Katedra informatiky
Přírodovědecká fakulta
Univerzita Palackého v Olomouci

BAKALÁŘSKÁ PRÁCE

Krása v šachu

Proč jsou některé trojtažky hezčí než jiné?

2024

Matěj Dluhoš

Vedoucí práce:
Mgr. Tomáš Urbanec, Ph.D.

Studijní program: Informatika,
Specializace: Programování a vývoj software

Bibliografické údaje

Autor:	Matěj Dluhoš
Název práce:	Krása v šachu (Proč jsou některé trojtažky hezčí než jiné?)
Typ práce:	bakalářská práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2024
Studijní program:	Informatika, Specializace: Programování a vývoj software
Vedoucí práce:	Mgr. Tomáš Urbanec, Ph.D.
Počet stran:	33
Přílohy:	elektronická data v úložišti katedry informatiky
Jazyk práce:	český

Bibliographic info

Author:	Matěj Dluhoš
Title:	Beauty in chess (Why some mates in 3 are prettier than others)
Thesis type:	bachelor thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2024
Study program:	Computer Science, Specialization: Programming and Software Development
Supervisor:	Mgr. Tomáš Urbanec, Ph.D.
Page count:	33
Supplements:	electronic data in the storage of department of computer science
Thesis language:	Czech

Anotace

Cílem práce je vytvořit aplikaci, jejíž funkcí je analyzovat uživatelem zadanou šachovou pozici a následně objektivně zhodnotit její atraktivitu. Aplikace je zaměřena na pozice, ve kterých je možný vynucený mat ve třech tazích. Šachovou partii je možné dohrát, popřípadě s dopomocí zobrazeného řešení. Práce je psána v jazyce C# a byla vyvíjena a testována v grafickém rozhraní Arena Chess GUI.

Synopsis

The aim of the thesis is to develop an application that analyses user-defined chessboard formation and objectively evaluates its attractiveness. The application is focused on formations, where a forced checkmate in three moves is possible. The user can play out the formation on their own or with the help of the displayed solution. The thesis is written in C# and was developed and tested using Arena Chess GUI.

Klíčová slova: šachy; algoritmus; atraktivita; šachová estetika

Keywords: chess; algorithm; attractiveness; chess aesthetics

Odevzdáním tohoto textu jeho autor/ka místopřísežně prohlašuje, že celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.

Obsah

1 Úvod

Cílem práce bylo vytvořit šachový engine, který dokáže ze zadané šachové pozice zjistit, zda se v ní nachází vynucený mat ve třech tazích. Pokud je takový mat nalezen, úkolem enginu je dále analyzovat atraktivitu celé sekvence vedoucí k matu a výsledná data předat uživateli v přehledné formě, která nachází rovnováhu mezi pochopitelností pro amatéra a užitečností pro pokročilého.

Mojí hlavní snahou při psaní této práce bylo pokusit se navrhnout systém pro hodnocení atraktivity šachových pozic, který vychází z obecně uznávaných metod pro hodnocení krásy v šachu [1, 7, 9], a současně tyto metody upravit a doplnit dle svých vlastních, subjektivních názorů na tuto problematiku. Zároveň jsem se snažil o vytvoření takového systému pro bodování atraktivity jednotlivých pozic, který je smysluplný, účelný, a přitom si zachovává jednoduchost.

1.1 Motivace

Jakožto amatérský šachista jsem při výběru bakalářské práce velmi uvítal vypsání právě tohoto tématu. Vývoj šachového enginu a jeho následné nástavby pro mě byl zajímavou výzvou, která vedla k mému širšímu porozumění a hlubšímu ocenění šachu jako takového. Proto bych chtěl poděkovat vedoucímu práce Mgr. Tomáši Urbancovi, Ph.D. za svěření mi tohoto tématu.

2 Teorie

V následující kapitole vysvětlím čtenáři stěžejní pojmy a technologie, kterým je nutné porozumět pro celkové pochopení vnitřní funkcionality aplikace.

2.1 Trojtažka

Pojem *trojtažka* je základní stavební kámen této práce. O šachové pozici je možné říct, že je trojtažkou, pokud hráč na tahu může dosáhnout matu ve třech tazích bez ohledu na to, jaké tahy zvolí oponent. Oponent navíc často žádnou možnost volby nemá, v trojtažkách bývá obvykle vnucen do situace, kdy disponuje pouze jedním tahem.

Řešení trojtažek je velmi důležitou disciplínou v oblasti šachové strategie, jedná se o velmi efektivní trénink v rozpoznávání standardizovaných motivů a taktik, které jsou právě v trojtažkách prominentní a umění je v šachových pozicích najít výrazně zvyšuje hráčovu šachovou dovednost.

V pozdějších kapitolách se podíváme na to, jakým způsobem jsem trojtažky hledal, jak jsem ukládal data o nich, a především na to, podle jakých kritérií jsem hodnotil jejich atraktivitu.

2.2 FEN string

Abychom mohli začít řešit jakýkoliv šachový problém strojově, je nutné vytvořit efektivní datovou strukturu pro vyjádření konkrétní šachové pozice. Tento problém vyřešili v průběhu 19. a 20. století David Forsyth a Steven Edwards, výsledkem jejich práce je takzvaná Forsyth-Edwards Notace [2] a zní vycházející *FEN string*.

Samotný FEN string se skládá ze 6 polí, která jsou oddělena mezerou.

- 1. pole reprezentuje rozestavení figurek na šachovnici. Pole má 8 částí oddělených lomítkem. Tyto části reprezentují řádky od osmého po první, každý řádek je reprezentován takto: Pokud je na políčku figurka, zapíšeme její písmennou zkratku, pokud ne, zapíšeme číslem počet souvislých prázdných políček a daná políčka přeskočíme, poté opakujeme. Velká písmena označují bílou barvu, malá písmena barvu černou. Zkratky figurek jsou následující:
  - pěšec jako P/p,
  - věž jako R/r,
  - kůň jako N/n,
  - střelec jako B/b,
  - dáma jako Q/q,
  - král jako K/k.
- 2. pole udává barvu hráče, který je právě na tahu. Značí se písmenem *w* nebo *b*.
- 3. pole nese informace o možných rošádách. V úplné formě *KQkq*, stejně jako u reprezentace figurek jsou velká písmena rošádami bílého hráče a malá písmena rošádami hráče černého. Písmeno K/k značí rošádu malou, tedy na stranu krále, písmeno Q/q rošádu velkou, na stranu dámy. Pokud některá z rošád dostupná není, písmeno je z pole vyjmuto a pole zkráceno. Pokud nelze provést žádnou rošádu, pole obsahuje znak -.
- 4. pole obsahuje políčko značící možný en passant (např. *b3*), pokud v dané pozici žádný není, je zde znak -.
- 5. pole se nazývá *halfmove clock*. Jde o počítadlo, které je nastaveno na 0 po tahu pěšcem nebo tahu, kdy proběhlo braní některé figurky. Při ostatních tazích je inkrementováno. Toto počítadlo existuje kvůli *remízovému pravidlu 50 tahů*, tedy když počítadlo dosáhne 50, hra je prohlášena za remízu.
- 6. pole se nazývá *fullmove clock*. Toto počítadlo slouží k počítání uplynulých celých tahů a je inkrementováno po každém tahu černého hráče.

Níže můžete vidět příklad FEN stringu, tento konkrétní reprezentuje standardní startovní pozici.

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
```

2.3 UCI protokol

The Universal Chess Interface Protocol (UCI) [3] je komunikační protokol, který umožňuje komunikaci mezi šachovými enginy a uživatelskými rozhraními. Komunikace probíhá výhradně pomocí standardního vstupu a výstupu, obě strany přitom mají k dispozici seznam příkazů, jejichž zasíláním a čtením se vzájemně dorozumívají. Tento seznam je ve své kompletní verzi značně rozsáhlý a pokrývá mnohem hlubší funkcionalitu, než můj engine potřebuje, v této kapitole budou proto uvedeny pouze příkazy, které byly použity při psaní této práce.

- GUI -> Engine:
  - `uci`
    Nařídí enginu, aby používal UCI. Odpovědí je identifikace enginu ve formě jeho názvu a jména autora.
  - `isready`
    Synchronizační prvek, kterým se dá zařídit, aby GUI počkalo na engine, pokud ještě neskončil s výpočtem.
  - `ucinewgame`
    Oznamuje enginu, že následující pozice začíná novou hru, pokud dosud analyzoval pozice z hry jiné. Slouží k resetu interních nastavení enginu.
  - `position [fen | startpos] moves ...`
    Nařídí enginu nastavit danou pozici a na ní následné tahy specifikované za `moves`. V případě možnosti `fen` je nutné následně zadat konkrétní FEN string.
  - `go`
    Standardně začíná výpočet na nastavené pozici. Můj engine začíná výpočet už při zadání pozice, na tento příkaz pouze odpovídá příkazem `bestmove`.
  - `stop`
    Zastavuje výpočet enginu a vrací momentální dosažený `bestmove`. Vzhledem k tomu, že délka výpočtů mého enginu je relativně malá, příkaz není potřeba využívat.
  - `quit`
    Ukončuje komunikaci a program.

- Engine -> GUI:
  - `id`
    Je odpovědí na `uci`. Vrací `id name` a `id author`.
  - `uciok`
    Dává GUI na vědomí, že je připraven v režimu UCI.
  - `readyok`
    Tímto odpovídá engine na `isready`, když je připravený na další vstup.
  - `bestmove`
    Jedná se o tah, který engine zahraje v pozici, která byla výsledkem tahu hráče.
  - `info string`
    Obvykle slouží k ukazování informací o počítaných tazích. Můj engine tento příkaz využívá k ukazování nápovědy a chybových hlášení.

Tyto příkazy na sebe vzájemně reagují v takzvaném *UCI loopu*, což je nekonečný cyklus zprostředkovávající obousměrnou komunikaci mezi entitami popsanými výše. K podrobnému popisu UCI loopu se vrátím v pozdější kapitole. Názornou ukázku komunikace mezi mým enginem a uživatelským rozhraním ukazuje obrázek 1.

```
uci
id name Dluhos Thesis Engine
id author Matej Dluhos
uciok
isready
readyok
position fen 5r1k/2r2P2/3qbRPp/1p1n4/p2pB3/4p1NP/PP6/6RK w - - 0 1 moves g6g7
info string g6g7 | h8g7 | g3h5 | g7h8 | f6h6
go infinite
bestmove h8g7
quit
```

Obrázek 1: Příklad začátku komunikace UCI protokolu.

2.4 Minimax algoritmus

Minimax algoritmus [4] je algoritmus využívaný v kompetitivních hrách pro 2 hráče. Jeho cílem je simulovat rozhodovací proces obou hráčů, kdy se každý hráč snaží maximalizovat svou šanci na vítězství. Z vnějšího pohledu na problém se tedy jeden hráč snaží skóre hry maximalizovat a druhý jej minimalizovat. Je důležité dodat, že algoritmus pracuje rekurzivně; z hlediska stromu, který vytváří, tedy ohodnocuje své uzly reprezentující pozice odspodu.

2.4.1 Příklad

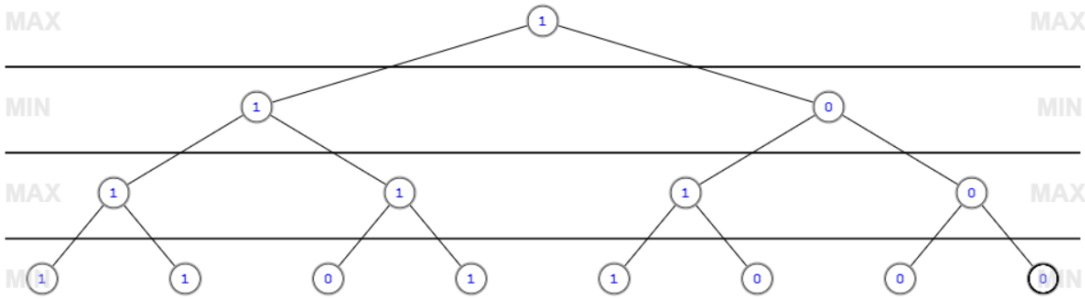
Algoritmus nyní demonstruji na jednoduchém příkladu. Naším cílem je zjistit, zda se v zadané šachové pozici nachází dvojtažka. Pro názornost budeme předpokládat, že oba hráči mají v jakýkoliv moment ve hře k dispozici právě 2 tahy. Minimax algoritmus prochází tahovým stromem stejným způsobem jako jiné *Depth-First Search* algoritmy, nejprve tedy zjišťuje ohodnocení listu, to pošle rodiči a algoritmus pokračuje k dalším potomkům. Rodič se chová rozdílně podle toho, jakého hráče reprezentuje. Je-li to tah hráče maximalizujícího, rodič se snaží ponechat si nejvyšší skóre obdržené od některého z jeho potomků. V případě hráče minimalizujícího se rodič snaží naopak mít skóre co nejnižší. V moment, kdy jsou všichni potomci vyhodnoceni, rodič předává své skóre svému rodiči a algoritmus pokračuje stejným způsobem až ke kořeni stromu.

Samotné ohodnocení listů stromu se liší v závislosti na typu hry, kterou algoritmus řeší. Pro hry, kde je výsledkem číselné skóre, může hodnota listů nabývat nejrůznějších číselných hodnot. V našem případě se jedná o hru, kterou lze jedině vyhrát, prohrát, nebo remízovat. V tomto zjednodušeném příkladu se omezíme pouze na možnost výhry a prohry, číselně tedy 1 a 0.

Na obrázku 2 můžeme vidět výsledný strom po průchodu minimax algoritmem. Ze dvou tahů, které má začínající hráč k dispozici, vede jeden k vynucenému matu ve dvou tazích (levý podstrom kořene), zatímco druhý tah matem nekončí, pokud minimalizující hráč zahraje správný tah.

2.4.2 Nedostatky

Je nutné podotknout, že výše popsaný příklad je velmi zidealizovanou verzí hledání matu. Minimalizující hráč sice často má k dispozici pouze jeden nebo několik tahů, zato hráč maximalizující může disponovat desítkami tahů v každé úrovni stromu. Rozšíření problému z dvojtažky na trojtažku navíc časovou náročnost výpočtu zvyšuje exponenciálně. Minimax algoritmus ve své základní podobě takový problém řeší řádově desítky sekund, což je nepraktické a zároveň neaplikovatelné pro případné hledání matu do větší hloubky. V následující sekci se proto podíváme, jak tento algoritmus zefektivnit.

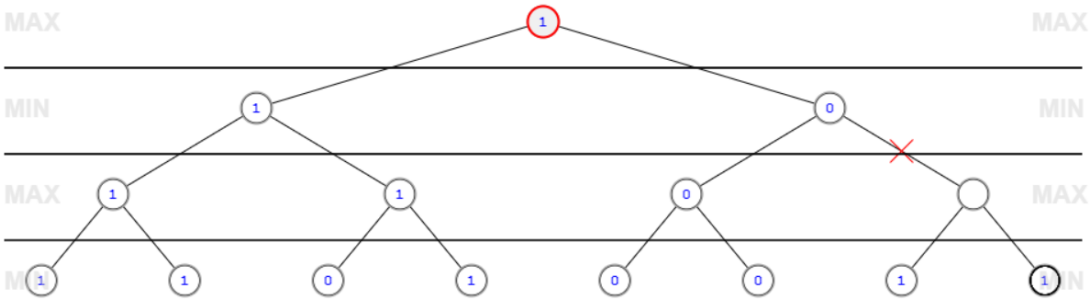


Obrázek 2: Minimax algoritmus pro zjednodušené hledání dvojtažky [5].

2.4.3 Tree pruning

Pro zrychlení výpočtu algoritmu využijeme *tree pruningu* [6], což je způsob, jak zredukovat počet navštívených uzlů při průchodu stromem. Cílem je navštívit co nejmenší počet uzlů, a přitom nalézt všechny možné cesty k matu.

Pokud se algoritmus nachází v uzlu maximalizujícího hráče, nezbývá mu nic jiného, než navštívit uzly všech jeho potomků. I kdyby některý potomek končil v přijímajícím stavu, ostatní, zatím nenavštívené uzly musí být také prohledány, protože mohou také vést k matu a algoritmus nesmí vynechat žádnou správnou cestu. V případě, že se algoritmus nachází v uzlu hráče minimalizujícího, ale máme možnost ořezat velkou část podstromů tohoto uzlu. Algoritmus postupně vyhodnocuje potomky uzlu a v momentu, kdy jeden z jeho potomků vrátí zamítající stav (potomek nevede k matu), další potomky už nemá smysl vyhodnocovat, protože i kdyby některé z nich končily matem, minimalizující hráč si zvolí onu nalezenou cestu nevedoucí k matu. Příklad podobný tomu z minulé sekce obohacený o tree pruning ilustruje obrázek 3.



Obrázek 3: Minimax algoritmus s využitím tree pruningu [5].

3 Hodnocení atraktivity

V moment, kdy je trojtažka nalezena, přichází na řadu analýza atraktivity šachových pozic obsažených v dané trojtažce. Každá trojtažka obsahuje 5 stavů. Stavem v tomto kontextu nazývám strukturu složenou ze šachové pozice před tahem, tahu zahraném v této pozici a z pozice po zahrání tahu. Motivy, ze kterých se výsledné hodnocení skládá, jsou hledány a vyhodnocovány pouze pro jim specifické stavy. Některé motivy se navíc nezabývají žádným konkrétním stavem, posuzují trojtažku jako celek. Trojtažky jsou hodnoceny bodovým systémem zdola ohraničeným nulou (nejnižší hranice atraktivity). Systém shora ohraničen není. Bodování jednotlivých motivů bylo motivováno veřejným vnímáním jejich atraktivity, odbornými texty [7] zmíněnými níže v této kapitole a mým subjektivním pohledem na věc. Obrázky příkladů této kapitoly byly vytvořeny s použitím stránky *lichess.org* [8].

3.1 Přítomnost kratšího matu

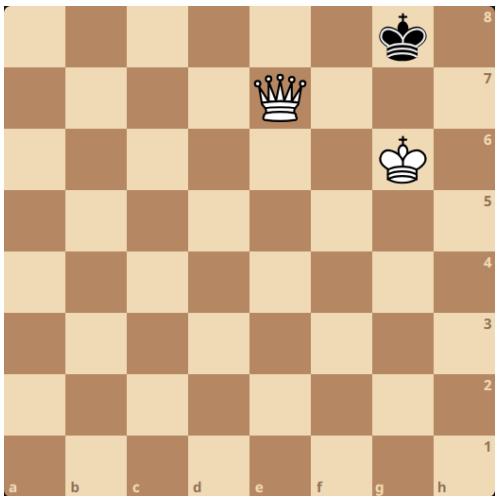
Před popisem motivů zvyšujících atraktivitu trojtažek zmíním jeden, který jim na kráse ubírá. Harold Osborne, zakladatel periodika British journal of aesthetics, se ve svém článku [9] zmiňuje o kráse v korektnosti trojtažky, konkrétně korektnosti v tom smyslu, kdy nejkratší sekvence dané trojtažky je délky 5 tahů, tedy právě 3 tahy začínajícího hráče. Možnost vyřešit trojtažku v menším počtu tahů trojtažce citelně ubírá na kráse, signalizuje zbytečnost, nepřesnost hráče.

Při bodování jsem bral v potaz to, zda se v trojtažce nachází mat v jednom tahu nebo ve dvou. Při nepřítomnosti kratšího matu přirozeně trojtažka body neztrácí. V případě přítomnosti matu v jednom tahu trojtažka ztrácí 15 bodů, mat ve dvou tazích ztrácí trojtažce bodů 10. Příklad je uveden na obrázku 4.

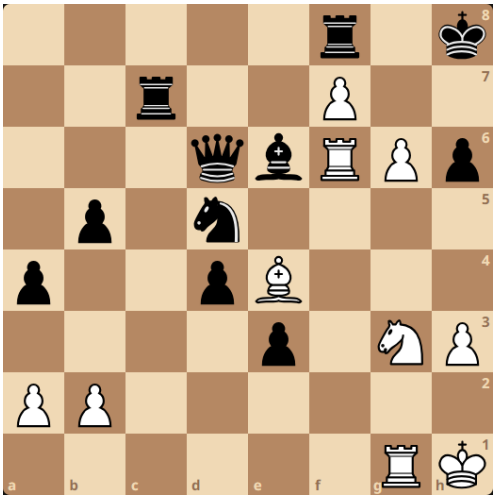
3.2 Výhra s méně materiálem

Výhra z pozice oslabení (často jen `WLM`) je považována za zajímavý zvrat napříč sportovním světem i mimo něj, v šachu tomu není jinak. Záleží však na tom, jak velké oslabení je.

Výpočet tohoto motivu je jednoduchý, stačí sečíst materiál oponenta a od něj odečíst materiál svůj. Pokud je výsledkem kladné číslo, získali jsme bodové ohodnocení tohoto motivu. Pokud ne, hráč není v oslabení a výsledkem je 0. Pro jednoznačnost zmíním, že můj engine používá standardní hodnocení figurek, tedy formát `1/3/3/5/9` (v pořadí pěšec, kůň, střelec, věž, dáma).



Obrázek 4: Přítomnost kratšího matu ve trojtažce (`e7e8`), -15 bodů.



Obrázek 5: Trojtažka bílého, Výhra s méně materiálem, 9 bodů.

3.3 Obětování materiálu

Obětování materiálu s očekáváním méně nebo žádného materiálu nazpět za účelem poziční nebo jiné výhody v šachové partii může být atraktivní motiv, který je často pastí vedoucí k matu proti méně zkušeným hráčům.

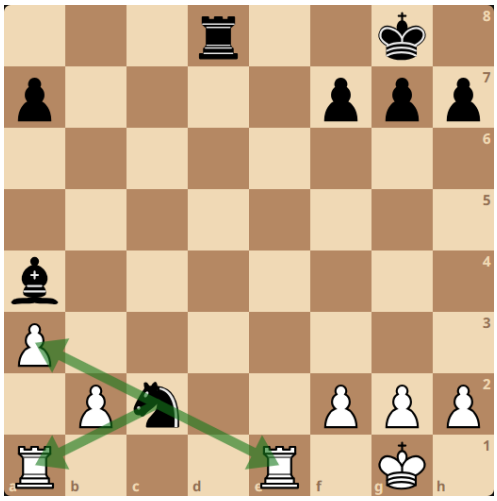
Výpočet hodnocení sestává z hodnoty obětované figurky a hodnoty figurky, kterou obětovaná figurka sebrala. Pokud nesebrala žádnou, přiřadíme na její místo 0. Výsledek je získán odečtením hodnot. Motiv je ilustrován na obrázku 6.

3.4 Vidlička

Tah, kterým se hráč dostane do pozice, kdy ohrožuje více cílů jednou figurkou, se nazývá *vidlička*. Soupeř v takovém případě většinou nemá k dispozici tah, kterým by zajistil neztracení materiálu. Bodování probíhá následovně:

- Typ útočníka:
  - Kůň: +4 body,
  - Pěšec: +3 body,
  - Ostatní: +1 bod.
- Typ cíle (pro každý cíl):
  - Král nebo dáma: +3 body,
  - Věž, kůň nebo střelec: +2 body,
  - Pěšec: +0,5 bodu.
- Pro každý cíl (od cíle č. 3) +2 body.

Obrázek 6: Obětování dámy, 8 bodů.



Obrázek 7: Vidlička, 10,5 bodu.

3.5 Vazba

Situaci, kde dáma, věž nebo střelec ohrožuje figurku soupeře, za kterou se nachází figurka s větší hodnotou, nazýváme *vazba*. Pokud je figurka s větší hodnotou král, můžeme o vazbě říct, že je absolutní, protože ohrožená figurka nemůže uhnout útoku kvůli potenciálnímu vystavení krále šachu. V opačném případě je vazba relativní. Vazba je bodována takto:

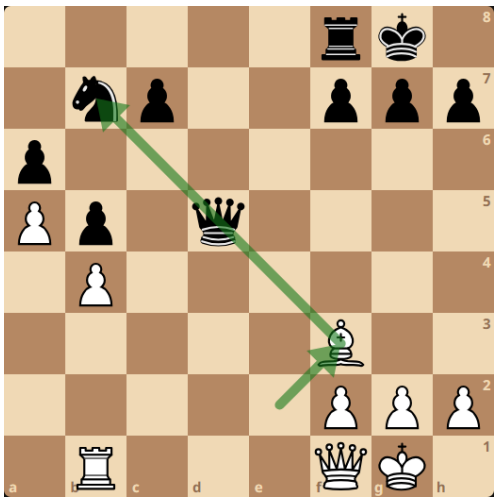
- Absolutní vazba: 10 bodů,
- Relativní vazba dámy nebo věže: 5 bodů,
- Relativní vazba koně nebo střelce: 3 body.

3.6 Špíz

Motiv, který je logickým opakem vazby, nazýváme *špíz*. Figurka, kterou hráč ohrožuje, má v tomto případě větší hodnotu než figurka za ní. Soupeř je tedy nucen zachránit cennější figurku a dovolit soupeři sebrat tu méně cennou. Špíz může, stejně jako vazba, být absolutní nebo relativní. Bodování vypadá obdobně:

- Absolutní špíz: 10 bodů,
- Relativní špíz dámy nebo věže: 5 bodů,
- Relativní špíz koně nebo střelce: 3 body.
- Relativní špíz pěšce: 2 body.

Obrázek 8: Relativní vazba, 5 bodů.



Obrázek 9: Relativní špíz, 5 bodů.

3.7 Rentgen obrana

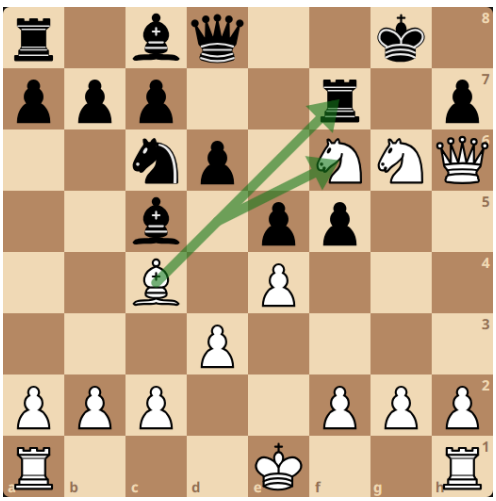
Rentgen obrana je taktický motiv, kde figurka hráče chrání jinou hráčovu figurku před útokem figurky, která se nachází mezi nimi. Pokud se obě figurky chrání navzájem, můžeme mluvit o *dvojité rentgen obraně*, v tomto hodnocení budeme však tuto situaci považovat za 2 instance motivu rentgen obrana. Tento motiv je subžánrem motivu *rentgen* [10], já se zaměřím pouze na zmíněný menší celek. Bodování je prosté, každá nalezená instance přidává 5 bodů.

3.8 Odhalený útok

Odhalený útok, často jen odhalení, je možnost útoku figurky, která se objeví po tahu figurky jiné. Tento objev hráči umožňuje jedním tahem zároveň dosáhnout nějakého cíle pohybem jedné figurky a ohrozit soupeře nepřímým útokem. Soupeř často nemá jak reagovat na obě hrozby současně a proto buď ztrácí materiál, nebo se dostává do nevýhodného postavení ve hře. Odhalený útok je bodován následovně:

- Odhalený útok krále: 10 bodů,
- Odhalený útok dámy: 8 bodů,
- Odhalený útok věže: 5 bodů,
- Odhalený útok koně nebo střelce: 3 body,
- Odhalený útok pěšce: 2 body.

Obrázek 10: Dvojitá rentgen obrana, 10 bodů.



Obrázek 11: Odhalení, 5 bodů.

3.9 Šach z šachu

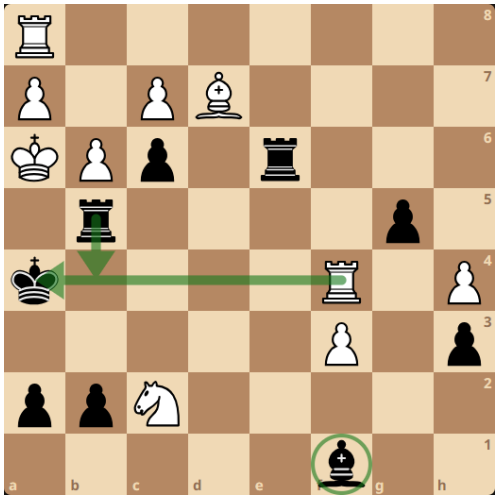
Šach z šachu představuje užitečný nástroj k převrácení iniciativy na šachovnici. Lze prohlásit, že byl proveden tento motiv, pokud z pozice, ve které byl hráč v šachu, zahrál tah šachující oponenta. Hráč tak jedním tahem vyřeší problém ve své obraně a zároveň dostává možnost zahájit vlastní útočnou taktiku. Tento motiv je často doprovázený motivem odhalený útok. Šach z šachu je bodován jednoduše, pokud je v pozici přítomen, přidáváme 10 bodů.

3.10 Dušený mat

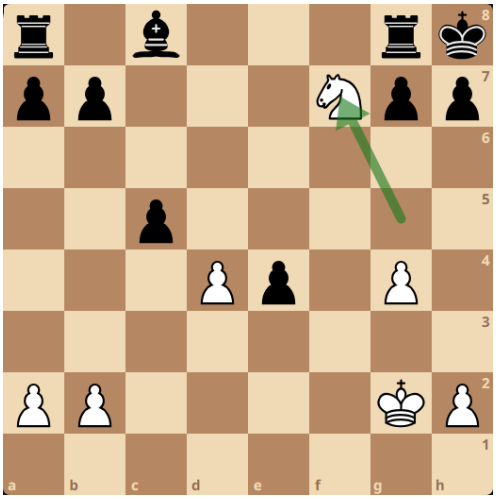
Tento motiv je speciální v tom, že je jako jediný z mého výběru vyhodnocován pouze v posledním stavu trojtažky, tedy u tahu vedoucímu k matu. *Dušený mat* [11] nastává, pokud hráč matuje koněm oponenta, jehož král není schopen uhnout kvůli kompletnímu obklopení figurek jeho vlastní barvy. Není, myslím, potřeba uvádět, že se jedná o kuriozitu, jedná se však o kuriozitu velmi atraktivní. Ačkoliv tedy není pravděpodobnost setkání se s tímto motivem velká, jeho přítomnost atraktivitu trojtažky znatelně posílí. Dušený mat je bodován 20 body.

3.11 Povýšení

Jak z názvu vypovídá, motiv *povýšení* nastává v situaci, kdy jeden z hráčů povýší svého pěšce. Takový tah ve většině šachových partií signalizuje klíčový moment ve hře a v tomto bodovacím systému je proto považován za velmi atraktivní. Ještě zajímavějším se motiv stává v případě povýšení na jiný typ figurky než dámu, tedy *minoritní povýšení*. Povýšení je hodnoceno 15 body, minoritní povýšení 20 body.



Obrázek 12: Šach z šachu, 10 bodů.



Obrázek 13: Dušený mat, 20 bodů.

3.12 Zpětný tah

Zpětný tah, často také *obrátka*, je sekvence dvou pohybů figurky, kterou hráč táhne za nějakým účelem na určenou pozici, a tahem druhým toutéž figurkou táhne na políčko, odkud původně vyrazila. Tyto dva tahy nemusí nastat okamžitě po sobě. Zpětný tah je vyhodnocován dvakrát, ve stavu prvního i druhého tahu. Bodování je rozhodováno dle účelu:

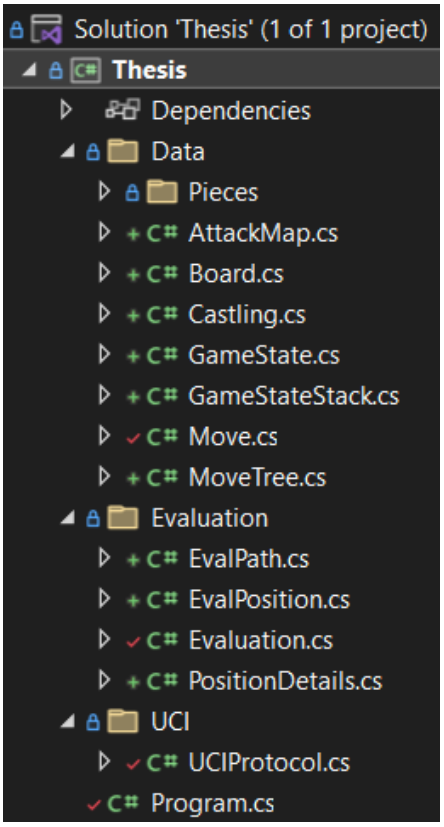
- Základní hodnota: 2 body,
- Za účelem šachování soupeře: +5 bodů,
- Za účelem sebrání figurky: +3 body,
- Za účelem ohrožení figurek soupeře: +1 bod za každou figurku.

Obrázek 14: 1. tah obrátky, 10 bodů.

Obrázek 15: 2. tah obrátky, 7 bodů.

4 Programátorská příručka

Projekt byl vyvíjen v prostředí Visual Studio 2022. Hlavním důvodem upřednostnění tohoto typu před webovou aplikací je existence standardizovaných grafických rozhraní pro šachové enginy, vývoj vlastního rozhraní by tedy byl redundantní. Projekt byl vyvinut a testován za použití grafického rozhraní *Arena Chess GUI* [12], zamýšlené použití projektu je společně s tímto rozhraním. Engine by měl být kompatibilní s jakýmkoliv rozhraním podporujícím UCI protokol, funkčnost enginu na jiných rozhraních však pro nemožnost otestování není zaručena. Struktura projektu je k nahlédnutí na obrázku 16.



Obrázek 16: Struktura projektu.

4.1 Útočné mapy

Šachovnice je reprezentována 2D seznamem políček, která následně obsazují jednotlivé figurky. Každá figurka nese kromě informací o sobě (barva, typ, pozice) také vlastní *útočnou mapu* [13]. Tato mapa je velmi podobná samotné šachovnici, její políčka však nesou pouze pravdivostní hodnotu o tom, zda dané políčko figurka ohrožuje. Útočné mapy tvoří jádro systému pro výpočet veškeré šachové logiky enginu. Útočné mapy musí být správně modifikovány při každém přidání nebo odebrání figurky, nebo při pohybu figurky po šachovnici. Pro maximální efektivitu těchto výpočtů jsem vytvořil pro engine algoritmus, který vyhledá a aktualizuje pouze útočné mapy těch figurek, které je potřeba změnit.

4.2 Reprezentace tahů

Engine a grafické rozhraní se mezi sebou potřebují dorozumívat, každý ale používá jinou reprezentaci tahů. Rozhraní využívá dlouhé algebraické notace [14], engine ale pro své výpočty potřebuje notaci schopnou uložit větší množství informací. Různé části enginu využívají různé notace, engine má k dispozici nástroje pro překlad mezi nimi. Obě notace vypadají následovně:

- GUI:
  - Základní tah: `[odkud] + [kam]`.
    - Příklad: `e2e4`.
    - Rošáda je považována za základní tah krále.
  - Povýšení: `[odkud] + [kam] + [druh povýšení]`.
    - Příklad: `d7d8q`.
    - Druh povýšení je malé písmeno bez ohledu na barvu.
- Engine:
  - Základní tah: `[odkud] + [kam] + [braná figurka]`.
    - Příklad: `e2e4` , `e3f5Q`.
    - U brané figurky záleží na velikosti písmena.
    - Pokud tah není braním figurky, místo písmena je mezera.
  - Rošáda: `[odkud] + [kam] + [S/L]`.
    - Příklad: `e1g1S`, `e8c8L`.
    - Uvádí se `L` v případě velké rošády, `S` v případě malé.
  - En passant: `[odkud] + [kam] + E`.
    - Příklad: `a4b3E`.
  - Povýšení braním: `[odkud.sloupec] + [kam.sloupec] + [braná figurka] + [povýšení] + P`.
    - Příklad: `bcRqP`.
  - Povýšení jinak: `[odkud.sloupec] + [kam.sloupec] + mezera + [povýšení] + P`.
    - Příklad: `ee QP`.

4.3 Řazení tahů

Při hledání matu pomocí minimax algoritmu s integrací tree pruningu (viz 2.4) je potřeba pro každou pozici, kterou algoritmus prochází, vygenerovat seznam všech možných tahů. Algoritmus poté rekurzivně pracuje na každém tahu, dokud nedorazí ke konci cesty nebo není ořezán.

Praktika, která zásadně zrychluje průchod algoritmu stromem, se nazývá *řazení tahů* [15]. Jedná se o vzestupné řazení podle toho, jak pravděpodobné je, že tahy povedou k matu ve třech tazích. Algoritmus tedy začíná vždy těmi nejperspektivnějšími tahy, což v případě minimalizujícího hráče téměř vždy vede k brzkému ořezání stromu. Mnou zvolené bodování je prosté, ale efektivní, vypadá takto:

- Tah vede k šachu: +1000 bodů,
- Tah je braním figurky:
  - Dáma: +900 bodů,
  - Věž: +500 bodů,
  - Kůň nebo střelec: +300 bodů,
  - Pěšec: +100 bodů.

4.4 Implementace UCI protokolu

Hlavním pilířem implementace UCI protokolu je takzvaný *UCI Loop* [16]. Jedná se o funkci, která v nekonečném cyklu čte vstup uživatele, zpracovává jej a následně podle významu vstupu reaguje. Reakcí může být předání vstupu enginu k výpočtu, nastavení pozice, zastavení výpočtu nebo ukončení programu. UCI Loop ale musí zůstat responzivní pro uživatele i během výpočtu, který může trvat při komplexních pozicích až 15 sekund. Zde přichází na řadu *multithreading*. Nastalý problém jsem vyřešil tím, že jsem na hlavním vláknu nechal běžet UCI Loop, který se stará o zpracovávání uživatelského vstupu, a vytvořil jsem vlákno nové, kterému je v případě potřeby přiřazen výpočet enginu.

4.5 Hodnocení atraktivity

Po tom, co algoritmus vyhodnotí uživatelem specifikovanou pozici a nalezne všechny možné cesty trojtažky, přichází na řadu analýza atraktivity trojtažky. Je třeba dodat, že generování informací o atraktivitě proběhne pouze v případě, že algoritmus potvrdí pozici jako trojtažku.

Algoritmus jako první vyhodnocuje WLM (viz 3.2). Důvodem je to, že tento motiv zůstává stejný bez ohledu na vyhodnocovanou cestu, protože závisí pouze na startovní pozici, kterou má každá cesta stejnou. Následně se postupně začínají vyhodnocovat jednotlivé cesty.

Jak už jsem zmiňoval v sekci 3, každá cesta se skládá z pěti stavů. Při analýze každé cesty začne algoritmus s motivem zpětný tah (viz 3.12), jedná se totiž o jediný motiv spojený s více stavy. Následně začíná algoritmus procházet postupně jednotlivé stavy a hodnotit motivy s nimi spojené. Dušený mat (viz 3.10) je jako jediný z dosud nezmíněných motivů vyhodnocován pouze ve finálním stavu, všechny ostatní jsou hodnoceny v každém ze stavů 1–4.

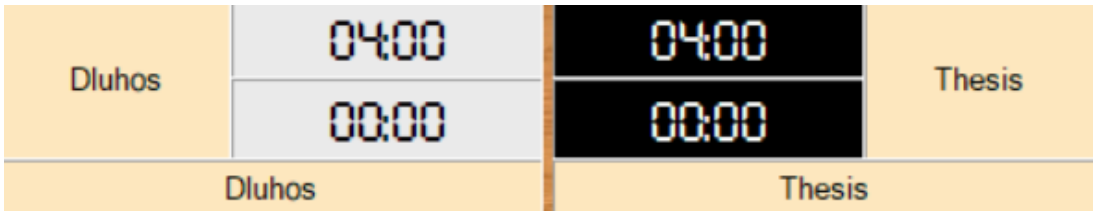
Po ohodnocení každé cesty musí program předat informace uživateli. Toto je realizováno pomocí samostatného souboru, do kterého je celková evaluace generována. Soubor `eval.txt` je k nalezení na adrese `Thesis/bin/Release/net7.0`. Podrobnou ukázku prezentace dat v souboru popíšu v následující kapitole.

5 Uživatelská příručka

V této sekci uživatele provedu instalací enginu do uživatelského rozhraní a následně detailně vysvětlím průběh hry a možnosti uživatele během ní.

5.1 Instalace do uživatelského rozhraní

Poté, co sestavíme program v terminálu příkazem `dotnet publish` ve složce `engine` a nainstalujeme grafické rozhraní Arena Chess GUI, v rozhraní zamíříme do `Engines -> Install New Engine` a vybereme spustitelný soubor `Thesis.exe`, nacházející se na adrese `Thesis/bin/Release/net7.0`. Pokud se engine automaticky nenačte, načteme engine manuálně v `Engines -> Load Engine`, kde vybereme engine s názvem `Thesis`. V tento moment by měl engine být připravený k použití, indikátorem je název enginu na některé ze stran ukazatele času (viz obrázek 17).



Obrázek 17: Časomíra po načtení enginu.

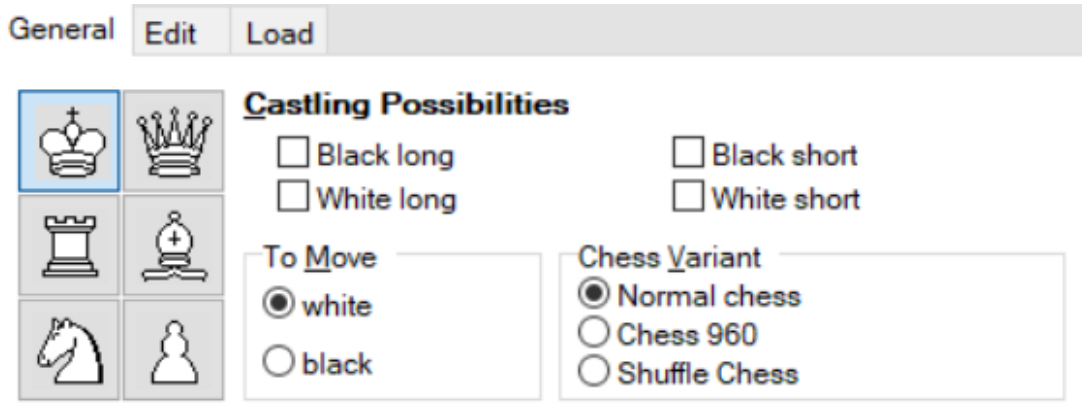
5.2 Načtení pozice

Po instalaci a načtení enginu musí uživatel zadat pozici, kterou chce analyzovat. Pozici lze nastavit v `Position -> Set-up a Position`. V okně pro nastavení pozice (viz obrázek 18) má uživatel několik možností, jak pozici nastavit.

Uživatel má možnost pomocí tabu `General` pozici zadat manuálně vybráním typu figurky a jejím postavením na šachovnici pomocí kliknutí na jedno z políček. Levé tlačítko myši vytvoří bílou figurku, pravé černou.

Mnou preferovaný a silně doporučovaný způsob načítání pozic je pomocí tabu `Load`. Uživateli zde stačí zadat platný FEN string (viz 2.2), šachovnice a její nastavení se nastaví podle něj.

Poslední možností, sloužící hlavně k modifikaci už nastavené pozice, je tab `Edit`. Uživatel zde má k dipozici nástroje pro vertikální a horizontální zrcadlení šachovnice, prohození barvy všech figurek a možnost posunout všechny figurky jakýmkoliv směrem o libovolný počet řádku nebo sloupců. Funkce tohoto tabu slouží jako příležitostné usnadnění a urychlení změny stávající pozice.

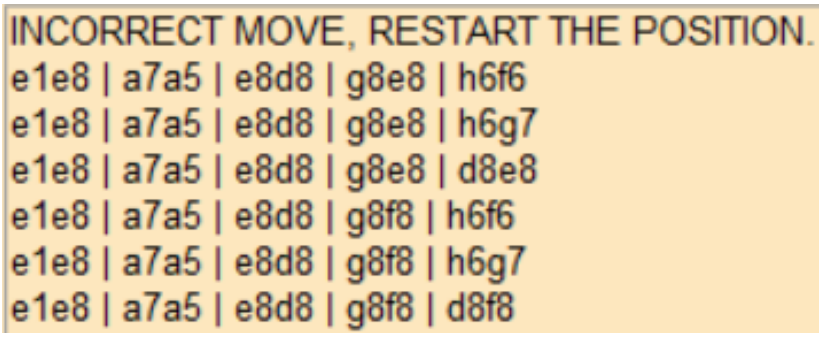


Obrázek 18: Okno pro nastavení pozice.

5.3 Začátek hry

Po nastavení validní pozice hra začíná prvním tahem hráče. V tento moment začíná algoritmus výpočet, ten může trvat dle komplexity pozice až 15 sekund. Důvodem délky výpočtu je především to, že algoritmus musí najít každou možnou cestu vedoucí k matu ve třech tazích, nemá proto mnoho možností pro ořezávání stromu, jak bylo vysvětleno v sekci 2.4.3.

Po dokončení výpočtu se v informačním panelu (viz obrázek 19) zobrazí informace o výpočtu. V případě, že pozice není trojtažkou, se v panelu objeví jediný řádek: `NO POSSIBLE MATE-IN-3 DETECTED IN THIS POSITION`. Pokud pozice trojtažkou je, informace panelu se déle liší podle toho, zda byl první tah hráče správný. Nesprávný tah vede k tomu, kdy panel vypíše hlášku `INCORRECT MOVE, RESTART THE POSITION.`, následovanou seznamem všech možných cest trojtažky. Správný tah zobrazí seznam uskutečnitelných cest, které začínají zahraným tahem.



Obrázek 19: Informační panel, špatný první tah.

5.4 Průběh hry

Zbytek hry probíhá jako normální šachová partie, ovšem s několika rozdíly. Engine hráči ohlašuje každý špatný tah, ale nemůže přímo zabránit hráči v hraní několika špatných tahů za sebou. Očekává se tedy, že po zahrání špatného tahu hráč využije navigačních šipek (viz obrázek 20) pro navrácení se do validní pozice. Tyto šipky mohou být použity pro navigaci oběma směry, jejich funkce přichází vhod například při vizualizaci motivů atraktivity, které algoritmus nalezl během hodnocení trojtažky. Je vhodné dodat, že cesty v informačním panelu jsou automaticky filtrovány podle průběhu hry. Pokud chce hráč prozkoumat jinou cestu k matu, stačí se vrátit v sekvenci šipkami a zkusit jiný tah. V případě toho, že hráč táhne špatně, je mu jako nápověda zobrazen celý seznam cest.

Obrázek 20: Šipky pro navigaci tahů.

5.5 Generování hodnocení

Součástí výpočtu po zahájení hry je zároveň kompletní analýza atraktivity trojtažky. Tato analýza proběhne pouze při pozitivním ukončení první části výpočtu, tedy zjištění, že se o trojtažku skutečně jedná. V tomto případě je vytvořen (nebo přepsán předchozí) soubor `eval.txt`, (viz 4.5), do kterého je následně hodnocení generováno.

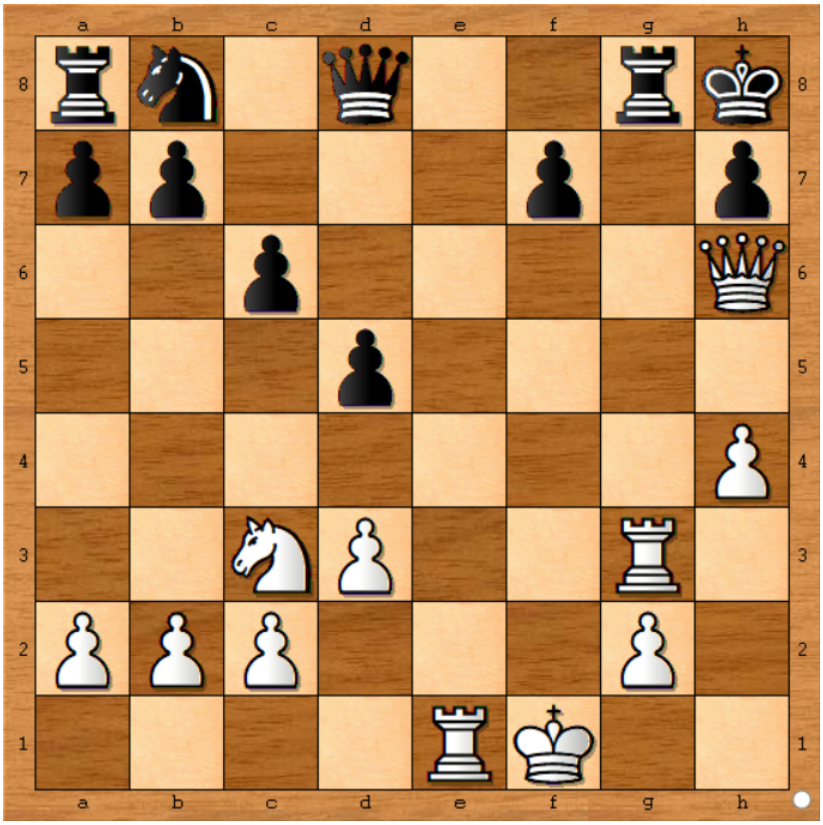
Výsledné hodnocení může nabírat dvou podob. Záleží, zda trojtažka obsahuje pouze jednu, a nebo více možných cest. Pokud se jedná o první variantu, je nejprve vygenerována tabulka hodnocení dané cesty. Tato tabulka je shrnutím hodnocení motivů napříč všemi stavy, a jejich součet společně s WLM tvoří finální skóre atraktivity trojtažky.

Existuje-li cest více, jako první se vygeneruje tabulka podobná té pro jednu cestu, každá hodnota v ní je ale zprůměrováním dané hodnoty v daném stavu napříč všemi možnými cestami. Výsledkem je potom opět součet součtů všech motivů v tabulce se statickým přidáním WLM, tento motiv je totiž, jak už jsem zmiňoval dříve, ve všech cestách neměnný. Po této tabulce následuje standardní tabulka pro jednu cestu, a to takovou cestu, kterou zvolí engine jako nejperspektivnější.

Po tabulkové části následují podrobná hodnocení stavů dané cesty. Každé z těchto hodnocení stavů obsahuje grafické znázornění pozice před a po provedení tahu daného stavu, a následně výpis všech motivů ve stavu nalezených, včetně podrobného popisu toho, které figurky jsou do motivu zapojené a jakou v něm hrají roli.

5.6 Příklad

Ukažme si příklad vyhodnocení celé trojtažky. FEN string startovní pozice příkladu je `rn1q2rk/pp3p1p/2p4Q/3p4/7P/2NP2R1/PPP3P1/4RK2 w - - 0 1`. Tato pozice obsahuje desítky možných cest k matu ve třech tazích, my se budeme zabývat tou hlavní. Následuje vizualizace příkladu a jeho hodnocení.



Obrázek 21: Startovní pozice příkladu.

```
e1e8 -> d8e8 -> h6f6 -> g8g7 -> f6g7
```

Obrázek 22: Hlavní cesta trojtažky.

Path: average of all possible move paths

Themes:	POS 1	POS 2	POS 3	POS 4	POS 5	TOTAL
SacrificeMaterial	0.00	0.00	0.00	0.17	0.00	0.18
Fork	6.00	2.69	3.94	1.13	0.00	13.76
Pin	10.00	3.99	5.16	2.00	0.00	21.15
Skewer	3.00	0.62	1.37	0.70	0.00	5.69
XRay	10.00	4.18	1.26	1.47	0.00	16.90
DiscoveredAttack	0.00	0.00	0.00	0.40	0.00	0.40
SmotheredMate	0.00	0.00	0.00	0.00	0.00	0.00
CrossCheck	0.00	0.00	0.00	0.00	0.00	0.00
Promotion	0.00	0.00	0.00	0.00	0.00	0.00
Switchback	0.00	0.06	0.09	0.07	0.18	0.40

Win With Less Material Bonus: 0

TOTAL AESTHETICS SCORE: 58.48

Obrázek 23: Tabulka průměrného hodnocení všech cest.

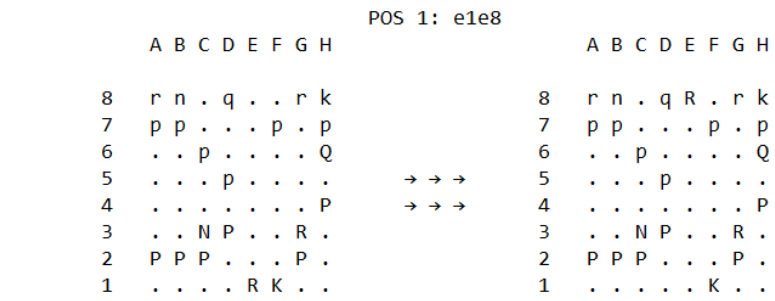
Path: e1e8 -> d8e8 -> h6f6 -> g8g7 -> f6g7

Themes:	POS 1	POS 2	POS 3	POS 4	POS 5	TOTAL
SacrificeMaterial	0.00	5.00	0.00	0.00	0.00	5.00
Fork	6.00	0.00	7.00	0.00	0.00	13.00
Pin	10.00	0.00	0.00	10.00	0.00	20.00
Skewer	3.00	0.00	0.00	2.00	0.00	5.00
XRay	10.00	0.00	0.00	0.00	0.00	10.00
DiscoveredAttack	0.00	0.00	0.00	0.00	0.00	0.00
SmotheredMate	0.00	0.00	0.00	0.00	0.00	0.00
CrossCheck	0.00	0.00	0.00	0.00	0.00	0.00
Promotion	0.00	0.00	0.00	0.00	0.00	0.00
Switchback	0.00	0.00	0.00	0.00	0.00	0.00

Win With Less Material Bonus: 0

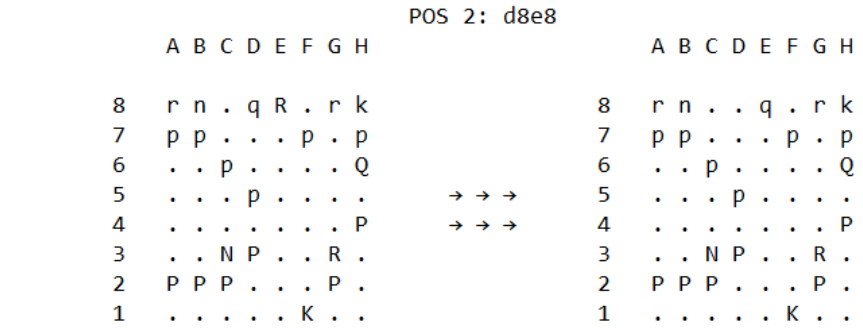
TOTAL AESTHETICS SCORE: 53.00

Obrázek 24: Tabulka hodnocení hlavní cesty.



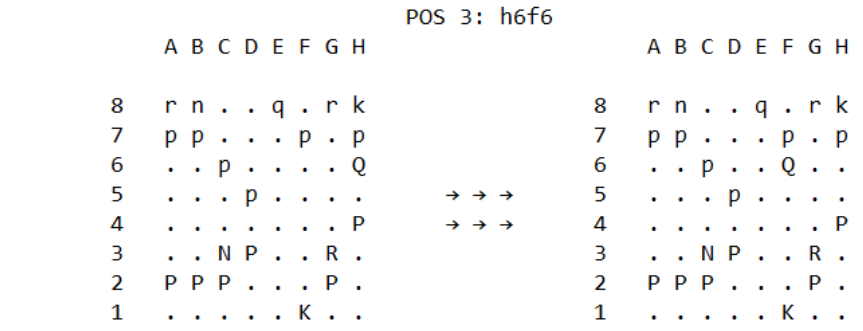
Fork: white Rook on e8 forks: black Queen on d8, black Rook on g8.
Skewer: white Rook on e8 skewers black Queen on d8.
Pin: white Rook on e8 pins black Rook on g8 to a black King on h8.
XRay: black Queen on d8 x-ray defends black Rook on g8 through white Rook on e8.
XRay: black Rook on g8 x-ray defends black Queen on d8 through white Rook on e8.

Obrázek 25: Hodnocení 1. stavu.



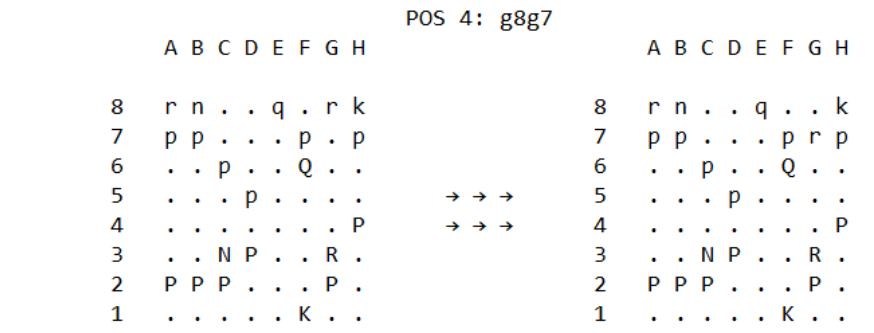
Sacrifice Material: Sacrificed white Rook on e8.

Obrázek 26: Hodnocení 2. stavu.



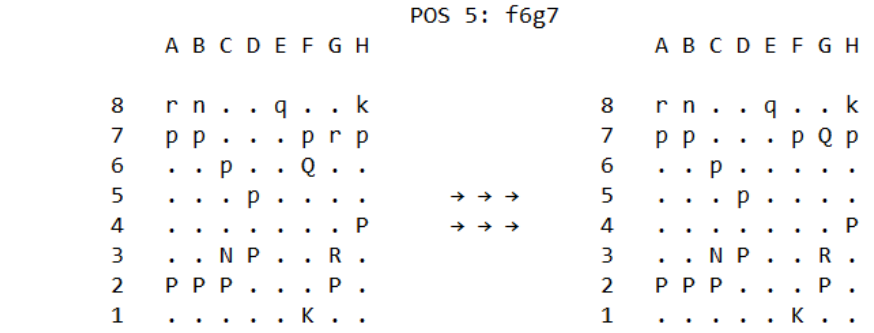
Fork: white Queen on f6 forks: black Pawn on c6, black Pawn on f7,
black King on h8.

Obrázek 27: Hodnocení 3. stavu.



Pin: white Queen on f6 pins black Rook on g7 to a black King on h8.
Skewer: black Rook on g7 skewers white Rook on g3.

Obrázek 28: Hodnocení 4. stavu.



Obrázek 29: Hodnocení 5. stavu.

Závěr

Výsledkem této práce je šachový engine, jehož funkcí je analyzovat šachovou pozici zadanou uživatelem, zjistit, zda se jedná o trojtažku, najít všechny možné cesty trojtažky a následně provést detailní vyhodnocení atraktivity trojtažky, které je vygenerováno do samostatného souboru. Systém hodnocení byl vytvářen tak, aby byl snadno pochopitelný pro méně zkušené hráče, a zároveň užitečný a nápomocný hráčům zkušenějším.

Vývoj práce byl pro mě zajímavou zkušeností, která mi přinesla nové znalosti, co se týče algoritmického řešení problémů, vývoje a správy relativně většího projektu a propojování back-end části projektu s uživatelským rozhraním. Zároveň mám po dokončení projektu hlubší porozumění softwarové implementace šachové logiky a problematiky šachu jako takové.

Conclusions

The result of this thesis is a chess engine designed to analyze user-defined chess positions, determine the presence of a forced mate-in-three, identify all possible paths leading to that mate, and perform a detailed evaluation of the position's aesthetics, which is then saved in a separate file. The evaluation system was created to be easily understandable for less experienced players while also providing valuable insights for more advanced players.

The development of this thesis was an interesting experience that provided me with new knowledge in several areas. I gained insight into algorithmic problem-solving, the development and maintenance of relatively large projects, and connecting the back-end parts of a project to the user interface. Additionally, after completing the thesis, I now have a deeper understanding of chess logic software implementation and the complexities associated with chess as a whole.

A Obsah elektronických dat

src/
: Složka obsahující zdrojové kódy aplikace.

text/
: Složka obsahující text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh. Složka obsahuje také soubory potřebné pro vygenerování PDF dokumentu textu v ZIP archivu, tedy vložené obrázky apod.

README.txt
: Soubor obsahující informace k instalaci aplikace a návod na její použití.

FENStrings.txt
: Soubor obsahující testovací data k dispozici uživateli.

Literatura

[1] chess.com. *Chess Tactics | 38 Definitions and Examples* [online]. [cit. 2024-8-2]. Dostupný z: ⟨https://www.chess.com/article/view/chess-tactics⟩.

[2] Isenberg, Gerd. *Forsyth-Edwards Notation* [online]. [cit. 2024-7-24]. Dostupný z: ⟨https://www.chessprogramming.org/Forsyth-Edwards_Notation⟩.

[3] Ridderkerk, WBEC; Kahlen, Stefan-Meyer. *Description of universal chess interface* [online]. [cit. 2024-7-24]. Dostupný z: ⟨https://wbec-ridderkerk.nl/html/UCIProtocol.html⟩.

[4] Mayefsky, Eric; Anene, Francine; Sirota, Marina. *Strategies and tactics for intelligent search* [online]. [cit. 2024-7-25]. Dostupný z: ⟨https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/intelligent-search/minimax.html⟩.

[5] Neumann, Leandro Ricardo; Beckemkamp, Eduardo Ivan; Peixoto, Jonathan Ramon; Rupp, Luiz Gustavo. *Minimax algorithm simulator* [online]. [cit. 2024-7-25]. Dostupný z: ⟨https://raphsilva.github.io/utilities/minimax_simulator/#⟩.

[6] Attard, Michael. *Pruning decision trees - easy examples* [online]. [cit. 2024-7-28]. Dostupný z: ⟨https://insidelearningmachines.com/pruning_decision_trees/⟩.

[7] Iqbal, Azlan; Guid, Matej; Heijden, Harold M.J.F. Van der; Makhmali, Ali. Evaluating the Aesthetics of Endgame Studies: A Computational Model of Human Aesthetic Perception. *IEEE Transactions on Computational Intelligence and AI in Games*. 2012.

[8] Lichess.org. *Board editor* [online]. [cit. 2024-7-26]. Dostupný z: ⟨https://lichess.org/editor⟩.

[9] Osborne, Harold. Notes on the aesthetics of chess and the concept of intellectual beauty. *British journal of aesthetics*. 1964.

[10] chessfox.com. *X Ray Attack* [online]. [cit. 2024-7-26]. Dostupný z: ⟨https://chessfox.com/x-ray/⟩.

[11] chess.com. *Smothered mate* [online]. [cit. 2024-7-26]. Dostupný z: ⟨https://www.chess.com/terms/smothered-mate⟩.

[12] Blume, Martin. *Arena Chess GUI* [online]. [cit. 2024-7-27]. Dostupný z: ⟨http://www.playwitharena.de/⟩.

[13] Isenberg, Gerd. *Attack and Defend Maps* [online]. [cit. 2024-7-27]. Dostupný z: ⟨https://www.chessprogramming.org/Attack_and_Defend_Maps⟩.

[14] Isenberg, Gerd. *Algebraic Chess Notation* [online]. [cit. 2024-7-28]. Dostupný z: ⟨https://www.chessprogramming.org/Algebraic_Chess_Notation⟩.

[15] Vanthoor, Marcel. *How move ordering works* [online]. [cit. 2024-7-28]. Dostupný z: ⟨https://rustic-chess.org/search/ordering/how.htmls⟩.

[16] Maughan, Steve. *UCI Protocol – Capturing User Input* [online]. [cit. 2024-7-28]. Dostupný z: ⟨https://www.chessprogramming.net/uci-protocol-capturing-user-input/⟩.

[17] Microsoft. *Visual Studio: IDE and Code Editor for Software Developers and Teams* [online]. [cit. 2024-7-27]. Dostupný z: ⟨https://visualstudio.microsoft.com/⟩.