VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

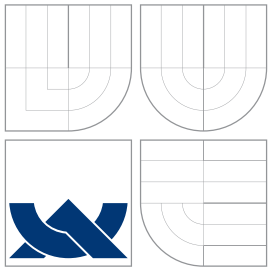
VÝUKOVÝ PROGRAM PRO ŠACHY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

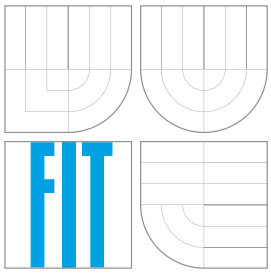
AUTOR PRÁCE
AUTHOR

KAMIL FLORYÁN

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

VÝUKOVÝ PROGRAM PRO ŠACHY

EDUCATIONAL CHESS PROGRAM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

KAMIL FLORYÁN

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. JAROSLAV ROZMAN

BRNO 2010

Abstrakt

Práce se zabývá tvorbou umělé inteligence a uživatelského rozhraní pro šachový program. V textu je také popsán návrh a implementace výukové části pro tuto aplikaci. Jsou zde analyzovány šachové motory a současné šachové programy zaměřující se na výuku.

Abstract

This bachelor thesis deals problems with an artificial intelligence and a user interface of a chess program. There is also a description of design and implementation of a tutorial part for this program. Chess engines and current chess programs focused on teaching are being analysed, too.

Klíčová slova

Šachy, Umělá inteligence, Minimax, C#, XML, Šachový motor, GUI.

Keywords

Chess, Artificial intelligence, Minimax, C#, XML, Chess engine, GUI.

Citace

Kamil Floryán: Výukový program pro šachy, bakalářská práce, Brno, FIT VUT v Brně, 2010

Výukový program pro šachy

Prohlášení

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

.....
Kamil Floryán
18. května 2010

Poděkování

Rád bych poděkoval svému vedoucímu Ing. Jaroslavu Rozmanovi za jeho vstřícný přístup, ochotu a věcné připomínky.

Obsah

1 Úvod **3**

2 Základní šachová pravidla **4**
2.1 Šachové figury 4
2.2 Důležité pojmy 5
2.3 Šachová notace 5

3 Výuka šachu **8**

4 Přehled šachových programů **10**
4.1 Vývoj šachu na počítači 10
4.2 Šachové motory 11
4.3 Současné vůdčí šachové programy 12
4.4 Ohodnocení šachového motoru 13
4.5 Turnaje šachových motorů 15
4.6 Výukové programy 17

5 Algoritmy určené k výběru tahu **19**
5.1 Algoritmus minimax 19
5.2 Algoritmus alfabeta prořezávání 20
5.3 Třídění tahů 21
5.4 Předpočítané tabulky 21
5.5 Zabijácká a historická heuristika 23
5.6 Nultahová heuristika 23
5.7 Vyhledání klidu 24
5.8 Metoda okna 24
5.9 Správa času 24

6 Návrh a implementace šachového motoru **25**
6.1 Šachovnice 25
6.2 Generátor tahů 26
6.3 Výběr nejlepšího tahu 27
6.4 Ohodnocovací funkce 27
6.5 Databáze zahájení a koncovek 28

7 Návrh a implementace výukové části **30**
7.1 Interaktivní výuka 30
7.2 Výuka tahů 31

7.2.1 Pseudo-náhodné vygenerovaní šachovnice . . . 31
7.2.2 Nalezení cesty figury . . . 31
7.3 Výuka zahájení . . . 32
7.3.1 Přidávání zahájení . . . 32
7.4 Matování a koncovky . . . 33
7.4.1 Aplikace pravidel pro koncovky . . . 33
7.5 Rádce v partii . . . 36
7.6 Databázový modul . . . 36
7.6.1 Databáze XML . . . 36
7.6.2 DOM parser . . . 37
7.6.3 Návrh XML dokumentu partie . . . 37
7.6.4 Přehrávání a uložení partie . . . 38
7.6.5 Návrh XML dokumentu databáze zahájení . . . 38

8 Návrh grafického rozhraní **39**
8.1 Šachovnice . . . 39
8.2 Report . . . 40

9 Návrh a implementace online šachové nápovědy **41**
9.1 Použité technologie . . . 41
9.1.1 Ajax . . . 41
9.1.2 jQuery . . . 42
9.2 Implementace . . . 42
9.2.1 Navigace . . . 42
9.2.2 Přehrávání partie . . . 42

10 Závěr **44**

A Seznam zahájení **49**

B Diagramy pro vyhodnocení ELA **50**

C Metriky kódu **52**

Kapitola 1

Úvod

Šachy jsou tradiční deskovou logickou hrou, řadící se mezi nejpopulárnější duševně zaměřené sporty. Kouzlo této královské hry spočívá v rozsáhlých možnostech uplatnění strategie a logického myšlení, což souvisí s rozvojem hráčova analytického myšlení a uvažování. Hraní šachu není ovlivňováno prvkem náhody, partie rozhodují pouze schopnosti a znalosti hráčů. Partie odehrané mezi lidmi se poněkud liší od partií, ve kterých se vyskytuje počítačový program. Program obsahuje umělou inteligenci založenou na hrubé síle. Generuje tedy většinu možných tahů, na které poté aplikuje hodnotící funkci.

Šachy se dají hrát na mnoha úrovních, od rekreačních partií pro zábavu či zkrácení dlouhé chvíle, až po mistrovskou úroveň v podobě soutěží či turnajů. Každá partie je však jiná, ovlivněna promyšlením každého z odehraných tahů a celkově zvolenou taktikou, kterou se hráč v průběhu partie ubírá.

Každý hráč, ať je jeho úroveň jakákoli, musel nějak začínat. Pochopit smysl hry, naučit se pohybovat figurami po šachovnici, vstřebat mnohá pravidla, které šachy obsahují. Oproti jiným logickým hrám, šachy nejsou hrou, kterou by si člověk dokázal osvojit v pár minutách. Postupně musí být zvládnuty základní dílčí kroky, na jejichž konci bude hráč schopen ovládat základní šachové konstrukce, ale tím to v podstatě všechno teprve začíná. Dále následuje vybudovávání si strategie, osvojení si analyzování pozice a předvídání soupeřových tahů. Právě počet tahů, jenž je hráč schopen předvídat, je úměrný hráčovým kvalitám a zkušenostem. Obdobně tomu je i v algoritmizaci umělé inteligence, kde je rovněž míra predikce, jedním ze základních aspektů při posuzování výkonnostní úrovně programu.

Tato bakalářské práce se zabývá návrhem a implementací šachového programu, který v sobě zahrnuje výukové prostředky. V druhé a třetí kapitole této technické zprávy je pojednáváno o pravidlech šachu a základní edukační osnově. Navržená výuková část programu se zabývá lekcemi: výuka tahů, výuka zahájení a výuka matujících koncovek. V kapitole čtyři jsou představeny existující šachové programy. V této kapitole je také popsán rozdíl mezi komplexními šachovými programy a programy specializujícími se výhradně na realizaci umělé inteligence. V závěru kapitoly jsou zmíněny programy, které obsahují výukové prostředky. Kapitola pět pojednává o typických algoritmech a metodách, které se uplatňují v šachových programech. Od šesté kapitoly se práce zabývá návrhem a implementací vlastního programu. Konkrétně kapitola šest pojednává o návrhu a implementaci inteligence programu. Kapitola sedm se dále věnuje návrhu a implementaci výukové části programu. V kapitole osm je popsán návrh a implementace grafického rozhraní a poslední kapitolou zabývající se tvorbou programu je kapitole devět, která se zabývá online webovou nápovědou. Předposlední kapitola deset popisuje ovládání a navigace v programu a zmíněno je také testování.

Kapitola 2

Základní šachová pravidla

Šachy jsou klasickou deskovou hrou pro dva hráče. Hrací deska je označována jako šachovnice, což je čtvercová deska rozdělena na 8×8 polí, střídavě tmavých a světlých (typicky černá a bílá pole). Šachovnice je při hře položena tak, aby oba hráči měli po své pravé ruce bílé rohové pole.

Jednotlivá pole šachovnice i soupeřící figury jsou od sebe odlišeny svou barvou. Označení hráčů v partii se řídí barvou jejich figur, tzn. bílý a černý. Každé pole šachovnice je jednoznačně identifikovatelné dvojicí písmene a číslice. Písmena A – H identifikují jednotlivé sloupce, číslice 1 – 8 udávají jednotlivé řádky šachovnice. Hráč hrající s bílými figurami rozmístní své figury na řady 1 a 2, naopak hráč hrající s tmavými figurami rozmístní figury na řady 7 a 8. Tato identifikace polí šachovnice je základem šachové notace.

2.1 Šachové figury

Každý z hráčů má na počátku partie šestnáct figur:

- *Král* – jedná se o nejdůležitější figuru, a i proto je její chod rozvážný, pohybuje se všemi směry(vodorovně, svisle i diagonálně), ale rozsah pohybu je pouze jedno pole. Král se vyskytuje v každé barvě na šachovnici pouze jedenkrát a nelze ho nikdy sebrat.
- *Dáma* (ve starší literatuře téže královna) – se podobně jako král pohybuje po šachovnici všemi směry, rozsah pohybu ovšem není nikterak limitován. Jedná se o nejsilnější figuru na šachovnici a každý hráč má tuto figuru pouze jedenkrát.
- *Věž* – se pohybuje po sloupcích a po řadách bez jakéhokoli omezení. Věže leží v základním rozestavění v rozích šachovnice a každý hráč má tedy tuto figuru dvakrát.
- *Střelec* – se pohybuje pouze po diagonálách šachovnice, taktéž bez jakéhokoli omezení v rozsahu pohybu. Každý hráč má dva střelce, jednoho černo polného(tzn. střelec se smí pohybovat pouze po černých polích a nikdy se nedostane na pole bílé) a druhého bílo polného.
- *Jezdec* – se pohybuje do tvaru písmene L, tzn. jezdec mine jedno pole sousední a pak uhne na další pole opačné barvy. Zvláštností této figury je, že jako jediná se smí pohybovat přes obsazená pole šachovnice. Stejně jako u věže a střelce je tato figura na šachovnici zastoupena dvakrát.

- *Pěšec* – se smí pohybovat pouze směrem vpřed. Typicky se pohybuje o jedno pole šachovnice ve směru sloupce (neplatí u vyhození soupeřovi figury), výjimkou je, zdali stojí pěšec v základním uspořádání šachovnice, kdy se může přenést vpřed o jedno či dvě pole. S postupem o dvě pole vpřed souvisí další pravidlo, a to že dostane-li se tímto tahem do sousedství z boku k protivníkově pěšci, může ho tento nepřátelský pěšec vzít mimochodem, tj. jako by byl pěšec ze základního postavení tažen o jedno pole. Pěšec je jediná figura, u něhož způsob chodu není totožný se způsobem braní. Pěšec táhne přímo vpřed, avšak bere šikmo vpřed, také jen o jedno pole. Dostane-li se pěšec na poslední řadu, tj. bílý pěšec na osmou, černý pěšec na první, čeká ho „odměna“. Musí se hned proměnit v jakoukoli jinou figuru téže barvy, kromě pěšce a krále.

Braní soupeřových figur se provádí na poli, kde daná figura leží. Mimo jezdce nesmí žádná jiná figura provádět tah přes pole, které je obsazeno jinou figurou.

V této kapitole bylo čerpáno z [49].

2.2 Důležité pojmy

Šach – jak jsem již výše zmínil, král má tu zvláštnost, že nemůže být nikdy vyhozen. V případě, že soupeř začne přímo ohrožovat krále, říkáme, že mu dává šach. Naopak král nesmí vstoupit na pole, které je ohroženo soupeřovou figurou. Jinými slovy, nesmí vstoupit do šachu. Šachu se musí ihned čelit, král nesmí zůstat v šachu ani po dobu jediného tahu.
Mat – úzce souvisí se šachem. Je ve své podstatě jeho speciálním případem, kdy se šachu už nelze nijak bránit.
Pat – je situace, v níž hráč je na tahu, nestojí v šachu a nemůže už táhnout žádnou ze svých figur.
Braní mimochodem – postoupí-li pěšec stojící v základním postavení o dvě pole a dostane-li se tím do bezprostředního sousedství, tedy ze strany, k nepřátelskému pěšci, může ho tento pěšec vzít mimochodem, tj. tak, jako by byl pěšec ze základního postavení táhl jen o jedno pole.
Rocháda – je speciálním tahem krále, ve kterém se přemístí dvě figury v rámci jednoho tahu. Známe dva typy rochád – malá a velká. Rozdíl mezi nimi souvisí zdali se rocháda provede s věží stojící na *královském křídle* (část šachovnice vymezena sloupci E – H) či *dámském křídle* (sloupce A – D). Provedení rochády je svázáno s několika předpoklady:

- Král nestojí v šachu.
- Pole přes které se rocháda provádí jsou prázdná a ani nejsou napadena žádnou ze soupeřových figur.
- Králem ani danou věží nebylo doposud v rámci partie pohnuto.

V této kapitole bylo čerpáno z knihy [49].

2.3 Šachová notace

Snad všichni známe citát „Neučte se z vlastních chyb, učte se z chyb druhých“ a u šachistů tomu není nijak jinak. Abychom se ovšem mohli poučit z již odehrané partie musí existovat prostředek, kterým lze provést záznam tahů partie. K tomu slouží šachová notace.

Podle způsobu záznamu partie můžeme notace rozlišovat:

- Algebraický (dlouhý i krátký zápis)
- International Correspondence Chess Federation (ICCF)
- Souřadnicový
- aj.

Už víme, že každé pole na šachovnici je jednoznačně identifikovatelné. Stejně se dají rozpoznávat i jednotlivé figury:

Figura	**Zkratka CZ**	**Zkratka EN**
Král	K	K
Dáma	D	Q
Střelec	S	B
Jezdec	J	N
Vež	V	R
Pěšec	bez značky	bez značky

Dlouhá algebraická notace (Long Algebraic): je notací využívající klasické indexování sloupců (písmena A – H) a řad (číslicemi 1 – 8) a značení figur jejich prvním písmenem z názvu. Zde ovšem nastává nevýhoda nepřenositelnosti zápisu algebraickou notací napříč různými jazyky, kde se názvy figur mohou lišit.

Ukázka zápisu CZ:

1. e2 – e4 e7 – e6
2. Jg1 – f3 f7 – f5
3. Sf1 – b5 Jg8 – h6

Ukázka zápisu EN:

1. e2 – e4 e7 – e6
2. Ng1 – f3 f7 – f5
3. Bf1 – b5 Ng8 – h6

Zkrácená algebraická notace (Short Algebraic): Vychází ze své nezkrácené verze, jen omezuje popis pozice pole na šachovnici, v zápisu nese jen cílové pole tahu, je-li takto zapsaný tah nejednoznačný (př. dva jezdci mohou jít na stejné pole na šachovnici), uvede se buďto zdrojová pozice tahu celým zápisem tedy indexem sloupce i řádku, nebo se používá zkrácený zápis označení pouze řádku či sloupce, odkud se daný tah uskutečňuje. Ukázka zápisu:

1. e4 e6
2. Jf3 f5
3. Sb5 Jh6

ICCF: je notace uzpůsobena strojovému zpracování. Každému poli šachovnice je přiřazeno jednoznačné číslo (indexování: 11 – 88). Každá pozice na šachovnici je tedy zapsána dvěma číslicemi (první indexuje sloupec, druhá řádek) nesoucí informace o řádku a sloupci šachovnice. Zápis jednotlivých tahů je zaznamenáván způsobem: zdrojové pole, cílové pole:

Ukázka zápisu:

1. 5254 5755
2. 7163 6765
3. 6125 7886

Souřadnicový zápis (Coordinate) zapisuje tahy stylem: souřadnice zdrojová, souřadnice cílová. Souřadnice jsou zde indexovány znaky (A – H) a číslicemi (1 – 8).

Pro zaznamenávání v naší výukové aplikaci jsem zvolil zkrácenou algebraickou notaci, pro její jednoduché pochopení i zápis. Zápis podobný zápisu ICCF jsem použil pro vnitřní reprezentaci demonstračních ukázek ve webovém prostředí.
Další používané značky ve zkrácené algebraické notaci:

Značka	**Význam**	**Příklad užití**
x	V případě že v provedeném tahu, jsme vyhodili soupeřovu figuru	Sxb5
+	V případě že v provedeném tahu, jsme dali soupeři šach	Sb5+
0-0	Malá rocháda	
0-0-0	Velká rocháda	
#	V případě že v provedeném tahu, jsme dali soupeři mat	Sb5#
!	Silný tah	Jf3!
!!	Velmi silný tah	b3!!
?	Chyba	b2?
??	Hrubá chyba	a1??
!?	Zajímavý tah	g4?!
?!	Pochybný tah	Ja3
	Jestliže pěšec projde na poslední řadu, dáváme za zápis tahu identifikátor figury, v jenž jsme pěšce proměnili	e8D

V této části mé práce bylo čerpáno z [31].

Kapitola 3

Výuka šachu

Šachy jsou hrou, kterou se nelze naučit za jedno či dvě odpoledne. Mnoho šachových začátečníků si myslí, že osvojením si základních pravidel šachu a naučením se pohybování figur po šachovnici mají vše potřebné za sebou. Ve skutečnosti jsou tyto dovednosti pouhým nezbytným odrazovým můstkem k tomu, aby z nich mohli stát praví šachisté. Výuku šachu lze rozdělit do několika samostatných kapitol.

Pro úplné začátečníky je důležité, aby pochopili základní principy a pravidla, osvojili si pohyb figur po šachovnici a dokázali jednoduše zanalyzovat postavení na šachovnici.

Po zvládnutí pravidel šachu můžeme přistoupit k hře samotné. Šachovou partii lze rozdělit do tří částí: na zahájení, střední hru a koncovku, přičemž každá část je stejně důležitá.

Úvod každé partie je označován jako zahájení, pod tímto pojmem se skrývá několik prvních tahů partie. Cílem této fáze hry je především boj o střed (který má být rychlý, aktivní, úplný, harmonický, plánovitý), vývin figur a zajištění bezpečnosti krále.

Šachová zahájení se klasicky dělí na:

- otevřená
- polootevřená
- uzavřená

Tato klasifikace je dnes užívána prakticky všude a její výhodou je jednoznačné určení oproti dříve užívaným označení, jako křídelní zahájení nebo nepravidelné zahájení, která jsou méně jasná. Každá z těchto tří základních skupin pak obsahuje množství jednotlivých zahajovacích systémů, které se dále dělí na jednotlivé varianty a subvarianty, z nichž řada nese své (vlastní) jméno. Tato pojmenování vznikala v průběhu mnoha let vývoje různými způsoby. Některá zahájení nebo varianty byly například pojmenovány po významných šachistech, kteří je objevili, používali ve svých partiích, případně je nějakým způsobem zpopularizovali (španělská hra, lotyšský gambit). Dalším častým případem je, že zahájení získalo své jméno podle jeho čistě šachového obsahu (hra dvou jezdců v obraně, střelcova hra). V oblasti šachových zahájení existuje mezinárodní kód označovaný ECO, který zařazuje jednotlivé varianty pod mezinárodní kombinaci písmene a čísel.

Po zahájení se partie přenáší do fáze hry zvané střední hra. Boj ve střední hře můžeme vést dvěma způsoby, které nazýváme poziční hra a kombinační hra. Tyto dva způsoby jsou spolu úzce spjaty, přičemž řídící roli má ten první. Při poziční hře se snažíme na základě obecných znaků dané pozice nalézt co nejlépe vyhovující plán hry. Souhrn zásad, kterými se řídíme při jejich stanovení, se nazývá strategie. Kombinační hra slouží k tomu,

abychom specifickým způsobem dosáhli cílů, které jsou stanoveny plánem hry. Souhrn všech prostředků, jimiž těchto cílů dosahujeme, se nazývá taktika.

Dalším dílčím prvkem každé partie je umění dáti protihráči mat. Pro hráče je nezbytně důležité, aby dokázal dát mat pomocí kombinace kterýchkoliv figur. K této kapitole se váže nejen matování, ale koncovky šachové partie obecně a procvičuje schopnost dokázat proměnit nepatrnou materiální převahu v koncovce ve vítězné postavení. Koncovky se elementárně dělí na koncovky: pěšcové, věžové, dáma proti pěšci na předposlední řadě, koncovky stejnobarevných střelců.

Významným pomocníkem mnohých šachistů jsou počítače. Šachovou hru lze provozovat na šachových speciálech (velmi zjednodušeně řečeno to jsou šachovnice, které samy hrají šachy) nebo na speciálních šachových programech určených pro osobní počítače.

Začátečníci budou moci počítač používat pro interaktivní formu výuky jednotlivých fází hry. Pro zkušené hráče je však počítač neocenitelným pomocníkem. Ukládání sehraných partií do databáze a jejich následné vyhledání podle zvoleného kritéria zpřehlední studium vlastních odehraných partií.

V této kapitole jsem se opíral o doporučení z trenérských příruček: [30] a [32].

Kapitola 4

Přehled šachových programů

V dnešní době již existuje velké množství šachových programů, které lze rozdělit do několika skupin. Programy můžeme rozlišit podle toho, zda jsou komplexní a obsahují uživatelské rozhraní, šachový motor a rozsáhlé databáze či se jedná o programy specializované. V takovém případě se může jednat o samostatné šachové motory, databázové aplikace či pouhá grafická rozhraní, které ke své funkčnosti vyžadují připojení šachového motoru.

4.1 Vývoj šachu na počítači

První šachové programy se začaly objevovat hned po vzniku prvních programovatelných počítačů. V roce 1949 americký matematik Claudie Shannon vydal článek *Programming a Computer for Platiny Chess* [45], pojednávající o základních principech programování šachu na počítači, ve kterém definoval dva zásadní body pro nalezení nejvhodnějšího tahu v partii, a to minimax proceduru dále popsanou v 5.1 a ohodnocovaní funkci, kterou naleznete v kapitole 6.4.

V letech 1948 – 1952 pracoval na svém šachovém algoritmu *TurboChamp* anglický matematik Alan Turing. Tento algoritmus obsahoval přesně stanovené podmínky pro výběr nejlepšího tahu. Ohodnocení pozice stanovoval na základě materiálního ohodnocení jednotlivých figur a na prověřování možných tahů. Tato analýza se obzvláště zaměřovala na útočné tahy. Pokud byl tah vyhodnocen jako útočný, byly následně prověřeny veškeré útočné tahy protivníka. Takto se rekurzivně procházelo vygenerovanými tahy, až vznikla situace, kdy nebyl možný ani jeden útočný tah, nebo se partie dostala do koncové pozice (pat, mat). V takto vzniklých situacích se provedlo materiální ohodnocení šachovnice a zvolil se nejlépe ohodnocený tah. Pro zjemnění ohodnocovací funkce zavedl Turing několik dalších vyhodnocovacích podmínek.[5]

- Bonusové ohodnocení pohyblivosti figur. Bonus se vypočetl jako odmocnina z počtu možných tahů figury.
- Bonusové ohodnocení pokud byl tah útočný.
- Bonusové ohodnocení pokud jsou vlastní figury chráněny (míra ohodnocení závisí na počtu figur, které danou figuru brání).
- Bonusové ohodnocení rochády. Toho ohodnocení bylo bráno ve třech možnostech, rochádu bylo stále možno provést, rochádu bylo možno provést v příštím tahu, rocháda byla již provedena.

- Bonusové avšak i penalizační ohodnocení na základě nebezpečí šachu.

Jednotlivé body ohodnocovací funkce byly převzaty z [43].

V době kdy Turing algoritmus dokončil, však doposud nebyl dostatečně výkonný počítač, který by program byl schopný spustit. Proto Turing sehrál partii s kolegou Alickem Glennie, ve které Turing simuloval počítač. Tuto partii nakonec TurboChamp prohrál [48].

Významný milník nastal v roce 1958 s příchodem nového algoritmu, který byl nazván alphabeta. Tento algoritmus pomocí prořezáváním značného množství tahů výrazně zefektivnil prohledávání jednotlivých větví stavového prostoru.

V letech 1970 – 1980 se v laboratořích výzkumné organizace Bell vyvíjel specializovaný šachový počítač s příslušným software. Tento počítač byl pojmenován Belle. Na tomto projektu pracovali Joe Condol a Ken Thompson. Počítač dokázal analyzovat 150 000 až 180 000 pozic za sekundu a stal se tehdejším nejsilnějším existujícím počítačovým programem na světě, když v roce 1980 zvítězil v World Computer Chess Championship (WCCC). [43]

Na konci osmdesátých let minulého století studenti z univerzity Carnegie-Mellon v Pittsburgu přišli s projektem *Deep Thought*, který doznal částečných úspěchů na mistrovství v počítačovém šachu. Mezi zajímavé přínosy patří i implementace kompletního generátoru tahů za pomoci VLSI na jediný čip. Tým autorů podílející se na vývoji Deep Thought přešel do výzkumného ústavu firmy IBM. Zde byl šachový počítač dále vyvíjen a právě z Deep Thought vzešel *Deep Blue*, který v roce 1997 sehrál šest partií proti šachovému mistru světa Garrymu Kasparovovi. Deep Blue s rozdílem jedné partie vyhrál 3.5 ku 2.5. Deep Blue je schopen vypočítat 200 miliónu pozic za sekundu, což při partii představuje schopnost propočtu tahů v průměru až 7 tahů dopředu. Při analýze využívá přibližně 6000 různých ohodnocovacích faktorů, realizovaných za pomoci specializovaného hardwaru. Konkrétně porážka Garry Kasparova je důsledkem specializovaně sestavené vyhodnocovací funkce, která byla zaměřena právě na slabiny tohoto hráče. [33], [43]

Ačkoli počítačové programy v současnosti dosahují velkých úspěchů s porážkami šachových velmistrů a jejich propočet s rychlostí analyzovaní tahů je závratný, mají v sobě už od dob, kdy pánové Shanon a Turing poprvé definovali své šachové algoritmy, zásadní znevýhodnění. Počítač není stále schopen vymýšlet nové strategie ani myslet. Vše je postaveno na kvalitě hodnotící funkce, která je postavena tak, aby zobecnila ohodnocení jakékoli pozice. Je nemožné sepsat takovou vyhodnocovací funkci, která by byla schopna obsáhnout a správně analyzovat veškeré možně vzniklé pozice na šachovnici.

4.2 Šachové motory

Co to vlastně šachový motor (chess engine) je? Jedná se o tu část programu, která má za úkol rozgenerovávat pozice do obecně zadané hloubky. Na základě ohodnocení vzniklých pozicí a principu užitých algoritmů musí zvolit nejlepší možný tah a ten předat další části aplikace pro zahrání. Šachové motory bývají mnohdy nezávislé na GUI aplikace. Zvláště pak motory určené pro unixové systémy, které nemívají GUI vůbec. Pro tyto případy vznikají specializovaná grafická prostředí, určená pro spouštění různých šachových motorů. Mezi takové aplikace se řadí např. Aréna, XBoard a WinBoard, které obsahují šachovnici, menu, ukládání a načítání partie aj. WinBoard jako první zavedl možnost spuštění dvou šachových motorů proti sobě, což posléze rozšířil o možnost pořádání turnajů šachových motorů [47]. Nejznámější šachové motory jsou komerční Rybka 3, open-source Crafty nebo český Phalanx. Amatérských šachových motorů však existuje obrovské množství, jejich výkonnost se pohybuje od ELO koeficientů blížících se ke 3000 až po programy hrající na úrovni

nejzákladnější výkonnostní třídy. Přehled vybraných motorů je v tabulce 4.1 a v následující kapitole jsou popsány přední profesionální šachové motory.

Název	ELO	Jazyk	Hledaní	DK	Šachovnice	Algoritmy
Beowulf	2300	C	Alfa-beta	Ano	Bitová pole	N,K,H,T,Q
ChEngine	-	C#	Alfa-beta	-	Bitová pole	T,Q
Crafty	2600	C	Alfa-beta	Ano	Bitová pole	T,K,Q,N
Gaia	2800	C	Alfa-beta	Ano	Bitová pole	N,K,H,T
Muse	2500	C	Alfa-beta	Ne	Bitová pole	N, H,K, T
Onno	2600	C++	Alfa-beta	Nalimov	Bitová pole	N, H
Rebel	2400	-	Alfa-beta	Ano	-	N, T
Ruffian	2400	C	Alfa-beta	Nalimov	Bitového pole a 8x8	N
Zappa	2700	C++	Alfa-beta	-	Bitová pole	-

Tabulka 4.1: Srovnání vybraných šachových motorů

Vysvětlivky

DK – databáze koncovek
N – Nultahová heuristika viz kapitola 5.6 na straně 23
H – History pruning viz kapitola 5.5 na straně 23
T – Transposition tables viz kapitola 5.4 na straně 21
K – Killer Moves viz kapitola 5.5 na straně 23
Q – Quiescence search viz kapitola 5.7 na straně 24
Nalimov - využívá Nalimovu databázi koncovek [38]
- Nebylo dohledáno

Jak si lze z tabulky 4.1 všimnout, při programování šachového motoru se typicky používají programovací jazyky C nebo C++. Lze se setkat i s jazyky jako Pascal nebo jazykem symbolických instrukcí a zřídka se vyskytují i motory psané v Javě nebo C#. Stejně tak lze na první pohled vidět, že nejtypičtější reprezentace šachovnice je pomocí bitového pole. Tato reprezentace vykazuje proti klasickým polím vyšší rychlost při generování tahů viz. 6.1

Pozn. Přehled užitých algoritmů je pouze orientační. Vycházel jsem z informací, které poskytli sami autoři na svých stránkách. Stejně tak i uváděné hodnoty ELA jsou zde pouze pro orientační přiblížení výkonnostní úrovně motoru. Při shromažďování informací o jednotlivých motorech bylo využito těchto pramenů: [19], [17], [29], [18], [44], [23], [12], [8], [4].

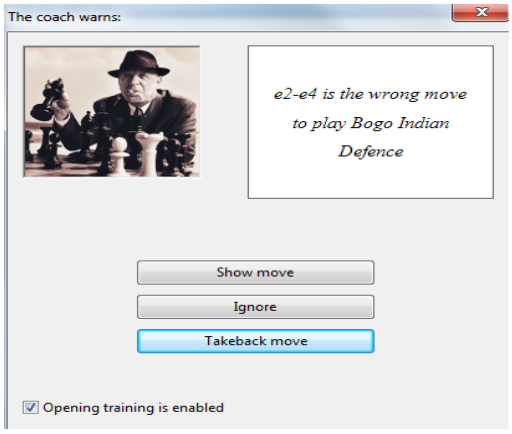
4.3 Současné vůdčí šachové programy

V komerční sféře se již řadu let udržuje své výsadní postavení na trhu německá společnost *ChessBase*, která dodává program *Fritz*, v současnosti s pořadovým číslem 12. Tento program se může pochlubit propracovanou umělou inteligencí, se kterou své síly měří světoví šachoví odborníci a odehrávají spolu vyrovnané partie. Databáze Fritze čítá 1,5 miliónu partií v epochálním rozpětí 1625 – 2009. Výuková část je zde ve formě tutoriál videí. Ve

12 hodinách k vám promlouvají světoví šachoví trenéři a šachoví velmistři a komentují šachové partie v několika výkonnostních třídách (komentáře namluvili: Alexei Shirov, Rustam Kasimdzhanov, Vladimir Kramnik, Vishy Anand, Garry Kasparov), jak je naznačeno na obrázku 4.1, který je převzat z [27]. Program také obsahuje hodinovou instruktáž pro úplné začátečníky, ve které vás zasvětí do pravidel šachu. Fritz také obsahuje tréninkové moduly pro zahájení, taktiku a koncovky (ukázka na obrázku 4.2, který je převzat z [27]). Uživatel vybere konkrétní zahájení a počítač odpovídá požadovanými tahy, dokonce lze zapnout varování v případě, že hráč neodehrál tah podle daného zahájení.

Při hraní partie proti umělé inteligenci lze zapnout tréninkový mód, ve kterém Fritz komentuje aktuální pozici na šachovnici. Aplikace umožňuje uživateli nastavit různorodou obtížnost protivníka, nastavit handicap, analýzu a vysvětlení aktuální pozice, zobrazování varování přímo na šachovnici a sběr statistik partií.

Obrázek 4.1: Video tutoriál

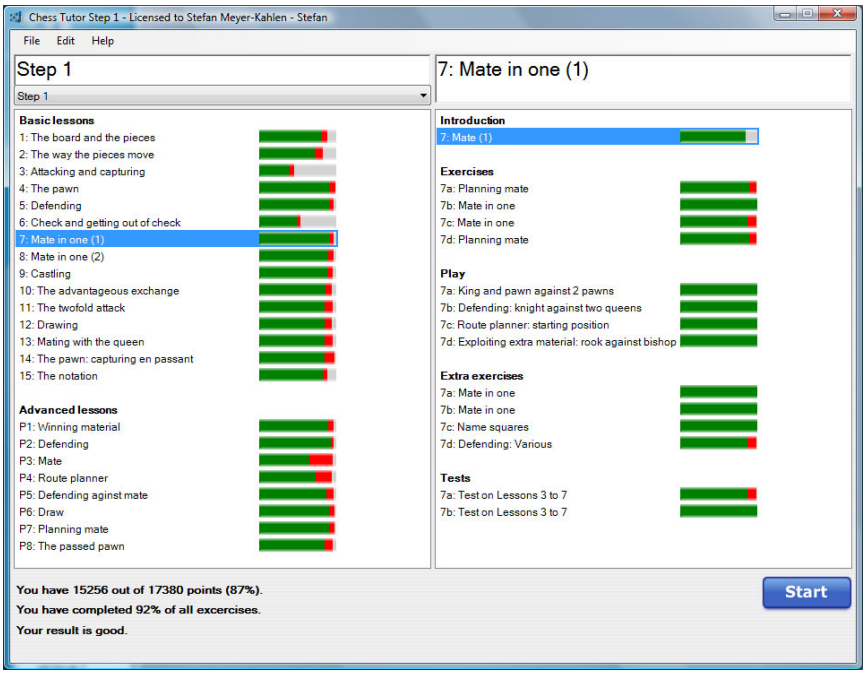


Obrázek 4.2: Upozornění na chybu při výuce zahájení

Stejná firma stojí i za programem *Shredder*, největším sběratelem mistrovských titulů, jehož autorem je *Stefan Meyer-Kahlen*. Shredder funguje ve stejném grafickém rozhraní jako Fritz 12. Shredder obsahuje i výukovou část, která se nazývá *Chess Tutor Step 1* (v současnosti je i Chess Tutor Step 2) a obsahuje výuku šachů od naprostých základů. Ve 23 lekcích lze nalézt na 1800 cvičení. Ukázka vyhodnocení úspěšnosti v jednotlivých lekcích je naznačena na 4.3, obrázek byl převzat z [6].

Dalším programem, který je v současnosti na vrcholu výkonnostního žebříčku, je program *Rybka*. Za jeho zrodem (v roce 2005 byla představena beta verze) stojí Američan s českým původem *Vasik Rajlich*, který svým nečekaným příchodem s motorem, který byl schopen porážet komerční konkurenci, způsobil doslova senzaci. Zpočátku byl Rybka vydáván pouze jako UCI motor, nyní je však k dispozici i v komerčním vydání s dodávaným rozhraní *Aquarium*, které je ukázaného na obrázku 4.4, který je převzat z [42]. Nyní dosáhl již svého třetího vydání, které kromě neobyčejně silného šachového motoru obsahuje nekonečnou analýzu šachového motoru, herní analýzu, kontrolu chyb, interaktivní hluboký rozbor či databázi s 3 300 000 partiemi.

Informace o těchto programech jsem čerpal ze zdrojů [27] a [42].



Obrázek 4.3: Ukázka výukové aplikace Chess Tutor

Obrázek 4.4: Ukázka programu rybka s rozhraní Aquarium

4.4 Ohodnocení šachového motoru

Šachy jsou sportem jako každý jiným a soutěživost je typickou lidskou vlastností. Proto bylo zapotřebí jistého ohodnocovacího systému na globální úrovni, který by odrážel aktuální výkonnostní úroveň hráčů. Pro tyto účely byl navržen koeficient *ELO*, což je statistické ohodnocení hráče na základě jeho výsledků. Mezinárodní šachová federace (FIDE) několikrát do roka vydává tabulku udávající pořadí šachistů v žebříčku podle ELA. Například ELO koeficient jednoho z nejnadanějších českých šachistů Davida Navary, je ke dni 17.4.2010 roven 2708 [28].

Nejen šachisté se rádi měří svou výkonností, ale i u počítačových motorů je sestavována tabulka na základě ELA. U šachových motorů není hodnota ELA tak jednoznačná, jako tomu bylo v tabulce od FIDE. Výsledné hodnoty jsou závislé na počítačové architektuře, na které jsou tabulky sestavovány. Podle několika zdrojů [50], [25] a [3] je ke dni 17.4.2010 jasným vítězem motor Rybka 3, jehož ELO se pohybuje v rozmezí 2900 – 3250. Ve srovnání s ELO českého mistra je jasně patrné, že výpočetní technika poskytuje šachovým motorům dostačující prostředky pro poměřování kvalit se světovými velmistry a mnohdy je již dokáží i překonat.

Pro testování síly šachových motorů se podle internetového zdroje [13] doporučuje provádět testovaní nejlépe na starším počítači, na kterém je jasně vidět kvalita heuristiky a hodnotící funkce. Také se doporučuje vypnout veškeré knihovny (zahájení, koncovek), aby všechny pozice musely být propočítány šachovým motorem.

Testování může být dvojího typu. Buďto jsou vyhodnocovány speciálně sestavované pozice, ve kterých má motor zvolit nejlepší tah. Při tomto testování se obvykle volí takové pozice, ve kterých je nejlepší řešení na první pohled skryto a mnohdy se jeví jako špatná volba (obětování figury, přehlédnutí zisku kvality). Existují již předpřipravené kolekce takovýchto taktických analýz, jako například BT 2450, BT 2630 nebo BS 2830. První dvě písmena v názvu určují jméno autora testů. V uvedených testech to jsou. B = Hubert Bednorz, T = Fred Tönissen a S = Heinz-Josef Schumacher. Demonstrace aplikování analýzy na šachové motory je naznačena na obrázku 4.5, který je převzat z [2].

Druhou možností jsou tzv. *test-matches*, neboli partie dvou šachových motorů proti sobě. Toto testování bývá mnohdy součástí turnaje šachových motorů. Pro důkladné otestování se doporučuje sehrát až na tisíce partií, při kterých jsou průběžně střídány barvy figur pro jednotlivé motory, nebo se uplatňují praktiky jako vymezení určitého počtu konkrétních zahájení pro obě dvě strany.

Při snaze ověřit si kvality svého programu jsem narazil na test založený na analýze 10 diagramu z reálných partií, který ověřuje výkonnostní úrovně amatérských šachových motorů (avšak i šachistů). Demonstrace těchto diagramů je uvedena v dodatku B.

Při spuštění testu s mým šachovým motorem mu při hloubce zanoření 3 půltahy bylo vypočteno ELO 1510. Při opakovaném spuštění testu s hloubkou 5 půltahů a bez uplatnění heuristik bylo ELO ohodnoceno na 1580. Test je převzat z internetového zdroje [1], kde bylo provedeno i vyhodnocení výsledků mého programu.

4.5 Turnaje šachových motorů

Turnajů šachových motorů existuje celá řada, ve kterých se utkávají profesionální i amatérské motory. Jednoduchý turnaj si může uspořádat kdokoli v rámci programu WinBoard [47]. Jedním z nejuznávanějších turnajů je *Chess Engines Grand Tournament* (CEGT).

LEGEND
Correct answer by Chess Tiger
Correct answer by ChessGenius
Correct answer by PocketChess
Same & incorrect answer by multiple programs
??? = Program crashed while analyzing

		Chess Tiger		ChessGenius		PocketChess	
(BT) ELO		2114		1812		1647	
#	ANSWER	EVALUATION	TIME	EVALUATION	TIME	EVALUATION	TIME
1	**Nxg7**	Nxg7	117	Qb3	900	Qb3	900
2	**Bxb6**	Bxb6	342	Bxb6	62	Be3	900
3	**Re6**	Re6	331	Re6	87	Nh3	900
4	**Qf7**	Rb1	900	Rf1	900	Qb1	900
5	**Ka6**	Ka6	69	Kb4	900	Bf3+	900
6	**e3**	e3	347	e3	81	Re8	900
7	**Rd6**	Rd6	53	Rc2	900	Nc4	900
8	**Rxc6+**	Rxc6	102	Rxc6+	2	Rxc6+	236
9	**g5**	g5	1	Rf8	900	Bxe5	900
10	**Rxg7+**	Bh6	900	c4	900	c4	900
11	**Qxh2+**	Qxh2	536	Qh5	900	Qh5	900
12	**Qe4**	Qe4	44	Qf5	900	b6	900
13	**Nb4**	Ne3	900	Ne3	900	Qc5	900
14	**Rxh7**	Rxh7	93	Qxb5	900	Qxb5	900
15	**Rg6**	Rg6	1	Rg6	1	Rg6	332
16	**g6**	g6	7	g6	327	???	900
17	**Qxf4**	Qxf4	68	Rxf4	900	Rxf4	900
18	**d6**	a5	900	d6	380	Kxd4	900
19	**f3**	Bxf1	900	Bxf1	900	Bxf1	900
20	**Ra2**	Ra2	106	Ng5	900	Ng5	900
21	**Re6**	Re6	226	Qe4	900	Qe4	900
22	**a3**	a3	13	cxd5	900	Bxf6	900
23	**Qf6**	Qf6	63	Qf6	192	Bh6	900
24	**g6**	Ra1	900	Ra5	900	Ra5	900
25	**Nd3**	Kc5	900	Kc5	900	g2	900
26	**f5**	f5	2	Qe2	900	f5	546
27	**e6**	e6	356	Qxe4	900	???	900
28	**Ne4**	Ne4	1	Ne4	1	Ne4	238
29	**Ke1**	Ke1	17	Ke1	1	Ke1	240
30	**f4**	d4	900	Rg8+	900	Rg8+	900

Obrázek 4.5: Taktická analýza testu BT 2450

Turnaje se účastní amatérské tak i profesionální motory a jsou rozděleny podle přiděleného času na tah do tří kategorií.

- 40 tahů ve 4 minutách (40/4)
- 40 tahů ve 40 minutách (40/40)
- 40 tahů ve 120 minutách (40/120)

Turnaje povolují *Symmetric multiprocessing*, tedy využití více procesorů na jedné základní desce, díky čemu vzrůstá výpočetní výkon počítače, což je jedním z hlavních faktorů u výkonnosti motoru.

CEGT je také organizátorem lig šachových motorů, které jsou odehrávány ve dvou výkonnostních třídách (Master Class a Higher Class). Liga je definována pro kategorii 40/40 s využitím procesoru o taktu 2 GHz. [3]

Další organizací zajišťující turnaje šachových motorů je *Swedish Chess Computer Association* (SSDF) [21]. Všechny partie jsou hrány systémem 40/120 se specifikací, že pro každých následujících 20 tahů je vymezena hodina hracího času.

4.6 Výukové programy

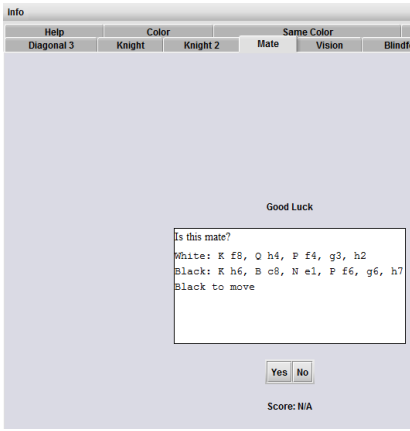
Kromě komerčních programů, které zpravidla obsahují výukovou část a disponují obrovskými knihovnami různorodých úkolů a cvičení, existuje řada dalších programů, které sice nedosahují takového věhlasu, ale oproti tomu jsou tématicky zaměřeny čistě na výuku šachu.

Jedním z takových programů je i *Chessmentor*. Tento program nabízí 2150 výukových cvičení, bohužel tyto příklady nejsou přehledně roztříděny tak, aby odpovídaly lekcím či výukovým modulům, jak jsme zvyklí u ostatních programů. Výuka vždy předpokládala pouze jedno možné řešení a každé bylo řádně okomentováno a objasněno. Na závěr je nutno podotknout, že se jedná o komerční projekt s možností vyzkoušení dema, které obsahuje 75 ukázkových cvičení. Více informací o tomto programu lze nalézt na [26].

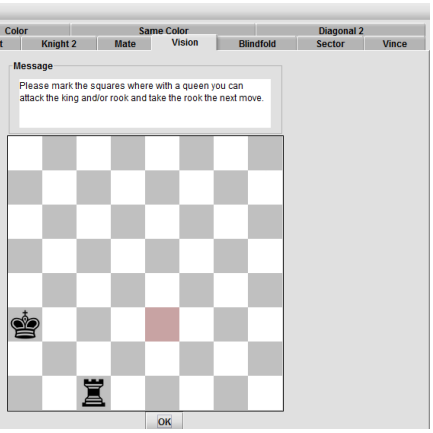
V oblasti freeware programů nalezneme zástupce v podobě *Chess Training Tools* [35], který se specializuje na výuku šachu pro úplné začátečníky. Jednotlivé lekce postupně představují rozložení šachovnice, výuku tahů figurami (patrné na obrázku 4.7), rozpoznání matové pozice (patrné na obrázku 4.6) a zajímavou lekcí je zkouška šachové představivosti, kdy je po uživateli požadováno určení pozice jednotlivých figur, při zadání několika tahů partie prostřednictvím textového zápisu. Tato kapitola nemá pevně předdefinované úkoly, nýbrž se aplikaci předá PGN[1] zápis partie, který je zanalyzován a po uživateli je požadováno koncové rozestavění partie.

Dalším komerčním produktem se specializací na výuku je Chessimo [24]. Program umožňuje již tradiční hru proti počítači. Dále obsahuje integrovanou formu tréninku. V plné program disponuje více než 4000 taktických cvičení. Podobně jako tomu bylo u Fritze, i zde si uživatel vybere výuku zahájení, strategie, koncovky či jiné z nabízených kapitol. Forma výuky je zde prezentována demo ukázkou či interaktivní cestou. Jednotlivé cvičení jsou sestaveny profesionálními šachisty, podobně jako tomu je i v odborné šachové literatuře. Mnohé cvičení jsou doprovázena patřičnými komentáři a vysvětleními. Ukázku programu můžete vidět na obrázku 4.8.

[1] Portable Game Notation. Jedná se o elektronický formát pro záznam partie.



Obrázek 4.6: Výuka matování



Obrázek 4.7: Výuka tahů

Obrázek 4.8: Výukový program chessimo

Kapitola 5

Algoritmy určené k výběru tahu

Obecně lze veškeré šachové partie vyjádřit jako obsáhlý strom, ve kterém jsou obsaženy veškeré možně vzniklé pozice na šachovnici, které nazveme uzly. Aktuální rozestavění šachovnice je tedy pro nás kořenem tohoto stromu. Listy tohoto stromu jsou koncové pozice partie (pat, mat).

Podle knihy [36] je na šachovnici přibližně 10^{43} možných pozicí, ve kterých mohou být figury rozestavěny. Shannon k tomuto číslu dospěl podle vzorce:

$$\frac{64!}{[32! \cdot (8!)^2 \cdot (2!)^6]}$$

a přesný výsledek odpovídá $4.63 \cdot 10^{42}$. Pozdějšími propočty bylo toto číslo upraveno na 10^{40}, kdy byly eliminovány pozice, ve kterých stojí králové bezprostředně vedle sebe či při kterých stojí oba králové v šachu.

Pro názornost tedy předpokládejme rozestavění šachovnice při začátku partie, které označíme za kořen. První úroveň stromu bude obsahovat 20 uzlů (16 tahů pěšci a 4 tahy jezdci). A již druhá úroveň stromu má uzlů 400. Podle [36] je celkový počet uzlů prohledávacího stromu přibližně 10^{100} korektních pozicí.

Uzly stromu lze procházet dvojím způsobem – do hloubky a do šířky. Jelikož počítač není schopen z aktuálního rozestavění figur na šachovnici zvolit jasnou strategii pro partii, kterou by se hra měla z jeho pohledu ubírat, je nucen zvolit procházení stromu do šířky, aby nebyla vynechána analýza ani jedné z možných variant. Pro jednotlivé varianty je pak strom procházen do hloubky. Algoritmus pro vyhledávání tahu prochází strom nejprve do hloubky (obvykle explicitně zadané), takže je vždy analyzována pouze jediná větev stromu. Tento princip má značnou nevýhodu v analyzování i neperspektivních uzlů stromu. Aplikací algoritmů (např. alfabeta) jsou prováděny rozsáhlé heuristiky, díky nimž je počítač schopen odřezávat jednotlivé nevyhovující větve, avšak proti heuristice šachových odborníků je zde stále vidět značná neefektivita. Šachisté si typicky vyberou několik (obvykle $2-3$) možných variant, které se jim jeví jako nejnadějnější, a pak je analyzují do hloubky až třiceti tahů.

5.1 Algoritmus minimax

Jedná se o velmi rozšířený algoritmus, na jehož základě je postaveno značné množství šachových programů. Název je odvozen od maximálního a minimálního ohodnocení pozicí, ke kterým se algoritmus rekurzivním zanořováním do hloubky propracuje.

Hloubka zanoření bývá pevně omezena hodnotou, či přiděleným časovým úsekem. V algoritmu platí, že hodnota, která je pro jednu stranu maximální (nejvýhodnější pozice), je

pro protihráče minimem (nejméně výhodná pozice). Funkce tedy vrací ohodnocení uzlů a pro hráče, jenž je právě na tahu, i tah s maximálním ohodnocením (tah vedoucí k nejvýhodnější pozici). Při aplikaci algoritmu jsou vygenerovány veškeré regulérní tahy a odpovědi na ně. Takto se pokračuje až do zadané hloubky, která udává počet vygenerovaných půltahů, přičemž každá z takto vzniklých pozicí je ohodnocena. Zvolení hodnoty hloubky prohledávání musí být optimální vzhledem k časové náročnosti při procházení všech uzlů prohledávacího stromu. Časová složitost je u tohoto algoritmu exponenciální, vyjádřitelná vztahem v^h, kde v značí hodnotu větvícího faktoru[1] a h odpovídá hloubce rekurzivního zanoření.

Pro zjednodušení volání funkce minimax je upravena hodnotící funkce tak, aby vracela vždy ohodnocení pro hráče jenž je právě na tahu. Rozdíl mezi hodnotami max a min zavisí jen na určení, který hráč je právě na tahu. Pokud by pozice na šachovnici byla vyhodnocena tak, že bílý má jednoho pěšce k dobru, stejná pozice by pro černého byla vyhodnocena jako ztráta jednoho pěšce. První ohodnocení by bylo kladnou hodnotou, druhé pak zápornou. Hodnota by v obou případech byla stejná a rozdíl by byl pouze ve znaménku. Z tohoto důvodu se při rekurzivním volání funkce minimax využívá střídání znaménka (negování). Tato úprava funkce minimax se v některých zdrojích nazývá *Negamax*[39].

Pseudokód:

```
int minimax( position p, int deep )
{
  Moves m;
  int value;
  best = -INF;

  if ( deep ==  0 )
    return Evaluate(p);

  m = GenerateMoves(p);

  for ( int i = 0;  i < m.count;  i++ )
  {
    SimulateMove(p, m[i]);
    value = -minimax(p, deep - 1);
    BackMove(p, t[i]);

    if (value > best)
        best = value;
  }
  return best;
}
```

5.2 Algoritmus alfabeta prořezávání

Algoritmus alfabeta prořezávání je zefektivněním časově náročného algoritmu minimax. Alfabeta zavádí do algoritmu dvě nové hodnoty, alfa a beta meze, pomocí nichž algoritmus

[1] průměrná hodnota počtu možných tahů

odřezává varianty, které jsou horší než alfa, díky čemuž dojde k redukci prohledávacího stromu. Alfa je dolní mezí a zabraňuje zbytečnému vygenerování tahů pro hráče, jenž je právě na tahu, a beta je mezí horní a zabraňuje vygenerování tahů protihráče.

Pseudokód:

```
int alphabeta( position p, int deep, int alpha, int beta )
{
  Moves m;
  int value;
  best = -INF;

  if ( deep ==  0 )
    return Evaluate(p);

  m = GenerateMoves(p);

  for ( int i = 0;  i < m.count;  i++ )
  {
    SimulateMove(p, m[i]);
    value = -alphabeta(p, deep ? 1, -beta, -alpha);
    BackMove(p, t[i]);

    if (value > alpha)
    alpha = value;
    if(value >= beta)
       return beta;
  }
  return alpha;
}
```

5.3 Třídění tahů

Efektivita algoritmu alfabeta je závislá na pořadí tahů při procházení stromu. Největší efektivity je dosaženo, jestliže na počátku prohledávání jsou procházeny nejnadějnější varianty. Proto existují metody na třídění tahů tak, aby byly potenciálně výhodnější tahy upřednostněny v seznamu všech vygenerovaných tahů. K takovémuto utřídění se využívá zavolání hodnotící funkce pro všechny vygenerované tahy. Na základě ohodnocení jsou tahy setříděny a v upraveném pořadí jsou předkládány metodě alfabeta.

Pro zvýšení rychlosti lze na vygenerované tahy volat upravenou hodnotící funkci. Tato funkce neprovádí kompletní heuristiku, ale provádí se jen analýza důležitých aspektů. Často se využívá zvýhodněné ohodnocení útočných tahů, při kterých dochází k výměně figur.

5.4 Předpočítané tabulky

V šachu vede mnoho různých cest do stejné pozice na šachovnici. Proto při rekurzivním volání minimax (či jiné) funkce se téže pozice analyzují hned několikrát. Řešením tohoto neduhu je ukládat výsledky ohodnocení a při dalším vyhodnocování nově vzniklé pozice se

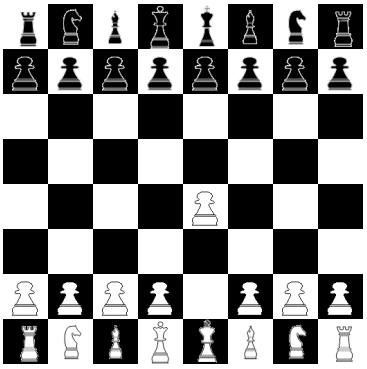
nejprve prohledají již uložená ohodnocení. Ukládání je řešeno pomocí hash tabulky (Transposition table), kde klíčem je ohodnocovaná pozice na šachovnici a hodnotou je dané ohodnocení.

Nicméně shodné rozestavění figur na šachovnici neznamená, že jsou dvě pozice totožné. V úvahu musíme vzít informace o příznacích platných pro rochádu a braní mimochodem.

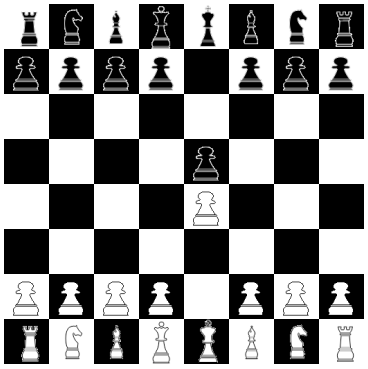
Pro vytvoření číselné hodnoty hashe se používá metoda Zobristova hashování [39]. Tato metoda si klade za úkol, aby generované hodnoty byly co nejvíce rozprostřeny, a to i pro velmi podobné pozice. Každé z 64 polí šachovnice může nabývat 12 hodnot odpovídající právě jednomu typu figury. Prázdná pole jsou rovna 0. Postupně je tedy vytvářeno 768 (12×64) 32 b nebo 64 b náhodných čísel. Několik dalších čísel je potřeba náhodně inicializovat pro příznaky rochády a braní mimochodem. Tato čísla zůstávají v tabulce po celou dobu stejná a pomocí nich se určí hash pro danou pozici. Pro získání výsledného hashe pozice provedeme nad každým neprázdným polem šachovnice logickou operaci XOR s příslušným náhodně inicializovaným číslem a danými hash hodnotami pro příznaky.

Názorná ukázka:

Varianta první

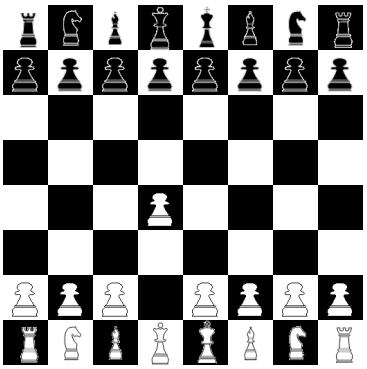


Obrázek 5.1: 1. e4

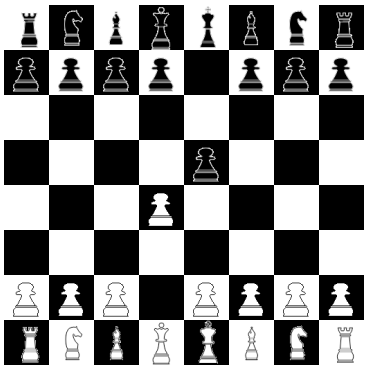


Obrázek 5.2: 1. ..e5

Varianta druhá

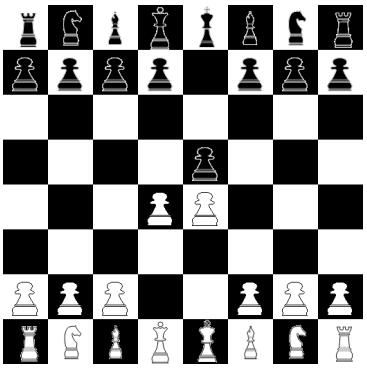


Obrázek 5.3: 1. d4

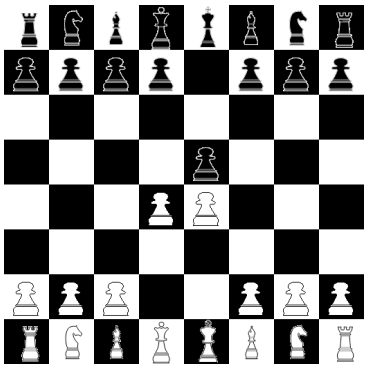


Obrázek 5.4: 1. ..e5

Totožná pozice



Obrázek 5.5: 2. d4



Obrázek 5.6: 2.e4

5.5 Zabijácká a historická heuristika

Tato metoda vylepšuje alfabeta prořezávání jednotlivých větví stromu. Princip je založený na podobnosti rozložení figur na šachovnici. Způsobí-li tedy tah prořezání, je pravděpodobné, že při jiném rozložení figur, které je na totožné úrovni stromu, bude také způsobeno prořezání. Jinými slovy, má-li tedy soupeř jeden dobrý tah, jsme nuceni mu v tomto tahu zabránit. Tah, kterým soupeřovi ve výhodném tahu zabráníme, se nazývá *killer move*.

Tyto speciální tahy, způsobující prořezávání větví, se ukládají zvlášť pro každou úroveň[2]. Při procházení tahů v určité hloubce se ověří, zdali pro danou hloubku jsou již některé tahy obsaženy v množině killer moves. Pokud ano, jsou tyto tahy při procházení upřednostněny, jelikož je u nich vyšší pravděpodobnost, že znovu způsobí průřez větve.

Killer moves lze ještě dále dělit na dvě podskupiny. Na tahy prořezávající větve s dobrými tahy protihráče a na tahy zabraňující matu. Tento princip využívá například program Rebel [44]. Druhá skupina killer moves je pak upřednostněna před běžnými tahy zabraňující prořezávání.

Historická heuristika funguje na podobném principu jako zabijácká heuristika. Narozdíl od ní se však neomezuje na aktuální úroveň zanoření, nýbrž využívá informace získané z celého procházeného stromu. I proto je též někdy nazývána jako zobecnění zabijácké heuristiky. V současnosti je tato metoda spíše na ústupu. V této kapitole bylo čerpáno z internetového zdroje [29].

5.6 Nultahová heuristika

Tato metoda heuristiky slouží k odfiltrování slabých tahů ještě před spuštěním rekurzivního analyzování do zadané hloubky. Princip je založený na provedení dvojtahu hráče[3], který je právě na tahu. Pokud se při takovémto zvýhodnění daný dvojtah pozitivně neprojeví na ohodnocení, je prohlášen za slabý a dále se neanalyzuje. Některé programy aplikují nultahové heuristiky i při rekurzivním zanořování. Tato heuristika je nejsilnější ve střední hře partie, v koncovkách je dokonce vypínána. V této kapitole bylo čerpáno z internetového zdroje [29].

[2] Některé programy ukládají killer moves ze dvou úrovní.
[3] Soupeřův tah je přeskočen.

5.7 Vyhledání klidu

Algoritmus minimax provádí rekurzivní zanořování do pevně zvolené hloubky prohledávání. V partii ovšem může nastat stav, kdy algoritmus vyhodnotí pozici na šachovnici vysokým hodnocením, avšak v příštím půltahu protihráč zahraje tah, kterým získá ještě lepšího postavení. Tento stav se nazývá tzv. efekt horizontu.

Omezený dohled algoritmu se pokouší eliminovat algoritmus nazývaný vyhledání klidu (quiescence search). Tato metoda je založena na procházení tahů i za pevně stanovenou hloubku zanoření. Analyzovány jsou však už jen tahy, které potencionálně mohou změnit vývoj partie v jasnou převahu jednoho z hráčů. Toto dodatečné procházení je ukončeno v okamžiku, kdy jsou všechny potenciálně nebezpečné tahy vyhodnoceny. Tento okamžik je nazýván *stav klidu*.

Za klidnou pozici lze obecně považovat takovou, ve které v příštím tahu nedojde k vyhození figury či prosazení pěšce na poslední řadě. V této kapitole bylo čerpáno z internetového zdroje [29].

5.8 Metoda okna

Při aplikaci alfabeta prohledávání jsou na počátku hodnoty alfa a beta nastaveny na maximální možný rozsah ohodnocení. Pro urychlení vyhledávání se používá metoda tzv. okna. V této metodě se před prohledáváním odhadne rozsah, do kterého spadne výsledek. Tento rozsah se vhodně volí podle výsledku předchozí iterace. Nevýhodou ovšem je, když se výsledek nevleze do zvoleného rozsahu (okna). V takovémto případě je nezbytné celý proces vyhledávání opakovat s větším rozsahem. V této kapitole bylo čerpáno z internetového zdroje [29].

5.9 Správa času

Šachové partie mají mnohdy pevně přidělený čas. Tento čas se tradičně přiděluje na celou partii nebo na určitý počet odehraných tahů (40/5, 40/40, 40/120) a při jeho překročení hráč automaticky prohrává. Šachové algoritmy proto musí být schopny kontrolovat zbývající čas a přizpůsobovat mu i hloubku prohledávání. Počítač má časové zvýhodnění na začátku partie, kdy tahy obvykle volí podle uložené databáze zahájení a ze svého přiděleného času spotřebuje jen časový interval, který zabere režie spojená s prohledáním databáze.

Ve střední hře už musí počítač aplikovat efektivní správu času. Jedna z možných metod je, že počítač algoritmicky odhadne možný počet tahů do konce partie a poté zbývající čas rovnoměrně rozdělí mezi odhadovaný počet tahů. Obecně se pak používá iterační prohlubování ve spojení s hodnotou efektivního větvícího faktoru. Tyto metody usnadňují okamžité vyhodnocení situace, kdy je možno zrušit právě prováděnou iteraci a zvolit nejlepší dosud nalezený tah nebo v závislosti na zbývajícím přiděleném čase na tah rozhodnout, zdali se má spustit další iterace (je pravděpodobné, že vyhodnocení proběhne ve zbývajícím čase kompletně) nebo novou iteraci již nespouštět (je pravděpodobné, že by iterace ve zbývajícím čase neproběhla kompletní a musela by být zrušena). Algoritmy obvykle počítají i s rezervou na kritické situace. Proto si v průběhu partie alokují časovou rezervu pro takovéto situace, které vyžadují důkladnou analýzu. V této kapitole bylo čerpáno z internetového zdroje [5].

Kapitola 6

Návrh a implementace šachového motoru

Návrh šachového motoru lze rozdělit do několika základních kategorií. První z nich je vhodná volba reprezentace šachovnice. O této problematice pojednává podkapitola 6.1. Dalšími body návrhu jsou zvolení vhodných metodik pro výběr tahu 6.3 a ohodnocení pozice na šachovnici 6.4. Důležitou částí je i generátor tahů, který je popsán v 6.2. Poslední bod 6.5 návrhu pojednává o využití databází v šachovém motoru.

6.1 Šachovnice

Reprezentace šachovnice v rámci programu je značně důležitým aspektem, od kterého se poté nadále odvíjí techniky pro generování tahu a ohodnocovací funkce. K reprezentaci šachovnice lze přistupovat v zásadě dvojím základním způsobem. Prvním způsobem je tzv. mailbox struktura [36]. Jedná se o reprezentaci, tak jak ji původně popsal Shanon. Šachovnice je reprezentována 64 proměnnými typu integer. Proměnná nabývá hodnot v rozsahu -6 až 6 a nese informace o jednotlivých figurách. Záporná čísla jsou černé figury a kladná jsou bílé. Číslo nula reprezentuje prázdné pole šachovnice. Jednotlivé tahy jsou pak reprezentovány třemi údaji: index počáteční pozice pole, index cílové pozice pole a poslední údaj je volitelný a skrývá v sobě informace o povýšené figuře z pěšce. Tato forma reprezentace je jednoduchá a člověku přirozená, avšak nevýhodou je vyšší režie při generování tahů při detekci meze šachovnice.

Toto znevýhodnění eliminuje užití šachovnice s rozměry 10×12 polí. Efektivně se samozřejmě používá opět pouze 8×8 polí jako v předešlém případě, zbylá jsou pouze doplňková. Rozšířením šachovnice na 10×12 polí získáváme jakési mantinely, ohraničující naší šachovnici. Ohraničení je dáno dvěmi bočními sloupci (z každé strany jeden) a čtyřmi řádky (z každé strany dva). Tímto rozšířením eliminujeme detekci mezí šachovnice. Pokud figuře byl vygenerován tah mimo meze šachovnice, dopadne na doplňkové pole a podle speciální identifikace těchto polí je ihned rozpoznáno překročení mezí a tah je zamítnut.

Druhým přístupem k reprezentaci šachovnice je použití struktury *bitových polí* (bit boards). Tato reprezentace se zakládá na užití 64 b slov (bitové pole). V rámci popisu šachovnice použijeme takových polí 12. Každé pole zastupuje jeden typ figury šachovnice. Index pole výskytu této figury koresponduje s jedním bitem bitového pole na téže indexu v rámci pole. Bity náležící obsazeným polím šachovnice jsou nastaveny na logickou hodnotu 1 zbylé jsou nulovány. Tento postup zrychluje přístup k obsahu i generování tahů a to

zvláště na 64 b architekturách. Nevýhodou této reprezentace je neintuitivní odraz abstrakce v lidském chápání. Více informací o implementaci lze dohledat na [41].

V mém programu je zvolen prvně zmiňovaný způsob reprezentace s klasickým rozložení 8×8 polí – `boards[8, 8]`. Abstrakce šachovnice je umístněna ve tříde `BaseChessboard`, stejně tak jako metody s ní manipulující (DefaultChessboard, InitChessboard, SetSquare, RotateChessboard aj.) a metody reprezentující tah (Move, BackMove, tmpMove, SimulateMove, aj.). Tuto konvenční reprezentaci jsem zvolil pro její nezanedbatelnou podobnost s šachovnicí reálnou. Tato podobnost značně usnadnila návrh výukové části programu, především sestavování konkrétního rozložení figur na šachovnici. Oproti užití šachovnice 10×12 polí je nevýhodou již zmiňovaná režie při generování tahů a také jisté zpomalení při přístupu k dvojrozměrnému poli oproti poli jednorozměrnému, které je v rámci frameworku .NET silně optimalizováno.

Na rozdíl od Shanonova návrhu je pozměněna abstrakce informací o jednotlivých polí šachovnice. Jedno pole šachovnice zde není chápáno jako číslo, avšak jako struktura o dvou prvcích:

- CHESSMAN chessman
- STATE state

Položka chessman s vymezením daným výčtovým typem `CHESSMAN` (Pawn, Bishop, Knight, Rook, Queen, King, Nothing) informuje o figuře umístněné na daném poli šachovnice. Položka state je opět dána výčtovým typem `STATE` (Black, White, Free) a nese informace o barvě figury nebo udává, že pole je prázdné.

6.2 Generátor tahů

Generátor tahů je jednou z nejvytíženějších částí programu, která je využívána takřka v každém modulu programu. Generátor je implementován v třídě `GenerateMove`, která obsahuje metody pro vygenerování tahů na základě několika požadavků. Nejzásadnějším požadavkem je, zdali generování tahů probíhá v závislosti na rozložení figur šachovnice. V takovémto případě jsou vraceny jen tahy v souladu s pravidly, naopak pokud obsah šachovnice není směrodatný, jsou vygenerovány tahy všechny. Další specifikací je pro jaký účel jsou tahy generovány. Zde rozlišujeme generátor pro standardní průběh partie a reverzní generátor uplatnitelný při převodu zápisu partie z algebraické zkrácené notace do posloupnosti souřadnicových dvojic odkud – kam.

Dále je rozlišováno, zdali se generují tahy pro všechny figury hráče, jenž je na tahu, nebo pouze pro zadanou figuru, či zda má generátor vracet tahy jen na prázdná pole šachovnice, nebo i tahy při kterých byla vyhozena protihráčova figura.

Generátor vrací seznam tahů. Tah je zde reprezentován třídou `generatePcMove`, jejichž atributy jsou řádkové a sloupcové souřadnice polí odkud a kam figura tah provede. Také je zde obsažena hodnota `value`, která obsahuje ohodnocení tahu. Pro jednoduché třídění tahů, které jsou uspořádany v seznamu, je v této třídě doimplementována metoda `CompareTo`, která tahy třídí podle hodnoty atributu `value`. Třídění tahů v seznamu je pak možné použitím metody `Sort`.

6.3 Výběr nejlepšího tahu

Zvolení nejlepšího tahu z vygenerovaného seznamu korektních tahů je realizováno pomocí algoritmu alfa-beta prořezávání, popsaného v kapitole 5.2. Pro dosažení vyšší efektivity prohledávání je využíváno utřídění tahů za pomoci jednoduché heuristiky, která upřednostní při samotném rekurzivním prohledávání ty tahy, které vedou k vyhození soupeřových figur. Tato metoda je více popsána v kapitole 5.3. Další heuristikou, která je uvnitř motoru využívána je tzv. *nultahová heuristika* detailněji popsána v 5.6.

6.4 Ohodnocovací funkce

Ohodnocovací funkce (Evaluation function), také známá jako heuristická hodnotící funkce. Jedná se o obecné ohodnocení jakékoli vzniklé pozice na šachovnici, přičemž je kladen důraz na rychlost vyhodnocení.

Na ohodnocení vzniklé pozice při analýze nahlížíme ze dvou pohledů. Prvním aspektem je vyhodnocení podle materiálu, jímž jednotliví hráči disponují. Tento způsob vyhodnocování je způsobem nejzákladnějším a užívají ho snad všechny šachové programy. Základní myšlenkou je sečtení materiálu obou hráčů. Figury hráče, z jehož pohledu je analýza prováděna, jsou ohodnoceny kladnými hodnotami, figury protivníka naopak zápornými. Ohodnocení jednotlivých figur je naznačeno v tabulce 6.1 převzaté z [5] a doplněné o ohodnocení figur v mém programu.

Zdroj	Pěšec	Jezdec	Střelec	Věž	Dáma
Claude Shannon [45]	100	300	300	500	900
Max Euwe	100	350	350	550	1000
Hans Berliner	100	320	333	510	880
Chess 4.5	100	325	350	500	900
Larry Kaufman	100	325	325	500	975
Fruit	100	400	400	600	1200
Má aplikace	100	330	330	520	980

Tabulka 6.1: Přehled materiálního ohodnocení jednotlivých figur

Dalším možným pohledem při analýze je postavení figur na šachovnici. Figury, které stojí při analýze ve výhodných pozicích, jsou ohodnoceny kladně, kdežto figury, jejichž postavení je nevýhodné, dostávají z hodnocení bodovou srážku.

Postavení figur je vyhodnocováno pro každou figuru zvlášť. Podle vhodnosti postavení dané figury je k celkovému ohodnocení přičítán bonus či odečítaná penalizace v rozsahu ± 75 bodů, což odpovídá $\pm\frac{1}{3}$ hodnoty nejslabší figury šachovnice – pěšce (100) . Toto ohodnocení se týká postavení jednotlivých figur na šachovnici, ještě je však potřeba zhodnotit vzájemnou souhru při rozestavění figur.

Nejzákladnějším požadavkem při zahajovacím systému je obsadit centrum šachovnice. Proto i ohodnocovací funkce při analýze pozice jednotlivých figur tento fakt zohledňuje a nejlépe hodnocená pole jsou právě ta ve středu šachovnice. Analýze však podléhají pouze tzv. lehké figury (jezdec, střelec), pěšci a král. Lehké figury hrají hlavní roly při vývoji partie a postavení ve středu šachovnice jim umožňuje napadnutí nejvíc polí. Pěšci (zvláště pak ti centrální) otevírají svým postupem vpřed manipulační prostor figurám z poslední řady šachovnice a postupně posouvají obranný val směrem kupředu a zmenšují hrací prostor

protihráči. U krále je výhodné postavení závislé na fázi partie. Při zahájení a střední hře se považuje za správný základ, když je král bezpečně uschovaný a chráněný ostatními figurami. Při zlomu partie do koncovky se z krále stává aktivní figura, která podporuje útok (typicky pěšcové koncovky). Pokud se partie přenese z koncovky do čistě matující koncovky, uplatňuji dynamické ohodnocování pozic po každém odehraném tahu.
Kladné ohodnocení:

- pěšec je volný
- blok pěšců vedle sebe
- král stojí v bezpečí (chráněn pěšcovým valem) nejlépe po rochádě
- barva střelce je barvou čtverců na kterých stojí protivníkovi pěšci
- střelec dává protihráčovu figuru do vazby
- věž stojí na volném sloupci (či polovolném)
- zdvojení věží
- dva střelci jsou výhodnější než střelec a jezdec
- dva střelci jsou výhodnější než dva jezdci

Záporné ohodnocení:

- dvojpěšec (a více)
- izolovaný pěšec

	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
8	0	0	0	0	0	0	0	0
7	75	75	75	75	75	75	75	75
6	10	10	20	50	50	20	10	10
5	5	5	10	30	30	10	5	5
4	0	3	5	25	25	5	3	0
3	0	1	3	12	12	3	1	0
2	0	0	-5	-25	-25	-5	0	0
1	0	0	0	0	0	0	0	0

Tabulka 6.2: Statická poziční ohodnocovací tabulka bílého pěšce

6.5 Databáze zahájení a koncovek

Zahájení jsou z pohledu šachového motoru obtížnou částí partie. V partii jsou zahájení volena podle daného strategického plánu, kdežto šachový motor volí tahy podle hodnotící funkce. Z tohoto důvodu se k zahájení přistupuje poněkud odlišně oproti zbylým částem partie. Program obsahuje databázi skutečných partií (obvykle ve formátu PGN). Poté se jedním průchodem vyberou potenciálně nejlepší zahájení (toto vyhodnocení je založeno na

	a	b	c	d	e	f	g	h
8	-75	-40	-20	-10	-10	-20	-40	-75
7	-40	- -20	-5	0	0	-5	-20	-40
6	-30	0	10	15	15	10	0	-30
5	-30	10	15	25	25	15	10	-30
4	-30	5	25	30	30	25	5	-30
3	-30	5	20	30	30	20	5	-30
2	-40	-20	0	10	10	0	-20	-40
1	-75	-50	-30	-20	-20	-30	-50	-75

Tabulka 6.3: Statická poziční ohodnocovací tabulka bílého jezdce

porovnání ELA hráčů, kteří partii sehráli) a sestaví se orientovaný graf hry, ve kterém jsou vrcholy pozice, které se alespoň jednou vyskytly ve vybraných partiích.

Obdobný způsob je využit i u koncovek. Základem je vygenerování databáze koncovek, která obsahuje veškerá možná rozestavění, do kterých je možno se ze zadané pozice dostat, včetně ohodnocení takto vzniklých pozicí a délky nejrychlejšího řešení. Vzhledem k obrovskému rozsahu takto vzniklých pozicí jsou kladeny obrovské nároky na paměťovou náročnost databáze (podle pramenu [20] jsou kapacitní požadavky pro 6 figurové koncovky přibližně 1.12 TB a připravované 7 figurové koncovky se budou kapacitně pohybovat mezi 50 – 60 TB), a to i přes aplikaci řady optimalizací využívající vertikální symetrii šachovnice a u nepěšcových koncovek i horizontální symetrii a diagonální symetrii (podle pramenu [41] je udáváno zhruba 6,4 násobné zrychlení a úsporu paměti). Vzhledem k tomuto omezení jsou dnes zvládnuty databáze obsahující vygenerované koncovky až pro šest figur. U šesti figurových koncovek jsou z databáze vynechány kombinace pěti figur proti králi, pro takovéto pozice je dostačující užití obecného šachového motoru.

Správně navržené šachové motory jsou schopny jednoduché koncovky vyřešit také, bohužel v netriviálních koncovkách začíná být patrný efekt omezeného zanoření a motor se k ohodnocení závěrečné pozice nedostane.

Databáze koncovek si lze nalézt na [10] nebo použít online analyzátor náležící k programu *Shredder* na adrese [46] nebo od *Lokasoft* na adrese [38].

V mé aplikaci je databáze zahájení realizována s omezením na předdefinovaná zahájení dle ECO kódu (seznam všech zahájení seřazený dle ECO naleznete zde [9]). Tímto však není uživatel jakkoliv omezen. Jakékoli další zahájení si uživatel podle svého vlastního uvážení může do databáze vložit a aplikace s ním bude už nadále pracovat. Stejně tak výčet všech zahájení s definovaným ECO kódem není kompletní (množství všech zahájení mající svůj vlastní ECO kód je obrovské, přibližně 500 variant), ale jedná se pouze o jeho demonstrační podmnožinu. Uživatel si i tato zahájení může libovolně připojovat k databázi s přidělením příslušného ECO kódu. Databáze zahájení, se kterými aplikace pracuje, je obsažena v přiloženém souboru `zahajeni.xml`.

Pro názornost jsou názvy zahájení i jejich variant v aplikaci předvyplněny a stačí prostřednictvím formuláře zadat ECO kód partie a seznam tahů podle zkrácené algebraické notace. Aplikace poté už rozpozná variantu v zahájení a může s ní plně pracovat.

Pro přehlednost je přidán i jednoduchý filtr, který umožní zobrazit všechna zahájení a varianty, a to i ty, které nejsou v databázi nebo jen varianty, které jsou uloženy v databázi.

Kapitola 7

Návrh a implementace výukové části

Výuková aplikace pro hru šachy by měla obsahovat alespoň tři základní prostředky pro různorodou formu výuky. První částí je modul reprezentující interaktivní formu výuky, jenž uživateli zprostředkovává základní šachové situace, jejichž úspěšné zvládnutí a tím pádem i osvojení si vzniklých typických šachových konstrukcí, vede uživatele k připravenosti na dané momenty v reálné partii.

Další, a z praktického hlediska snad i nejdůležitější, částí výuky je možnost odehrání vlastní partie. Tuto možnost může uživatel zvolit buď proti živému oponentovi, kde je ovšem nezbytné, aby oba hráči byli přítomni u stejného počítače, nebo si zahrát partii proti počítači. Počítač při partii využívá *šachový motor*, který zprostředkovává generování tahů počítače v podobě protihráče a který z těchto tahů volí nejlépe ohodnocenou variantu.

Poslední částí aplikace je databázový modul, jenž je schopen ukládat a znovu přehrávat odehrané partie.

7.1 Interaktivní výuka

Výukový modul uživateli předkládá několik tématicky rozdělených kapitol výuky. Každá z kapitol se váže k jednomu ze zásadních okruhů šachové teorie. Kapitoly pak dále obsahují několik dalších možností, podle kterých lze konkrétně navolit způsob výuky dané kapitoly.

Rozdělení kapitol je následovní:

- Tahy
- Zahájení
- Koncovky a matování

Volba možností je specifická pro každou kapitolu. Typicky se jedná o konkrétní určení tématu (výuka tahů jezdcem, výuka zahájení – hra čtyř jezdců, matování věží aj.) a způsob, kterým bude výuka probíhat (demo ukázka, interaktivní forma výuky).

Při výuce je důležité, aby uživatel pochopil základní princip probíraného tématu a rozpoznal, čeho se po něm vlastně žádá. Proto je u každé probírané kapitol v úvodu přidána názorná demo ukázka obsahující typické situace a jejich řešení. Průběh těchto demo ukázek je z velké části naskriptován a uživateli je zde odepřena forma interaktivního zásahu do dění.

Naopak interaktivní část výuky je postavena na uživatelově aktivitě. Podle zvolené kapitoly je po uživateli vyžadována aplikace znalostí získaných při shlédnutí náležící demo ukázky.

7.2 Výuka tahů

Výuka tahů pro každou z figur obsahuje kromě tradiční demo ukázky další dva interaktivní módy. Oba jsou principiálně založené tak, aby se uživatel přemístil s danou figurou z jednoho místa na druhé. Základní mód tento úkol definuje na zcela prázdné šachovnici. Druhý pokročilejší mód definuje tentýž úkol na šachovnici obeskládané figurami obou barev. Přičemž platí, že hráč nesmí vstoupit ani na jedno obsazené pole (barva figury zde nehraje roly) a taktéž nesmí vstoupit na pole napadnutelné soupeřovými figurami. Tento mód využívá metodu `RandomInitChessboard` a lze ho spustit v pěti různých obtížnostech. Tato metoda vygeneruje pseudo-náhodné rozestavění figur na šachovnici. Úroveň obtížnosti je právě jedním z parametrů této metody, který udává mimo jiné i stupeň obsazenosti šachovnice. Při každém vygenerovaném rozestavění šachovnice nám počítač vypočte minimální počet tahů, které je zapotřebí k regulérnímu přesunu figury na požadované místo. K tomuto výpočtu vznikla třída `Solver`, která realizuje procházení statického stavového prostoru prostřednictvím algoritmu BFS dále popsaného v podkapitole 7.2.2. Úroveň zvoleného levelu rovněž odpovídá minimálnímu počtu tahů potřebných k přesunu na požadované pole šachovnice.

7.2.1 Pseudo-náhodné vygenerovaní šachovnice

Hráč musí být schopen pohybovat figurami i po rozestavěné šachovnici. Pro tento účel byla vytvořena metoda pseudo-náhodně generující rozestavění i množství figur na šachovnici.

```
void RandomInitChessboard( int rate, CHESSMAN keyfigure );
```

Parametr `rate` udává přibližné množství vygenerovaných figur na šachovnici. Parametr `keyfigure` definuje figuru, pro jejíž výcvik má být šachovnice vygenerována, a je udáván, aby generátor zohlednil vynucenou existenci této figury (př. aby nevznikla situace, kde jsou dva králové nebo devět pěšců).

Počet generovaných figur je rozložen na čtyři samostatné úkony. Samostatně se generují figury podle své barvy a taktéž se zvlášť generuje počet pěšců a ostatních figur. Při každém z těchto čtyř generování je náhodně určena odchylka v rozsahu $0-2$. Hodnota této odchylky určuje meze vygenerovaného počtu figur v každé ze čtyř kategorií. Tedy výsledný počet figur získáme vygenerováním náhodného čísla z rozsahu $rate - odchylka$, $rate + odchylka$.

7.2.2 Nalezení cesty figury

Při náhodném rozgenerování pozice šachovnice pro výuku tahů figur, je nutné určit, zdali vygenerovaná pozice je řešitelná, tedy existuje-li minimálně jedna cesta se zvolenou figurou z místa A na místo B. K tomuto úkonu je využívaná třída `Solver`, která obsahuje metodu pro nalezení řešení. Tato třída využívá k nalezení řešení algoritmus *Breadth-first search* (neboli také zkráceně BFS) s úplnou eliminací dříve expandovaných uzlů. Jedná se o algoritmus určený k postupnému prohledávání stavového prostoru do šířky. Algoritmus začíná u kořene uzlu a postupně prochází vždy nejbližší sousedící uzly, které doposud nebyly prozkoumány. Při expanzi uzlu jsou jeho sousedící uzly umístněny do seznamu uzlů, které je zapotřebí ještě prozkoumat, a právě prozkoumávaný uzel je označen jako již projitý. Takový uzel už

není do konce prohledávání vložen do seznamu uzlů k expanzi. Detailnější popis algoritmu BFS lze najít ve studijní opoře: Základy umělé inteligence [34].

Metoda `solveMoves` hledá řešení k nalezení cesty podle specifikace pravidel pro výuku tahů. Pro zopakování to jsou tato pravidla. Při pohybu po šachovnici nesmí být figurou vstoupeno na obsazené pole šachovnice ani pole napadnutelné soupeřovou figurou. Z tohoto důvodu je metodě předán parametrem výčet všech soupeřem napadnutelných polí šachovnice. Kromě tohoto parametru je metodě předávána specifikace figury, pro niž je řešení provedeno, dále souřadnice pole, na kterém je umístněna figura (kořen), a cílové pole, na které je třeba se dostat (match). Podle výše popsaného algoritmu metoda vkládá pole, na které může figura vstoupit, do seznamu. Pokud vkládané pole v seznamu již je, není znovu přidáváno. Takto metoda projde množinu všech dostupných polí, která vyhovují pravidlům. Pokud je cesta nalezena, je vrácen minimální počet tahů k dosažení cílového pole. V opačném případě je vrácena 0, což indikuje že cílového pole nelze dosáhnout.

Úroveň	**Rozsah minimálního počtu tahů**
2	2 – 3
3	4 – 5
4	6 – 7

7.3 Výuka zahájení

Při tréninku zahájení si uživatel nerozšiřuje své šachové dovednosti a logiku, nýbrž je potřeba, aby si jednotlivé varianty vryl hluboce do paměti. Z tohoto důvodu jsou možnosti při výběru výuky omezeny na demo ukázku a samostatné zopakovaní zahájení.

Výběr zahájení je rozdělen do několika podskupin. Nejobecnější dělení zahájení je určeno podle volby prvního tahu partie a má tři skupiny: otevřené, polozavřené a zavřené zahájení. Mimo toto klasické rozdělení je zde i čtvrtá možnost, kam spadají uživatelem definované zahájení, která nemají svůj ECO kód, avšak uživateli se daný postup jevil zajímavý a rozhodl se jej uchovat a případně se k němu časem vrátit a studovat jej. Na základě zvolení jedné ze čtyř možností se v nabídce už pro konkrétní zahájení zobrazí zahájení spadající do vybrané kategorie. V rámci jednoho zahájení však může existovat několik sice podobných, ale běžně v posledním tahu se lišících variant zahájení. Tyto varianty mívají své konkrétní pojmenování velice zřídka, a proto jsou v nabídce variant rozlišena pomocí ECO kódu. Tato nabídka se podobně jako tomu bylo i při výuce tahů přizpůsobuje vybranému zahájení a jsou tedy vypsány jen ECO kódy zahájení spadají pod danou hru.

Realizace databáze zahájení je popsána v podkapitole 6.5 uvedené na straně 28.

7.3.1 Přidávání zahájení

Pro názornost jsou názvy zahájení i jejich variant v aplikaci předvyplněny a stačí prostřednictvím formuláře zadat ECO kód partie a seznam tahů podle zkrácené algebraické notace. Aplikace poté už rozpozná variantu v zahájení a může s ní začít pracovat.

Pro přehlednost je přidán i jednoduchý filtr, který umožní zobrazit všechna zahájení a varianty a to i ta, která nejsou v databázi, nebo jen varianty které jsou uloženy v databázi.

Uživatel, jak jsem již zmínil, zadává jednotlivé tahy pomocí zkrácené algebraické notace, která je však strojově přímo nezpracovatelná, zato uživatelsky přívětivá. Při zpracování takto zadaných tahů se využívá třídy `ParseNotation`, která provádí konverzi jednotlivých

tahů do formátu [Z: XY] – [KAM: XY], kde první údaj udává souřadnice X, Y pole odkud figura táhla, a druhý údaj značí souřadnice X, Y pole na které táhla. Zde však nastává problém, protože tah zapsaný zkrácenou algebraickou notací v sobě běžně skrývá jen informace o souřadnicích cílového pole a figuře, která tah provedla. Proto převod každého tahu vyžaduje jistou kontextovou závislost na posloupnosti tahů předešlých, s tím, že při provedení prvního tahu posloupnosti musí být jednoznačně definované rozestavění figur na šachovnici. Běžně však toto rozestavění bývá totožné s rozestavěním při počátku partie.

Při konverzi samotné se tedy postupuje takto: cílové pole se převede v algebraické notaci na souřadnice X, Y a poznačí se figura, která tah provedla. Pak se zpětným generátorem vygeneruje seznam polí, odkud by se mohla zadaná figura dostat na pole cílové. Tato pole se porovnají, zdali se daná figura na tomto poli vyskytuje. Pokud ano, stává se toto pole polem zdrojovým.

7.4 Matování a koncovky

Podobně jako tomu bylo u zahájení, i tato kapitola nabízí dvě základní možnosti spuštění, a to demo ukázku a interaktivní aplikaci matovacího postupu. Tyto možnosti lze spustit v základních variantách a to matovaní za pomoci: dámy, dvou věží, věže, dvou střelců a střelce společně s jezdcem. Matování každým z uvedených způsobů má svoje specifické postupy. Vygenerování pozice pro matování je náhodné a funguje na podobném principu, jako tomu bylo u kapitoly výuka tahů. Rozestavění generuje tedy metoda `RandomInitChessboard` s jediným parametrem typu `ENDPLAY`, rozeznávající variantu matové koncovky partie.

7.4.1 Aplikace pravidel pro koncovky

Podobně jako při hře proti počítači i tady je princip založený na vygenerování všech dostupných tahů počítače a následném ohodnocování výsledného rozestavění šachovnice. Při matování se však už nemusí používat obecně platná hodnotící funkce, nýbrž se může specifikovat podle dané situace. Takto speciálně navržené ohodnocovací funkce aplikují analýzu na základě pravidel pro matující koncovky. Pro tento účel vznikla třída `Matte`, která dědí od třídy definující základní šachové sounáležitosti `BaseChessboard`. Tato třída obsahuje základní metodu `MakeMatte`, jejímž parametrem je definice šachové koncovky dle výčtového typu `ENDPLAY`, podle něhož se aplikuje vhodná ohodnocovací funkce a inicializují se potřebné statické ohodnocovací tabulky pro dané figury.

Ohodnocení u všech variant koncovek zohledňuje materiální hodnotu jednotlivých figur. Podobně tomu je i u partie proti počítači. Ve všech případech, ve kterých je nalezen tah vedoucí do matové pozice je vráceno maximální ohodnocení. Oponentův král pracuje se statickou tabulkou `KingOppTable`, která ohodnocuje jednotlivá pole šachovnice, kde jsou nejlépe hodnocena středová pole.

Podobně je tomu i u krále hráče, jenž má materiální převahu. Ten však nepracuje se statickou tabulkou ohodnocených polí, ale tuto tabulku si po každém tahu vždy znovu generuje na základě aktuální pozice oponentova krále. Platí, že mimo koncovky dvou věží, je král při matování aktivní a nezbytnou figurou. Proto jsou hodnocení polí šachovnice tomu aspektu přizpůsobena následovně:

Podle tabulky reprezentující pole šachovnice 7.2 se na poli D5 nalézá bránící král, jenž čelí matové pozici. Hodnoty těsně sousedící s tímto polem jsou ohodnoceny jako jediné zápornou hodnotou, toto ohodnocení je zde uváděno spíše symbolicky, protože generátor tahů

	a	b	c	d	e	f	g	h
8	-50	-40	-30	-20	-20	-30	-40	-50
7	-30	-20	-10	0	0	-10	-20	-30
6	-30	-10	20	30	30	20	-10	-30
5	-30	-10	30	40	40	30	-10	-30
4	-30	-10	30	40	40	30	-10	-30
3	-30	-10	20	30	30	20	-10	-30
2	-30	-30	0	0	0	0	-30	-30
1	-50	-30	-30	-30	-30	-30	-30	-50

Tabulka 7.1: Statická ohodnocovací tabulka bránícího krále v koncovce

	a	b	c	d	e	f	g	h
8	4000	400	400	400	400	400	400	300
7	4000	500	500	500	500	500	400	300
6	400	500	-1000	-1000	-1000	500	400	300
5	400	500	-1000	K	-1000	500	400	300
4	400	500	-1000	-1000	-1000	500	400	300
3	400	500	500	500	500	500	400	300
2	400	400	400	400	400	400	400	300
1	300	300	300	300	300	300	300	300

Tabulka 7.2: Dynamická ohodnocovací tabulka útočícího krále v koncovce

nesmí vygenerovat tah do bezprostřední blízkosti protivníkova krále, avšak pro názornost jsem toto pole ohodnotil zápornou hodnotou. Zbylá pole mají hodnotu úměrnou vzdálenosti od protivníkova krále. Hodnocení je vypočteno ve dvou krocích:

1. Určení, zdali ohodnocované pole má větší rozdíl X nebo Y souřadnice vzhledem k poli, na kterém se nachází protivníkův král.
2. Výpočet hodnoty pole vzhledem k jeho vzdálenosti od protivníkova krále.

Výpočet ohodnocení pole šachovnice pro útočícího krále v koncovce:

```
diffX = abs( blackKing.x - actualSquare.x )
diffY= abs( blackKing.y - actualSquare.y )

if( diffX >= diffY )
  value = ( 7 - diffX ) * 10;
else
  value = ( 7 - diffY ) * 10;
```

Také je v koncovce, zvláště při matování věží, důležité postavení králů v opozici. Bránící král by nikdy neměl do opozice vstupovat, neboť tím hrozí následný šach a zkrácení manipulačního prostoru, či přímo šach mat. Naopak útočící král se snaží přinutit protivníkova krále ke vstupu do opozice. Tento efekt se nazývá *tempování*. Útočící král by rovněž

za žádných okolností neměl dobrovolně vstupovat do opozice, jelikož pak hrozí možný útěk protivníkova krále. Tomuto faktu jsou přizpůsobeny i příslušné hodnoty ohodnocení daných polí, na kterých by králové stáli v opozici. Ovšem pro útočícího krále jsou zvýhodněny pozice těsně sousedící s polem opozičním. Když se na těchto polích nachází útočící král, tlačí tím protivníka do opozice.

Matování dámou

Při matování dámou je princip založen na odřezávání prostoru soupeřova krále a dotlačení ho k boční stěně šachovnice. V momentě, kdy je král odříznut na okrajovém řádku či sloupci, se dává do pohybu král aktivní (útočící) strany, dokud nedojde do blízkosti protivníkova krále tak, aby mohl být dámou dán mat.

I dáma využívá dynamicky vygenerovanou tabulku ohodnocení polí šachovnice. K vygenerování se používá speciální metoda `findBestOppPos`, která na základě souřadnic protivníkova krále určí nejbližší roh a nejbližší stěnu šachovnice, ke které bude soupeřův král tlačen.

Na základě určení nejbližší stěny šachovnice je vygenerována hodnotící tabulka polí pro dámu, která je naznačena tabulkou 7.3. Tato tabulka je vygenerována tak, že po vyhodnocení osmé řady šachovnice, jako nejbližší okrajové řady, je král horizontálně odříznut od zbytku šachovnice směrem ke kraji (právě k zmiňované osmé řadě). Z určení levého horního rohu (pole šachovnice a8) jako nejbližšího jsou přidány další bonusy na pole `e5` a `f5` odřezávající krále směrem k jeho nejbližšímu rohu.

Levý horní roh				Pravý horní roh			
	a	**b**	**c**	**d**	**e**	**f**	**g**	**h**
8	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
6	0	0	0	K	0	0	0	0
5	7000	7000	7000	7000	9000	8500	7000	0
4	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
Levý dolní roh				Pravý dolní roh			

Tabulka 7.3: Dynamická ohodnocovací tabulka dámy v koncovce

Matování dvěma věžemi

Při matování věžemi se využívá stejných metod jako v předchozí kapitole. Při každém tahu je dynamicky vygenerována tabulka ohodnocení pro jednotlivá pole šachovnice. Princip je založen na postupném odřezávání polí protivníkova krále, směrem k nejbližší stěně šachovnice. Vnitřní řada či sloupec ve směru k nejbližší stěně šachovnice, která sousedí s aktuální pozicí soupeřova krále, je ohodnocena hodnotou +7000. Pokud se v této řadě či sloupci již věž nalézá, je soupeřův král odříznut. V takovém případě je potřeba dát králi šach, respektive přímo mat. Taková pole jsou ohodnocena hodnotou +9000.

Matování věží

Matování jednou věží je oproti předcházející variantě značně složitější. V takovém případě se stejně jako při matování s dvěma věžemi aplikuje obdobné ohodnocení odřezávajícího sloupce či řady, avšak pole na kterých věž dává šach jsou ohodnocena pouze za předpokladu, že oba králové stojí v opozici. Ohodnocování polí krále je sestaveno tak, aby v kombinaci s věží byl soupeřův král nucen vstoupit do opozice.

Zhodnocení koncovek

Profesionální šachové programy přistupují k vyhodnocování koncovek zcela odlišným způsobem, jak jsem již popsal v podkapitole 6.5. Mé řešení nevyžaduje obsáhlé databáze a vzniklé situace se snaží řešit algoritmicky. Nevýhodou ovšem je značná složitost při implementaci algoritmů tak, aby obsáhly všechny postupy aplikované při matování. V aplikaci jsem tedy implementoval jen základní matové kombinace.

7.5 Rádce v partii

Uživateli je napovídáno i při samotné partii. V průběhu partie mu jsou průběžně oznamovány nechráněné figury protivníka i jeho vlastní. Tuto analýzu zprostředkovává metoda `Help` ve třídě `Report`, která při analýze využívá speciálně upraveného generátoru `AllMoveAttack`, který vrací tahy jen na soupeřem obsazená pole šachovnice. Takto získaný seznam tahů už je jen předložen metodě `CheckSquare`, zjišťující, jestli je dané pole hlídané jinou vlastní figurou či nikoli. Tento postup se aplikuje pro oba dva hráče, čímž docílíme získání soupisu napadnutých i nechráněných figur.

Zároveň metoda `Help` analyzuje dosavadní průběh partie a informuje uživatele o názvu aktuálně hraného zahájení a jeho možných pokračováních, kterými se může v zahajovacím procesu pokračovat. Tato analýza má obdobné omezení rozsahu databáze zahájení, jako měla tréninková část.

7.6 Databázový modul

Pro účel archivace odehraných partií, ale i seskládání jakéhosi depozitáře zahájení, je zvoleno co nejednodušší řešení, ale tak, aby ve formátu ukládaných dat byla stanovena hierarchie a struktura. Se zohledněním těchto dvou kritérií jsem zvolil XML formát dokumentu. Bezesporná výhoda XML je ve velké programátorské podpoře při zpracování XML.

7.6.1 Databáze XML

XML (Extensible Markup Language) je značkovací jazyk standardizovaný konsorciem W3C [22]. Jedná se o předchůdce SGML (Standard Generalized Markup Language), což je značkovací metajazyk, který standardizuje definice obecného značkovacího jazyku (XML). Jazyk se například využívá pro výměnu dat mezi aplikacemi (jabber) a obecně pro tvorbu všech dokumentů, u kterých je kladen požadavek na definovanou strukturu jednotlivých částí.

XML dokument je textový a skládá se z kombinace značek a textového obsahu. Základním požadavkem na strukturovaný XML dokument je, aby byl celý uzavřen v jednom elementu. Všechny elementy musí být párové. Pro počáteční tag `<tag>` musí vždy existovat i ukončovací tag `</tag>`. Výjimku tvoří prázdné elementy, které nemají žádný obsah. Jim

odpovídající tag však musí být ukončen znaky `/>` místo pouhého `>`. Vlastnosti jednotlivých elementů jsou popsány pomocí atributů. V XML jsou jména všech elementů a atributů citlivá na velikost písmen.

```
<výrobek>
    <název>Hifi věž SuperSound PX 537</název>
    <cena>8356</cena>
    <měna>Kč</měna>
</výrobek>
```

Ukázka XML zápisu

V kapitole je čerpáno z internetového zdroje [37], včetně převzaté ukázky XML zápisu.

7.6.2 DOM parser

DOM (Document Object Model) je objektově orientovaná reprezentace XML nebo HTML dokumentu. DOM je API umožňující přístup či modifikaci obsahu, struktury nebo stylu dokumentu, či jeho částí. Jedná se o standard W3C, který je nezávislý na programovacím jazyku. V kapitole je čerpáno z internetového zdroje [37].

7.6.3 Návrh XML dokumentu partie

Pokud uživatel potvrdí při začátku nové partie, že si přeje partii ukládat, je po jejím odehrání proveden výstup záznamu partie do dokumentu ve formátu XML. Dokument se nachází ve složce odkud byl program spuštěn a je pojmenován ve tvaru: white-black-dd.mm.yyyy.xml, kde white značí jméno bílého hráče partie, black jméno černého hráče a dd.mm.yyyy je aktuální datum, které je samo předvyplněno.

Samotná struktura dokumentu je následující: na prvním řádku dokumentu je vždy očekávána XML hlavička `<?xml version="1.0" encoding="utf-8"?>`, následuje kořenová značka `<game>`, která obsahuje nepárové elementy `<player1>`, `<player2>`, `<result>`, `<specification>` a párový element `<moves>`. Elementy player1 a player2 obsahují atribut `name="xxxx"`, result obsahuje atribut `result="White/Black win"` a element specification obsahuje atributy `date="dd.mm.yyyy"` a `place="xxxx"`. V párovém elementu tahy je obsaženo 1–N elementů `<move>` s povinnými atributy `moveNumber="x"`, `white="yy"` a `black="yy"`. Následuje konkrétní ukázka jedné varianty zahájení.

```
<?xml version="1.0" encoding="utf-8"?>
<game>
  <player1 name="Kamil Floryán" />
  <player2 name="cpu" />
  <result result="draw" />
  <specification date="26.4.2010" place="Uherský Ostroh" />
  <moves>
    <move moveNumber="0" white="e4" black="e6" />
    <move moveNumber="1" white="Jf3" black="Jc6" />
    <move moveNumber="2" white="Sc4" black="e5" />
    <move moveNumber="3" white="0-0" black="d6" />
  </moves>
</game>
```

Struktura XML dokumentu pro záznam partie

7.6.4 Přehrávání a uložení partie

Aplikace je schopna přehrávat již odehrané partie. Záznam partie je uložený v XML dokumentu. Výhodou tohoto formátu je velice jednoduché zpracování využívající DOM parseru XML. Nevýhodou je, že tento formát je interní, navržený jen pro účely aplikace, tudíž nelze očekávat podporu při zpracování v jiných šachových programech.

7.6.5 Návrh XML dokumentu databáze zahájení

Aplikace očekává databázi zahájení uloženou v externím souboru `zahajeni.xml` v adresáři odkud je program spuštěn. Formát struktury XML dokumentu je pevně daný. Po XML hlavičce `<?xml version="1.0" encoding="utf-8"?>` následuje kořenová značka `<zahajeni>`, která obsahuje jednotlivé varianty zahájení `<hra>` (párový element) s povinnými atributy `eco="xxx"`, `jmeno="yyyyy"` a `typ="zzz"`. Kde zzz může nabývat jedné ze tří hodnot: Otevřené zahájení, Uzavřené zahájení nebo Polootevřené zahájení. Každá hra ma opět povinný párový element `<tahy>`, který obsahuje 1–N elementů `<tah>` s povinnými atributy `cisloTahu="x"`, `bily="yy"` a `cerny="yy"`. Následuje konkrétní ukázka jedné varianty zahájení.

```
<?xml version="1.0" encoding="utf-8"?>
<zahajeni>
  <hra eco="A10" jmeno="Anglická hra" typ="Uzavřené zahájení">
    <tahy>
      <tah cisloTahu="1" bily="c4" cerny="" />
    </tahy>
  </hra>
</zahajeni>
```

Struktura XML dokumentu pro záznam zahájení

Kapitola 8

Návrh grafického rozhraní

Pro realizaci mého šachového programu jsem původně zvolil vývojové prostředí Visual Studio 2005, které jsem později zaměnil za jeho novější verzi 2008 [16]. Při samotné realizaci programu jsem se rozhodl pro formu window form aplikace. Při návrhu jsem se rozhodl pro jeden hlavní globální form, který bude ostatní okna zapouzdřovat a zároveň bude plnit formu globálního menu nad celou aplikací. V jednotlivých oknech jsou použity dekorační obrázky, které byly převzaty z internetových stránek [7] a [14].

8.1 Šachovnice

Vizuální prezentace šachovnice je vedena jako samostatné okno. Pro své vykreslování využívá rozhraní GDI+, které je součástí .NET frameworku. Minimální rozměry okna jsou nastaveny na 620×620 pixelů, proto aby se šachovnice zobrazovala bez deformací. Při zvětšování tohoto okna reaguje aktivní vykreslená plocha šachovnice vždy na menší stranu okna. Pokud se tedy bude měnit rozměr pouze jedné strany okna, zůstává vykreslená šachovnice neměnná. Grafické provedení šachovnice je možno měnit podle tří barevných kombinací, kterými jsou klasická černo-bílá, dále modro-bílá a červeno-bílá. Textury jednotlivých figur byly do aplikace vloženy ve formátu PNG o rozměrech 65×65 pixelů a převzaty byly z internetového zdroje [11].

Objekt okna šachovnice lze vytvořit v několika různých variantách. Základní rozdělení je, zdali byla její inicializace zahájena v hracím módu či výukovém. Výukový mód lze dále rozdělit podle způsobu výuky, tedy na demo ukázky a interaktivní část. Nepatrné rozdíly jsou i v jednotlivých kapitolách výuky, ale ty se týkají spíše ovládání, přesněji řečeno pohybu figur po šachovnici.

Kvůli těmto výše zmíněným odlišnostem má tedy objekt `Chessboard` dva konstruktory, které se uplatní podle toho, zdali je požadován hrací či výukový mód. Výukový mód si specifikaci volí dle předaných parametrů při inicializaci, které se sice mezi jednotlivými výukovými kapitolami mírně liší, avšak jejich formu jsem se snažil co nejvíce unifikovat.

Obecně tedy objekt třídy `Chessboard` funguje pouze jako jistý vizualizátor nad abstraktním modelem šachovnice definovaným třídou `chess` a zároveň s daným objektem manipuluje. Mezi základní metody třídy `Chessboard` patří `DrawChessboard` pro vykreslení hrací plochy, dále pak metody komunikující s informačním oknem *Report*, obsluha interaktivních událostí (zvláště pak stisk levého a pravého tlačítka myši nad hrací plochou) a v neposlední řadě metoda `makePcMove`, která vyvolává generování tahů počítačového protihráče (avšak toto generování probíhá už prostřednictvím metod obsažených ve třídě `chess`), které

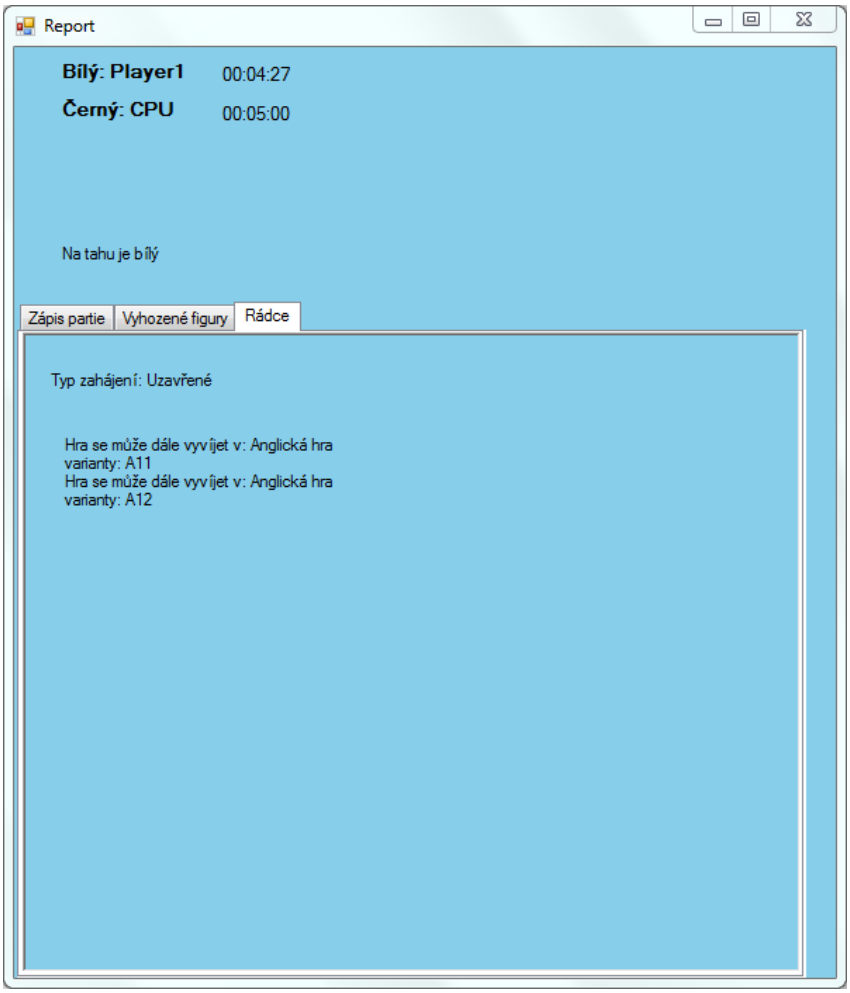
probíhá vůči běhu okna v paralelním vláknu za pomoci komponenty `BackgroudWorker`.

8.2 Report

Informačním prostředníkem ohledně dění na šachovnici je okno Report. Toto okno uživateli prezentuje zápis právě hrané partie, zbývající čas do konce partie, přehled vyhozených figur a informace v rámci šachového rádce (viz obrázek 8.1). Tedy informace o právě hraném zahájení a variantách, do kterých se z právě odehraného průběhu může stále zahájení dostat. Uživatel zde také nalezne seznam svých i protivníkových nechráněných figur, kterým hrozí v příštím tahu sebrání.

Z důležitých metod lze zde nalézt metodu `Help` pro vypsání informací šachového rádce, dále `SetInfo` pro nastavení základních informací o aktuálním průběhu, `DrowDropFigure` pro vykreslení vyhozených figur a `UpdateRecordMatch` pro zápis průběhu partie.

Tohoto okna je podobně jako okna šachovnice využíváno dvojím způsobem při hracím a výukovém módu. Výše popsaný způsob náleží hracímu módu. Výukový mód tohoto okna využívá především pro zadávání úkolů a informování o úspěchu či neúspěchu zvoleného řešení.



Obrázek 8.1: Ukázka informačního okna při zvolené záložce rádce partie

Kapitola 9

Návrh a implementace online šachové nápovědy

Šachy jsou při výuce hodně upovídané a jejich pravidla obsáhlá a plná definic. Aby výukový program nebyl příliš upovídaný a zabýval se pouze aktivní částí výuky a také, aby si uživatel nemusel dohledávat potřebné znalosti z jiného zdroje, je program obohacen o online nápovědu. Tato nápověda ve zkratce zahrnuje všechny potřebné teoretické informace, které by začínající šachista mohl potřebovat.

9.1 Použité technologie

V implementaci komunikace mezi klientem a serverem se využívá technologie *Ajax* společně s *Hypertext Preprocessor*, neboli zkráceně PHP. K interaktivní demonstraci ukázek je využívána technologie *javascript*. Efekty jsou realizovány pomocí javascriptové knihovny *jQuery*.

9.1.1 Ajax

Zkratka AJAX je z anglického pojmenování Asynchronous Javascript And XML. Jedná se o technologii zajišťující asynchronní komunikaci se serverem.

Při užívání webových stránek je velmi často zapotřebí interaktivita. Při návrhu webu, který nevyužívá Ajax technologie, musí být pro každou interaktivní činnost načten celý dokument znovu a znovu zpracován, byť je změna mezi prvotním dokumentem a znovu načteným a již upraveným dokumentem sebemenší. Web pracující s Ajax technologií mění pouze potřebnou část dokumentu a zbytek zůstane neměnný. K tomuto nám poslouží objekt `XMLHttpRequest` prostřednictvím něhož sestavíme HTTP požadavek pouze pro vyžadovaná data. Zatímco server zpracovává odpověď, webový klient dále reaguje na podněty od uživatele.

Mezi nevýhody technologie Ajax spadá fakt, že objekt `XMLHttpRequest` dosud nebyl standardizován W3. Další nevýhodou je, že nelze vytvářet odkazy na jednotlivé dokumenty uvnitř webu a jednotlivé dokumenty nelze ani oindexovat roboty vyhledávačů. V této kapitole bylo čerpáno ze studijní opory: Internetové aplikace (WAP) V. část AJAX [40].

9.1.2 jQuery

JQuery je javascriptovou knihovnou, která dbá na náležitý důraz mezi interakcí javascriptu a HTML. Pomocí jQuery lze jednoduše animovat, zpracovávat události, upravovat CSS, využívat AJAX technologii a mnohé další. Jedná se o open-source volně šiřitelný projekt [15].

9.2 Implementace

Veškerá činnost naprogramována na webovém klientu je řešena skrze javascript. Implementace funkcí je obsažena v souboru `chess.js` a v knihovně jQuery `jquery.js`.

9.2.1 Navigace

Navigace na webu je řešena pomocí vertikálního animovaného menu při levé straně obrazovky. Menu je několika úrovňové, seřazené podle kapitol. Animace při zvolení jedné kapitoly a následné zobrazení zanořených podkapitol jsou řešeny pomocí jQuery. Využívá se funkcí `show` a `hide`.

Volání jednotlivých odkazů probíhá pomocí Ajaxu, který si ze serveru načte potřebná data z náležitých php souborů. Komunikace se serverem prostřednictvím Ajaxu je implementována ve funkcích `getPage` a `getData`. `GetPage` dostává jako parametr zadané URL. Tato funkce předzpracuje URL pro funkci `getData`, která již realizuje přímou komunikaci se serverem.

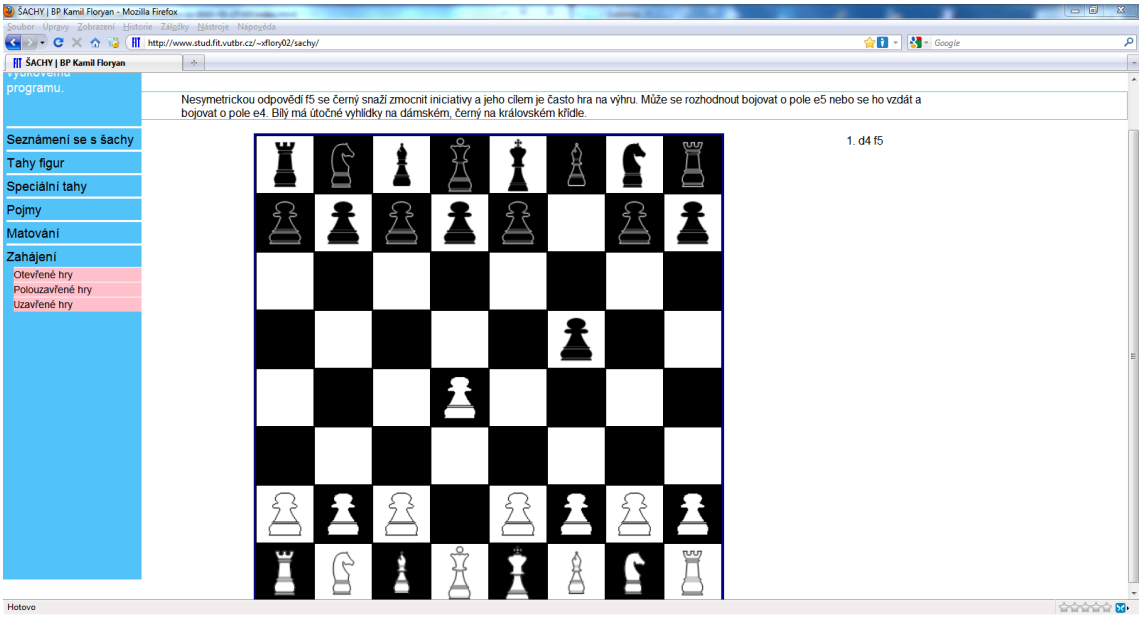
9.2.2 Přehrávání partie

Podobně jako u programu, i zde musíme vyřešit, jak bude reprezentována šachovnice a informace o jednotlivých polích. Šachovnice je v rámci javascriptu realizována jako jednorozměrné pole o 64 prvcích. U každého pole jsou uchovávány příznaky o figuře, kterou je pole obsazeno, respektive zdali je pole prázdné. Pokud je pole obsazeno, je důležitý i druhý příznak, kterým je barva figury.

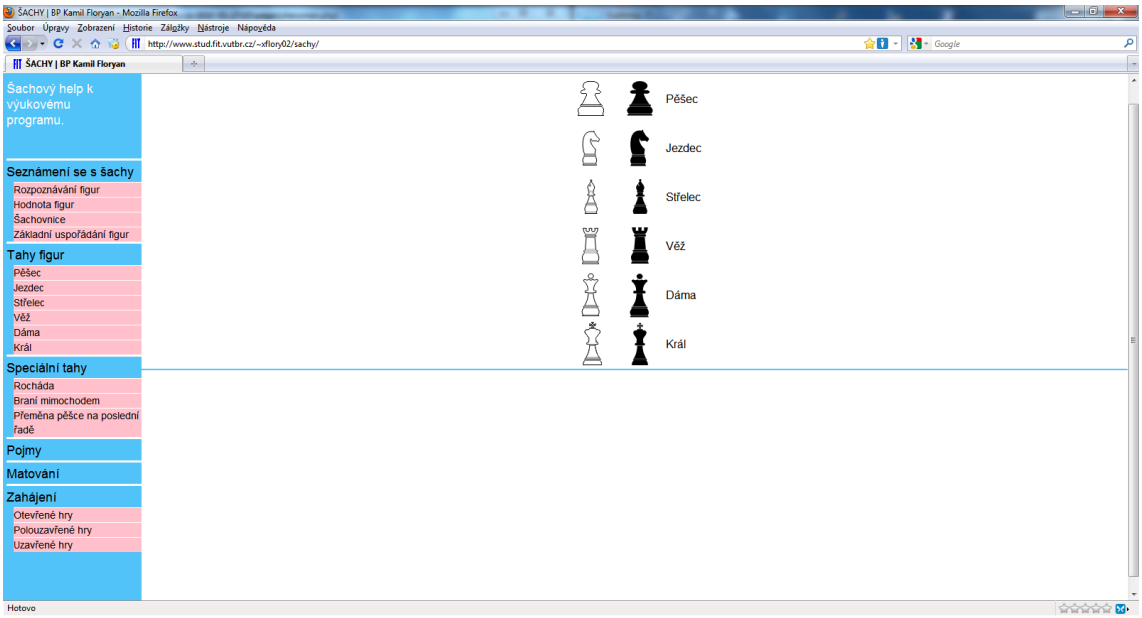
Při načtení šachovnice je provedena prvotní inicializace polí. Po spuštění přehrávání záznamu je spuštěn periodický časovač, který v pevně daný okamžik provede změnu hrací plochy podle načtených dat průběhu partie a následně podle uchovávaných příznaku vykreslí šachovnici.

Prvotní inicializace jsou implementovány ve funkcích `start` a `initChessboard`. Periodický čítač je spouštěn funkcí `play`, která si průběžně převádí textový záznam tahu na souřadnice `xFrom, yFrom, xTo, yTo`. Tyto převedené souřadnice předává funkci `makeMove`.

V javascriptové variantě využívám strojově jednodušeji zpracovatelnou notaci ICCF dále popsanou v 2.3.



Obrázek 9.1: Ukázka přehrávání zahájení v online nápovědě



Obrázek 9.2: Ukázka online šachové nápovědy

Kapitola 10

Závěr

Cílem práce bylo vytvoření programu pro výuku šachu, zaměřující se na různé varianty zahájení a na koncovky. Výsledný program disponuje dvěma spustitelnými módy, a to pro spuštění partie proti počítači (avšak i lidskému protihráči) a spuštění výukové části.

Umělá inteligence využívaná při partii proti programu, je založena na alfabeta prořezávání a uplatňuje několik základních heuristik pro optimalizaci vyhledávání nejlepšího tahu. Dalšími vylepšeními, které by bylo možno v budoucnu doimplementovat jsou pokročilější heuristické metody, z nich některé jsou popsány v kapitole 5, reimplementace abstrakce šachovnice pomocí bitových polí pro značné urychlení generování tahů, připojení databáze koncovek pro efektivní vyhodnocování složitých rozestavění šachovnice pro koncovky až o šesti figurách. Při rozšiřování databáze zahájení, by bylo vhodné zvolit jiných prostředků pro jejich ukládání než stávající formát XML dokumentů.

Výkonnostní ohodnocení programu již bylo částečně vyhodnoceno v kapitole 4.4, při otestování pozičních diagramů. Naměřené ELO bude jistě nadhodnoceno, protože byly analyzovány jen pozice střední hry. Program se jeví nejslabší v zahajovacích fázích partie. Za tento důsledek může obecně sestavená hodnotící funkce, která sice v hodnocení reflektuje rozestavění figur, avšak v hodnocení se nepromítne taktické rozestavění a souhra všech figur (vývoj partie), která je nutná pro dosažení maximálně výhodného postavení důležitého pro snadný přechod do střední fáze partie.

Pro částečné řešení tohoto negativního vlivu aplikace využívá knihovnu zahájení. Toto řešení se jeví jako velice efektivní (podobný princip je využíván i v komerčních programech), ale je podmíněno rozsáhlou databází zahajovacích systémů. Pokud průběh hry nezapadá do žádného zahájení, které by bylo zahrnuto v databázi, jsou tahy voleny opět na základě ohodnocení pozice.

Výuková část obsahuje tři lekce. Všechny lekce disponují několika spustitelnými kombinacemi, které při výuce vnášejí více různorodosti v definování zadaných úkolů. První lekce je pro začínající šachisty – výuka tahů. Druhá a třetí lekce se již zaměřuje na výuku zahájení a koncovek. Při zadávání ukázkových příkladů je kladen důraz na neopakovatelnost předkládaných diagramů. Do budoucna by tato část programu mohla být dále rozšířena o kapitolu pro výuku střední hry. Také by mohla být rozšířena kapitola věnující se koncovkám. Při tomto rozšíření by bylo velmi vhodné k programu připojit již výše zmiňovanou databázi koncovek.

Výuka náročností spadá do kategorie mírně pokročilých šachistů, kteří již překonali úplné začátky a začínají mít trochu šachového vidění. Oproti již existujícím programům aplikace nenabízí stovky různých cvičení a úloh, tak jak nám jej předkládají šachové učebnice. Má aplikace preferuje různorodost úkolů a při jejich zadávání je tedy vnesen prvek

náhody při generování počáteční pozice. Avšak kapitola zahájení je naopak celá vybudována na předdefinované posloupnosti řešení. Ovšem ani zde není množství variant jednotlivých her limitováno konstrukcí programu, ale uživatel prostřednictvím programu získává prostředek pro jednoduchou správu databáze (ve formátu XML), díky kterému si může libovolné varianty zahájení přidávat

Dalším podpůrným prvkem výuky je online šachová nápověda, která obsahuje nezbytné teoretické informace týkající se pravidel šachu.

V technické zprávě jsou popsány stěžejní aspekty návrhu a implementace jednotlivých částí programu. Pro srovnání jsou zde také uváděny již existující šachové programy. Při popisu byl kladen důraz na zhodnocení úrovně inteligence šachového programu a na popis jejich prostředků pro výuku šachu. V samostatné kapitole pak poukazuji na metody využívané při implementaci šachového motoru a u vybraných šachovým motorů jsem provedl také souhrnné srovnání těchto metod, které je naznačeno v tabulce 4.1.

Aplikace byla testována na operačním systému Windows XP, Windows Vista a Windows 7 a ke svému běhu vyžaduje .NET 3.5. Kromě spustitelného souboru `sachy.exe` je vyžadován konfigurační soubor `sachy.exe.config`, který obsahuje informace o uživatelově nastavení programu. Dále program předpokládá pro správnou funkčnost databázi zahájení `zahajeni.xml`.

Internetová nápověda ke své funkčnosti vyžaduje, aby byl v prohlížeči povolen Javascript. Webové rozhraní bylo testováno v prohlížečích Mozilla Firefox 3.6.3 a Google Chrome 4.1.

Tato práce pro mě byla velmi přínosnou. Zpracováním teoretické části týkající se šachu, jsem hlouběji pronikl do principů, na kterých funguje většina existujících šachových programů. Také jsem si rozšířil své povědomí o mnohé další šachové programy či šachové motory, na které jsem při shromažďování materiálu ke své práci narazil. Při implementaci programu jsem se naučil pracovat s rozsáhlou softwarovou platformou Microsoft .NET Framework a rozšířil své zkušenosti s tvorbou GUI aplikace.

Literatura

[1] The Automated Chess Rating Utility [online]. http://www.chessmaniac.com/ELORating/ELO_Chess_Rating.shtml .

[2] BT2450 Tactical Analysis [online]. http://www.fishdev.net/chess/palmchess/bt2450.html .

[3] CEGT [online]. http://www.husvankempen.de/nunn/ .

[4] ChEngine [online]. http://chengine.codeplex.com/ .

[5] Chess Programming [online]. https://chessprogramming.wikispaces.com/ .

[6] Chess Tutor Step 1 [online]. http://www.brothersoft.com/games/chess-tutor-step-1.html .

[7] Clker.com - the royalty free clip art. http://www.clker.com/ .

[8] Crafty: Online technical papers [online]. http://www.cis.uab.edu/info/faculty/hyatt/pubs.html .

[9] ECO : Kniha zahájení [online]. http://www.celysvet.cz/sachy/sachy-eco .

[10] Endgame Tablebases [online]. http://kirill-kryukov.com/chess/tablebases-online/.

[11] Free Vector Silhouettes. http://all-silhouettes.com/ .

[12] Gaia Chess Engine [online]. http://gaiachess.free.fr/ .

[13] How to test a chess engine? [online]. http://lozibaldonedinicola.blogspot.com/2007/08/how-to-test-chess-engine.html .

[14] Icon Archive. http://www.iconarchive.com/ .

[15] jQuery [online]. http://jquery.com/ .

[16] Microsoft Visual Studio [online]. http://www.microsoft.com/cze/msdn/produkty/vstudio/default.mspx .

[17] Muse [online]. http://www.fierz.ch/muse.htm .

[18] Onno Chess Software [online]. http://www.onnochess.com/ .

[19] Ruffian interview [online]. http://www.top-5000.nl/int/ruffian.htm .

[20] Rybka Chess Community Forum [online]. http://rybkaforum.net/ .

[21] Swedish Chess Computer Association [online]. http://ssdf.bosjo.net/ .

[22] World Wide Web Consortium (W3C) [online]. http://www.w3.org/ .

[23] Zappa Chess Engine [online]. https://netfiles.uiuc.edu/acozzie2/www/zappa/ .

[24] 8 x 8 Media AG: Personal Chess Trainer - Chessimo [online]. http://www.chessimo.com/trainer/index.php?lang=en&val=en.

[25] CCRL: Elo rating list [online]. http://www.computerchess.org.uk/ccrl/4040/.

[26] Chess Mentor: Chess Mentor Chess Software [online]. http://www.chessmentor.com/.

[27] ChessBase: Fritz 12 [online]. http://www.chessbase.com/shop/product.asp?pid=467.

[28] FIDE - World Chess Federation: Chess Ratings [online]. http://ratings.fide.com/.

[29] Frayn, C.: Computer Chess Programming Theory [online]. http://www.frayn.net/beowulf/theory.html .

[30] Herejk, I. P.: Základní šachový výcvik začátečníků. http://www.chess.cz/www/mladez/metodika/zakladni-sachovy-vycvik.html.

[31] Hrček, J.: *Nástroj pro reprezentaci a záznam šachových partií*. Bakalářská práce, MUNI-FI, Brno, 2006.

[32] Hurta, M.: *Rukověť trenéra šachu*. Pliska, 1997, iSBN 80-85232-39-1.

[33] IBM Research: Deep Blue [online]. http://www.research.ibm.com/deepblue/meet/html/d.3.1.html.

[34] doc. Ing. František Zbořil: *Základy umělé inteligence Studijní opora*. FIT VUT Brno, 2006.

[35] Kappe, D.: Chess Training Tools 1.5.1. http://linux.softpedia.com/get/GAMES-ENTERTAINMENT/TBS/Chess-Training-Tools-11877.shtml.

[36] Kent, A.; Williams, J.: *Encyclopedia of Computer Science and Technology*. Marcel Dekker Incorporated, 44 vydání, 2001, iSBN 0-8247-2297-3.

[37] Kosek, J.: XML [online]. http://www.kosek.cz/xml/index.html.

[38] Lokasoft: Nalimov Tablebase server [online]. http://www.lokasoft.nl/uk/tbweb.htm.

[39] Millington, I.: *Artificial intelligence for games*. Morgan Kaufmann Publishers, první vydání, 2006, iSBN 0124977820.

[40] Máčel, L.; Kužela, A.; Prof. Ing. Tomáš Hruška, C.: *Internetové aplikace (WAP) V. část AJAX, Studijní opora*. FIT VUTBR, 2007.

[41] Němec, J.: Seriál - Šachové myšlení [online].
http://www.linuxsoft.cz/article.php?id_article=1109.

[42] Rajlich, V.: Rybka 3 [online].
http://products.convekta.com/1569/rybka_3_aquarium.htm.

[43] Řehulka, P.: Počítač hraje šachy [online].
http://www.fi.muni.cz/usr/jkucera/pv109/xrehulka_p109.html, 1999.

[44] Schröder, E.: Programmer Stuff [online].
http://www.top-5000.nl/authors/rebel/chess840.htm#MATE%20THREATS .

[45] Shannon, C.: Programming a Computer for Playing Chess,. *Philosophical Magazine*, , č. 41, 1950.

[46] Shredder: Endgame Database [online].
http://www.shredderchess.com/online-chess/online-databases/endgame-database.html.

[47] Tim Mann's Chess: XBoard and WinBoard [online].
http://www.tim-mann.org/xboard.html.

[48] Turing, A.: Turing Test [online].
http://www.chessgames.com/perl/chessgame?gid=1356927.

[49] Veselý, J.: *Jak hrát šachy?* ADONAI, 2002, iSBN 80-7337-005-0.

[50] YAPOFFRL: EloStat [online]. http://www.compchess.de/html/elostat.html.

Dodatek A

Seznam zahájení

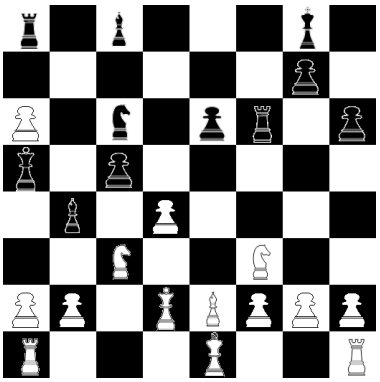
Otevřené zahájení
Střední gambit
Vídeňská hra
Královský gambit
Střelcova hra
Lotyšský gambit
Philidorova obrana
Ruská obrana
Skotská hra
Hra tří jezdců
Hra čtyř jezdců
Uherská obrana a Italská hra
Evansův gambit
Italská hra
Hra dvou jezdců v obraně
Španělská hra
Královským pěšcem
Ponzianiho hra

Polootevřené zahájení
Francouzská obrana
Sicilská hra
Caro-Kannova hra
Skandinávská obrana
Aljechinova hra
Pircova obrana
Nimcovičova obrana
Gründfeldova indická obrana
Albinův protigambit dámského gambitu
Anglická hra
Birdova hra
Blumenfeldův protigambit
Bogoljubova indická obrana
Budapešťský gambit
Dámská indická obrana
Dámský gambit

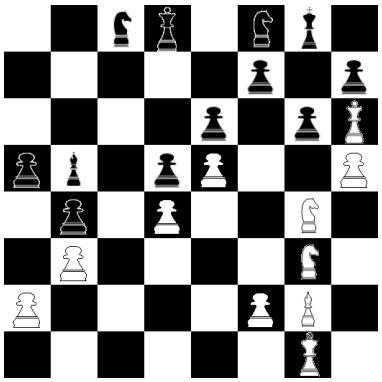
Uzavřené zahájení
Grünfeldova indická obrana
Holandská obrana
Katalánský systém
Keresova obrana
Královská indická obrana
Nimcovičova indická obrana
Obrana Benoni
Rétiho hra
Útok Richter-Veresova
Slovanská obrana
Staroindická obrana
Torreho útok
Trompowského útok
Volžský gambit

Dodatek B

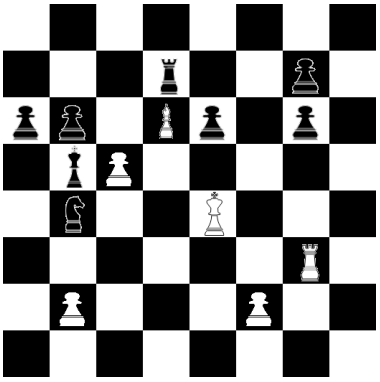
Diagramy pro vyhodnocení ELA



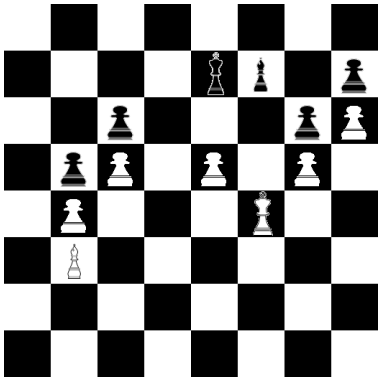
Obrázek B.1: Černý na tahu



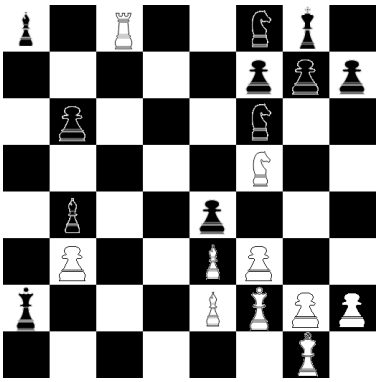
Obrázek B.2: Bílý na tahu



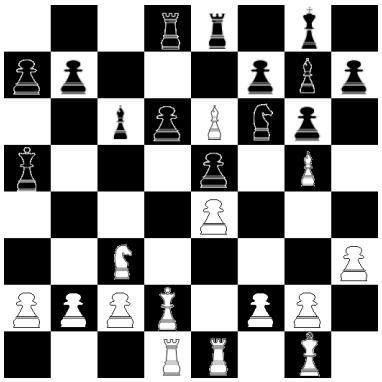
Obrázek B.3: Bílý na tahu



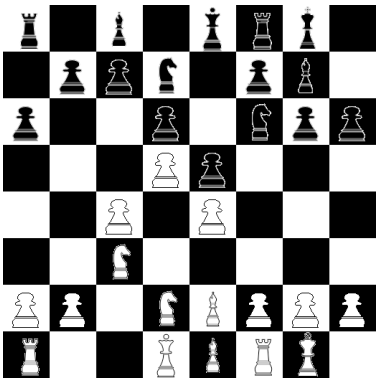
Obrázek B.4: Bílý na tahu



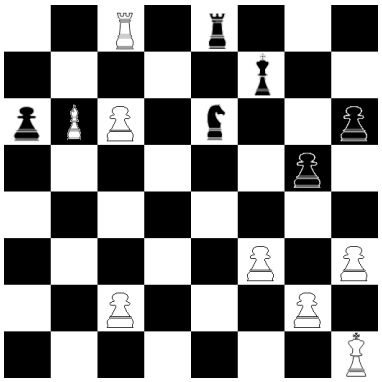
Obrázek B.5: Bílý na tahu



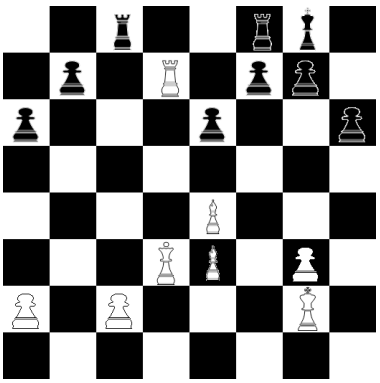
Obrázek B.6: Bílý na tahu



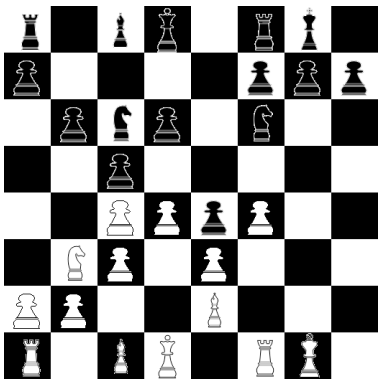
Obrázek B.7: Černý na tahu



Obrázek B.8: Bílý na tahu



Obrázek B.9: Bílý na tahu



Obrázek B.10: Černý na tahu

Dodatek C

Metriky kódu

Ke dni 17.5.2010

Počet řádků celkem:	18 606
Počet řádků kódu:	14 210 (76%)
Počet řádků komentářů:	2 413
Počet souborů:	44

Zbylé řádky jsou prázdné. Pro získaní statistiky projektu byl použit plugin pro Microsoft Visual Studio DPack.
Seznam souborů s vlastním kódem:

About.cs
AddOpening.cs
BaseChessboard.cs
DesignForm.cs
Game.cs
GenerateMove.cs
chess.cs
chessboard.cs
MainForm.cs
Matte.cs
MenuForm.cs
Move.cs
NewGame.cs
OnlineHelpForm.cs
Opening.cs
OpeningMove.cs
ParseNotation.cs
PreferOpenings.cs
Program.cs
Report.cs
RulesForm.cs
SetUserName
Solver.cs
Train.cs
TrainForm.cs
TransformPawnForm.cs