialení na jeden krok, vyššiu prioritu má hráčova výhra ako blokovanie súpera. Prípady čísla 4 a 5 sú si v princípe ekvivalentné, líšia sa len tým, ktorý súper je práve vyhodnotený ako lepší (ktorý je bližšie k cieľu). Preto koeficient blokovania lepšieho hráča je nastavený na vyššiu hodnotu ako koeficient horšieho hráča. Avšak koeficienty sú stále nastavené tak, že ak existuje spôsob, ktorým sa dá blokovať horšieho protihráča výrazne efektívnejšie ako toho lepšieho (hráč slabšiemu súperovi vie predĺžiť cestu aspoň o dve políčka viac ako silnejšiemu súperovi), tak ten ťah vykoná.

Poslednou možnosťou, ako môže nastať viac špeciálnych situácii naraz je, že nastanú súčasne prípady 4 a 6 (resp. 5 a 6). Môžeme si všimnúť, že postup bude rovnaký bez ohľadu na to, či so 6. prípadom nastane spolu 4. alebo 5. prípad, preto môžeme ďalej hráča B a C označiť obecne ako o protihráča. Rozhodovanie o tom, ktorá situácia sa rieši prioritne závisí na tom, či je hráč bližšie k svojmu cieľu ako protihráč. Ak áno, uvažuje sa posledná, šiesta, situácia, teda hráč bráni súperovi v blokovaní. Úvaha vychádza z toho predpokladu, že vzhľadom k vzdialenostiam hráčov od ich cieľov bude protihráč hráča blokovať. Na druhej strane, ak je pred vykonaním ťahu protihráč bližšie k cieľu ako hráč, uvažuje sa situácia číslo 4 (resp. 5), pretože ak by hráč súpera neblokoval, ten by v nasledujúcom ťahu vošiel do chodbičky a keďže z definície chodbičky už súper nemôže byť blokovaný a je bližšie k cieľu, protihráč by sa dostal do cieľa ako prvý a hráč by hru prehral.

5.2. Typy a úrovne hráčov

5.2.1. Testovanie

Ako je spomenuté v predošlých sekciách, jednotlivé typy počítačových hráčov sa líšia v použitom algoritme a ohodnocovacej funkcii. Vzhľadom na to, že variant hry pre troch hráčov doteraz neexistoval, a teda neboli ani skúmané žiadne herné stratégie, nie je k dispozícii žiadna ohodnocovacia funkcia, ktorá sa dá pri vyhodnocovaní ťahov použiť. Preto bola na účely tejto práce vymyslená vlastná funkcia tak, ako bola zadefinovaná v časti 5.1.1.1. Typy hráčov závisia na zvolených koeficientoch vo funkcii, teda rozlišujú priority jednotlivých možností ťahu. Po početnom testovaní rôznych možností koeficientov boli za najúspešnejšie zvolené tri varianty, ktorých hodnoty sú v nasledujúcich tabuľkách.

1. Variant – hráč uprednostňujúci pohyb vlastného panáčika

a	***b***	***c***	***d***
5	3,5	0	1,5

Tabuľka 5.1

2. Variant – blokujúci súpera s rovnakou prioritou ako pohybom vlastného panáčika

a	***b***	***c***	***d***
4	4	1	1

Tabuľka 5.2

3. Variant – koeficienty meniace sa na základe presnej vzdialenosti hráča od cieľa

Vzdialenosť hráča od cieľa	***a***	***b***	***c***	***d***
viac ako 3	5	3,5	2	1
3	5,5	4	3	1
2	6	4,5	3,5	1

Tabuľka 5.3

V Tabuľke 5.3 je vidieť, že koeficienty určujúce prioritu pohybu hráčovho panáčika a blokovania súperov závisí od vzdialenosti hráčov od cieľa. Pre každý z koeficientov *a, b, c* sa však vždy uvažuje len vzdialenosť príslušného hráča, čiže pre koeficient *a* uvažujeme vzdialenosť hráča, ktorý je na ťahu, pre koeficient *b* lepšieho protihráča a pre koeficient *c* horšieho protihráča. Koeficient *d* určujúci cenu straty steny hráča sa so vzdialenosťou nemení.

Nakoniec je treba spomenúť, že koeficienty sa počas hry menia aj na základe špeciálnych situácii tak, ako je to napísané v predošle podkapitole. Vzhľadom na to, že špeciálne situácie sa riešia vždy rovnako, bez ohľadu na typ hráča, nie je potrebné venovať sa v tejto časti konkrétnym hodnotám koeficientov, ktoré sa v takých prípadoch nastavujú.

Po tom, ako máme zadefinované typy hráčov a ich ohodnocovacie funkcie, treba ich medzi sebou vzájomne porovnať. Najprv boli porovnávaní len počítačoví hráči bez použitia algoritmu Max^n. Z nich bol vybraný najúspešnejší typ hráča, na ktorom bol následne testovaný algoritmus Max^n. Všetky testy prebiehali na menšom type dosky, teda na hracej ploche so stenou dĺžky 5.

Úvodné testy boli zamerané na vplyv poradia hráča na výsledky hry. Pre každý typ hráča prebehli tri testy, pričom v každom bolo odohratých 20 hier a všetci hráči mali nastavený rovnaký typ. Z výsledkov ako najvýhodnejšia pozícia vyšla práve druhá. Hráč, ktorý začína ako druhý vyhral 50% hier, prvý približne 35% a tretí len zvyšných 15%. To z toho dôvodu, že hráč, ktorý ide ako prvý, sa najskôr

dostane do situácie, ktorou ohrozí súperov svojou šancou na výhru. Avšak druhý hráč sa často vo svojom ťahu spoľahne na to, že tretí bude blokovať toho prvého. Tým získa pred svojimi súpermi jeden ťah k dobru a dostáva sa do vedúcej pozície.

Ako prvé boli vzájomne porovnávané varianty č. 1 a 2. Výsledky tohto testovania sú v nasledujúcej tabuľke.

Poradie hráča	**1**	**2**	**3**
Variant	1	1	2
Počet výhier	13	13	14
Variant	1	2	1
Počet výhier	9	17	14
Variant	2	1	1
Počet výhier	11	9	20
Variant	2	2	1
Počet výhier	4	16	20
Variant	2	1	2
Počet výhier	14	20	6
Variant	1	2	2
Počet výhier	6	20	14

Tabuľka 5.4

Keďže poradie hráčov môže ovplyvniť výsledky, varianty č. 1 a 2 boli testované vo všetkých možných kombináciách, vždy po 40 odohratých hrách. V Tabuľke 5.4 je pre každú sadu 40 hier, pre každý typ hráča a pre každé jeho poradie v hre uvedený počet výhier. S prihliadnutím na to, že ako sa v úvodnom teste ukázalo, má hráč na druhej pozícii zvýhodnenú pozíciu, z výsledkov môžeme vyvodiť záver, že typ č.1 je v priemere úspešnejší ako typ č.2, aj keď výsledky sú veľmi tesné a nie úplne rozhodné. Navyše si môžme všimnúť, že úspešnosť počítačového hráča závisí nie len na poradí a variante ako takých, ale najmä na ich kombinácii. Teda napríklad v Tabuľke 5.4 vidíme, že hráč na tretej pozícii je úspešnejší, ak je iného typu ako prví dvaja hráči.

V nasledujúcom teste sa porovnával tretí typ s prvým, teda s tým úspešnejším. Test prebiehal rovnako ako v predošlom prípade, výsledky sú v Tabuľke 5.5.

Poradie hráča	1	2	3
Variant	1	1	3
Počet výhier	2	10	28
Variant	1	3	1
Počet výhier	6	26	8
Variant	3	1	1
Počet výhier	20	10	10
Variant	3	3	1
Počet výhier	10	14	16
Variant	3	1	3
Počet výhier	14	14	12
Variant	1	3	3
Počet výhier	10	12	18

Tabuľka 5.5

Z výsledkov v tabuľke vidíme, že tretí typ hráča, teda typ, ktorý počíta svoje koeficienty počas hry v závislosti od vzdialenosti hráčov, v priemere vyhráva oveľa častejšie ako variant č. 1. Je teda zo všetkých troch typov najúspešnejší. Z toho dôvodu bol práve tento typ zvolený na testovanie algoritmu Max^n.

Úspešnosť algoritmu Max^n sa testovala rovnakým spôsobom ako v predošlých prípadoch. Pre počítač najnáročnejší bol počiatočný test, v ktorom proti sebe hrali dvaja hráči bez spomínaného algoritmu a jeden s ním. Hráč, ktorý využíval na rozhodovanie algoritmus Max^n hral schválne až na tretej pozícii, aby sa zistilo, či je úspešný aj z najnevýhodnejšej štartovacej pozície. Navyše, maximálnu hĺbku stromu, s ktorou počítal bola nastavená na 4. To je síce hodnota, ktorá je pre hru kvôli svojej časovej náročnosti nepoužiteľná (20 odohraných hier trvalo až 15 hodín), no bola testovaná, aby sa zistilo, či má vôbec význam používať algoritmus Max^n a do akej hĺbky. Výsledky testu hovoria jasne v prospech algoritmu. Z 20 odohratých hier až 13 vyhral hráč využívajúci Max^n, a to aj napriek nevýhodnej štartovacej pozícii. Tento istý test bol prevedený ešte jedenkrát s tým rozdielom, že hĺbka bola znížená na hodnotu 3. Pri tejto hĺbke trvá jeden ťah hráča okolo 8 sekúnd, čo je doba čakania, ktorá neznemožňuje hru a oplatí sa čakať ak je hráč dostatočne dobrý. Test prekvapivo skončil úplne rovnako ako ten s väčšou hĺbkou, teda opäť 13 výhier hráča s Max^n. Z toho vyplýva, že použitie algoritmu Max^n je určite

výhodnejšie v porovnaní s jednoduchým ohodnocovaním ťahov a stačí počítať stavový strom len do hĺbky 3.

Pre istotu boli vykonané testy algoritmu aj pre iné možnosti rozdelenia pozícií hráčov. Nasledujúca tabuľka zobrazuje výsledky porovnania hráča s algoritmom Max^n a hráčov používajúcich jednoduché vyhodnocovanie ťahu, konkrétne najúspešnejší variant č. 3.

Poradie hráča	**1**	**2**	**3**
Použitý algoritmus	Max^n	Vyhodnocovanie ťahov variant č.3	Vyhodnocovanie ťahov variant č.3
Počet výhier	16	8	16
Použitý algoritmus	Vyhodnocovanie ťahov variant č.3	Max^n	Vyhodnocovanie ťahov variant č.3
Počet výhier	10	20	10
Použitý algoritmus	Vyhodnocovanie ťahov variant č.3	Vyhodnocovanie ťahov variant č.3	Max^n
Počet výhier	6	8	26

Tabuľka 5.6

5.2.3. Úrovne hráčov

Výsledky testovania priamo určujú poradie typov hráčov podľa ich úspešnosti. V konečnej implementácii som sa rozhodla len pre tri úrovne hráčov. Dôvod je ten, že varianty 1 a 2 mali natoľko podobné výsledky, že je zbytočné ich rozdeľovať do zvláštnych úrovní. Preto variant č. 2 nie je v implementácii zahrnutý vôbec a variant č. 1 je použitý ako najnižšia, prvá úroveň. Variant č. 3 je o niečo úspešnejší, keďže do svojich výpočtov dynamicky zahŕňa aktuálne vzdialenosti hráčov od cieľa, preto je zvolený za druhú, strednú úroveň. Na najvyššiu, čiže tretiu úroveň, bol na základe výsledkov testovania zvolený variant č. 3, ktorý navyše využíva algoritmus Max^n do hĺbky 3.

Záver

Táto práca sa venovala doskovej hre Quoridor, jej rozšíreniu pre troch hráčov a počítačovému riešeniu. Cieľom bolo navrhnúť zmenu hry a jej pravidiel tak, aby ju mohli hrať traja hráči namiesto pôvodného počtu dvoch, resp. štyroch hráčov. Hlavným zameraním práce bola samotná implementácia hry tak, aby ju mohol užívateľ hrať proti počítačovým protivníkom. To zahŕňalo analýzu a implementáciu herných stratégií a rozhodovacích algoritmov.

V úvode práce bola predstavená hra Quoridor vo svojom pôvodnom variante spolu s pravidlami pre dvoch hráčov. Nasledoval rozbor zmien potrebné na rozšírenie pre troch hráčov. Hracia plocha bola upravená na hexagonálnu sieť dvoch rôznych veľkostí. Steny, ktoré môže hráč na plochu pokladať sa zmenili z rovných prekážok na prekážky lomené v 120° uhle. Počet stien, ktoré má hráč k dispozícii bol upravený 7 alebo 16, v závislosti od veľkosti hracej plochy. Vzhľadom na zvýšený počet hráčov a rozdielnu hraciu plochu boli mierne pozmenené pravidlá o preskakovaní figúrok.

Ďalej bolo popísané počítačové riešenie hry pre dvoch hráčov. To obsahovalo odhad zložitosti hry Quoridor a jeho porovnanie so šachom a popis najzákladnejšieho algoritmu používaného na hry dvoch hráčov – MiniMax.

Najrozsiahlejšou časťou práce bola analýza počítačového protivníka vo verzii pre troch hráčov. Jej súčasťou bolo popísanie použitých algoritmov, menovite Dijkstrov algoritmus a Max^n. Ďalej bola navrhnutá ohodnocovacia funkcia, na ktorej bol postavený celý rozhodovací proces počítačového hráča. Zvlášť rozobraté boli špeciálne situácie, ktoré počas hry môžu nastať a ich spôsoby riešenia. Na záver boli vytvorené jednotlivé typy hráčov na základe ich ohodnocovacích funkcií a použitých algoritmov, ktoré boli následne medzi sebou testované a usporiadané do úrovní podľa obtiažnosti. Testy ukázali, že rozhodovanie počítačového hráča je možno až príliš priamočiare. To je však spôsobené najmä novým druhom stien, a teda iným spôsobom blokovania súpera.

Výsledkom práce je hotová aplikácia, ktorá umožňuje užívateľovi hrať rozšírenú verziu hry Quoridor tak ako proti hráčom ovládaným človekom, tak aj proti počítačovým protivníkom. Z počítačových protivníkov má užívateľ možnosť zvoliť si z troch obtiažností. Najľahšia úroveň uprednostňuje pohyb vlastného panáčika a súpera blokuje len ak je to vyslovene potrebné. Stredná úroveň si priority pohybu panáčika a blokovania súpera dynamicky prepočítava počas hry na základe aktuálnych pozícií hráčov. Najvyššia úroveň používa algoritmus Max^n, teda prehľadáva stavový strom do hĺbky a dosahuje tým najlepšie výsledky. Na úkor úspešnejšieho rozhodovacieho algoritmu trvá ťah tohto hráča až niekoľko sekúnd.

V budúcnosti by sa dalo moje riešenie zlepšiť v rôznych ohľadoch. Vo výslednej aplikácii sa môže napríklad pridať voľba poradia hráčov a nastavenia obtiažnosti každému počítačovému protivníkovi zvlášť. V návrhu riešenia by si špeciálnu pozornosť zaslúžila rozsiahlejšia analýza ohodnocovacej funkcie. Použitý algoritmus Max^n by pre zlepšenie hry mohol z rovnako ohodnotených stavov vyberať výsledný stav náhodne, prípadne použiť prídavnú funkciu na výber vhodného ťahu. Mohli by sa premyslieť aj iné heuristiky ako tie uvedené v tejto práci, čo by snáď viedlo k lepším výsledkom tým, že by algoritmus mohol počítať do väčšej hĺbky.

Použitá literatúra

1. **Alden, Scott a Solko, Derk**. Quoridor | Board Game. BoardGameGeek. [Online] [Dátum: 8. 5 2015.] http://www.boardgamegeek.com/boardgame/624/quoridor.

2. **Esser, Markus**. BEST-REPLY SEARCH IN MULTI-PLAYER CHESS. Maastricht, 2012.

3. **Matoušek, Jiří a Nešetřil, Jaroslav**. Kapitoly z diskrétní matematiky. Praha : Univerzita Karlova v Praze, 2009. ISBN 978-80-246-1740-4.

4. **Mertens, P.J.C**. A Quoridor-playing Agent. 2006.

5. **Sturtevant, Nathan, Zinkevich, Martin a Bowling, Michael**. Prob-Maxn. s.l. : Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada T6G 2E8.

6. **Töpfer, Pavel**. Algoritmus minimaxu. [Online] [Dátum: 8. 5 2015.] http://ksvi.mff.cuni.cz/~topfer/.

7. **Töpfer, Pavel**. ALGORITMY a programovací techniky. Praha : PROMETHEUS, 2007. ISBN 978-80-7196-350-9.

8. **Wikipedia, The Free Encyclopedia**. Game complexity. [Online] 2. 5 2015. [Dátum: 8. 5 2015.] http://en.wikipedia.org/wiki/Game_complexity.

Prílohy

Príloha 1 – CD

Priložené CD obsahuje:

- Spustiteľný súbor „Quoridor.jar“ vo formáte JAR obsahujúci aplikáciu Quoridor pre troch hráčov
- Súbor „Implementace hry Quoridor ve variantě pro tři hráče.pdf“ obsahujúci text tejto práce
- Adresár „doc“ obsahujúci vygenerovanú dokumentáciu programu
- Adresár „src“ obsahujúci zdrojový kód programu

Príloha 2 – užívateľská dokumentácia

Program sa spustí automaticky po otvorení súboru „Quoridor.jar“. Aplikácia je naprogramovaná v programovacom jazyku Java, hlavne kvôli prenositeľnosti, a je spustiteľná na každom z troch najpoužívanejších operačných systémov: MS Windows, Unix a OSX. Na beh aplikácie nie je nutné nič inštalovať, potrebná je len Java verzia 1.8.

Po spustení programu sa otvorí úvodné okno (obr. P2.1), v ktorom má užívateľ na výber z troch možností. Po stlačení tlačítka „Quit“ sa celá aplikácia vypne. Po stlačení tlačítka „Game Rules“ sa zobrazí nové dialógové okno, v ktorom sú napísané pravidlá hry.

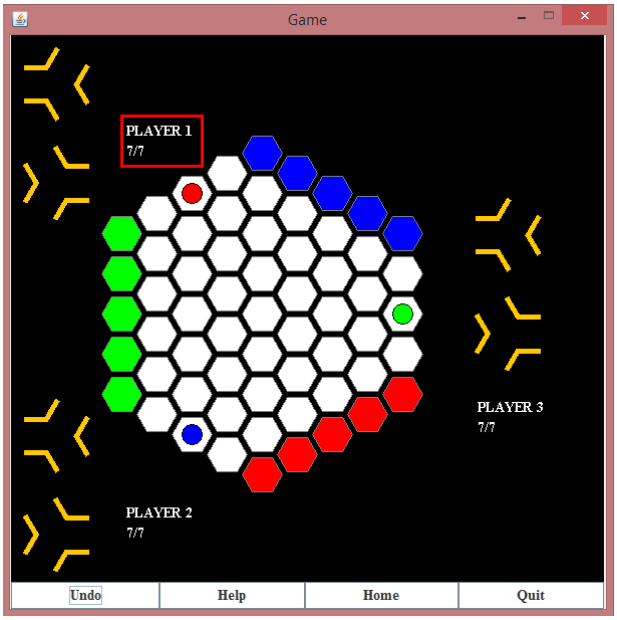
Obrázok P2.1 Úvodné okno

Ak užívateľ zvolí tlačítko „Start a new game“, súčasné okno sa zavrie a otvorí sa nové (obr. P2.2). Tu má užívateľ možnosť konfigurácie vlastnej hry. Vybrať si môže jednu z dvoch veľkostí hracej plochy, počet ľudských hráčov v hre a obtiažnosť počítačových protivníkov, kde „1“ reprezentuje najnižšiu úroveň a „3“ najvyššiu. Užívateľ nemá možnosť zvoliť poradie hráčov, ani rôzne úrovne jednotlivých počítačových protihráčov v jednej hre. Toto rozšírenie je zahrnuté v návrhu na budúci vývoj aplikácie.

Obrázok P2.2 Konfigurácia hry

Tlačítkom „Back“ sa dá vrátiť do úvodného okna a tlačítkom „OK“ užívateľ svoj výber potvrdí a spustí samotnú hru.

Hra sa otvorí v novom okne (obr. P2.3). Každý hráč má na hracej ploche svoju figúrku a vedľa plochy má k dispozícii svoje steny. Existuje šesť možností ako položiť stenu na hraciu plochu, preto má každý hráč všetkých šesť polôh zobrazených. Farebný rámik hráča indikuje, kto je práve na ťahu.

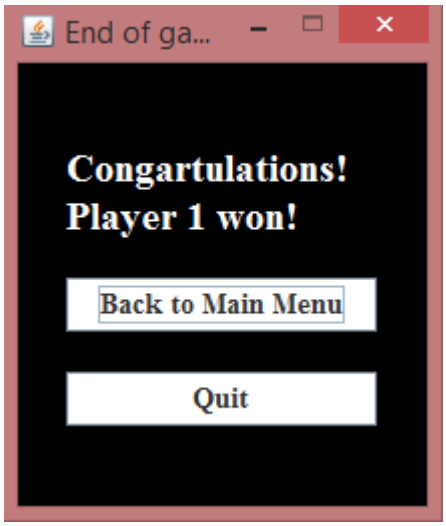


Obrázok P2.3 Herné okno

Hra sa ovláda len myšou. Hráč má počas svojho ťahu na výber, či posunie svoju figúrku alebo položí na hraciu plochu stenu. Obidve akcie sa vykonajú kliknutím na svoju figúrku (reprezentovanú farebným krúžkom) alebo svoju stenu a potiahnutím na požadovanú pozíciu.

Užívateľ má možnosť vrátiť späť svoj ťah a hrať znova tak, že klikne na tlačítko „Undo“. Vždy je možné vrátiť sa len o jeden krok späť. Tlačítko „Help“ slúži na zobrazenie pravidiel hry, tlačítkom „Home“ sa hra skončí a znova sa otvorí úvodné okno a tlačítko „Quit“ vypne celú aplikáciu.

Hra sa končí, keď jeden z hráčov vyhrá. To nastane, ak sa so svojou figúrkou dostane na ľubovoľné políčko zo steny oproti jeho počiatočnej pozície. Otvorí sa dialógové okno s informáciou, ktorý hráč vyhral (obr. P2.4).



Obrázok P2.4 Koniec hry

Po skončení hry má užívateľ možnosť vrátiť sa do hlavného menu, teda znova otvoriť úvodné okno (tlačítko „Back to Main Menu“) alebo ukončiť aplikáciu (tlačítko „Quit“).