VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Desková hra Dáma

Checkers

2014 Miroslav Ježík

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Miroslav Ježík**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Desková hra Dáma
Checkers

Zásady pro vypracování:

Cílem práce je implementovat počítačovou podobu deskové hry Dáma.

1. Popište stručně varianty hry Dáma a zaměřte se na jednu vybranou variantu.
2. Pro vybranou variantu popište pravidla hry.
3. Proveďte objektově orientovanou analýzu zvolené varianty s využitím návrhových vzorů.
4. Hru implementujte (hra člověk-počítač).

Seznam doporučené odborné literatury:

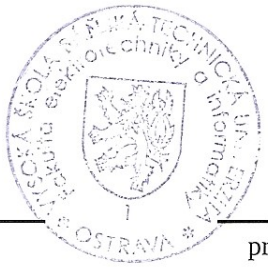
Zapletal, M. Velká kniha deskových her. Mladá fronta Praha, 1991, ISBN 80-204-0188-1

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Mgr. Jiří Dvorský, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014



doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2014 .....

Rád bych zde poděkoval svému vedoucímu doc. Mgr. Jiřímu Dvorskému, Ph.D. za všechny důležité rady a věcné připomínky, které mi pomohly při vypracování této práce.

INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Poděkování

Tato práce byla vypracována s podporou projektu Rozvoj lidských zdrojů ve výzkumu a vývoji moderních soft computingových metod a jejich praktického využití, reg. č. CZ.1.07/2.3.00/20.0072 podpořeného Operačním programem Vzdělávání pro konkurenceschopnost, financovaného ze strukturálních fondů EU a státního rozpočtu ČR.

Abstrakt

Cílem práce bylo popsat jednotlivé varianty dámy a poté si jednu variantu vybrat k implementaci a provést její objektově orientovanou analýzu. Jako výsledek vznikla aplikace s uživatelským rozhraním, kdy je možné si nastavit u jednotlivých soupeřů umělou inteligenci a také je možné dodatečně hru uložit, či načíst.

Klíčová slova: Dáma, teorie her, truel, umělá inteligence, Minimax, Negamax, popis implementace, bakalářská práce

Abstract

The main aim of this thesis was to describe single variations of board game Checkers and then to choose one of these variations for implementation and create object-oriented analysis for this implementation. As result, the application with user interface was created, where you are able to set possibility of artificial intelligence for your rivals and in addition, you've got possibility to save or load your game.

Keywords: Checkers, game theory, truel, artificial intelligence, Minimax, Negamax, implementation description, bachelor's thesis

Obsah

1 Úvod

V dnešní moderní době se ztrácí mnoho aktivit, které v minulosti byly běžné. Příkladem může být například čtení knih, které vystřídalo sledování televize, či jiné aktivity na internetu. Hraní deskových her je ale stále ještě oblíbené u velké skupiny lidí, ať už se jedná o šachy, či jiné varianty deskových her. Tato práce se konkrétně bude zabývat deskovou hrou Dáma, která je také známa pod anglickým názvem Checkers.

V České republice je nejvíce známa naše klasická verze dámy, ačkoliv existuje také mnoho jiných zahraničních verzí, které v této práci zmíním. Hlavním tématem této práce ovšem bude Dáma pro tři hráče, kterou jsem si zvolil k implementaci v programovacím jazyce C#. Mimo jiné bude práce taktéž obsahovat objektově orientovanou analýzu s využitím návrhových vzorů a bude zde popsáno celkové řešení mé implementace.

2 Dáma

Dáma je desková hra, která je obvykle hrána na čtvercové šachovnici různých rozměrů. Pohyb herních kamenů je většinou povolen diagonálně vpřed a zajímání je povoleno jejich přeskokem. Pokud se herní kámen dostane na konec herního plánu, je povýšen na dámu a získává speciální vlastnosti.

Jak bylo již ale v úvodu zmíněno, existuje na světě mnoho variant této deskové hry, kdy se jednotlivé varianty liší různými způsoby. Většinou se jedná o rozměry herního plánu, jeho tvar, pohyby herních kamenů, či styl jejich zajímání. Někdy jsou změny od tradiční dámy tak rozsáhlé, že by se dalo diskutovat o tom, zda se nejedná o jinou hru.

Následující stručný popis jednotlivých verzí dámy vychází z knihy „Velká knihy deskových her od Miloše Zapletala“ [7].

Česká dáma

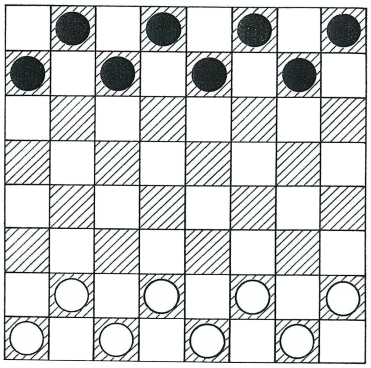
Jedná se o klasickou verzi dámy pro dva hráče, kde na šachovnici o velikosti 8×8 je umístěno 8 černých a 8 bílých herních kamenů na tmavých polích ve dvou řadách na opačných koncích šachovnice (viz obr. 1). Na tahu je jako první bílý, poté dochází k pravidelnému střídání. Pohyb herních kamenů je povolen pouze šikmo po tmavých polích buď doprava nebo doleva o jedno pole vpřed. Přeskoky kamenů jsou povinné a jsou povoleny jednoduché i několikanásobné přeskoky. Pokud se kámen dostane na opačný konec šachovnice, tak je povýšen na dámu a může se pohybovat neomezeně ve všech směrech šikmo po černých polích. Vítězem je hráč, který zajme všechny soupeřovy herní kameny.

Anglická dáma

V Anglii je tato verze dámy známa pod názvem draughts. Od české dámy se liší v počtu herních kamenů, který je zvýšen na 12. Dále jsou kameny, které dojdou na opačný konec šachovnice povyšovány na krále, který se může pohybovat všemi směry, ale pouze o jedno pole. Král také při skákání nemusí upřednostnit vícenásobný skok před jednoduchým, ale pokud se pro něj rozhodne, tak ho musí provést až do konce.

Turecká dáma

Herní plocha je klasicky 8×8, ačkoliv všechna pole jsou bílá. Ve hře je 16 bílých a 16 černých herních kamenů, které jsou nazývány keny a jsou umístěny v druhé a třetí řadě pro bílého hráče, a poté v šesté a sedmé pro černého. Pohyb je povolen ortogonálně směrem dopředu, nalevo a napravo. Když herní kámen dojde na konec herního plánu, tak je povýšen na perce. Pohyb perce platí do všech čtyř směrů a je neomezený. U přeskoků percem jsou herní kameny okamžitě odstraněny z plochy, což znamená, že pokud má perc před sebou i za sebou v jedné řadě soupeřovy kameny, může zajmout jedním tahem oba soupeřovy kameny.



Obrázek 1: Česká dáma [7]

Žravá dáma

Pohyb herních kamenů se řídí klasicky pravidly české dámy. Jediný rozdíl je v cíli hry, kdy hráč nemá za úkol zajmout všechny soupeřovy kameny, ale naopak si musí nechat zajmout všechny vlastní kameny.

Mezinárodní dáma

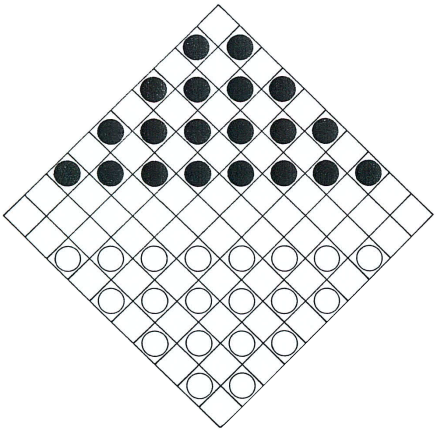
Hraje se na herní ploše 10×10. Oba hráči mají k dispozici 20 herních kamenů, které jsou rozestaveny na černých polích v prvních a posledních čtyřech řadách. Pohyb herních kamenů je stejný jako u české dámy, až na přeskoky, které lze provádět i směrem dozadu. Hráč musí provést tah herním kamenem, kterého se dotkl nejdříve. Přeskoky herních kamenů jsou povinné a vždy je nutno provést tah s herním kamenem, který je schopen udělat nejvíce přeskoků. Pokud hráč neprovede povinný skok, soupeř smí hráče vyzvat k dokončení tahu, jinak bude přeskok promlčen. Zbytek pravidel pohybu herních kamenů lze aplikovat z tradiční české dámy. Na mezinárodní dámu je také nastaveno časové omezení 50 tahů za 2 hodiny.

Rohová dáma

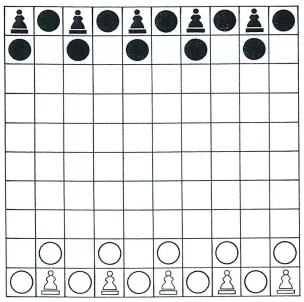
Hraje se na stejné herní ploše jako Mezinárodní dáma. Herní plocha je ovšem otočená rohy k hráčům a herní kameny jsou opět rozestaveny na bílých polích v prvních a posledních čtyřech řadách viz obrázek 2. Většinu pravidel lze opět aplikovat z mezinárodní dámy, až na některé výjimky. Kameny postupují dopředu, doleva nebo doprava. Povyšování na dámu lze provést pouze na dvou polích v protějším rohu herního plánu.

Dvacet proti deseti

Řídí se opět stejnými pravidly jako Mezinárodní dáma. Rozdíly od mezinárodní dámy jsou, že černý hráč má pouze 10 herních kamenů, zatímco bílý jich má 20. Tato nevýhoda je vyrovnána pravidlem, které umožňuje černému hráči vždy táhnout dvakrát, než je na



Obrázek 2: Rohová dáma [7]



Obrázek 3: Středověká bitva [7]

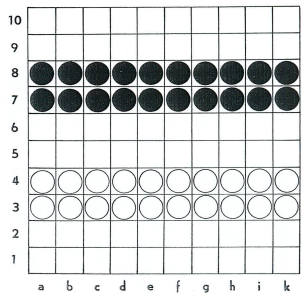
řadě bílý hráč. Vítězství bílého nastává v případě, že dokáže černého zablokovat tak, že nemůže provést dva po sobě jdoucí tahy, přičemž černý se snaží zajmout všechny herní kameny protivníka.

Středověká bitva

Herní plochou je opět čtvercová šachovnice 8×8. Oba hráči mají k dispozici 10 pěšáků a 5 rytířů, kteří jsou rozmístěni, jak je uvedeno na obrázku 3. Pěšáci se pohybují jako klasické kameny v dámě, přičemž rytíři se pohybují ortogonálně směrem dopředu, doleva a doprava. Na opačné straně šachovnice se pěšák promění ve velitele a má standardní pohybové vlastností dámy. Rytíř je povyšován na krále, který má pohybové vlastnosti dámy v šachu.

Jezdecká dáma

Dvacet bílých a černých jezdců je rozestaveno na herním plánu 8×8 v třetí, čtvrté, sedmé a osmé řadě viz obrázek 4. Jezdci mají pohybové vlastnosti jako jezdec v šachu, ale musí



Obrázek 4: Jezdecká dáma [7]

se vždy pohybovat vpřed. Na konci herního plánu se jezdec mění na dámu, která má stejné pohybové vlastnosti jako jezdec, ale smí se pohybovat všemi směry. Přeskoky herních kamenů probíhají tak, že se jezdec, či dáma dostane na pole, kde je soupeřův jezdec a odtud skáče znovu na další pozici. Až poté je soupeřův herní kámen odstraněn z hrací plochy. Vícenásobné přeskoky jsou zde také povoleny.

Dám

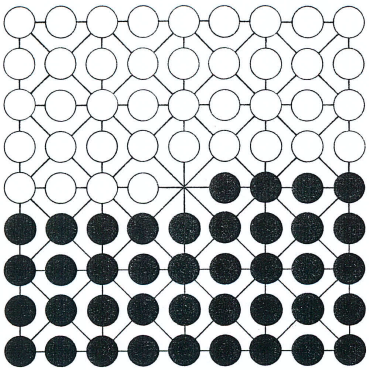
Tato verze dámy se hraje na herním plánu 12×12 a oba hráči mají k dispozici 30 herních kamenů rozestavených v prvních a posledních pěti řadách na bílých polích. Herní kameny se pohybují pouze o jedno pole úhlopříčně vpřed i vzad. Při postupu na konec herního plánu se herní kámen mění na dámu, která se pohybuje úhlopříčně bez omezení na počet polí. Přeskoky dámy probíhají klasickým způsobem až na výjimku, kdy je dáma schopna přeskočit všechny herní kameny soupeře na diagonále, pokud je poslední pole této diagonály volné.

Alquerque

Herní plán má rozměry 5×5 a je složen z šesti úhlopříčných, pěti svislých a pěti vodorovných čar. Oba hráči mají k dispozici 12 herních kamenů které se mohou pohybovat dopředu i dozadu, ale jen ve směru čar. Přeskoky fungují jako u české dámy.

Čoko

Jedná se o hru ze Srí Lanky, která má princip dámy, ale celkově probíhá ve dvou fázích. Hraje se na herním plánu 5×5, kdy každé pole herního plánu je důlek v půdě. Jeden hráč má 12 krátkých a druhý 12 dlouhých hůlek. V první fázi hry hráči postupně zapichují hůlky do prázdných polí a když se jeden z hráčů rozhodne první fázi hry ukončit, tak to oznámí svému soupeři. Poté oba hráči ještě umístí poslední hůlku do herního plánu a zbylé odloží. Pohyb hůlek je ortogonální a jsou povoleny pouze jednoduché přeskoky. Při přeskoku má hráč právo vyřadit jak přeskočenou hůlku, tak také další hůlku dalšího výběru. Hra pokračuje dokud jeden z hráčů neztratí všechny hůlky, nebo pokud oběma hráčům nezůstane pouze po jedné hůlce.



Obrázek 5: Africká dáma [7]

Africká dáma

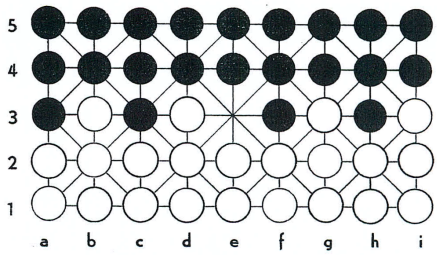
Herní plán a výchozí situace je vyznačena na obrázku 5. Jako hrací kameny se používá 40 malých světlých a 40 tmavých hůlek, a poté je k dispozici 5 velkých světlých, a také tmavých hůlek. Hůlky se opět pohybují ve směru čar a provádějí v tomto směru také přeskoky. Pokud se hůlka dostane na opačný konec herního plánu, tak je povýšena na kouzelníka a má možnost se přesouvat přes neomezený počet polí.

Osetinská dáma

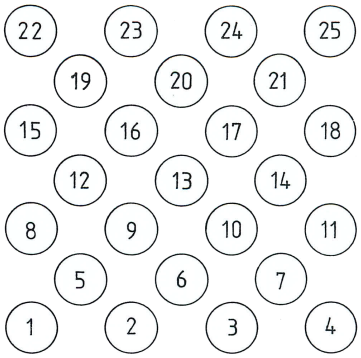
Je hrána na čtvercovém herním plánu, kde je dohromady vyznačeno 85 možných pozic pomocí vodorovných, svislých a úhlopříčných čar. Je zde také možnost využití většího plánu, který má dohromady 155 možných pozic. Oba hráči mají k dispozici buď'to 21 nebo 33 herních kamenů na malém herním poli. Pro velký herní plán se používá buď'to 27 nebo 43 herních kamenů. Herní kameny se posouvají ve směru čar pouze kupředu. Skoky jsou povoleny jak jednoduché, tak několikanásobné, přičemž skoky je možné provést i směrem dozadu. Pokud herní kámen dosáhne konce herního plánu, tak se nemění na dámu, ale má možnost čekat na přeskok směrem vzad. Hru vyhrává ten, který zajme, či totálně zablokuje soupeřovy herní kameny.

Fanorona

Herní plán a výchozí pozice herních kamenů je opět vidět na obrázku 6. Oba hráči mají na počátku hry 22 herních kamenů, které se opět pohybují ve směru čar. Zajmutí herních kamenů zde ovšem funguje naprosto jinak, než u klasické dámy. Herní kámen může zajmout soupeřův buď'to přiblížením a nebo odsunem. Při prvním způsobu může hráč zajmout soupeřův herní kámen tím, že se k němu přiblíží do těsné blízkosti ve směru, ve kterém leží. Zajaty jsou také soupeřovy herní kameny, které leží v daném směru v těsné blízkosti za vyřazeným kamenem. Při druhém způsobu se musí herní kámen odsunout z těsné blízkosti soupeřova herního kamene v opačném směru. Opět jsou také vyřazeny herní kameny, ležící na dané linii v těsné blízkosti. Pokud má hráč při vyřazení soupeřova



Obrázek 6: Fanorona [7]



Obrázek 7: Laska [7]

herního kamene možnost vyřadit další, má možnost daným kamenem táhnout znovu. Toto pravidlo není povoleno při prvním tahu hry.

Laska

Jedná se o obměnu anglické dámy, kdy je herní síť složena z 25 kruhových polí. Hráči mají na počátku hry 11 herních kamenů rozmístěných na pozicích 1 až 11 a 15 až 25 (viz obr. 7). Směr pohybu je jako u klasické dámy. Při přeskoku není herní kámen odklizen z plochy, ale je zasunut pod kámen, který ho zajal. Při zajmutí sloupce kamenů je zajat pouze horní kámen, který ho ovládá, přičemž horní kámen určuje směr pohybu. Při přesunu na konec herního plánu probíhá korunovace na krále a horní kámen je vyměněn u bílého hráče za modrý a u černého za červený. Král se pohybuje jako u anglické dámy. Pokud je král zajat, může být stále osvobozen a pohybovat se jako král.

3 Dáma pro tři hráče

Jedná se o verzi dámy, které budu věnovat největší pozornost, jelikož jsem si ji zvolil k implementaci v programovacím jazyce C#. Důvodem volby je hlavně netradičnost, ať už v počtu hráčů, tak také ve tvaru herního plánu, který je pro standardní dámu netypický. Následující přesné znění pravidel, které se skládá z pravidel české dámy a dámy pro tři hráče, jsem opět převzal z knihy „Velká kniha deskových her od Miloše Zapletala“ [7].

Pravidla

Počet hráčů: 3.

Potřeby: Herní plán má tvar rovnostranného trojúhelníku a je rozdělen na 144 trojúhelníkových polí. Z toho je 78 polí tmavých, 66 bílých. Hracích kamenů je 21 bílých, 21 černých a 21 červených.

Výchozí situace: Hráči sedí tak, aby měli před sebou jeden roh herního plánu. Kameny jsou rozestaveny na tmavých polích, v každém rohu jiná barva (viz obrázek 8).

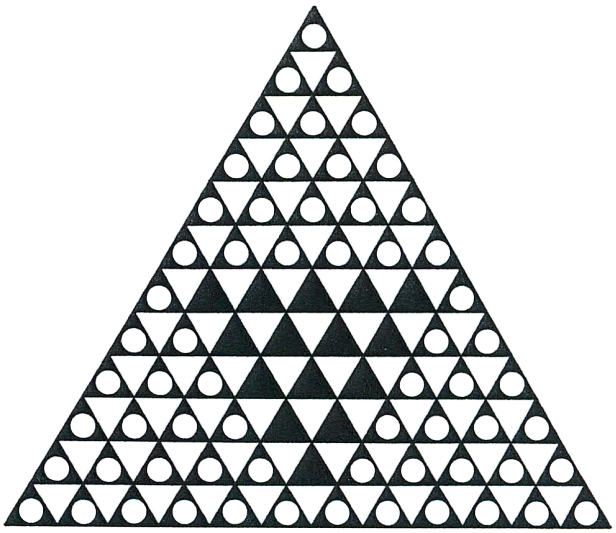
Úkol hráčů: Vyřadit co nejvíc kamenů obou soupeřů ze hry.

Tahy: Začíná bílý, druhý je na řadě černý, třetí červený. Dál se pravidelně střídají v tomto pořadí.

Pohyb kamenů: Všechny kameny postupují po tmavých polích úhlopříčně vpřed. Dovoleny jsou posuny o jedno pole nebo přes pole obsazené soupeřovým kamenem.

Přeskoky: Skákání je povinné. Kdo úmyslně nebo přehlédnutím poruší toto pravidlo a místo přeskoku jen posune některý svůj kámen, může být na toto opomenutí upozorněn a soupeř má právo mu vzít kámen, který pravidlo o povinném skákání nerespektoval. Obvykle se přitom vyslovuje formule: "Zapomněl jsi skákat!"Hráči musí provést možný vícenásobný přeskok v plném rozsahu. Jestliže ho nedokončí (například místo možných tří přeskoků udělají jen dva), může jim soupeř odebrat kámen, který se provinil proti tomuto pravidlu.

Povyšování: Kámen, který dojde do dlouhé řady polí na opačném konci herního plánu než je roh, z něhož vyšel, mění se v dámu a má její obvyklá práva.



Obrázek 8: Dáma pro tři hráče [7]

Pohyb a boj dámy: I dáma se pohybuje jen po tmavých polích, ale na rozdíl od prostých kamenů má právo postupovat všemi směry. Dáma se smí přemístit v jednom úhlopříčném směru přes jakýkoli počet volných polí a zastavit se na kterémkoli z nich. Stojí-li dámě v cestě kámen nebo dáma jiné barvy, za nimiž je aspoň jedno pole volné, musí je přeskočit. Přeskočený kámen je ihned odklizen z desky. Tento skok lze provést přes jakýkoli počet volných polí. Ani dáma však nesmí přejít přes kámen stejné barvy, ten je pro ni nepřekonatelnou překážkou. Má-li dáma možnost přeskočit víc než jeden kámen, je povinna provést skok v plném rozsahu. Mezi přeskakovanými kameny musí být vždy aspoň jedno pole volné. Dva kameny stojící za sebou ani dáma nemůže přeskočit. Zato má právo po každém dopadu změnit směr pohybu. Povinné skákání platí i pro dámu. Porušení tohoto pravidla se trestá stejně jako u obyčejných kamenů. Dáma, která skok vůbec neprovede nebo ho neuskuteční v plném rozsahu, je soupeřem odklizena z desky.

Zakončení hry: Když jeden z hráčů ztratí všechny kameny, ostatní dva pokračují v boji do konečného rozhodnutí.

4 Teorie her

Teorie her se zabývá řešením konfliktů, u kterých jsou nutná určitá rozhodnutí, s využitím matematiky, přičemž se konfliktem myslí situace, u které je nutné vybrat určitou strategii k dosažení cíle. Konfliktem nemusí být nutně hra jako taková, ale může se jednat o jakoukoliv běžnou situaci, či spor, který může nastat a je nutné ho řešit. Základními pojmy teorie her jsou hra, hráči, strategie a výplata. Hra je souhrn pravidel, do níž vstupují hráči, kdy si každý hráč zvolí určitou strategii, pomocí které hru hraje. Pomocí zvolených strategií dojde k ohodnocení hráčů a jsou jim přiřazeny výplaty [2].

Typy her dle podskupin

Hry se dále mohou dělit do jistých podskupin, dle určitých vlastností. Může se jednat o kooperativní, či nekooperativní hry, kdy to závisí na možnosti hráčů uzavírat dohody. Dále se může jednat o jednokolové, či vícekolové hry, kdy hráči volí své strategie v závislosti na počtu kol, kdy při více kolech mají hráči možnost tyto strategie měnit, či reagovat na strategie protihráčů. Symetrické a asymetrické hry se navzájem liší skutečností, zda jsou hráči od začátku ve stejném postavení, či nikoliv. U her s nulovým součtem je součet výplat hráčů roven nule, či jiné konstantě, zatímco u her s nenulovým součtem není možné finální součet výplat hráčů předem určit. Hry s úplnou informací a částečnou informací jsou rozdílné v závislosti na informovanosti hráče o možném průběhu hry, kdy pokud hráč předem nezná všechny informace o možných strategiích, či výplatách hráčů, jedná se o hru s částečnou informací. Nekonečně dlouhé hry nemají určený konečný počet kol a jsou většinou využívány pro výzkumné účely. U konečných, diskrétních a spojitých her záleží na počtu možných strategií, kdy pokud ve hře není konečný počet strategií, jedná se o spojitou hru. Pokud hráči nemají informace o průběhu hry protihráčů, jedná se o simultánní hry. V opačném případě se jedná o metahry [2].

Vězňovo dilema je konflikt, kde jsou dva vězni vyslýchání v separovaných celách. Oba mohou být usvědčeni ze spáchání menšího přestupku, ale aby byli usvědčení ze spáchání horšího trestného činu, tak je nutné, aby jeden z nich na toho druhého vypovídal. Pokud by oba mlčeli, oba budou odsouzeni na rok. Pokud jeden usvědčí toho druhého, získá svobodu, zatímco druhý vězeň bude odsouzen na 4 roky. Pokud budou na sebe oba vzájemně svědčit, budou oba odsouzeni na 3 roky. Ačkoliv by bylo pro oba lepší mlčet, stále mají oba šanci nebýt odsouzeni, pokud budou svědčit za předpokladu, že druhý vězeň bude mlčet [5].

Truel

Jedná se o speciální situaci, kdy je na rozdíl od duelu do konfliktu zapojen třetí hráč. Jako známý příklad truelu je uváděn souboj tří střelců A, B a C, kteří mají rozdílnou šanci na zásah, kdy A je nejlepší střelec a C nejhorší, přičemž pořadí, ve kterém střelci střílejí je C, B, A. Zároveň mají také střelci nekonečný počet nábojů. Při zahájení souboje je pro střelce C nejvýhodnější vystřelit do vzduchu, jelikož pokud bude na tahu B, bude se chtít

rozhodne zbavit nejsilnějšího soupeře A. Pokud střelec B mine, bude střelec A opět mířit na největší hrozbu, kterou je střelec B. V takovém případě má střelec C největší šanci zůstat naživu. Kromě tohoto případu se dá také uvažovat nad různými obměnami s různými pravidly. Pokud by se předpokládalo, že střelci vždy zasáhnou, přičemž mají pouze jednu ránu, bylo by rozhodování střelců stejné, až na skutečnost, že by zůstali dva živí střelci. Také je možné uvažovat nad případem, kdy má každý střelec opět rozdílnou šanci na zásah, ale není mu povoleno střílet do vzduchu. V takovém případě musí střelec vždy uvážit své šance na přežití, pokud vystřelí na určitého soupeře a zvolit si lepší možnost. V tomto případě bude první střelec vždy mířit na silnějšího soupeře, s kterým by poté měl složitější souboj v duelu. Pokud mine, jeho situace se nemění, ale pokud zasáhne, tak zůstane v duelu se slabším střelcem. V případě možnosti uzavírání spojenectví mezi střelci je možné uvést případ, kdy je pořadí střelců A, B, C a pravděpodobnost zásahu je $P(a) > P(b) > P(c)$, přičemž mají všichni střelci vysokou šanci na zásah (větší, než 60%). Pokud střílí nejprve A, je pro něj nejlepší vystřelit na druhého nejsilnějšího střelce, což je B. Pokud zasáhne, má C vysokou šanci na přežití. Pokud A mine, střelec B bude střílet na A, přičemž má střelec C stále ze všech střelců největší šanci na přežití, zatímco pravděpodobnost na přežití střelce B je nejmenší. V takovém případě je pro střelce A a B výhodné uzavřít spojenectví a střílet nejprve na C, čímž se šance A a B na přežití zlepší [1].

Kromě sekvenčního fixního pořadí střelců, které bylo uvedeno v tomto příkladu je v truelu možné také náhodné pořadí, kdy je po každém výstřelu náhodně zvolen jeden z přeživších střelců. Existují také případy, kdy mohou všichni střelci střílet vždy současně. Na rozdíl od sekvenčního pořadí střelců, kdy vždy zůstane alespoň jeden střelec naživu, tak u truelu, kdy všichni střílí při každém kole současně může nastat situace, kdy se všichni zastřelí do kruhu navzájem a živý nezůstane nikdo. V případě takového truelu opět závisí na různých pravidlech, kdy pokud mají střelci nekonečný počet nábojů a je možnost střílet do vzduchu, je pro všechny nejvýhodnější střílet do vzduchu neustále, jelikož tímto způsobem zůstanou naživu všichni [1].

5 Umělá inteligence

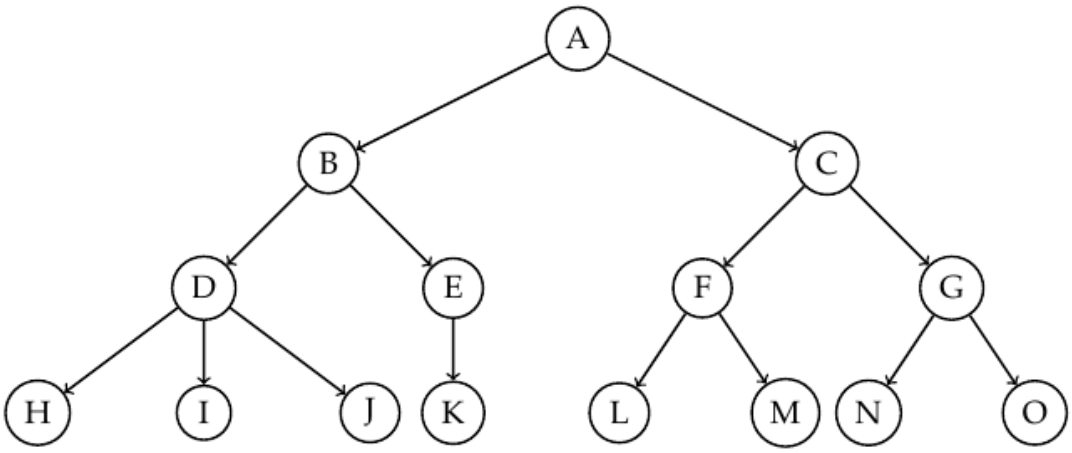
Inteligence, jako taková, je všeobecně přiřazována živým bytostem, které jsou schopny určitým způsobem reagovat na různé podněty a lze o nich tvrdit, že jsou jistým způsobem inteligentní. Ačkoliv se dá například testovat IQ člověka, tak není snadné jasně definovat pojem inteligence, jako takový. Otázkou, zda jsou stroje schopny myšlení se zabývali již filozofové v 17. století, ale jednalo se pouze o filozofické myšlenky. Velice známým testem umělé inteligence je Turingův test, který je založen na rozhovoru člověka se strojem, kdy aby byl stroj uznán, jako inteligentní, tak nesmí člověk rozeznat, zda komunikoval pouze se strojem, či s jiným člověkem přes terminál [3].

Stavový prostor

Stavový prostor je datová struktura, využívaná při implementaci umělé inteligence. Jedná se o množinu všech stavů, které mohou nastat při řešení určené úlohy, kdy se pomocí operací snažíme dostat od počátečního bodu k cílovému. Posloupnost takovýchto operací označujeme jako plán a metody, které využíváme při tvoření plánů jsou označovány jako metody řešení úloh. Stavový prostor poté můžeme zobrazit orientovaným grafem, kde uzel představuje stav a hrana mezi uzly představuje přechod mezi těmito stavy (viz obr. 9). Příkladem může být hra Lišák, u které máme čtvercové herní pole o rozměrech 3×3 a 8 postupně očíslovaných herních kamenů, které jsou ve výchozí pozici náhodně rozmístěny. Cílem hry je dosáhnout pomocí posloupnosti posunů těchto kamenů určité cílové pozice všech kamenů [3].

Minimax

Minimax je algoritmus využívaný u her s nulovým součtem, kterými jsou například piškvorky, šachy, či právě dáma, kdy algoritmus prohledává všechny možné tahy, což tvoří datovou strukturu stromu. Minimax používá dvě funkce, které se nazývají MIN a MAX. Při vykonávání algoritmu je nejprve na tahu MAX, který provede všechny možné tahy a ohodnotí je, poté zkouší tahy hráč MIN a opět se provede ohodnocení. Algoritmus se tímto dostává do určité hloubky stromu podle toho, kolik tahů dopředu má uvažovat. Pokud algoritmus dosáhne dané hloubky, provede se vyhodnocení nejlepšího tahu, kdy každý list stromu představuje možný tah. Hráč MAX si poté vždy vybírá tah s nejlepším ohodnocením, zatímco hráč MIN si vybírá nejhorší ohodnocení, které je pro hráče MAX nevýhodné. Při vybírání nejlepšího tahu algoritmus uvažuje, že hráči budou volit vždy optimální tah. Pokud tedy MAX zvolí nejlepší možnost, ale MIN poté nezvolí naprosto optimální tah, tak se výhoda tahu hráče MAX ještě více navyšuje. Jako MINIMAX se poté označuje hodnota tahu, který byl zvolen jako optimální. Při volbě hloubky počtu tahů, do které má algoritmus zajít musíme brát v potaz skutečnost čas výpočtu algoritmu, který se při každém dalším tahu exponenciálně navyšuje. Pokud je hloubka prohledávání tahů tohoto algoritmu rovna 1, tak lze potom hovořit o „naivním algoritmu“ [6].



Obrázek 9: Stavový prostor

Negamax

Negamax je varianta algoritmu podobná minimaxu, kdy se opět prohledávají tahy hráče a reakce soupeře. Rozdíl mezi těmito algoritmy je ale v jiném zápisu algoritmu, kdy je využívána jen jedna funkce NEGAMAX, která při vyhodnocení tahu vždy pouze obrátí znaménko. Při tahu hráče získá hráč kladné body a při tahu soupeře se body zase odečítají [4].

Optimalizace pomocí alfa-beta ořezávání

Jedná se o optimalizaci Minimaxu, kdy ačkoliv má exponenciální složitost, dá se částečně touto metodou optimalizovat. Jde typicky o alfa-beta ořezávání aplikované na strom všech tahů, kdy například pokud začne s tahem hráč MAX a dojde k určitému ohodnocení těchto tahů, tak při tahu hráče MIN se zjistí, že druhý tah hráče MAX má horší ohodnocení, než první, a tudíž se ho nevyplácí dále procházet [6].

6 Popis implementace

Aplikace je vyvíjena jako formulářová aplikace, fungující na .NET Frameworku 4. Projekt je rozdělen na namespace Playground, který obsahuje třídy a metody, které zajišťují vytvoření herního plánu včetně herních kamenů. Druhý hlavní namespace je Game, který obsahuje přídavné metody uživatelského rozhraní a také se stará o předávání tahů mezi hráči. Třetí namespace je Players, který se stará o vytvoření hráčů a k ním patřící logiku pohybu figurek. Posledním namespace je AI, který se stará o umělou inteligenci hráčů. Rozložení tříd je znázorněno na třídním diagramu viz obrázek 10.

6.1 Uživatelské rozhraní

Aplikace je složena z hlavního menu na horním okraji okna, stavového řádku na dolním okraji, výčtu statistik na pravé straně a hlavního plátna pro zobrazení hry. Náhled uživatelského rozhraní s vykresleným počátečním stavem hry je vidět na obrázku 11.

Menu

V hlavním menu se nacházejí volby File a Controls. Ve volbě File máme na výběr možnost New Game pro spuštění nové hry, kdy je vytvořen nový formulář, který nám umožní si zvolit, jakým hráčům přiřadit umělou inteligenci. Dále možnost Save Game pro otevření dialogového okna, díky kterému je možné aktuálně rozehranou hru uložit do XML souboru. Pomocí Load Game se opět otevře dialogové okno pro nahrání uložené hry z XML souboru. Poslední volbou v položce menu je Quit pro vypnutí aplikace. V nabídce Controls jsou možnosti Step Back a Step Forward. Step Back je nastaven na klávesovou zkratku CTRL + Z a slouží k navrácení hráče o jeden tah nazpět. Step Forward je nastaven na klávesovou zkratku CTRL + Y a umožňuje hráči posunout se o tah vpřed. Tuto volbu lze použít pouze v případě, kdy se hráč posunul alespoň o jeden tah zpět.

Status Bar

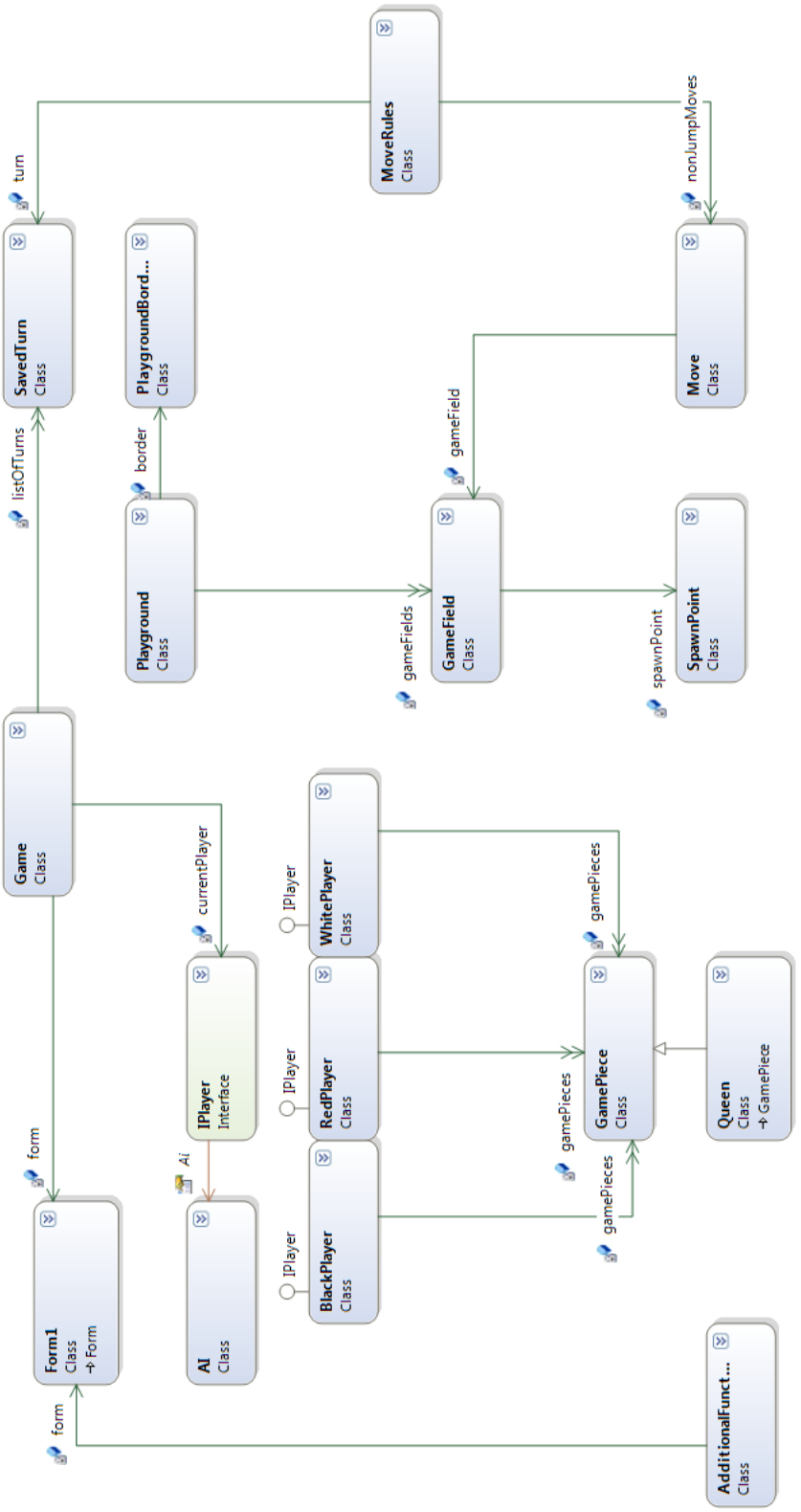
Tento stavový řádek zobrazuje aktuální stav hry (hráč na tahu, vítězství hráče).

Statistika

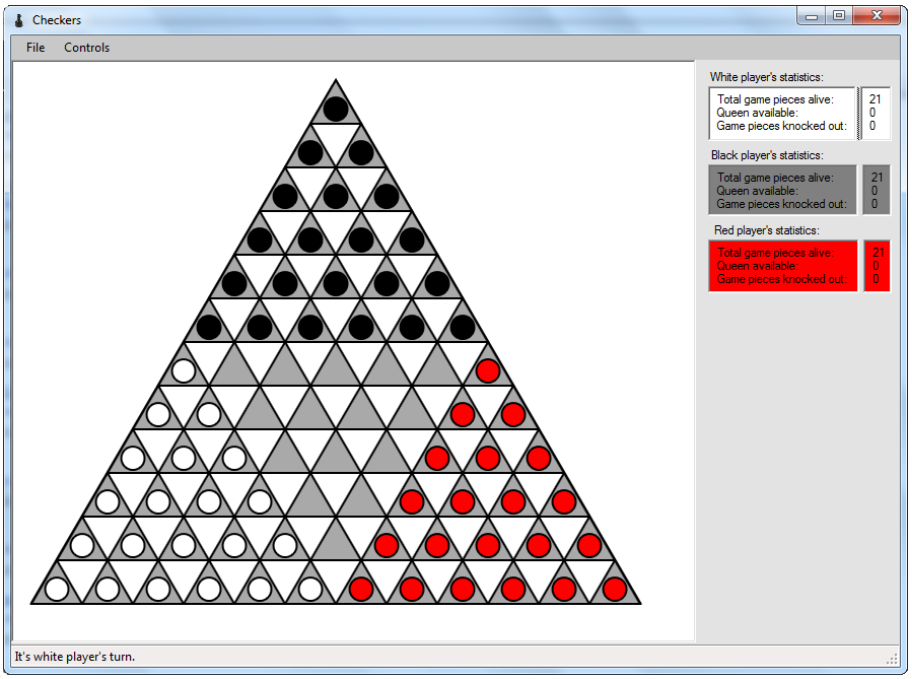
Je zobrazena na pravé straně uživatelského rozhraní a zobrazuje aktuální počty herních kamenů pro každého hráče. Statistika se vždy znovu přepočítává z počtu existujících objektů herních kamenů v polích, ve kterých jsou uloženy. Přepočítávání probíhá po každém provedeném tahu.

Canvas

Jedná se o hlavní plochu uživatelského rozhraní, kde je se zobrazuje herní plán. Jedná se o Canvas, který je převzat z WPF aplikace a je hostován v prvku ElementHost, který umožňuje využití canvasu ve formulářové aplikaci. Objekty vykreslené na této herní



Obrázek 10: Třídní diagram



Obrázek 11: Náhled aplikace

ploše reagují na událost kliknutí myši. Reakce na kliknutí myši na herní kámen je provedena pouze v případě, kdy je hráč dané barvy na tahu. Reakce na kliknutí na určitou pozici v herním plánu je uskutečněna pouze v případě, že byl hráčem již označen herní kámen, a že se jedná o validní pohyb herního kamene. Všechny validní tahy po označení herního kamene jsou vyznačeny tmavě zelenou barvou, několikanásobné přeskoky jsou vyznačeny světle zelenou. Oranžová a růžová barevná indikace je použita při rozhodování použité cesty přeskoku, pokud lze zvolit více, než jednu cestu, přičemž se cesty setkávají na stejném koncovém herním poli. Poslední barevnou indikací je světle a tmavě modrá, která indikuje tah, který provedla umělá inteligence. Překreslování tohoto prvku probíhá vždy po každém tahu hráče.

6.2 Playground

Tento namespace obsahuje třídy, které vytvářejí základní prvky grafického zobrazení herního plánu a kamenů, které jsou vykreslovány na společný Canvas.

PlaygroundBorder

Jedná se o třídu zajišťující vytvoření okraje herního plánu. Využívám zde třídu Polygon z balíku System.Windows.Shapes, po jejímž vytvoření je možné na Canvas daný objekt vykreslit.

GameField

Tato třída se stará o vytvoření jednotlivých polí v herním plánu, po kterých se poté figurky mohou pohybovat. Třída obsahuje atributy, obsahující pozici objektu na Canvasu, poté objekt třídy Polygon, který představuje samotné grafické znázornění objektu, a dále objekt třídy SpawnPoint. Kromě konstruktoru tato třída obsahuje metodu, reagující na událost kliknutí na tento objekt. Další metodou je FindIndex, která funguje jako pomocná metoda pro aplikační logiku a vyhledá index v poli, na kterém je uloženo dané pole herního plánu. Zbytek metod je určeno ke grafické indikaci možných tahů po označení herního kamene.

Queen

Třída dědí z GamePiece a obsahuje také stejné metody, jako třída GamePiece. Rozdíl mezi těmito třídami ale je v jejich grafické reprezentaci, kdy GamePiece je znázorněna instancí třídy Ellipse, ale Queen se znázorňuje pomocí dvou instancí třídy Ellipse, které se překrývají a vytváří označení dámy. Všechny metody jsou tomu tedy přizpůsobeny a jsou implementovány tak, aby se elipsy pohybovaly po herním plánu jako jeden herní kámen.

SpawnPoint

SpawnPoint je pomocná třída pro třídu GameField, která ukládá reálné souřadnice X a Y na canvasu, po kterých se poté herní kameny pohybují. Dále obsahuje atribut gamePiece pro uložení instance herního kamene, který se právě nachází na aktuálním poli na herním plánu.

GamePiece

Objekty této třídy představují základní herní kameny na šachovnici. O grafické zobrazení kamenu se stará objekt třídy Ellipse z balíku System.Windows.Shapes. Objekt si dále ukládá informace o aktuální pozici na herním plánu (pamatuje si aktuální objekt typu GameField) a barvu. Také je zde pomocný atribut pro určení, zda byl proveden povinný přeskok. MarkOnClick je metoda, která se stará o reakci na kliknutí uživatelem. Reakce se provede pouze v případě, že je hráč dané barvy na tahu. AllowTurn a DisallowTurn jsou metody, které přidělí, či odeberou danému hernímu kameni možnost tahu. Metoda Deallocate způsobí smazání objektu typu Ellipse. Metoda SetPosition přijímá parametry, které jsou typu double a způsobí změnu pozice objektu na Canvasu. DrawGamePiece vykreslí samotný objekt elipsy na Canvas. SaveGamePiece přijímá jako parametr objekt XmlTextWriteru, pomocí kterého se aktuální herní kámen uloží do XML souboru.

Playground

Tato třída se stará o kompletní vytvoření herního plánu na canvas. Je implementována jako Singleton a ukládá si jednotlivé instance typu GameField do pole. Následující me-

toda CreateGameFields se stará o vytvoření instancí pro nakreslení samostatného herního plánu. Metoda využívá metod CreateBorder, která vytvoří okraj herního plánu a CreateGameField vytváří jednotlivá herní pole.

```
public void CreateGameFields()
{
    int tmp = 0;
    CreateBorder();

    for (int i = 0; i < 12; i++)
    {
        for (int j = 0; j <= i; j++)
        {
            this.gameFields[tmp] = CreateGameField(i, j);

            tmp++;
        }
    }
}

private void CreateBorder()
{
    this.border = new PlaygroundBorder(thickness, new Point(xBorder, canvasHeight - yBorder),
        new Point(canvasWidth/2, yBorder), new Point(canvasWidth - xBorder, canvasHeight -
        yBorder), Brushes.White);
}

private GameField CreateGameField(int i, int j)
{
    return new GameField(thickness,
        new Point(((canvasWidth / 2 - gameFieldBaseSize / 2) - i * gameFieldBaseSize / 2 + j *
            gameFieldBaseSize),
            ((gameFieldHeight + yBorder) + i * gameFieldHeight)),
        new Point((canvasWidth / 2 - i * gameFieldBaseSize / 2 + j * gameFieldBaseSize),
            (yBorder + i * gameFieldHeight)),
        new Point(((canvasWidth / 2 + gameFieldBaseSize / 2) - i * gameFieldBaseSize / 2 + j *
            gameFieldBaseSize),
            ((gameFieldHeight + yBorder) + i * gameFieldHeight)), Brushes.DarkGray,
        new SpawnPoint((((canvasWidth / 2 - gameFieldBaseSize / 2) + gameFieldBaseSize / 4)
            - i * gameFieldBaseSize / 2 + j * gameFieldBaseSize), ((yBorder + gameFieldHeight *
                1.3) / 2 + i * gameFieldHeight)), 11 - i + j * 2, i);

}
```

Další metoda CreatePlayground vytvoří herní kameny pro všechny tři hráče a umístí je do výchozí pozice na herním plánu. Poté také nastaví pro bílé herní kameny možnost tahu.

```
private void CreatePlayground()
{
    CreateGameFields();
    int tmp = 0;

    for (int i = 0; i < 6; i++)
```

```
        {
            for (int j = 0; j <= i; j++)
            {
                CreateGamePieces(i, j, tmp);
                Game.GameInstance.White.MoveRules.GamePieces[tmp].AllowTurn();
                Game.GameInstance.CurrentPlayer = Game.GameInstance.White.MoveRules;
                Game.GameInstance.Black.MoveRules.GamePieces[tmp].Game_Field = gameFields[tmp];
                this.gameFields[tmp].SpawnPoint.Game_Piece = Game.GameInstance.Black.MoveRules.
                    GamePieces[tmp];
                Game.GameInstance.White.MoveRules.GamePieces[tmp].Game_Field = gameFields[21 +
                    tmp + 6 * i];
                this.gameFields[21 + tmp + 6 * i].SpawnPoint.Game_Piece = Game.GameInstance.White
                    .MoveRules.GamePieces[tmp];
                Game.GameInstance.Red.MoveRules.GamePieces[tmp].Game_Field = gameFields[27 +
                    tmp + 6 * i];
                this.gameFields[27 + tmp + 6 * i].SpawnPoint.Game_Piece = Game.GameInstance.Red.
                    MoveRules.GamePieces[tmp];
                tmp++;
            }
        }
    }
```

Jako poslední navazující metoda má název InitializePlayground. Její samotné vyvolání způsobí prvotní vytvoření a vykreslení hry do canvasu. Tato metoda je využívána při spuštění nové hry.

```
    public void InitializePlayground ()
    {
        CreatePlayground();

        Form1.Control.canvas1.Background = Brushes.White;

        Form1.Control.canvas1.Children.Add(Border.Triangle);

        for (int i = 0; i < GameFields.Length; i++)
        {
            Form1.Control.canvas1.Children.Add(GameFields[i].Triangle);
        }

        for (int i = 0; i < Game.GameInstance.Black.MoveRules.GamePieces.Length; i++)
        {
            Form1.Control.canvas1.Children.Add(Game.GameInstance.Black.MoveRules.GamePieces[i].
                Game_Piece);
            Form1.Control.canvas1.Children.Add(Game.GameInstance.White.MoveRules.GamePieces[i].
                Game_Piece);
            Form1.Control.canvas1.Children.Add(Game.GameInstance.Red.MoveRules.GamePieces[i].
                Game_Piece);
        }
    }
```

Poslední metoda této třídy se nazývá RedrawPlayground. Využívá se pro překreslení canvasu pro již existující hru. Překreslení probíhá vždy po provedení tahu.

6.3 GameControl

Tento namespace obsahuje třídy pro ovládání přídavných funkcí uživatelského rozhraní jako je uložení a načtení hry z XML souboru, a poté možnosti navrácení hráče o tah zpět nebo posunutí hráče o tah vpřed. Dále je zde také zahrnuto předávání tahů mezi hráči a vyhodnocení vítězství na konci hry.

SavedTurn

Jednotlivé instance této třídy jsou používány na ukládání provedených tahů, pro možnost navrácení tahu, či posunutí se o tah vpřed. Třída má atributy pro uložení počáteční a koncové pozice figurky, a také seznam pozicí figurek, které byly při přeskoku smazány. Všechny pozice jsou uloženy jako integer hodnoty, které odpovídají hodnotě indexu v poli, kde jsou uložena herní pole. Nachází se zde jediná metoda, která uloží daný tah do XML souboru.

```
public void SaveTurnXML(XmlTextWriter w)
{
    w.WriteStartElement("savedTurn");
    w.WriteAttributeString("startPosition", gamePieceStartPosition.ToString());
    w.WriteAttributeString("endPosition", gamePieceEndPosition.ToString());
    foreach (int a in deletedGamePieces)
    {
        w.WriteStartElement("deletedGamePiece");
        w.WriteAttributeString("position", a.ToString());
        w.WriteEndElement();
    }
    w.WriteEndElement();
}
```

AdditionalFunctions

Konstruktor této třídy přijímá jako parametr odkaz na hlavní formulář. Třída tedy obsahuje metody pracující přímo s hlavním formulářem aplikace. Hlavními metodami zde jsou StepBack, StepForward, SaveGame, LoadGame.

Metody StepBack a StepForward jsou využívány k posunutí hráče v tahu zpět, či vpřed. Metoda StepBack využívá list instancí typu SavedTurn a při tahu zpět navrátí všechny figurky do počáteční pozice, a potom rychle provede všechny tahy až do tahu, který danému tahu předcházel. Metoda StepForward funguje obráceným způsobem, a to že posune hráče o tah vpřed vzhledem k aktuálnímu tahu. Tato metoda má efekt pouze v případě, že byla původně využita metoda StepBack. Při využití metody StepBack, ať už jednou, či vícekrát v po sobě se při následném provedení tahu smažou všechny tahy, které byly navráceny a poslední tah je nahrazen aktuálně provedeným tahem.

```
public void StepBack()
{
    GameField.TurnAllDarkGray();
    if (Game.GameInstance.ListOfTurnsIndex >= 0)
```

```
        {
            Game.GameInstance.ListOfTurnsIndex--;
        }
        Playground.Playground.PlaygroundInstance.InitializePlayground();
        Game.GameInstance.CurrentPlayer = Game.GameInstance.White.MoveRules;
        for (int i = 0; i <= Game.GameInstance.ListOfTurnsIndex; i++)
        {
            Players.MoveRules.MoveRulesInstance.MakeTurn(Game.GameInstance.ListOfTurns[i]);
            Game.GameInstance.SetNextPlayer();
        }
        Game.GameInstance.AllowPlayerTurn();
        Game.GameInstance.UpdateStatusBar();
        Playground.Playground.PlaygroundInstance.RedrawPlayground();
    }
```

Metody SaveGame a LoadGame slouží k ukládání a načítání hry do XML souboru. Při volbě Save Game v uživatelském rozhraní se otevře dialogové okno pro uložení a daný vstup názvu souboru se poté pošle do parametru dané metody. Stejným způsobem funguje metoda LoadGame, která ovšem přečte XML soubor a dle něj poté nahraje hru. Pokud by byl XML soubor nějakým způsobem chybný, tak je odchycena výjimka XmlException. Poté se vyvolá chybové hlášení, že hru nebylo možné načíst a místo toho je vytvořena nová hra s výchozím postavením figurek. Do XML souboru je ukládána barva hráče, který je aktuálně na tahu, pozice jednotlivých herních kamenů, informace o typu herního kamenu a jako poslední jsou uloženy všechny provedené tahy.

```
    public void SaveGame(string fileName)
    {
        XmlTextWriter xml = new XmlTextWriter(fileName, Encoding.UTF8);
        xml.WriteStartDocument();
        xml.WriteStartElement("game");

        xml.WriteElementString("currentPlayer", Game.GameInstance.ReturnCurrentPlayer());

        Game.GameInstance.White.SavePlayer(xml);

        Game.GameInstance.Black.SavePlayer(xml);

        Game.GameInstance.Red.SavePlayer(xml);

        foreach (SavedTurn turn in Game.GameInstance.ListOfTurns)
        {
            turn.SaveTurnXML(xml);
        }

        xml.WriteEndElement();
        xml.WriteEndDocument();
        xml.Close();
    }

    public void SavePlayer(XmlTextWriter w)
    {
        w.WriteStartElement("white");
```

```
        if (this.Ai != null)
        {
            w.WriteAttributeString("AI", "1");
        }
        else
        {
            w.WriteAttributeString("AI", "0");
        }
        for (int i = 0; i < this.GamePieces.Length; i++)
        {
            if (this.GamePieces[i] != null)
            {
                this.GamePieces[i].SaveGamePiece(w);
            }
        }
        w.WriteEndElement();
    }
```

Game

Třída slouží k ovládání hry a je implementována jako Singleton. Obsahuje instance jednotlivých hráčů a seznam, ve kterém se ukládají jednotlivé tahy. Metody této třídy se starají o předávání tahů mezi hráči, update status baru, který určuje hráče, který je aktuálně na tahu a také metodu, která sleduje, zda nějaký hráč nevyhrál.

6.4 PlayersControl

Daný namespace obsahuje třídy pro vytvoření hráčů a adekvátní logiky pohybu herních kamenů pro každého hráče.

Players

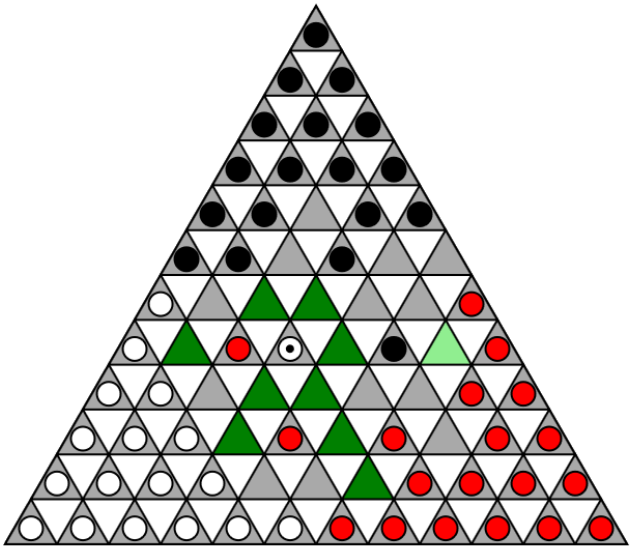
Jedná se o namespace, který obsahuje třídy pro vytvoření hráčů. S vytvořením hráče také souvisí inicializace pole pro následné uložení herních kamenů, nastavení směrových vektorů pro správný pohyb figurek a také nastavení barvy danému hráči. Třídy tohoto namespace implementují interface IPlayer.

MoveRules

Namespace obsahuje třídy, které se starají o kompletní logiku pohybu herních kamenů. Směr pohybu závisí na hráči, který je aktuálně na tahu, jelikož jednotlivé instance hráčů zde předávají své směrové vektory.

Move

Jednotlivé instance této třídy slouží jako pomocné ke správnému fungování logiky přeskoků figurek. Instance této třídy si ukládají odkaz na herní pole, kam se herní kámen má



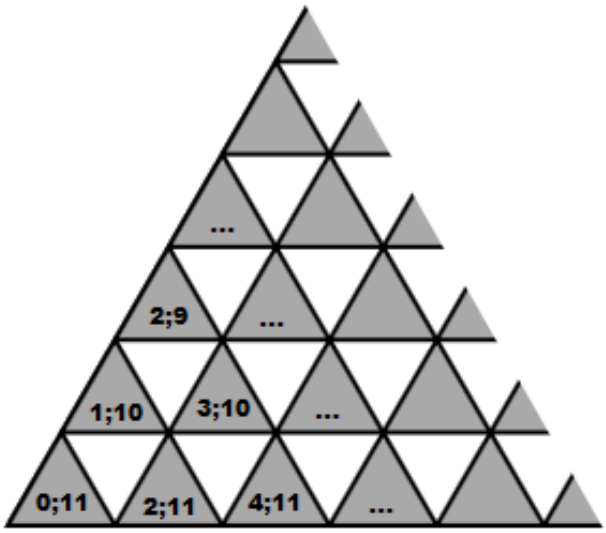
Obrázek 12: Příklad barevného značení

přesunout a také odkaz na herní kámen, který se při přeskoku má smazat. Tyto instance se ukládají do seznamů a vytvářejí cesty, které herní kámen musí projít, až do cílového herního pole, přičemž smaže herní kameny, které přeskočil.

MoveRules

Tato třída obsahuje atributy a metody, které obstarávají kompletní herní logiku pohybu herních kamenů. Obsahuje atributy pro ukládání jednotlivých cest při přeskocích, také pomocné atributy pro sledování prozkoumání těchto cest, z důvodu kolizí daných cest, kdy dvě cesty jsou rozdílné, přičemž se navzájem kříží ve stejném bodě. Také jsou zde statické atributy, které si dočasně uloží odkaz na herní kámen, který byl označen pro tah a poté odkaz na herní pole, které bylo označeno pro provedení tahu. Třída také zajišťuje barevné značení možných tahů, kdy se možné tahy vypočítají a vyznačí po kliknutí na herní kámen, u kterého je možný tah.

Barevné značení: Tmavě zelená barva označuje pole, na které je možné provést tah s aktuálně označeným herním kamenem. Světle zelená indikuje několikanásobný přeskok tím způsobem, že se všechny přeskoky označí touto barvou a závěrečné pole je značeno tmavě zelenou. Oranžové pole znamená, že nelze přesně určit, kterou cestou se má figurka do tohoto pole vydat. Odemčení tohoto pole se provádí pomocí zaznačení jedné z cest, které jsou vyznačeny růžovou barvou. Příklad značení tahu dámy je vidět na obrázku 12.



Obrázek 13: Indexace herního plánu

Herní plán: Jak již bylo výše zmíněno, tak se herní kameny mohou pohybovat pouze po tmavých polích. Všechny tmavé políčka jsou tedy uloženy v jednom poli typu GameField. Každé toto pole je tedy indexováno v dvourozměrném souřadnicovém systému, z čehož vycházejí všechny algoritmy, přičemž indexaci je možné vidět na následujícím obrázku 13.

MoveGamePiece

MoveGamePiece je hlavní metodou v této třídě která se stará o pohyb herních kamenů. Metoda nejprve provede kontrolu, zda byla před označením herního pole již označena figurka a poté, zda herní pole je označeno jako korektní tah. Metoda poté provede nastavení nové pozice herního kamene a zjistí cestu, kterou herní kámen použil k přesunu. Podle zvolené cesty dojde také k případnému smazání patřičných herních kamenů soupeřů. Pokud nebyl proveden žádný přeskok, resp. nebyl smazán žádný herní kámen, tak je metodou CheckRequiredJump provedena kontrola, zda některá z hráčových figurek neměla provést povinný skok. Je-li tomu tak, jsou poté smazány všechny herní kameny, které se provinily tomuto pravidlu. Následně se zruší barevné značení z herního plánu a předá se tah dalšímu hráči. V metodě se mimo jiné tah také ukládá.

CheckViableMove

Metoda se rozhoduje, zda označený herní kámen je dáma, či nikoliv. Poté zavolá patřičné metody pro výpočet všech možných tahů.

Výpočet pozic pro možný tah

Pro tyto výpočty se používají jednotné metody, do kterých jsou vždy dosazeny směrové vektory aktuálního hráče. Dle směrových vektorů se vypočítají pozice pro daný herní kámen. Pokud označený herní kámen má možnost pouze obyčejného přesunu bez možnosti přeskoku, jsou tyto pozice barevně označeny jako kandidáti pro přesun.

Při přeskoku jsou tyto pozice uloženy do seznamu, obsahující instance typu Move, které si pamatují cíl přeskoku a referenci na herní kámen, který má být smazán. Při zjištění přeskoku, je daná metoda opakovaně volána, pro vyšetření možnosti několikanásobného přeskoku.

Po vyšetření všech možností přesunu herního kamene je dále provedena kontrola, která zjišťuje, zda nedochází ke křížení cest. Křížení cest znamená, že se herní kámen může vydat dvěma cestami, které vedou přes navzájem odlišné pozice, ačkoliv končí na stejném herním poli. V takovém případě by se algoritmus nemohl sám rozhodnout, kterou cestou se má herní kámen vydat pouze označením cílové pozice. V takovém případě dojde k zablokování cílového pole a hráč je nucen označit celou cestu, kterou se herní kámen vydá, než dojde k odemčení cílové pozice. Logiku řešení této situace řeší metody CheckPathBlock, MarkPath, CheckMarkedPath, BlockOtherPaths a UnblockOtherPaths.

Výpočet tahů pro pěšáka probíhá způsobem, že se sleduje, zda je následující pole obsazené. Pokud není, je zde možný přesun. Zjistí-li se, že je dané pole obsazené, je provedena kontrola, jestli se jedná o cizí herní kámen. Při potvrzení se jako poslední zkontroluje, zdali je následující pole v daném směru volné. Při detekci možného přeskoku algoritmus následně kontroluje, zda není přeskok možný provést několikanásobně. Tuto funkčnost vytvářejí metody CheckViableMove, CheckJumpOverGamePiece a CheckMultipleJump. Příkladem uvedu metodu CheckViableMove a část metody CheckJumpOverGamePiece, která zjišťuje přeskok pro jeden směrový vektor.

```
private void CheckViableMove(GamePiece gamePiece)
{
    ResetValues();
    CheckJumpOverGamePiece(gamePiece);
    foreach (GameField gameField in Playground.Playground.PlaygroundInstance.GameFields)
    {
        if ((gamePiece.Game_Field.Ypos + Game.GameInstance.CurrentPlayer.MoveDirection[0]
            == gameField.Ypos) && (gamePiece.Game_Field.Xpos + Game.GameInstance.
            CurrentPlayer.MoveDirection[1] == gameField.Xpos) && (gameField.SpawnPoint.
            Game_Piece == null))
        {
            nonJumpMoves.Add(new Move(gameField, null));
        }

        if ((gamePiece.Game_Field.Ypos + Game.GameInstance.CurrentPlayer.MoveDirection[2]
            == gameField.Ypos) && (gamePiece.Game_Field.Xpos + Game.GameInstance.
            CurrentPlayer.MoveDirection[3] == gameField.Xpos) && (gameField.SpawnPoint.
            Game_Piece == null))
        {
            nonJumpMoves.Add(new Move(gameField, null));
        }
    }
```

```
}

private void CheckJumpOverGamePiece(GamePiece gamePiece)
{
    foreach (GameField gameField in Playground.Playground.PlaygroundInstance.GameFields)
    {
        if ((gamePiece.Game_Field.Ypos + Game.GameInstance.CurrentPlayer.MoveDirection[0]
            == gameField.Ypos) && (gamePiece.Game_Field.Xpos + Game.GameInstance.
            CurrentPlayer.MoveDirection[1] == gameField.Xpos) && (gameField.SpawnPoint.
            Game_Piece != null))
        {
            if (gameField.SpawnPoint.Game_Piece.Color != Game.GameInstance.CurrentPlayer.
                Color)
            {
                foreach (GameField gameField2 in Playground.Playground.PlaygroundInstance.
                    GameFields)
                {
                    if ((gamePiece.Game_Field.Ypos + Game.GameInstance.CurrentPlayer.
                        MoveDirection[0] * 2 == gameField2.Ypos) && (gamePiece.Game_Field.Xpos +
                        Game.GameInstance.CurrentPlayer.MoveDirection[1] * 2 == gameField2.Xpos)
                        && (gameField2.SpawnPoint.Game_Piece == null))
                    {
                        if (Game.GameInstance.CheckMode)
                        {
                            gamePiece.ViableJump = true;
                        }

                        else
                        {

                            gameFieldMarked = gameField2;
                            jumpPaths.Add(new List<Move>());
                            jumpPaths.Last().Add(new Move(gameField2, gameField.SpawnPoint.
                                Game_Piece));
                            numberOfPaths++;
                            do
                            {
                                multipleJump = false;
                                CheckMultipleJump();
                            } while (multipleJump);
                        }
                    }
                }
            }
        }
    }
}
```

Dáma se v této variantě pohybuje neomezeně šesti směry. Dojde tedy k zaznamenání těchto cest až po okraje herního plánu do jednotlivých seznamů. Seznamy se poté prohledávají a dochází k výpočtům podobně jako u pěšáka. Tuto funkčnost zajišťují metody QueenCheckViableMove, QueenCheckPaths, QueenCheckMultipleJump.

Ostatní metody

FindPathAndDelete je metoda, která provede vyhledání správné cesty ze seznamu cest a následné smazání, což je hlavně důležité pokud dochází ke křížení cest. Pro samotné smazání herních kamenů využívá metody DeleteGamePiece a FindAndDelete, kdy se nejprve rozhodne barva smazaného herního kamene, který se následně vyhledává v patřičném poli.

Metoda CheckPromotionForQueen se volá po každém provedeném tahu a kontroluje, zda se některá z figurek nedostala na konec herního plánu a je tedy ji nutné povýšit na dámu.

Další přídavnou metodou je ColorViableMoves, která zajišťuje barevnou indikaci tahů. Metoda je vždy volána po výpočtu všech možných cest označeného herního kamene.

```
public void ColorViableMoves()
{
    foreach (List<Move> path in jumpPaths)
    {
        foreach (Move move in path)
        {
            if  (move == path.Last())
            {
                move.Game_Field.TurnGreen();
            }
            else
            {
                move.Game_Field.TurnLightGreen();
            }
        }
    }

    foreach (Move move in nonJumpMoves)
    {
        move.Game_Field.TurnGreen();
    }

    CheckPathBlock();
}
```

6.5 AI

Tento namespace obsahuje logiku, která řeší výpočty a následné provedení tahů umělé inteligence. Veškerá tato funkcionalita je uložena ve stejnojmenné třídě AI. Jelikož jsou v této variantě dámy tři hráči, nelze jednoduše využít algoritmy, které jsou při řešení umělé inteligence u těchto deskových her běžné (Minimax, Negamax, ...).

Pokud má některý z hráčů využívat umělou inteligenci, musí mu být přiřazena instance této třídy, kdy se do parametrů v konstruktoru posílají odkazy na instanci aktuálního hráče, a poté na instance soupeřů. Pokud hráči nebyla přiřazena instance AI, automaticky se čeká na akci uživatele, resp. na provedení tahu.

Samotná třída obsahuje atributy pro uchovávání aktuálně kontrolovaného tahu, nejlepšího tahu, a také obsahuje atributy které určují bodové ohodnocení jednotlivých tahů, přičemž je vždy pouze ukládána množina tahů, které mají nejlepší bodové ohodnocení. Dané hodnoty bodů jsou přiřazovány jednotlivými metodami, které provádějí vzájemné ohodnocování pro jednotlivé hráče. Hodnoty, které jsou vidět v následující tabulce jsem zvolil na základě vlastního uvážení, kdy také během testování došlo k následným optimalizacím.

Akce	Body
Přeskok pěšáka	5
Přeskok dámy	15
Přeskok cizího pěšáka soupeřem	2
Přeskok cizí dámy soupeřem	8
Povýšení na dámu	11
Tah	1

Tabulka 1: Bodové ohodnocení tahů

Jak je v tabulce vidět, musí být v tomto algoritmu odlišováno, zda se jedná o figurku aktuálního hráče, či o figurku dalšího soupeře. Je tedy zřejmé, že hráč získává bodové hodnocení za přeskok určitých typů herních kamenů soupeře. Když se ovšem předpovídají tahy soupeřů a soupeř provede přeskok, tak se musí zjistit, zda šlo o herní kámen hráče, či zda byl proveden přeskok soupeřovy figurky. V případě přeskoku figurky druhého soupeře opět dochází ke zvyšování bodového hodnocení, namísto snížení, jelikož došlo ke kladnému jevu pro aktuálního hráče, kdy jeho soupeř přeskočil herní kámen druhého soupeře. Tento parametr je využíván hlavně v případě, kdy je například jisté, že hráč o určitý herní kámen přijde, ale stále ještě může provést takový tah daným herním kamenem, že se postaví do pozice několikanásobného přeskoku, který vyřadí i cizí herní kámen. Posledním zohledňovaným parametrem je možnost povýšení na dámu, kdy pokud má herní kámen v dalším kole být povýšen na dámu, získá tento jev bodové hodnocení 11 bodů. Hodnotící funkce je dále rozdělena do tří funkcí, kdy každá funkce ohodnotí jednoho hráče. Rozdělení jsem provedl hlavně z důvodu přehlednosti, jelikož při ohodnocování jednotlivých hráčů se musíme dívat na různé parametry a také musíme brát u každého hráče vždy v potaz i druhého soupeře. Aktuální hráč tedy sleduje pouze, zda nepřeskočil nějaký herní kámen a dle typu herního kamene si přidá body, dále sleduje, zda by tahem nedošlo k povýšení na dámu a dále musí sledovat povinné přeskoky. U ohodnocení tahu prvního soupeře musíme sledovat, čí herní kámen přeskočil, jelikož přeskok kamene druhého soupeře je kladný jev pro aktuálního hráče. Při hodnocení druhého soupeře se opět sleduje, jakou barvu má herní kámen, který byl vyřazen. Pokud byl vyřazen herní kámen prvního soupeře, tak je také nutné snížit jeho osobní hodnocení daného tahu, jelikož musíme předpokládat, že první soupeř nebude úmyslně dělat tahy, které by jeho samotného poškodily. Kód hodnotící funkce pro aktuálního hráče je vidět v následující ukázce.

```
private void EvaluatePlayerTurn(List<Move> path, ref int value, GamePiece gamePiece)
{
    if (path.First().MarkedForDelete == null)
    {
        value += EvaluateRequiredJump();
    }
    else
    {
        foreach (Move move in path)
        {
            if (move.MarkedForDelete is Queen)
            {
                value += queenValue;
            }
            else
            {
                value += gamePieceValue;
            }
        }
    }
    if (IsPromoted(path.Last().Game_Field) && !(gamePiece is Queen))
    {
        value += possibleQueenPromotion;
    }
}
```

Nejlepší tahy se uchovávají jako instance struktury BestTurn, která obsahuje pouze referenci na herní kámen a také na tah, který by měl provést.

Hlavní metodou, která provádí tah umělé inteligence je MakeTurnAI. Tato metoda nejprve provede výpočet všech nejlepších tahů a poté si jeden vybere k provedení. Metoda je volána vždy, pokud je po ukončení hráče zjištěno, že následující hráč má přiřazenu umělou inteligenci.

Samotný výpočet všech nejlepších možných tahů je prováděn metodami CalculateTurn, CalculateFirstEnemyTurn a CalculateSecondEnemyTurn. Metoda CalculateTurn je hlavní metoda pro ohodnocování tahů, přičemž je nejprve ohodnocen možný tah daného hráče. Pokud je hodnota tahu větší nebo rovna hodnotě nejlepšího tahu, tak jsou pomocí metod CalculateFirstEnemyTurn a CalculateSecondEnemyTurn vypočítány nejlepší možné tahy soupeřů. Při ohodnocení soupeřových tahů jsou také využívány metody MakeTurn a RevertTurn, které provádějí simulace tahů.

7 Závěr

Cílem této bakalářské práce bylo nejprve popsat jednotlivé varianty deskové hry dámy a poté si vybrat jednu z těchto variant k implementaci. Zvolil jsem si variantu dámy pro tři hráče, kterou jsem implementoval v programovacím jazyce C#, kdy jsem pro aplikaci vytvořil uživatelské rozhraní s průběžnými statistikami všech hráčů. Při vytváření nové hry je možnost hrát bud'to proti lidským soupeřům nebo je možnost nastavit zvoleným hráčům umělou inteligenci, která byla implementována formou Negamaxu, který dokázal vypočítat tahy pouze jeden tah dopředu. Hlavním důvodem k tomu byla skutečnost, že Minimax i Negamax nejsou přímo vytvářeny pro hry, kde účinkují více, než dva hráči, což vedlo k nutnosti přizpůsobit k tomu daný algoritmus. Jako příklad by se dala uvést skutečnost, že při klasickém Negamaxu, by se od hodnocení tahu hráče poté odečetlo hodnocení tahu soupeře, z čehož vyplyne výsledné celkové hodnocení tahu. V této variantě dámy ale musíme brát v potaz skutečnost, že hráči netvoří koalice, takže po ohodnocení tahu aktuálního hráče nelze následně sečíst hodnoty soupeřů. Musíme se tedy dívat na skutečnost, že pokud se snažíme předpovědět ideální tah obou soupeřů, tak musíme brát v úvahu fakt, že všichni hráči stojí proti sobě. Velkým problémem také je, že ačkoliv by bílý hráč vypočítal ideální tah, tak pokud černý provede neočekávaně špatný tah, může tím nejen zvýhodnit červeného hráče, ale zároveň také poškodí prvního, pro kterého se tímto jeho původně „ideální“ tah může změnit v průměrný. Rozehranou hru je poté možné uložit a následně načíst z XML souboru. Dále byla v aplikaci implementována možnost procházení jednotlivých tahů, respektive možnost vrácení tahu, či zopakování již provedeného tahu. Součástí hry je také barevná indikace možných tahů, kdy je také ošetřena výjimka kdy dochází ke křížení dvou různých cest. V poslední řadě byla kompletně popsána implementace dané aplikace v kombinaci s ukázkami částí zdrojových kódů.

8 Reference

[1] Kilgour, D. Marc, Brams, Steven J. *Mathematics Magazine Vol. 70, No. 5,* Mathematical Association of America, 1997.

[2] Lachout, Petr. *Pokročílá teorie her ve světě kolem nás,* Praha: Grada Publishing, a.s., 2013.

[3] Lažanský, Jiří, Mařík, Vladimír, Štěpánková, Olga. *Umělá inteligence (1),* Praha: Academia, 2000. ISBN: 80-200-0496-3.

[4] Millington, Ian, Funge, John. *Artificial Intelligence for Games, Second Edition,* CRC Press, 2009. ISBN 0123747317.

[5] Osborne, Martin J. *An Introduction to Game Theory,* Oxford University Press, 2009.

[6] Russell, Stuart J., Norvig, Peter. *Artificial Intelligence. A modern Approach, Third Edition,* Prentice Hall, 1995. ISBN 0-13-103805-2.

[7] Zapletal, Miloš. *Velká kniha deskových her,* Brno: Mladá fronta, 1991.