VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

UMĚLÁ INTELIGENCE PRO DESKOVOU HRU HNEFATAFL

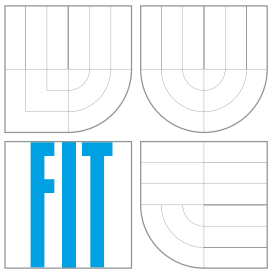
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

LENKA STRATILOVÁ

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INTELLIGENT SYSTEMS

UMĚLÁ INTELIGENCE PRO DESKOVOU HRU HNEFATAFL

THE HNEFATAFL BOARD GAME ARTIFICIAL INTELLIGENCE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — LENKA STRATILOVÁ
AUTHOR

VEDOUCÍ PRÁCE — Ing. DAVID KUBÁT
SUPERVISOR

BRNO 2012

Abstrakt

Obsahem této práce je návrh, vytvoření a otestování umělé inteligence pro deskovou hru Hnefatafl. Začátek písemné práce popisuje pravidla Hnefataflu včetně nejznámějších variant a použité algoritmy z oblasti umělé inteligence, především algoritmus Alfa-Beta s dohledáváním do klidové pozice. Následuje popis implementace a testování. Program umožňuje hru dvou hráčů, hráče proti počítači a hru dvou počítačů.

Abstract

The main task of this bachelor thesis is to design, create and test the Hnefatafl board game artificial intelligence. In the beginning of thesis are description of Hnefatafl rules with its variations and the most common algorithms of artificial intelligence. The following describes the implementation and testing. The game allows two players mode, player against computer mode and two computers mode.

Klíčová slova

Hnefatafl, umělá inteligence, C++, Qt, Alfa-Beta, optimalizace, ohodnocovací funkce, grafické uživatelské rozhraní.

Keywords

Hnefatafl, artificial intelligence, C++, Qt, Alpha-Beta, optimization, evaluation function, graphical user interface.

Citace

Lenka Stratilová: Umělá inteligence pro deskovou hru Hnefatafl, bakalářská práce, Brno, FIT VUT v Brně, 2012

Umělá inteligence pro deskovou hru Hnefatafl

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Davida Kubáta.
Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Lenka Stratilová
15. května 2012

Poděkování

Chtěla bych poděkovat svému vedoucímu Ing. Davidu Kubátovi za rady a odborné vedení.

Obsah

Seznam obrázků

Kapitola 1

Úvod

Deskové hry provázejí člověka již od nepaměti. V mnohých starověkých a středověkých kulturách byly deskové hry považovány nejen za libou kratochvíli, ale jejich znalost za nutnou výbavu vzdělaného člověka. Přestože toto již dnes neplatí, obliba deskových her rozhodně neklesá. Ať už se jedná o klasické hry jako šachy nebo moderní deskovky jako například Osadníci z Katanu a Carcassonne.

Jednou z velmi starých deskových her je i Hnefatafl, hra původem ze Skandinávie, která se díky vikingským nájezdům rozšířila po celé Evropě. Tato hra byla natolik oblíbená, že vzniklo mnoho různých variant, které dnes označujeme jako taflové hry. Obsahem této práce je seznámit čtenáře s Hnefataflem a vytvořit program s umělou inteligencí, který čtenáři umožní si hru zahrát a otestovat si tak své herní schopnosti.

V první kapitole seznamuje čtenáře s pravidly a historií Hnefataflu. Zároveň ukazuje, jak rozsáhlá a rozmanitá je rodina taflových her. Druhá kapitola nastíní algoritmy a techniky potřebné pro naprogramování umělé inteligence této deskové hry a je především zaměřena na vysvětlení jednotlivých strategií pro černého a bílého hráče. To, jakým způsobem byla umělá inteligence pro Hnefatafl naimplementována, je obsahem čtvrté kapitoly. A nakonec pátá kapitola srovnává dosažené výsledky v důkladném testování.

Obrázek 1.1: Ukázka figur

Kapitola 2

Hnefatafl

Tato kapitola představuje deskovou hru Hnefatafl, popisuje její historii a pravidla. Dále obsahuje stručný přehled nejznámějších dochovaných variant, obecně označovaných jako taflové hry.

2.1 Představení a historie

Hnefatafl je desková hra původem ze Skandinávie přibližně z 8. století n.l. Byla oblíbenou kratochvílí Vikingů, kteří ji díky svým dobyvatelským výpravám rozšířily téměř po celé Evropě. Hra se na mnoha místech uchytila, byť s různými obměnami. Dnes tyto varianty souhrně nazýváme taflové hry.

Význam slova Hnefatafl pochopíme hned po jeho rozložení na slova hnefi a tafl. Hnefi v staronorštině znamená pěst, avšak díky vlivu šachu jej v přeneseném významu překládame jako krále. Tafl je severské slovo označující stůl a v literatuře se často uvádí jeho odvození od latinského *tabula*.[3]

Přestože hra je několikrát zmiňována v severských ságách (např. *Edda*), o konkrétním původu Hnefataflu toho mnoho nevíme. Někteří autoři se proto domnívají, že hra má své kořeny ve starověkém Římě, konkrétně ve hře *Ludus Latrunculorum* [7] (česky *Hra vojáků*). Ta je s taflovými hrami spojována hlavně kvůli principu zajímání figur, který není znám z žádné jiné dochované hry. Toto spojení nikdy nebylo prokazatelně dokázáno.

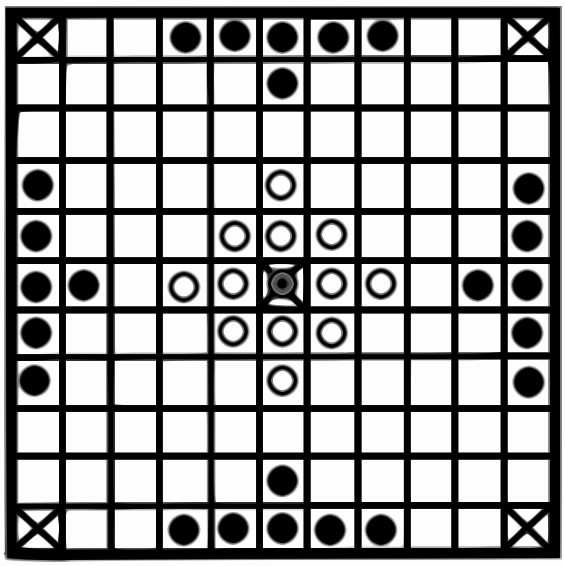
Oblíbenost taflových her rostla až do 12. století, kdy do Evropy dorazily šachy. Ty málem způsobily úplné vymizení Hnefataflu [5].

Naštěstí existují některé dokumenty, které pravidla hry pomohly zachovat až do současnosti. Jedním takovým dokumentem jsou například zápisy přírodovědce a lékaře Carla Linné, který při své cestě po Laponsku v roce 1732 [2] zapsal pravidla hry, kterou hráli jeho průvodci. Bohužel tento dokument nepopisuje hru úplně přesně. Carl Linné neuměl jazyk místních a proto pravidla popsal jen na základě pozorování.

V současné době existuje mnoho variant. Nejčastěji se liší velikostí desky, počtem figur, základním rozestavěním a cílem bílého. Některé varianty dokonce vyžadují kostku. V této práci jsou jako základní pravidla brána tzv. pravidla skandinávských muzeí [3]. Jedná se o kompilaci pravidel z více zdrojů, která tato muzea uchovávají.

2.2 Pravidla hry

K základní hře Hnefataflu potřebujeme desku o velikosti 11 x 11, 24 černých a 12 bílých figur a dále pak figurku krále. Král vždy začíná uprostřed hrací desky (toto pole se nazývá trůn) a ostatní figury jsou rozestavěny kolem podle následujícího schématu.



Obrázek 2.1: Základní rozestavení

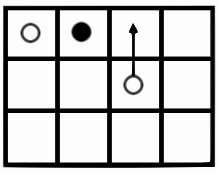
Hru vždy začíná útočník, v našem případě reprezentován černými figurami. Hráči se v tazích střídají.

Všechny figury se pohybují jako šachové věže, tedy mohou se libovolně pohybovat ve svém řádku nebo sloupci, dokud nenarazí na jinou figuru nebo na speciální pole. Pohyb po diagonále není povolen.

Speciálních polí je na desce celkem 5 (na obrázku 2.1 znázorněny přeškrnutím), konkrétně jsou to rohy a středový trůn. Na tato pole může vstoupit pouze král. V případě trůnu mohou toto pole ostatní figury přeskočit, pokud trůn není králem obsazen.

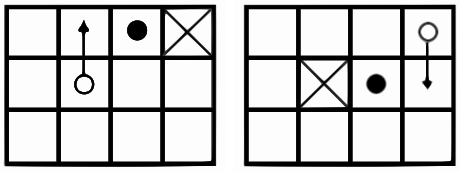
Cílem bílého hráče je dostat krále do libovolného rohu, naopak cílem černého je krále zajmout.

Zajmout lze libovoulnou figuru, pokud je obklíčena ze 2 protilehlých stran soupeřovými figurami.



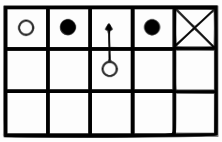
Obrázek 2.2: Základní zajmutí figury

K zajmutí figury je možné také využít střed nebo rohy. Pokud však chceme použít trůn k zajmutí bílé figury, je nutné, aby trůn nebyl obsazen králem. Pokud je král na trůnu, trůn se počítá jako běžná bílá figura.



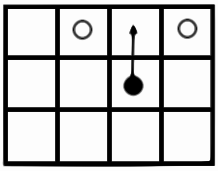
Obrázek 2.3: Zajmutí figury pomocí rohu a trůnu

Je také možné jedním tahem zajmout více figur, jako je to například možné v klasické dámě.



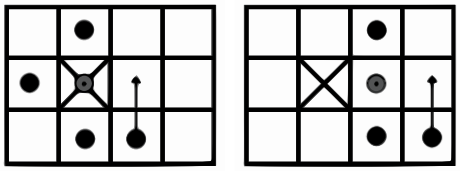
Obrázek 2.4: Dvojité zajmutí figury

K zajetí dochází pouze v případě, že figura je obklíčena na konci pohybu protivníka. Toho mohou využít ohrožené figury a „schovat“ se mezi dvě protivníkovy figury, případně protivníkovu figuru a speciální pole.



Obrázek 2.5: Schování figury mezi soupeřovými figurami

Zajetí krále se řídí stejnými pravidly jako zajetí pěšce, existuje však jedna výjimka. Pokud je král na trůnu, může být zajat pouze, pokud je obklíčen ze 4 stran. Pokud král stojí na některém ze 4 polí sousedících s trůnem, musí být obklíčen ze 3 zbývajících stran.



Obrázek 2.6: Zajetí krále na trůnu a v sousedství trůnu

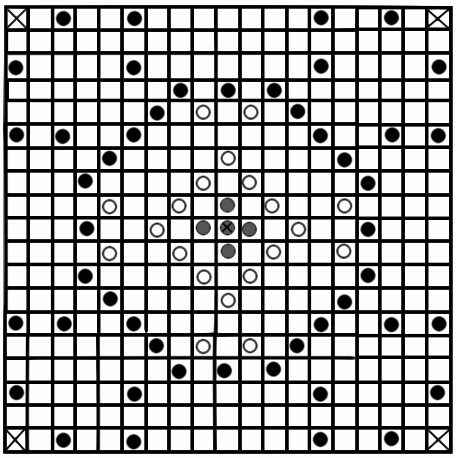
Pokud hráč nemůže pohnout s žádnou svou figurou, prohrál.

2.3 Další varianty

Jak již bylo zmíněno v podkapitole pojednávající o historii Hnefataflu, existuje mnoho různých variant, které jednotně nazýváme taflové hry. Zde je výčet několika nejznámějších.

2.3.1 Alea Evangelii

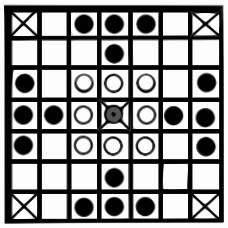
Alea Evangelii, v překladu „hra evangelíků“, je saskou variantou Hnefataflu. Dle [7] se dokonce jedná o jedinou deskovou hru, kterou Sasové hráli. Zmínky o této variantě pocházejí z irského rukopisu z 10. století n.l. Úvodní věta tohoto dokumentu dala hře i jméno: „*Incipit alea evangelii quam Dubinsi...* “. Hra se hraje na desce o velikosti 19 x 19 polí s 25 bílými a 48 černými figurami. Hlavním rozdílem oproti Hnefataflu je přítomnost tzv. strážců (součást bílých figur). Tyto figury nemohou být zajmuty, pouze blokovány. Král je zajímán ze 4 stran, bílý vítězí útěkem do rohu.



Obrázek 2.7: Startovní rozestavění varianty Alea Evangelii

2.3.2 Ard-Ri

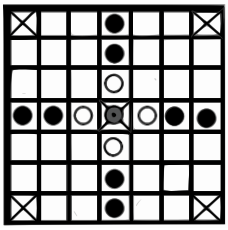
Překlad skotských slov Ard-Ri znamená nejvyšší král a je to také název tamější varianty Hnefataflu. Od toho se liší především velikostí desky, pouze 7 x 7, a počtem figur, 9 bílých a 16 černých. Král je zajímán ze 4 stran, bílý vítězí po dosažení okraje.



Obrázek 2.8: Startovní rozestavení varianty Ard-Ri

2.3.3 Brandubh

Název irské varianty Hnefataflu je překládán jako „havran černý“. Zmínky o této hře lze najít např. v básni Abair riom a Éire ógh připisované básníku Maoil Eóin Mac Raith z období přibližně 13.-14. století n.l. [3]. Zajímavostí je, že hlavní bílá figura zde není nazývána králem, ale jako „branán“, což v doslovném překladu znamená šéf. U této varianty je deska velká pouze 7 x 7 polí a hraje se s 5 bílými a 8 černými figurami. Král může být zajímán ze 2 i ze 4 stran, bílý vítězí dosažením libovolného rohu.

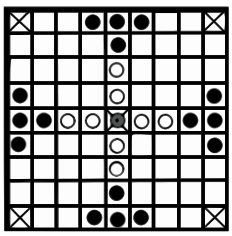


Obrázek 2.9: Startovní rozestavení varianty Brandubh

2.3.4 Fidchell, Fitchneal, Fitcheall

Stejně jako Brandubh i Fidchell pochází z Irska. Přestože hra je velmi podobná Tablutu (viz. níž), někteří autoři ji již neřadí do rodiny taflových her, např.[3], a spíše ji přirovnávají k římské hře *Ludus Latrunculorum.*

Hra používá desku velikosti 7 x 7 i 9 x 9 polí. Pro variantu 9 x 9 se používá 9 bílých a 16 černých figur. Král je zajímán ze 4 stran, bílý vítězí po dosažení okraje.



Obrázek 2.10: Startovní rozestavení varianty Fidchell

2.3.5 Gwyddbwyll

Gwyddbwyll je pouze welšský název pro 7 x 7 variantu Fidchellu.

2.3.6 Tablut

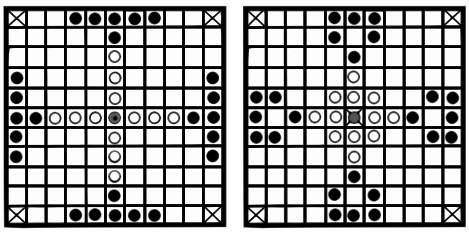
Z Finska pochází Tablut, další varianta, která má k Hnefataflu asi nejblíže. Pravidla této konkrétní obměny zapsal jako první Carl Linné v 18. století n.l. [2], jedná se tedy o nejmladší pravidla taflové hry vůbec. Hra samotná měla znázorňovat bitvu mezi existujícími národy, Švédy (bílý hráč) a Rusy (černý hráč).

Hra probíhá na desce o velikosti 9 x 9 polí s 9 bílými a 16 černými figurami. Král je zajímán ze 4 stran, bílý vítězí po dosažení rohu. Základní rozestavení je stejné jako u fidchellu.

2.3.7 Tawlbyund

Welšská varianta Hnefataflu je nazývána Tawlbyund nebo také Tawl Bwrdd, což v překladu znamená „házecí deska.“ K této variantě máme jedny z nejpřesnějších pravidel díky popisu v dokumentu *Peniarth Manuscript 158* z roku 1587 [3], které se dochovaly až do dnešních dní a jsou uloženy ve Welšské národní knihovně.

Velikost desky u této varianty je 11 x 11 a hraje se s 13 bílými a 24 černými figurami. Počáteční rozložení existuje ve dvou variantách. Král je zajímán ze 2 stran a bílý hráč začíná. Existuje zde obdoba zvolání „Šach!“ při ohrožení krále. Bílý vítězí po dosažení okraje.



Obrázek 2.11: Dvě varianty startovního rozestavení Tawlbyundu

Kapitola 3

Umělá inteligence

Tato kapitola pojednává o pojmech a algoritmech z oblasti umělé inteligence, které poslouží k naprogramování inteligentního soupeře v Hnefataflu. Jsou tu popsány nejčastěji používané algoritmy včetně modifikací použitých při této konkrétní implementaci. Kapitola je zpracována podle [4], [8] a [9].

3.1 Klasifikace Hnefataflu

Při hraní Hnefataflu spolu hráči nespolupracují, vítězství jednoho hráče znamená prohru druhého. Hráči se ve svých tazích střídají a není možné vzít zpět nebo jinak opravit svůj již uskutečněný tah. Na základě těchto faktů můžeme hru zařadit do skupiny nekooperativních her v normální formě s nulovým součtem [4].

Další třídou, kam můžeme Hnefatafl zahrnout, jsou složité hry [9]. Díky velikosti desky a způsobu pohybu, nejsme schopni prohledat celý stavový prostor, proto jsme nuceni použít algoritmy, které jej prohledávají jen do určité zadané hloubky. Takovými algoritmy jsou například Minimax a Alfa-Beta.

3.2 Základní pojmy

- Stavový prostor - množina všech stavů dané úlohy a přechody mezi nimi, k jeho reprezentaci využíváme datovou strukturu stromu
- Tah - provedení pohybu obou hráčů
- Půltah - provedení pohybu jednoho hráče
- Strategie - výběr konkrétního půltahu

3.3 MiniMax

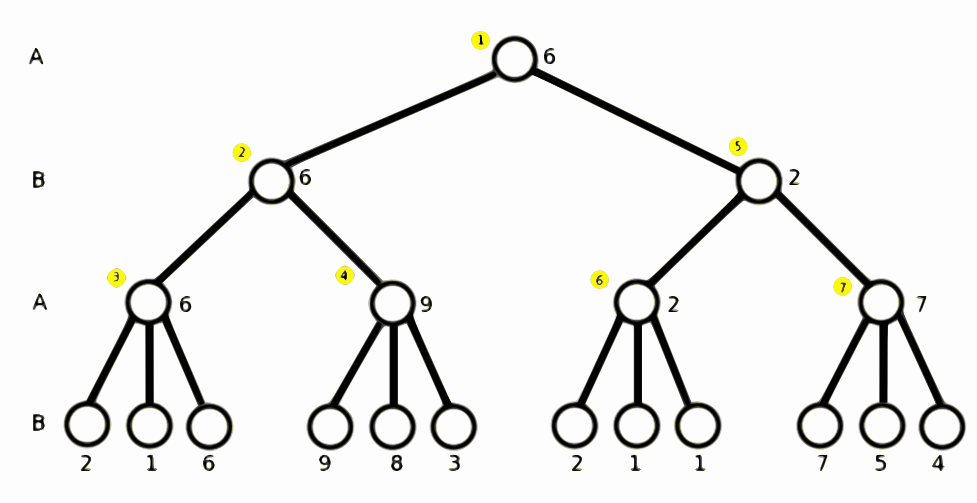
Jedním ze základních algoritmů, které prohledávají stavový prostor do omezené hloubky, je MiniMax. Algoritmus využívá rekurzivní volání sebe sama pro aktuální stav hry a právě hrajícího hráče. Na výstupu funkce vrací ohodnocení uzlu a pro aktuálního hráče i nejvýhodnější půltah. Hodnotu v daném uzlu získáme pomocí statické ohodnocovací funkce. Hloubka prohledávání musí být omezena, aby algoritmus skončil v rozumném čase.

3.3.1 Pseudokód

```
int minimax(uzel, hloubka)
{
   if (uzel == list || hloubka <= 0 )
     return hodnota_uzlu;

   α = -MAX;

   for (každý potomek uzlu)
      α = max(α, -minimax(potomek, hloubka-1);

   return α;
}
```

3.3.2 Příklad

Fungování algoritmu si můžeme ukázat na následujícím příkladu:



Obrázek 3.1: MiniMax

Na tahu je hráč A, který zavolá proceduru MiniMax na svůj první tah (uzel 1) a hráče B (uzel 2). Hráč B pak volá MiniMax na svůj tah (uzel 2) a tah hráče A (uzel 3). Ten opět volá funkci na všechny své možné tahy a hráče B. Protože všechny následující uzly jsou již listy stromu, MiniMax vrátí pouze ohodnocení těchto listů a hráč A z nich vybere a vrátí výš maximální ohodnocení. Hráč B přijme tuto hodnotu a zavolá funkci MiniMax na svůj další tah (uzel 4). V tomto uzlu hráč A zavolá funkci MiniMax a protože následující uzly jsou opět listové, získá zpět nejvyšší ohodnocení a to předá výš. Protože tato hodnota je větší než hodnota, která je v tuto chvíli v uzlu 2 ($6 < 9$), hráč B toto ohodnocení nepřijme a vrátí výš svou aktuální hodnotu. Tento systém procházení se aplikuje i na zbývající část stromu.

Jedním ze zjednodušení MiniMaxu je metoda NegaMax [1], která je použita i v této konkrétní implementaci. Negamax je možné použít jen v případě, že hodnota uzlu hráče A je zápornou hodnotou uzlu hráče B. To znamená, že aktuální hráč na tahu hledá pohyb

s největší zápornou hodnotou soupeřova pohybu.

V praxi se toho využívá tak, že můžeme zjednodušit implementaci MiniMaxu a použít pouze jednu funkci pro oba hráče, kdy hledáme pouze maximální ohodnocení, místo abychom střídavě hledali minimální a maximální ohodnocení uzlů. Minimalizaci u soupeře totiž dosáhneme změnou znaménka.

Výsledek, který vrátí NegaMax, je úplně stejný, jaký vrací základní MiniMax.

3.4 Alfa-Beta řezy

Dříve uvedený algortimus MiniMax prochází všechny uzly do zadané hloubky, ale protože vyšetřování některých uzlů je zbytečné, můžeme použít tzv. Alfa-Beta řezy, které pomohou MiniMax zrychlit.

Alfa-Beta funguje tak, že zastaví vyhodnocování tahu, pokud je horší než libovolný jiný již dříve zkoumaný tah. Ořízne tak větve stromu, které nemohou ovlivnit výsledek.

3.4.1 Pseudokód

```
int alfabeta(uzel, hloubka, α, β, hrac)
{
 if (hloubka == 0 || uzel == list)
  return (hrac * hodnota_uzlu);
 else
 {
  for (každý potomek uzlu)
  {
    hodnota_uzlu = -alfabeta(potomek, hloubka - 1, −β,−α,-hrac);
    if hodnota ≥ β
      return hodnota;
    if hodnota ≥ α
      α = hodnota;
  }
  return α;
  }
}
```

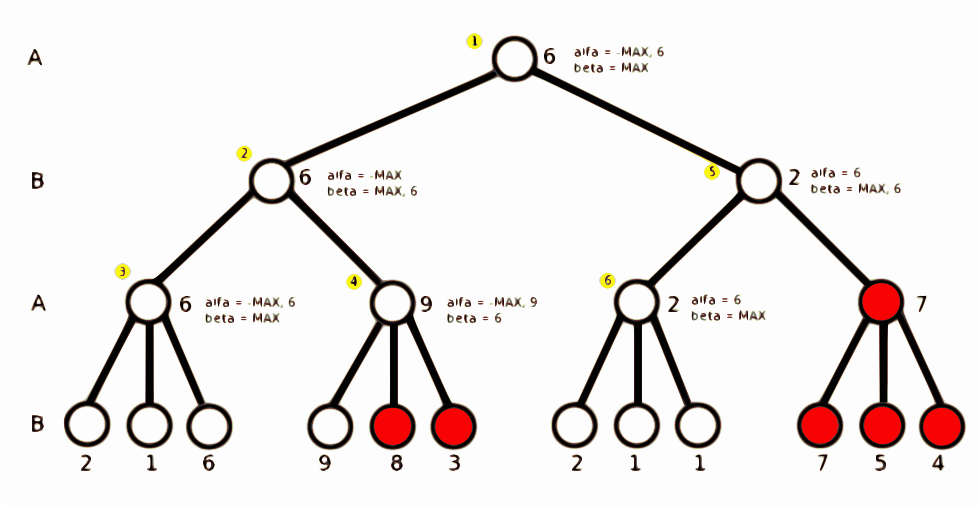
První zavolání funkce:

```
alfabeta(koren, hloubka, -MAX, +MAX, aktualni_hrac);
```

3.4.2 Příklad

I zde si předvedeme fungování algoritmu na příkladu:

Na začátku nastavíme $\alpha = -MAX$, $\beta = +MAX$. Hráč A, který je na tahu (uzel 1), zavolá funkci Alfa-Beta na svůj první tah a tah hráče B (uzel 2) s hodnotami $\alpha = -MAX$, $\beta = +MAX$. Hráč B poté zavolá tuto funkci na svůj tah a na tah hráče A (uzel 3) opět s hodnotami $\alpha = -MAX$, $\beta = +MAX$. V tuto chvíli hráč A zavolá Alfa-Betu na své potomky. Protože to jsou listy stromu, nejvyšší vrácená hodnota se uloží do $\alpha = 6$. Protože $\alpha < \beta$, pokračujeme v prohledávání.



Obrázek 3.2: Alfa-Beta řezy

Hráč A (uzel 3) vrátí o úroveň výš hodnotu α, hráč B tuto hodnotu přijme a uloží si ji do proměnné β, proměnná α se nemění. Hráč B nyní zavolá funkci Alfa-Beta s hodnotami $\alpha = -MAX$, $\beta = 6$ na další svůj možný tah a hráče A (uzel 4). Hráč A opět zavolá funkci na své potomky. Protože z prvního potomka se mu vrátí hodnota 9, která je větší než aktuální hodnota v proměnné α, tuto vrácenou hodnotu přijme. Tím je však porušena podmínka $\alpha < \beta$: $(9 < 6)$ a vyšetřování uzlu se ukončí. O úroveň výš se vrací hodnota $\alpha = 9$, ale protože hráč B vybírá menší číslo z původní a vrácené hodnoty, zůstávají v uzlu 2 hodnoty $\alpha = -MAX, \beta = 6$.

Protože není žádný další možný tah hráče B, vrátíme o úroveň výš hodnotu β, kterou si v uzlu 1 uložíme do proměnné α, tj. $\alpha = 6, \beta = +MAX$. S těmito hodnotami zavoláme funkci Alfa-Beta na zbývající část stromu.

Jak můžeme vidět, použitím Alfa-Beta řezů místo základního MiniMaxu jsme ušetřili prohledávání 6 uzlů (v grafu červeně zbarvené).

3.4.3 Modifikace

Protože princip Alfa-Beta řezů existuje již od 60. let 20. století, bylo objeveno mnoho metod a technik [8], které algoritmus zrychlují. Zde je krátký souhrn metod, které jsou v projektu využity:

- **třídění tahů** - protože účinnost Alfa-Beta řezů závisí na konkrétním pořadí procházených tahů, je výhodnější, pokud procházíme tahy od nejlepších po nejhorší. Toto není možné vždy dosáhnout úplně přesně, přesto lze vhodným řazením nebo heuristikou eliminovat určité, pro daný typ hry nezajímavé, tahy a tím prohledávání stromu zrychlit.
- **hledání klidové pozice** - jedná se o prohledávání tahů dál než je zadaná maximální hloubka. Takto však nejsou zkoumány všechny tahy, ale pouze vybrané. Při vhodné volbě tahů dovolí tato metoda umělé inteligenci zkoumat *matové* pozice do mnohem větší hloubky bez výrazného časového nárůstu.

- **metoda okna** - tato technika se snaží dosáhnout co nejpřesnějšího odhadu výsledku a díky zúžení vzálenosti mezi α a β ořezat co největší množství pozic, které musíme prozkoumat. Pokud použijeme nulové okno, tj. $\alpha = \beta$, bude se nám vracet pouze informace, zda je zkoumaný tah horší nebo lepší než aktuální, ale nebudeme vědět o kolik lepší či horší je.

- **iterativní prohlubování** - při použití této modifikace nevoláme Alfa-Beta funkci s pevnou hloubkou, ale postupně prohledáváme všechny pozice pro postupně se zvyšující hloubku do nekonečna, případně do limitní hloubky. Toho se využívá především, když potřebujeme zavést časové omezení. V případě, že vyprší čas, už máme nalezen nejlepší tah pro hloubku o jedna nižší než právě prohledávanou a částečně i pro hloubku právě zkoumanou.

3.5 Taktiky hráčů

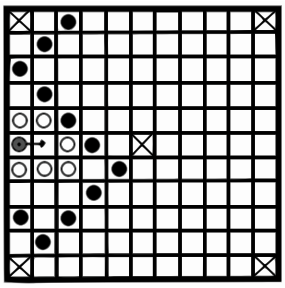
V této podkapitole si popíšeme vybrané základní taktiky, které byly zohledněny i v implementaci. Souhrn těchto taktik byl vypracován na základě zkušeností a přehledu Tima Millara [6], dvojnásobného mistra světa v quickplay[1] Hnefataflu za rok 2009 a 2010.

3.5.1 Taktiky bílého hráče

Většina taktik pro bílého hráče staví na tzv. malé věži, což je uskupení 4 bílých figur do čtverce. Tato formace je černým hráčem nerozbitná, dokud se figury navzájem kryjí. Nevýhodou téměř všech těchto rozestavení je jejich pasivita. Prakticky není možné zajímat černé figury bez toho, aby bylo rozestavění rozbito.

- **Draw fort**

  Pasivní taktika bílého hráče, která se vyplatí pouze ve chvíli, kdy je alespoň jedna bílá figura mimo pevnost (*fort*). Tato figura pak pomáhá zajímat černé a tím uvolnit okolní barikádu.

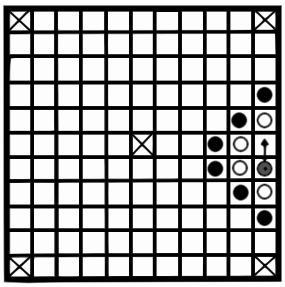


Obrázek 3.3: Draw fort

- **Bunker**

  Pasivní okrajová taktika, jejíž výhodou je malé množství potřebných figur, které chrání krále. Opět se vyplatí pouze, pokud je alespoň jedna bílá figura volná a může pomoci rozbít černou bariéru.

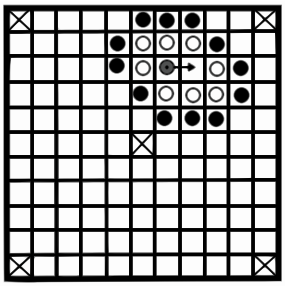
[1] *Quickplay* je varianta Hnefataflu, kdy hráči mají na tah pouze 10 s.



Obrázek 3.4: Bunker

- **Fortress**

  Jedná se o středovou variantou rozestavění bunker. Její výhodou je, že se celá formace může pokusit postupně přesunout k vybranému rohu, příp. okraji, obzvlášť pokud ještě nebyla uzavřena ze všech stran černými figurami.

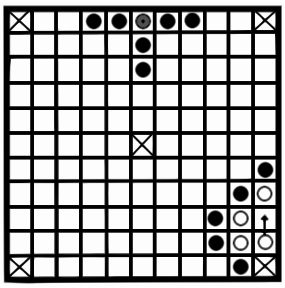


Obrázek 3.5: Fortress

- **Garbo**

  Jméno tohoto rozestavení údajně pochází od herečky Grety Garbo, jejíž jedna známá filmová replika zní: "*I want to be alone...*".

  I u této taktiky platí, že je pasivní a bez podpory volné figury značně nevýhodná.

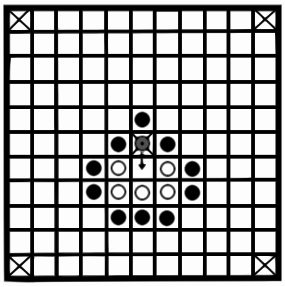


Obrázek 3.6: Garbo

- **Royal citadel**

  Royal citadel je taktika, která maximálně využívá vlastnosti trůnu. Jedinou figurou, která může na trůn vstoupit, je král. To mu umožňuje s podporou jen několika dalších figur pohyb bez toho, že by byl v této pozici ohrožen.

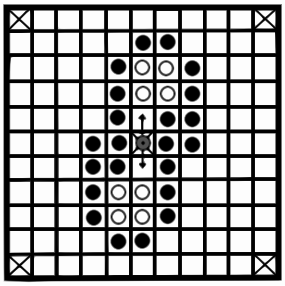
Bohužel i tato taktika je neefektivní bez alespoň jedné bílé figury mimo barikádu.



Obrázek 3.7: Royal citadel

- **Two towers**

  Tato taktika v sobě kombinuje rozestavení 2 malých věží a pohybujícího se krále. Ze všech uvedených taktik je nejúčinnější, protože neobsahuje pouze pasivní obranu krále, ale zároveň umožňuje zajímání černých figur.

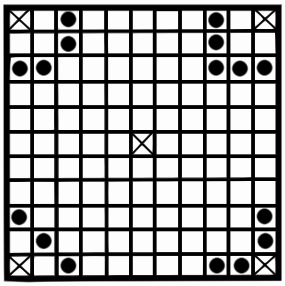


Obrázek 3.8: Two towers

3.5.2 Taktika černého hráče

Pro černého hráče je nejvýhodnější v začátku hry obsadit rohy za pomoci tzv. malých barikád. Výhodné je to především proto, že černý má početní převahu, navíc bílému na začátku hry trvá nejméně 8 tahů, než uvolní krále. To dává dost času na zabezpečení rohů.

Barikády existují ve čtyřech variantách, jak je vidět na obrázku 3.9. Nejlépe vychází obklíčení rohu 3 figurami.



Obrázek 3.9: Barikády černého hráče

Kapitola 4

Implementace

Tato kapitola detailně popisuje použité nástroje, grafické rozhraní, datovou reprezentaci a nejzajímavější algoritmy využité v této práci.

4.1 Nástroje

Pro implementaci programu jsem zvolila jazyk C++ s využitím toolkitu Qt. Tato kombinace mi dovolila vytvořit rychle jednoduché uživatelské rozhraní a díky tomu se pak dále zabývat především implementací umělé inteligence.

4.2 Uživatelské rozhraní

Uživatelské rozhraní bylo zamýšleno vytvořit co nejjednodušší a nejpřehlednější. Hlavní okno aplikace se skládá z menu a centrálního widgetu. Ten zajišťuje vykreslování a řízení celé hry. Menu se skládá ze dvou složek, a to `Game`, obsahující položky `New game` a `Exit`, a `Help`.

Po kliknutí na položku `New game` v menu se spustí dialog nastavení hry, kde je možné vybrat, zda proti sobě budou hrát dva hráči, dvě umělé inteligence nebo hráč a umělá inteligence. Po odkliknutí tohoto dialogu se spustí samotná hra. Ovládání hry pro hráče je velmi jednoduché a spočívá v kliknutí na vybranou figuru a poté kliknutí na pole, kam chce s figurou pohnout. Program hráči ukazuje možné tahy s vybranou figurou. To, který hráč je na tahu, je zobrazeno na liště programu.

V případě hry dvou umělých inteligencí se nevykresluje hrací deska do okna aplikace, ale vypisují se podrobné výpisy do konzole.

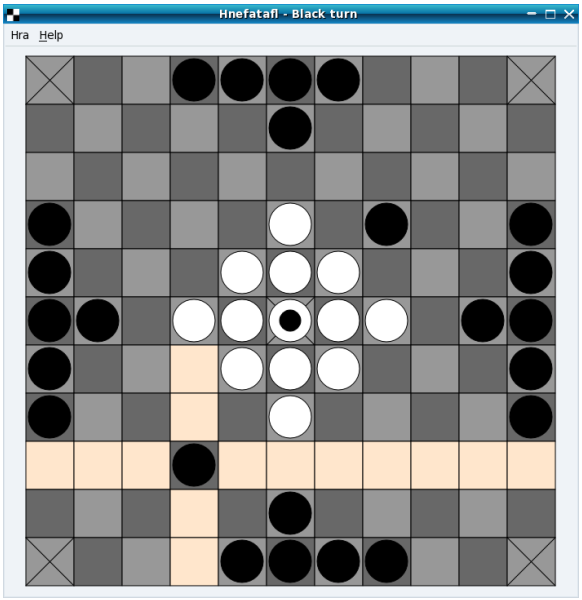
V programu je zpracována jednoduchá nápověda.

4.3 Datová reprezentace

4.3.1 Hrací deska

Deska Hnefataflu má rozměry 11 x 11 a obsahuje tedy 121 polí, z toho dva druhy speciálních: čtyři rohy a trůn. Na obyčejném poli může být právě jeden z čtyřiadvaceti černých kamenů nebo jeden z dvanácti bílých kamenů nebo král. Speciální pole jsou buď prázdná, nebo obsazená králem.

Existuje mnoho variant, jak uspořádat vnitřní reprezentaci desky:



Obrázek 4.1: Uživatelské rozhraní

- Od minimální, 259-bitové, kde máme uspořádaný seznam kamenů se 7-bitovým indexem a zvolenou speciální hodnotou pro zajmutý stav.
- Přes přirozenou, pole sto dvaceti 8-bitových indexů s mnoha variantami reprezentace speciálních polí, trůnu, kamenů a krále.
- Až po, pro paralelní výpočet vhodnou, variantu s několika 128-bitovými poli, např. pro bílé a černé kameny, případně i pro krále zvlášť.

Po delší úvaze jsem zvolila variantu s třinácti 64-bitovými čísly reprezentující jednotlivé řádky desky a s 2 řádky navíc pro okraje. Kvůli zrychlení generování tahu a úspoře velikosti pole nerozlišuji krále od bílé figury a trůn od rohů. To znamená, že pokud se figury snaží přeskočit roh, narazí na okraj. Každé pole má velikost 4 bitů, přestože může nabývat pouze pěti hodnot: bílá figura, černá figura, okraj, prázdné pole, speciální pole (roh nebo trůn).

Nejnižší dva bity každého pole reprezentují, zda lze pole použít jako zarážku při zajímání soupeřovy figury, kde se pod pojmem zarážka myslí figura stejné barvy, prázdný trůn nebo roh ležící za protivníkovou figurou. Každá barva má vyhrazen vlastní bit. Nejvyšší dva bity reprezentují figury hráčů, jeden pro černého a jeden pro bílého, takže figury mají nastaveny 2 bity na jedničku, bit pro figuru dané barvy a bit pro zarážku dané barvy. Okraj má nastaveny oba bity figur na jedničku. Prázdně pole má vše nulové. Rohy a trůn mají oba bity zarážky nastavené na jedničku. Je to do určité míry kompromisní varianta, protože zde občas chybí pátý bit vyhrazený pro prázdné pole jako daň, za možnost mít levé a pravé okrajové pole, oba se totiž do 64-bitového čísla pro řádek nevlezou. Pátý bit se dá nahradit bitovým operátorem *or* nad řádkem a řádkem posunutým o 2 bity doprava, a následně bitovou negací výsledku. Z tohoto důvodu jsou pozice barev u vrchních dvou bitů figury a spodních dvou bitů zarážky vhodně prohozeny.

Tato reprezentace dokáže najít brací prázdné pole pro celý řádek najednou, bez zpomalujících smyček se skoky. Stačí zvolit výsledný bit, shodný pro každém pole, který bude značit zajmutí. Poté udělat kopii řádku obsahující zarážky a v něm posunout potřebnou zarážku na pozici výsledného bitu. Dále udělat druhou kopii řádku obsahující soupeřovy

	Bity figury		**Bity zarážky**	
Barva	Černá	Bílá	Bílá	Černá
Černá figura	1	0	0	1
Bílá figura	0	1	1	0
Roh nebo trůn	0	0	1	1
Prázdné pole	0	0	0	0
Okraj	1	1	0	0

Tabulka 4.1: Datová reprezentace jednoho pole

figury a posunout vybraný bit figury na pozici výsledného bitu. Nakonec udělat třetí kopii řádku a provést operaci pro nastavený výsledného bitu na jedničku u prázdného pole. Výsledný řádek se získá pomocí operace bitový *and* nad řádkem s prázdnými poli, řádkem se soupeřovými kameny posunutý o celé jedno pole vlevo nebo vpravo (opačně než je směr zajmutí) a řádkem se zarážkami posunutým o celé dvě pole stejným směrem jako předchozí. Zbyvá už jen použít masku na vynulování ostatních bitů v poli, než je výsledný bit. Tento postup funguje obdobně i pro směr nahoru a dolů.

Protože nerozlišuji krále od bílé figury, musím si uchovávat index pole, na kterém král je. Tento index je od 0 do 119, kde 0 je levý horní roh, 10 pravý horní roh, 60 trůn, 110 levý spodní roh a 119 pravý dolní roh. Index krále -1 značí, že král byl zajat. Kvůli optimalizaci mám některé informace uloženy v polích, např. rohy[index] vrací jedničku pro jakékoli jiné pole než roh, `index_na_radek[index]` vrací $1 + \texttt{index}/11$, obdobně `index_na_sloupec[index]` vrací `index` mod 11. Opačný převod není potřeba, protože generující funkce prochází pole postupně od 119 k 0 a při zajmutí tak znám index skoku, stačí tedy přičíst nebo odečíst 1 nebo 11. Dále mám uchovány informace o jednoduchém hashi a cenu pozice z pohledu právě hrajícího hráče. Hash je potřeba pro kontrolu zacyklení v dané pozici.

4.3.2 Reprezentace tahu

Struktura `typ_pohyb` obsahuje dvě 8bitové hodnoty, a to index, odkud se figura pohnula, a index, kam hráč táhnul. Dále obsahuje stavové slovo s příznaky pro zajmutí daným směr, pohyb krále a příznak pro přednostně prohledávané pohyby (použité pro blokovaní krále a pohyb krále do rohů). Poslední položkou struktury je cena tahu z pohledu hrajícího hráče, která obsahuje rozdíl ohodnocení od předchozí pozice, přesněji statickou hodnotu políčka, kam bylo taženo, mínus statickou hodnotu políčka, odkud bylo taženo, a případně je ještě přičtena cena zajatých figur soupeře. Odečtení statické hodnoty soupeře a kontrola zajetí krále se tak provadí mimo funkci generování tahů, až ve funkci provádějící tah.

4.4 Vybrané funkce

4.4.1 Funkce generování tahů

Funkce pro generování tahů vytvoří seznam tahů pro danou pozici a barvu. Ve variantě pro bílého při zjištění, že král může do rohu, se přeruší generování a vrátí se jen tento tah. Funkce na začátku vytvoří pomocné pole s příznaky zajmutí pro každé políčko a následně prochází desku od nevyššího indexu k nejnižšímu a hledá figury hráče na tahu. Získá tak index, odkud se případně táhlo. Nalezne-li funkce figuru, hledá ve všech směrech prázdné pole,

kam lze táhnout. Varianta pro bílého je opět složitější, protože rozlišuje, zda netáhneme králem, kterým můžeme skončit i na trůn, případně roh a přerušit generování. Pohyby po řádku se provádí pomocí masky pro daný sloupec, která odmaže ostatní pole. Pohyby po řádcích změnou indexu řádku.

Figura se hledá pomocí bitového *and* s bitem figury dané barvy, prázdné pole bitovým *and* s daným řádkem a maskou s výsledkem nula, konec cesty bitovým *and* s daným řádkem a maskou pro okraj. Nezáleží, zda se zarazí o okraj nebo o figuru libovolné barvy. Malá modifikace je u hledání prázdných polí vpravo. Protože procházíme desku sestupně, pohybujeme se v řádku zprava doleva. Můžeme si tedy cestou počítat velikost souvislého prázdného pole a tuto hodnotu použít. Musí se jen ošetřit, aby figura neskočila na trůn.

4.4.2 Funkce generování zajímavých tahů

Funkce generování zajímavých tahů (zajímavé tahy jsou tahy, které se zkoumají při dohledání do klidové pozice) je obdobná funkci generování běžných tahů. I ona na začátku vytvoří pomocné pole s příznaky zajmutí, ale při procházení deskou nehledáme prázdná pole, ale pole brací, a od nich správnými směry figuru hráče, který je právě na tahu. Správným směrem se rozumí jakýkoliv jiný směr, než je směr braní. Pokud se bere jako směr vlevo, figura může ležet nahoře, vpravo nebo dole. Pokud se bere vlevo i vpravo, pak to znamená, že prázdné pole leží mezi soupeřovými kameny, v tom případě tedy prohledáme jen směr nahoru a dolů. Směr hledaní tedy zjistíme pouhou negací stavového slova pro zajmutí.

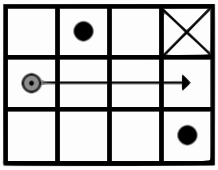
Bracích tahů není v jednotlivých pozicích příliš mnoho, a proto na začátku každého řádku zjistím, zda mě vůbec tento řádek zajímá, respektive zda pomocné pole pro braní na konkrétním řádku je nenulové.

Mezi zajímavé tahy také řadím tahy krále do rohu nebo na místo, odkud se dá skočit do rohu, případně pro černého tahy blokující krále. Aby se zároveň kontrolovalo, zda má král volný roh nebo může skočit na pole, z kterého doskočí do volného rohu, tak použiji pomocné stavové slovo s deseti příznaky. Představme si krále na desce mimo okraj. Umístíme-li na jeho pozici velký kříž, tak protne okraje na čtyřech místech. Z každého místa zkontroluji oba rohy, na které lze dojít. Vznikne tak osm příznaků: horní vlevo je volný, horní vpravo je volný, levý nahoru je volný, levý dolů je volný, atd. Na cestu vlevo i vpravo mám vytvořeny masky pro každý sloupec, tedy mi stačí jediná operace s bitovým *and*. Cesta nahoru se dá spojit pro obě vertikály, cesta dolů také. Aby to bylo ještě efektivnější, zjistím zároveň, zda má král volnou cestu na horní nebo spodní řádek daném sloupci, kde se nachází. Místo masky pro levý nebo pravý okraj použiji masku pro daný sloupec. Ta nastaví poslední dva příznaky. Pak zjistím, zda je král podle indexu na okraji, použitím pole vracícího číslo okraje nebo nulu. Podle okraje zkontroluji příslušné příznaky.

Pokud král může do rohu a hraje bílý, tak se funkce ukončí a vrátí tento jediný konkrétní tah. Hraje-li černý, snaží se jen zajmout krále nebo ho zablokovat, ostatní tahy negeneruje. Pokud král neleží na okraji nebo leží, ale nemá volný roh, zkontoluji postupně, zda může jít na čtyři okrajové pole s použitím příznaků nebo masky vlevo či vpravo, a zda má z daného okraje volný roh. Pokud už ležel král na okraji, tak sice duplicitně zkontroluji, že to nejde, ale zároveň tak zkontroluji i protilehlý okraj. Pokud má král z nově dosaženého okraje oba rohy volné a na tahu je bílý hráč, tak ukončím prohledávání a vrátím tah skoku na okraj. Krále nejde blokovat na oba protilehlé směry a neleží-li těsně u rohu, tak nejde ani zajmout. Pokud má jen volný roh, přidám ho do seznamu nových tahů a zpětně přidám příznaky braní. Hraje-li černý, snaží se zase blokovat jakékoliv volné pole na králově cestě.

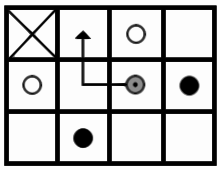
Tahy krále na okraj kontroluji z toho důvodu, že se často stává, že krále na okraji

už nemusí jít zablokovat. Většinou je to pole těsně vedle obklíčeného rohu, a přitom tah předtím blokovat šel a algoritmus by ho považoval za jistou výhru nebo prohru podle hráče. Na obrázku 4.2 se král zdánlivě nezajímavým tahem dostane do neblokovatelné pozice před rohem.



Obrázek 4.2: Dva tahy krále do rohu

Obdobně by šlo kontrolovat krále o tah předtím. Stačí, aby král kličkoval mezi figurami u rohu, jak je vidět na obrázku 4.3.



Obrázek 4.3: Tři tahy krále do rohu

4.4.3 Funkce provedení tahu

Tato funkce zkopíruje desku, otočí její cenu a odmaže cenu tahu, protože cena je už brána z pohledu následujícího hráče. Dále smaže pole, odkud se táhlo, a na pozici, kam se táhlo, vloží figuru dané barvy. Provede dvakrát *xor* s hashem, jednou s indexem *odkud* a podruhé s indexem *kam*. Pokud příznak zajmutí je zapnut, odmaže i zajmutou figurku. Odečte hodnotu statické pozice zajmuté figurky a provede *xor* s hashem a indexem zajmuté figury.

Černý hráč na konci zjistí, zda nesmazal krále, a pokud ano, zda to byl platný tah. Pokud leží král v chráněné oblasti okolo trůnu, potřebuje černý hráč čtyři zarážky, tj. své tři figury a trůn. Pokud byl král zajat proti tomuto pravidlu, je oživen, jinak funkce vrací nenulovou hodnotu. Bílý hráč zjistí, zda netáhl s králem, a pokud ano, zda netáhl z trůnu, aby ho mohl vložit na smazané pole. Zkontroluje také, zda nedosáhl rohu, a pokud ano, vrací nenulovou hodnotu.

4.4.4 Funkce Alfa-Beta

Funkce Alfa-Beta zjistí, zda pozice, kterou obdržela po tahu protihráče, se neopakuje. Pokud ano, zjistí kolikrát se opakovala, a to od začátku hry až k právě zkoumanému tahu. Pokud je prozatimní cena kladná, tak se od této ceny odečte násobný postih za cyklení. Protože hráči na tahu se daří, ale přitom se pozice opakuje pořád dokola, proto dostává tento postih. Jinak by dostal bonus, že se mu daří zdržovat hru. Díky tomu, že se kontroluje od začátku hry, může časem vyhrávajícímu hráči přijít tah jako nevýhodný a zvolit pro něj předtím horší variantu.

Pokud nebyla dosažena zvolená hloubka, funkce generuje všechny tahy hráče. Nemá-li hráč žádný možný tah, prohrál. Pokud byla hloubka překročena, generuje dál jen zajímavé

tahy. Přestože nemá už žádné zajímavé tahy, neukončí ihned prohledávání, protože soupeř může ještě mít zajímavé tahy, a proto provede nulový tah a zavolá protihráče s příznakem *končím*.

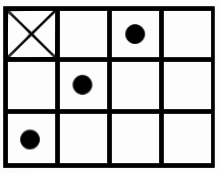
Následně funkce seřadí tahy podle ceny a postupně prochází jednotlivé větve. Až všechny prohledá a vyprší jí hloubka, zjistí, zda si nepohoršil oproti současné ceně pozice. Pokud ano, tak se mohlo stát, že vygenerovaný omezený počet tahů byl příliš omezující. Např. černý hráč uvolní roh, bílý má jen tah krále před roh a následně černý bere o roh krále. Aby se odlišilo přirozené zhoršení vlivem špatné pozice od předchozího příkladu, zkusím, zda žádný tah není lepší než nejlepší dosažený. Protože chci vědět pouze, zda libovolný tah není lepší, volám Alfa-Betu s nulovým oknem nastaveným na hodnotu nejlepšího výsledku. Pokud výsledek je lepší, vracím cenu aktuální pozice.

Pokud je skončeno regulérní prohledávání do zadané hloubky, tedy příštím voláním Alfa-Bety začíná generování pouze zajímavých tahů, a zároveň příznak koncové hry je nastaven, ohodnotím cenu pozice podle dodatečných podmínek snažící se krále zajmout, což jsou počet zarážek a počet možných pohybů krále. Pod příznakem koncové hry je myšleno, že všechny bílé figury s vyjímkou krále jsou zajmuty.

Vzhledem k tomu, že podle barvy hráče se drobně liší výpočet, nemám jednu rekurzivní funkci Alfa-Beta, ale dvě funkce navzájem se volající pro každou barvu. Dále dvě funkce generující tahy, dvě funkce generující zajímavé tahy a dvě, které provádějící tah. Za cenu delšího kódu je omezen počet podmínek.

4.4.5 Funkce volající Alfa-Betu

Na začátku výpočtu nastavím statické tabulky polí pro jednotlivé hráče podle polohy krále. Chci tím dosáhnout, aby se figurky shlukovali v místě krále a ne náhodně u protilehlého rohu. Je připraveno 9 tabulek pro každého hráče a vybírám jednu podle toho, zda je král uprostřed, nahoře, dole, vlevo, vpravo, nebo v jednotlivých rozích. Statická tabulka pro krále se v této fázi hry nemění, čím blíže k rohu je, tím lepší bonus získá. Občas se stane, že bílý hráč se mačká u špatného rohu a netuší, že je už nepřekonatelně zablokován, protože to kvůli zjednodušení nezkoumám. Černé figury mají obdobný bonus za blízkost ke králi a navíc, bez ohledu na jeho polohu, jsou velmi silně ohodnoceny 3 pozice u rohu tvořící nepřekonatelnou hradbu. Hráč může raději obětovat figuru, než přijít o tuto pozici.



Obrázek 4.4: Blokáda rohu pouhými třemi černými figurkami

U koncové hry černého proti osamělému králi, se stávalo, že černý hráč nebyl schopen a ani se nepokusil zajmout krále. Proto je tabulka pro krále přepsána na zvýhodnění středových pozic a černé kameny se snaží vytvořit na desce diagonálu. Pokud je král pod diagonálou u pravého dolního rohu, tak černý hráč ztrácí bonus za blokování protilehlého rohu a naopak.

Vygeneruji všechny možné tahy hrajícího hráče a k jejich počáteční ceně přičtu nepatrné náhodné číslo. Tím dosáhnu toho, že pokud existuje několik nejlepších variant, např. na počátku hry zrcadlově shodné tahy, tak nebude pokaždé vybrána ta varianta,

kterou generátor vytvoří jako první. A to navíc bez dodatečné ceny, kterou by se muselo zaplatit, pokud by se prohledávalo s oknem, kdy alfa bude neustále snížená o jedna než je hodnota nejlepšího tahu. Pak by se zjistily všechny nejlepší tahy, ale výpočet bude trvat déle.

Seřadím tahy podle počáteční ceny a začnu prohledávat do hloubky 0 od prozatím nejlepšího tahu. Najdu-li lepší, ihned ho umístím v seznamu nahoru. V prohledávání ve vyšší hloubce bude první nebo na lepších pozicích, najdu-li ještě lepší. Tím částečně udržuji nejlepší tahy nahoře, aby se prohledávali jako první. Není to úplně dokonalé, protože pokud druhý nejlepší tah má stejné ohodnocení jako nejlepší, nebude nalezen. Nemusí se dát odlišit od průměrného, okno totiž může vrátit hodnotu shodnou s alfa.

Hloubka hledání se postupně zvyšuje, dokud nenarazí na limit nebo nevyprší čas. Potom se vrátí tah ležící na vrcholu seznamu tahů.

4.5 Ostatní

4.5.1 Zjištění nejlepší cesty

Kromě výběru nejlepšího tahu umí program zjistit i nejpravděpodobněji hranou cestu až k listu obsahující hodnotu tohoto nejlepšího tahu, tedy nejlepšího tahu hráče, nejlepšího protitahu soupeře, atd.

Dosahuje se toho pomocí pole, které obsahuje pro každou hloubku seznam pokračujících tahů. Např. jsme-li v hloubce 3 a nalezli jsme prozatím nejlepší variantu, tak zkopírujeme z právě volané hloubky 4 připravenou cestu a přidáme před ní náš tah. Cesta se musí zkopírovat ihned, protože při zkoumání dalšího tahu v hloubce 3, bude v hloubce 4 uložena jiná cesta, která je odpovědí na nový tah z hloubky 3. Před ukončením prohledávání v hloubce 3 máme tedy pro hloubku 2 nejlepší cestu připravenou. Takto neustále kopírujeme a přepisujeme nejlepší cesty, dokud se nevrátíme ke kořeni.

4.5.2 Hodnoty statických tabulek

Pomocí statických tabulek dosahuji slabou strategickou hru počítače. Vhodnou volbou hodnoty statických tabulek a cenou figur, lze dosáhnout rozdílného pojetí hry. Černý hráč se snaží o blokádu všech rohů i za cenu své figury. Rohy blokuje diagonálou tří kamenů tvořící s rohem pravoúhlý trojúhelník 3 x 3. Bílý hráč se snaží dosáhnout rohu a podle umístění krále se zlepšuje nebo zhoršuje bonus pro jeho kameny. V koncovce černého proti králi vytváří černý diagonálu a snaží se krále vytlačit ze středu, kde má nejvyšší ohodnocení a zamknout ho na jedné polovině desky. Pak může uvolnit figury blokující teď už nepřístupný roh a zvyšit pravděpodobnost, že černý zahlédne mat.

Cena figur je zvolena 75 pro bíleho a 50 pro černého. Černých kamenů je sice 2x více, ale pořebuje aspoň 12 kamenů, aby trvale zablokoval roh, případně 10, pokud hraje už jen proti králi. Bílý hráč se nedostane k žádnému rohu, aniž by neobětoval kámen, a naopak černý nezablokuje všechny čtyři rohy, aniž by si nenechal něco vzít. Proto zde nemá materiál úplnou převahu nad statickou pozicí. Statické ohodnocení černé figury, která blokuje roh, je rovna ceně černé figury. V pokročilé fázi hry se díky tomu stává, že bílý hráč obětuje zbytečně svou figuru, protože v zajímavých tazích neobsahují pohyby figur z malé barikády a opačně. Bílý hráč si tedy myslí, že černý se už nemůže vrátit, protože bracím tahem se ukončuje prohledávání.

Kapitola 5

Testování

Tato kapitola popisuje průběh testování aplikace a shrnuje získané poznatky.

5.1 Testování umělé inteligence s různým časem

Při tomto testování byly opakovaně proti sobě spouštěny umělé inteligence s různými časovými limity. Pro každý časový limit bylo odehráno 100 partií. Hloubka prohledávání byla pro obě barvy nastavena na nedosažitelnou hodnotu, konkrétně 20. Následující tabulka ukazuje procentuální úspěšnost černého hráče proti bílému, dále zobrazuje průměrnou délku hry v půltazích a medián z délky her v půltazích vyhrané danou barvou.

Černý/Bílý		**1 s**	**4 s**	**16 s**
1 s	Černý vyhrál	52 %	17 %	20 %
	Průměr	457	324	445
	Medián bílý	94	82	70
	Medián černý	382	321	1171
4 s	Černý vyhrál	71 %	54 %	22 %
	Průměr	148	300	160
	Medián bílý	77	78	68
	Medián černý	141	188	279
16 s	Černý vyhrál	80 %	50 %	80 %
	Průměr	167	144	335
	Medián bílý	58	74	92
	Medián černý	138	179	334

Tabulka 5.1: Testování umělé inteligence s různým časem

5.2 Testování umělé inteligence s různou hloubkou

V rámci tohoto testování byly opakovaně proti sobě spouštěny umělé inteligence, tentokrát s různými maximálními hloubkami. Pro každou hloubku bylo odehráno opět 100 partií. Následující tabulka taktéž ukazuje procentuální úspěšnost černého hráče proti bílému.

Černý/Bílý		**0**	**1**	**2**
0	Černý vyhrál	72 %	0 %	0 %
	Průměr	791	178	221
	Medián bílý	292	72	68
	Medián černý	609	-	-
1	Černý vyhrál	83 %	3 %	0 %
	Průměr	668	174	57
	Medián bílý	84	78	50
	Medián černý	515	340	411
2	Černý vyhrál	100 %	57 %	83 %
	Průměr	96	113	172
	Medián bílý	86	86	108
	Medián černý	94	115	163

Tabulka 5.2: Testování umělé inteligence s různou hloubkou

5.3 Testování umělé inteligence proti člověku

Každému hráči byly před samotným testováním důkladně vysvětlena pravidla Hnefataflu a předvedeno ovládání programu. Každý hráč pak samostatně odehrál 5 partií za černého a 5 partií za bílého hráče u každého konkrétního nastavení. Hráči byly různého pohlaví, věkový průměr 22,6 let. V následujících tabulkách je uvedena procentuální úspěšnost umělé inteligence při daném nastavení.

	hloubka = 1	hloubka = 2	čas = 1 s	čas = 2 s
Hráč 1	80 %	100 %	100 %	100 %
Hráč 2	80 %	100 %	80 %	100 %
Hráč 3	40 %	80 %	20 %	80 %
Hráč 4	80 %	60 %	60 %	100 %

Tabulka 5.3: Testování umělé inteligence proti člověku - černý hráč

	hloubka = 1	hloubka = 2	čas = 1 s	čas = 2 s
Hráč 1	80 %	80 %	100 %	80 %
Hráč 2	60 %	80 %	80 %	100 %
Hráč 3	40 %	80 %	60 %	60 %
Hráč 4	60 %	60 %	80 %	80 %

Tabulka 5.4: Testování umělé inteligence proti člověku - bílý hráč

Po skončení testování měli hráči stručně popsat své dojmy z programu. Všichni pochválili grafické rozhraní, především intuitivní ovládání programu, většina by si hru opět zahrála.

Kapitola 6

Závěr a zhodnocení

6.1 Závěr

V rámci této práce jsem se zabývala vytvořením umělé inteligence pro vikingskou deskovou hru Hnefatafl a následně naprogramováním přehledného a přívětivého uživatelského rozhraní. Tohoto cíle jsem dosáhla a aplikace tak splňuje všechny požadavky vytyčené v zadání.

Na začátku jsem nastudovala algoritmy z oblasti umělé inteligence a pravidla Hnefataflu. Na základě těchto poznatků jsem navrhla algoritmy, které jsem dále optimalizovala, aby co nejvíce vyhovovaly této konkrétní hře.

Jako základní kámen umělé inteligence jsem použila algoritmus Alfa-Beta s dohledáváním do klidové pozice, který byl mírně upraven tak, aby vyhovoval požadavkům této hry. Další velkou částí tohoto projektu bylo vytvoření vhodných ohodnocovacích funkcí, které byly uzpůsobeny potřebám jednotlivých barev. Při testování těchto funkcí jsem například zjistila, že ve hře existuje hranice, do které téměř bez vyjímky vyhrává bílý a od které naopak vyhrává především černý.

Nejvýraznější optimalizací použitou v programu bylo využití bitových masek a operací nad nimi při generování tahu. Toto pomohlo program výrazně zrychlit a provádět tak vypočet do větší hloubky. Další optimalizací byla prevence zacyklení.

Dále jsem naprogramovala aplikaci, která umožňuje pohodlné hraní Hnefataflu, ať už proti jinému hráči nebo proti umělé inteligenci, případně aplikace umožňuje hru dvou umělých inteligencí proti sobě. Tuto aplikaci jsem otestovala s několika hráči. Umělá inteligence při tomto testování měla úspěch a v drtivé většině případů živého hráče porazila.

Přínosem této práce pro mě bylo prohloubení znalostí o tvorbě umělé inteligence pro deskové hry a také zlepšení se v jazyce C++. Dalším benefitem by mohlo být získání rozsáhlých informací o hře, která není tak běžná a rozšířená jako například šachy.

6.2 Zhodnocení a návrhy na vylepšení

Přestože program splňuje vytyčené požadavky, lze najít několik oblastí, kde může být rozšířen nebo vylepšen. Jednou z těchto oblastí je uživatelské rozhraní. V současné chvíli není implementována možnost vrácení tahu nebo uložení a nahrátí aktuální pozice. Taktéž by mohl být přínosný modul pro ukládání statistik již odehraných her s možností porovnání s jinými hráči.

Protože v tuto chvíli aplikace plně podporuje lokální hru hráče proti hráči, hráče

proti umělé inteligenci a hru dvou inteligentních soupeřů, je jedním z dalších možných rozšíření podpora síťové hry. Ať formou webového rozhraní nebo vytvořením serveru běžícího v rámci aktuálně lokálně spuštěné aplikace.

Dalším možným rozšířením je implementace ostatních variant taflových her, především těch, které ke hře vyžadují kostku či obsahují speciální pravidla (např. Alea Evangelii).

Posledním vhodným rozšířením, které se přímo dotýká implementované umělé inteligence, je použití hashovací tabulky. Vymyšlením vhodné hashovací funkce by se mohl urychlit výpočet a umělá inteligence by tak mohla prohledávat stavový prostor do větších hloubek.

Literatura

[1] Negamax [online]. http://en.wikipedia.org/wiki/Negamax.

[2] Tafl games [online]. http://en.wikipedia.org/wiki/Tafl.

[3] HELMFRID, S.: Hnefatafl: The Strategic Board Game of the Vikings. 2005.

[4] HRUBÝ, M.: Doprovodné texty ke kurzu Teorie her. zima 2010, skripta do předmětu THE.

[5] KOŠNER, L.: Hnefatafl - stará severská hra [online]. http://www.skandinavskydum.cz/cs/hnefatafl-stara-severska-hra.

[6] MILLAR, T.: Hnefatafl - tactics and strategy [online]. http://www.tim-millar.co.uk/section509308.html.

[7] de NICÉE, B.: HNEFATAFL - The Viking Game [online]. http://www.gamecabinet.com/history/Hnef.html, [cit. 2012-05-13].

[8] STEINWENDER, D.; FRIEDEL, F.: *Šachy na PC*. UNIS, 1996, ISBN 3-87791-522-1.

[9] ZBOŘIL, F.; ZBOŘIL, F.: Základy umělé inteligence - studijní opora. březen 2006, skripta do předmětu IZU.

Příloha A

Obsah CD

- Zdrojové kódy
- Text této práce