Středoškolská odborná činnost

Umělá inteligence pro hru Pentago

Petr Kouba

Pardubice 2010

Středoškolská odborná činnost

Obor 18 – Informatika

Umělá inteligence pro hru Pentago

Artificial intelligence for Pentago

Autor: Petr Kouba

Škola: Gymnázium, Pardubice, Dašická 1083

Konzultant: -

Pardubice 2010

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně, použil jsem pouze podklady (literaturu, SW atd.) citované v práci a uvedené v přiloženém seznamu a postup při zpracování práce je v souladu se zákonem č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) v platném znění.

V Pardubicích dne …… *podpis: ……*

Poděkování

Děkuji všem pedagogům a studentům za pomoc při testování programu, především pak Ing. Milanu Volejníkovi a RNDr. Vladimíru Víchovi za podnětné připomínky, které mi během práce poskytovali.

Anotace

Tato práce popisuje tvoření umělé inteligence a grafického rozhraní pro hru Pentago. V první části je krátký úvod do pravidel hry Pentago a umělé inteligence. Následuje podrobný popis mnou vytvořeného programu. Na závěr je uveden souhrn všech poznatků a náhled budoucího vývoje.

Práce by měla posloužit všem, kdo se zajímají o problematiku umělé inteligence deskových her. Přečtením dokumentu by měl být každý poměrně rychle obeznámen s programem, principy použité umělé inteligence a hrou Pentago.

Klíčová slova: pentago, umělá inteligence, piškvorky

Annotation

This paper describes the process of creating an artificial intelligence and a graphic user interface for Pentago. The first part consists of Pentago rules and algorithms artificial intelligence. This is followed by detailed description of the program I created myself. Finally all the findings are summed up and plans for future development are set.

Paper should be useful for everyone interested in implementation of artificial intelligence for board games. After reading the paper everyone should be familiar with the program, principals of used artificial intelligence and the game Pentago.

Key words: Pentago, artificial intelligence, 5-in-a-row

Obsah

Slovník pojmů a zkratek

Zkratka	Pojem	Česky
AI	artificial inteligence	umělá inteligence
GUI	graphical user interface	grafické uživatelské rozhraní
-	benchmark	výkonnostní test
-	implementace	zavádění
-	array	pole
-	multithreading	paralelizace na více jader
VB.NET	Visual Basic .NET	(programovací jazyk)
-	On_Click	při kliknutí
-	On_Enter	při vstupu
-	On_Exit	při výstupu

Úvod

V této části stručně popíši, co je Pentago.

Pentago

O hře

Pentago je abstraktní strategická hra pro 2 hráče vytvořená Tomasem Flodénem. Práva na vývoj a komercializaci hry vlastní švédská společnost Mindtwister. Hra byla zatím oceněna 8 cenami, mezi jinými: Hra roku 2005 ve Švédsku, Hra roku 2006 ve Francii, Vítěz Mensa Select 2006.

Pravidla

První hráč položí kuličku své barvy do libovolného důlku na herním poli. Poté otočí jednu ze čtyř herních desek o 90 stupňů (ve směru hodinových ručiček nebo proti němu). Pak úplně stejně pokračuje jeho protihráč, tzn. položí kuličku do důlku a otočí jednou z desek. Jednou položená kulička se už nesmí přemístit. Vyhrává ten hráč, kterému se jako prvnímu podaří vytvořit řadu pěti kuliček vlastní barvy. Řada pěti kuliček může být vodorovná, svislá nebo úhlopříčná a může vést i přes dvě nebo tři herní desky. Když hráč položením kuličky do důlku docílí vítězné řady pěti stejných kuliček, herní deskou už neotáčí. Pokud hráč vytvoří řadu až otočením desky, jeho protihráč smí ještě položit kuličku. V tom případě může nastat situace, že oba hráči docílí vítězné řady - hra pak končí nerozhodně. Stejně jako v případě, kdy hráči položí všechny své kuličky, aniž by se některému z nich podařilo vytvořit vítěznou řadu.

Umělá inteligence

Použitý typ AI - aneb prohledávání stavového prostoru.

Jeho princip spočívá ve vhodném procházení stavů řešené domény za účelem nalezení požadovaného stavu. Nejdříve definujeme stavový prostor jako všechny možné hratelné tahy a požadovaný stav jako pětku (je to složitější než v piškvorkách, protože jsou zde obsaženy rotace a pravidlo pro remízu).

Algoritmy AI

Naive

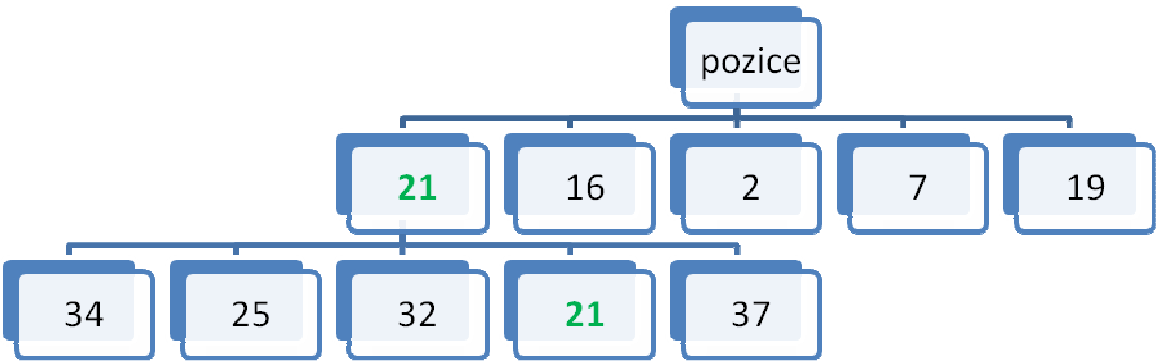
Vlastní algoritmus velmi jednoduchý. Vygenerujeme ze zadané pozice všechny tahy a zkoušíme je jeden po druhém zahrát. Vzniklou pozici vždy ohodnotíme statickou ohodnocovací funkcí a tah vrátíme. Průběžně si pamatujeme nejlepší tah a rovněž hodnotu pozice, ke které vede. Po otestování všech vygenerovaných tahů vrátíme ten nejlepší.

Obrázek 1 – naivní algoritmus

Minmax

Je zlepšením Naivního algoritmu. Místo statické ohodnocovací funkce po každém zahraném půltahu zavoláme kód, který vyzkouší všechny odpovědi soupeře na náš tah. Jednu po druhé zahraje, ohodnotí je statickou ohodnocovací funkcí, zahraje zpět a zapamatuje si hodnotu nejlepšího tahu, tentokrát z hlediska soupeře. Za cenu konkrétního našeho tahu z úvodní pozice je tak prohlášena cena nejlepší soupeřovy odpovědi.

Zobecněný algoritmus, který počítá do obecné hloubky n určené parametrem, se jmenuje minimax, neboť když se za jednoho hráče snažíme v propočtu maximalizovat cenu pozice, za druhého ji tím minimalizujeme a o půltah dále zase naopak. Implementuje se obvykle pomocí jedné rekurzivní funkce minimax, která má jako parametry pozici a hloubku propočtu, návratovou hodnotou je cena pozice po propočtu do dané hloubky. Místo statické ohodnocovací funkce HodnotaPozice volá minimax sám sebe do hloubky o jedničku menší. Pouze při hloubce nula zavolá HodnotaPozice a v koncových pozicí vrací konstanty pro výhru/prohru/remízu a v propočtu příslušné varianty nepokračuje.

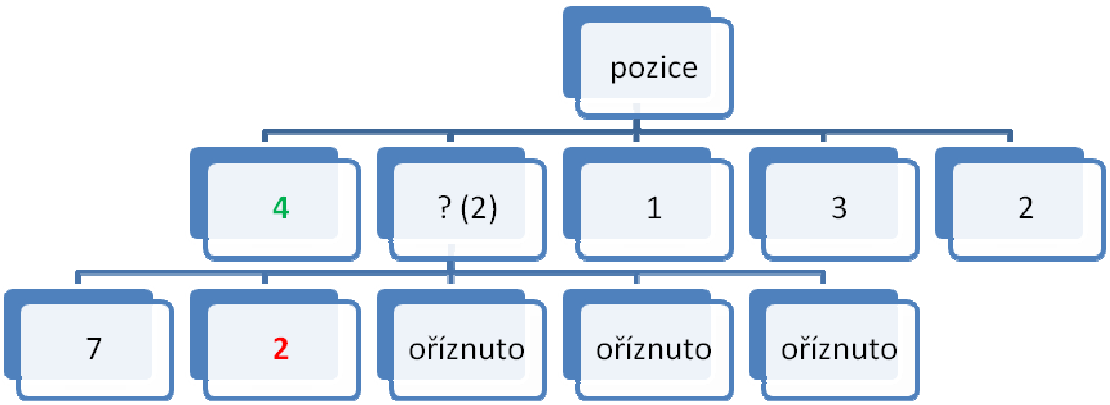


Obrázek 2 – Algoritmus minimax

Alfabeta

Alfa-beta ořezávání (angl. alpha-beta pruning) je vylepšení algoritmu minimax, které v průměrném případě zrychluje jeho běh. Je založeno na pozorování, že pokud právě zpracovávaný půltah už nemůže obstát v konkurenci s jiným, nemusíme dál prohledávat jeho důsledky. Metoda byla pojmenována alfa-beta, protože v ní rozšiřujeme původní minimaxový algoritmus o dvě další hodnoty alfa a beta, které nám určují potřebné meze.

Účinnost ořezání nepotřebných větví lze zvýšit vhodnou volbou pořadí, v němž jsou procházeny. Nejvýhodnější je projít nejprve ty větve, které by měly dát podle nějakého rychlého (ve srovnání s procházení kompletním minimaxem) odhadu nejlepší výsledky. Pokud budeme mít při volbě pořadí procházení tahů smůlu, alfa-beta ořezávání běh nezrychlí vůbec.



Obrázek 3 – Algoritmus alfabeta

Paměťová a časová náročnost

Propočítávání bohatých herních stromů s sebou přináší značné nároky na výkon. Pokud vezmeme třeba větvící faktor (to je kolik tahů lze z dané pozice provést), tak ten je pro prázdnou desku bez optimalizací 288 (36 tahů* 8 rotací), poté se každým tahem snižuje o 8 (1 tah* 8 rotací).

Obrázek 4 - Příklad propočtové náročnosti, algoritmy AI

Algoritmus	Situace (Tah)	Počáteční větvící faktor	Pozic k propočtu
Naive (hloubka1)	1	288	288
(hloubka1)	15	168	168
Minmax (hloubka2)	1	288	80.640
(hloubka3)	15	168	4.085.760
Alfabeta (hloubka2)	1	288	20.160
(hloubka3)	15	168	517.440

Zdroj: Přibližné výpočty.

Hlavním faktorem při procházení stromů je rychlost. Program nyní dosahuje rychlosti až 400.000 pozičních propočtů za sekundu. Této rychlosti bylo dosaženo optimalizací (kapitola - Provedené optimalizace), dále se domnívám, že rychlost lze ještě vylepšit.

Paměťová náročnost programu je minimální, pro příklad:

Proměnná reprezentující desku je Array typu Byte o velikosti 36 -> 36 Bytů.
Proměnná reprezentující možné tahy je Array typu Short o velikosti maximálně 288 -> 2,3 kBytů.

Postup při tvorbě programu

Jazyk, vývojové prostředí

Zvolil jsem jazyk Visual Basic .NET, protože už jsem ho na dobré úrovni ovládal a připadal mi srozumitelný.

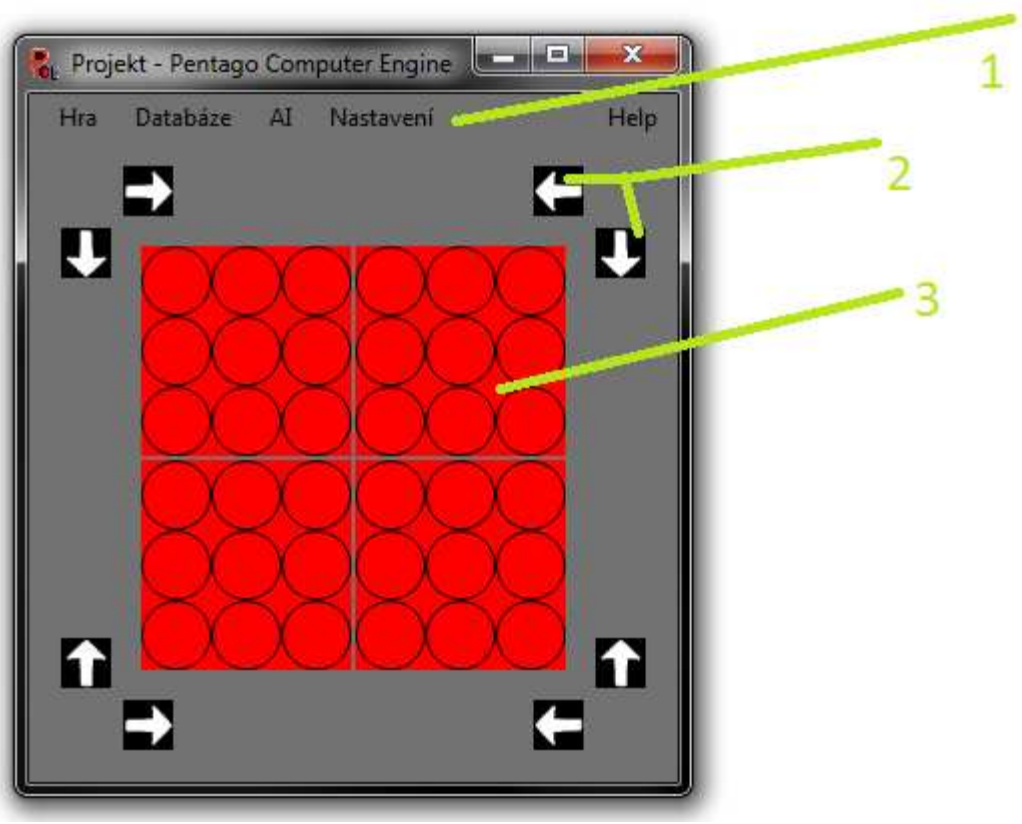
Jazyk využívá sady knihoven .NET Framework. .NET Framework umožňuje rychlou tvorbu aplikací připojených k datům, s nimiž jsou koncoví uživatelé velmi spokojeni. Poskytuje totiž stavební bloky (prefabrikovaný software) pro řešení častých programátorských úkolů. Připojené aplikace využívající .NET Framework efektivně modelují procesy podnikatelské sféry a usnadňují integraci systémů v heterogenních prostředích.


Na programování používám Visual Studio od Microsoftu. Visual Studio poskytuje pokročilé vývojové nástroje, ladicí funkce, funkce pro práci s databázemi a vynalézavé novinky pro rychlou tvorbu špičkových aplikací různých typů. Přináší četná vylepšení: vizuální návrháře pro rychlejší vývoj s využitím .NET Framework 3.5, podstatná zdokonalení webových vývojových nástrojů a vylepšení jazyka, která zrychlují vývoj pro data všech typů.

Zdroje informací

Prvním zdrojem informací byly články na internetu a knihy o programování ve VB.NET. Dále pak články o piškvorkách a tvorbě šachové AI. Dále pak servery microsoftu (dokumentace konkrétních funkcí a optimalizací).

Tvorba GUI



Obrázek 5 – GUI (1-Menu, 2-Reprezentace rotací, 3-Reprezentace hracího pole)

Grafické rozhraní je nejdůležitější částí programu z hlediska uživatelské přívětivosti, proto jsem se snažil udělat rozhraní co nejvíce intuitivní.

Rozhraní je vytvořeno pomocí návrhářské aplikace Visual Studia. Pro indikaci uživelské aktivity a provádění úkonů jsou používané hlavně eventy myši:

On_Click (při kliknutí)– provedení tahu, rotace

On_Enter (při vstupu do prvku)– náhled rotace, náhled tahu

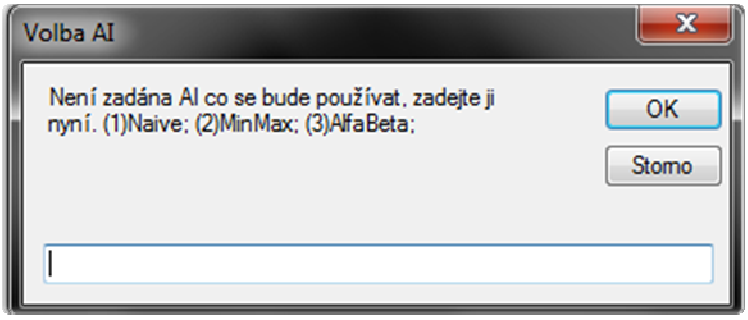
On_Exit (při výstupu z prvku)– návrat do původního stavu před náhledem

Menu

Menu je klasického typu známého z MS Office.

1. Hra
   1.1. Nová hra
      1.1.1. Jeden hráče

Začne hru proti zvolené umělé inteligenci.



Obrázek 6 – Volba AI

1.1.1.1. Za bílé
1.1.1.2. Za černé

1.1.2. Dva hráči

Umožňuje duel 2 živých hráčů.

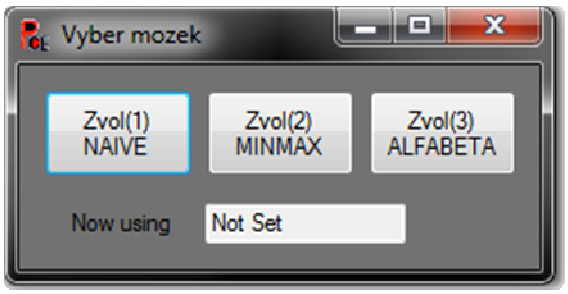
1.1.3. AI vs. AI

Duel umělé inteligence.

1.2. Uložit/Načíst hru (zatím není plně funkční)
1.3. Konec

2. Databáze (zatím není funkční)
3. AI
   3.1. Vybrat mozek

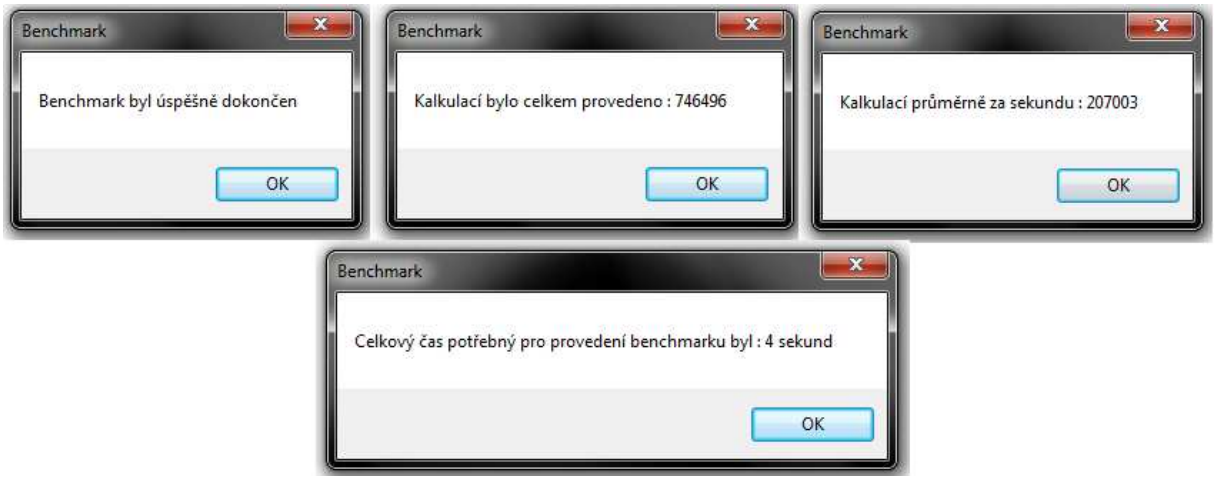
Menu výběru algoritmu AI a případné hloubky propočtu.



Obrázek 7 – Vyber mozek

3.2. Výpis přemýšlení (zatím není funkční)
3.3. Benchmarking

Provede základní úkony AI a vypíše hodnoty jako počet provedených kalkulací a čas výpočtu, dále pak hlavní hodnotu – počet pozičních kalkulací za sekundu



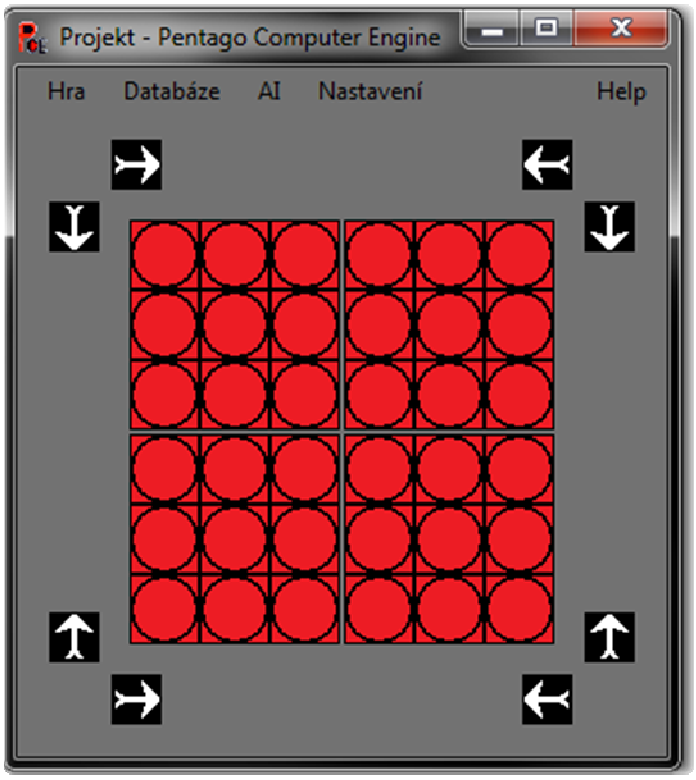
Obrázek 8 - Benchmark

3.4. AI Override

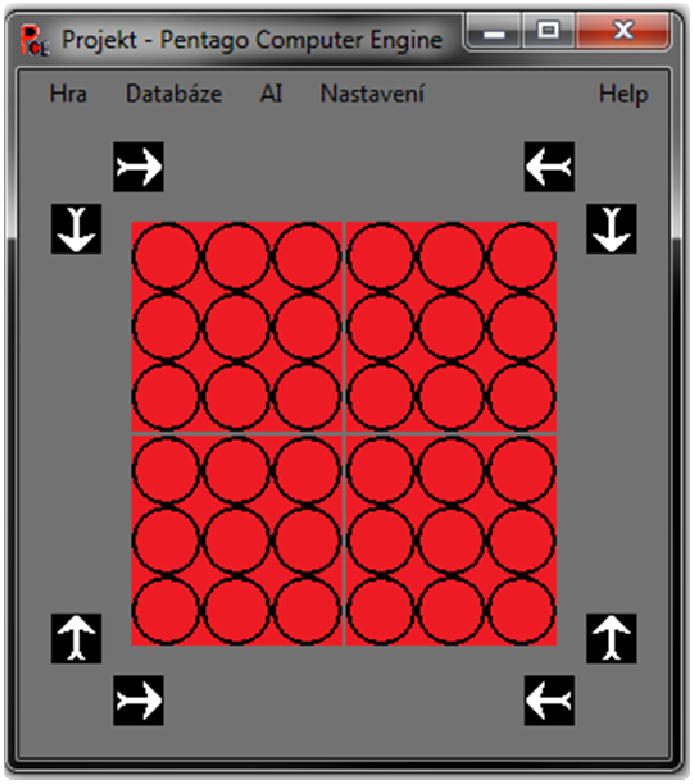
Umožňuje donutit AI zahrát tah za uživatele.

4. Nastavení
   4.1. Nastavení barev pozadí
   4.2. Nastavení grafických sad

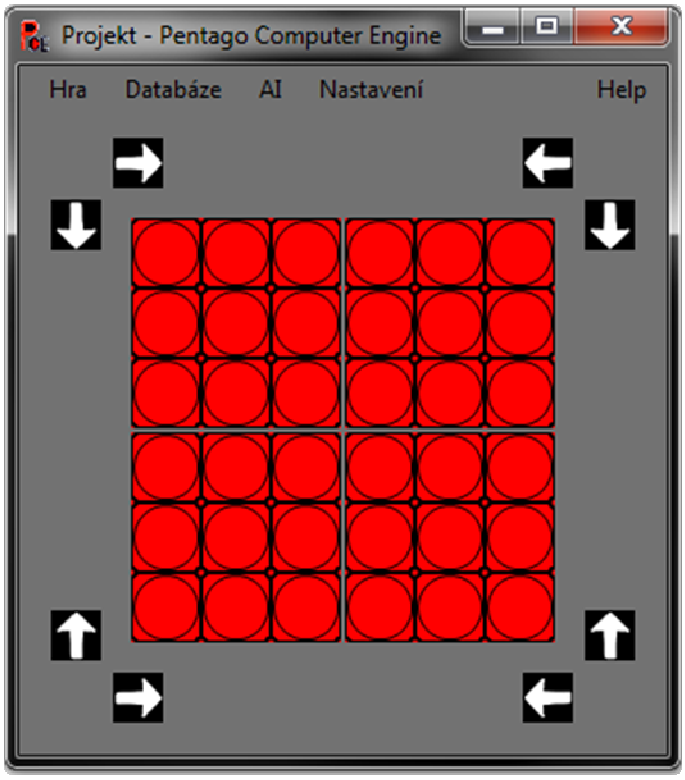
Prozatím existují 4 grafické sady.



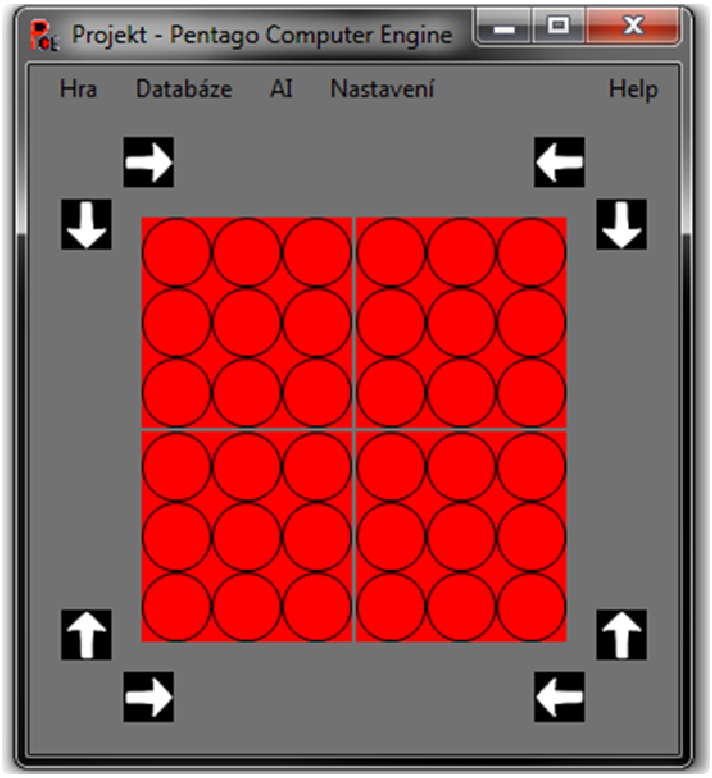
Obrázek 9 - Klasický grafický set



Obrázek 10 - Klasický grafický set bez linek



Obrázek 11 - Moderní grafický set



Obrázek 12 - Moderní grafický set bez linek

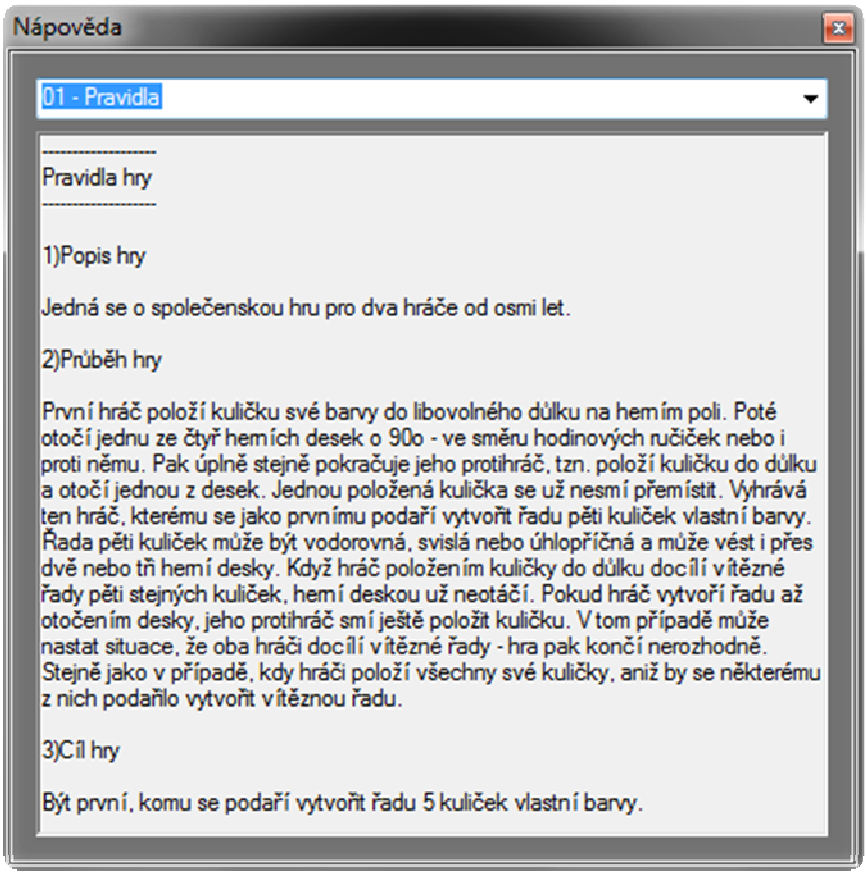
4.3. Resetování cesty k .exe

Užitečné při kopírování souborů aplikace do jiného umístění.

5. Help

5.1. Nápověda

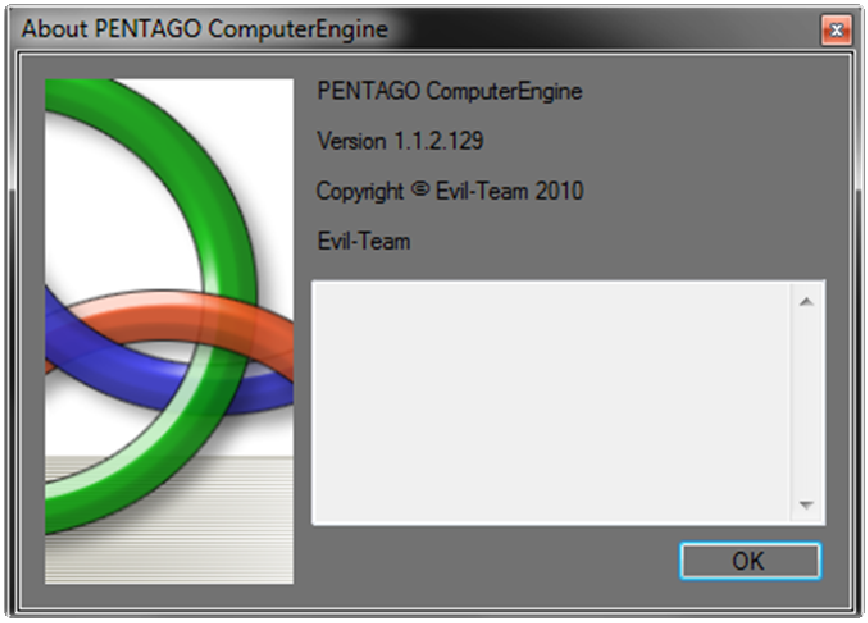
V současné době obsahuje pravidla hry.



Obrázek 13 – Nápověda

5.2.O Programu

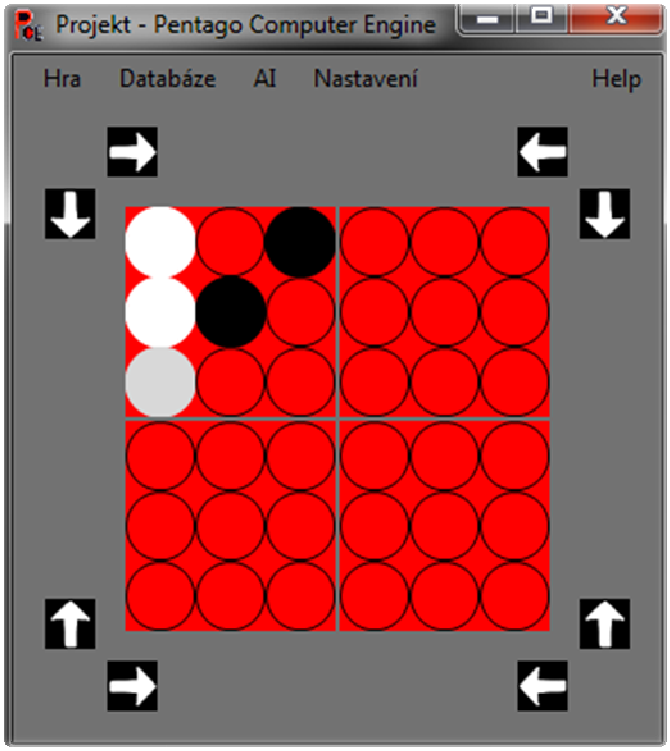
Vypíše současnou verzi programu.



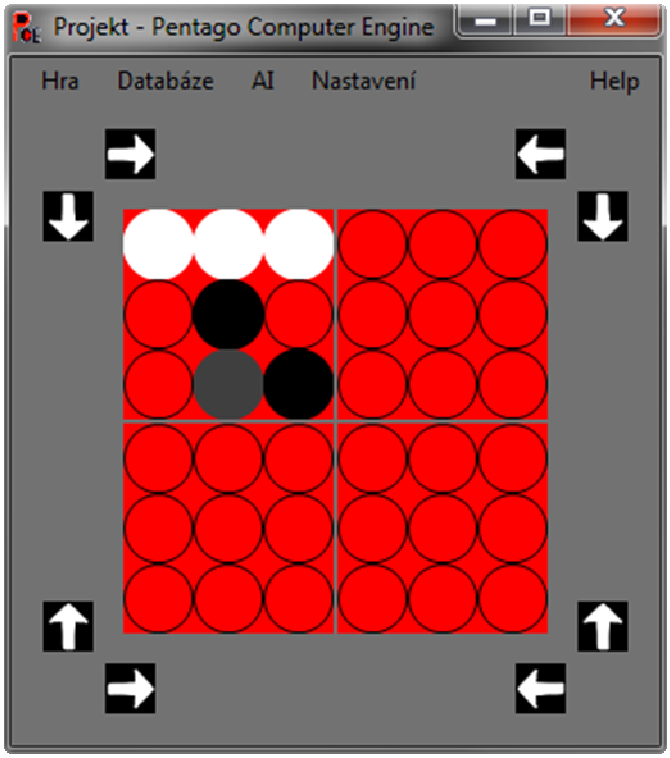
Obrázek 14 – O Programu

Usnadnění

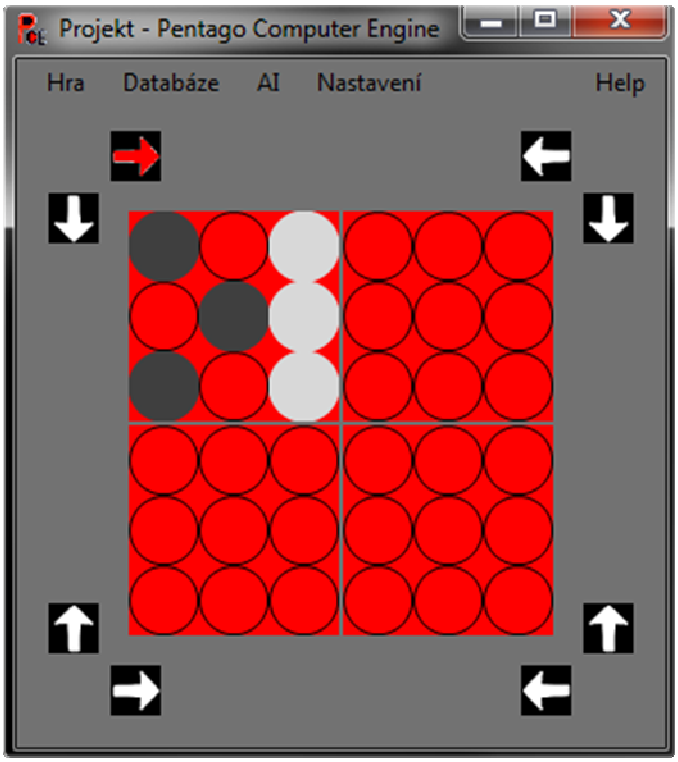
Protože je hra na člověka z hlediska představivosti značně složitá, řeší grafické rozhraní některé tyto limitace užitečným náhledem tahu před provedením.



Obrázek 15 – Tah bílého (Tmavší bílá představuje pole, kam plánuje bílý táhnout.)



Obrázek 16 – Tah černého (Světlejší černá přestavuje pole, kam chce táhnout černý.)



Obrázek 17 – Rotace (Zobrazení rotace před provedením, konkrétně rotace vpravo.)

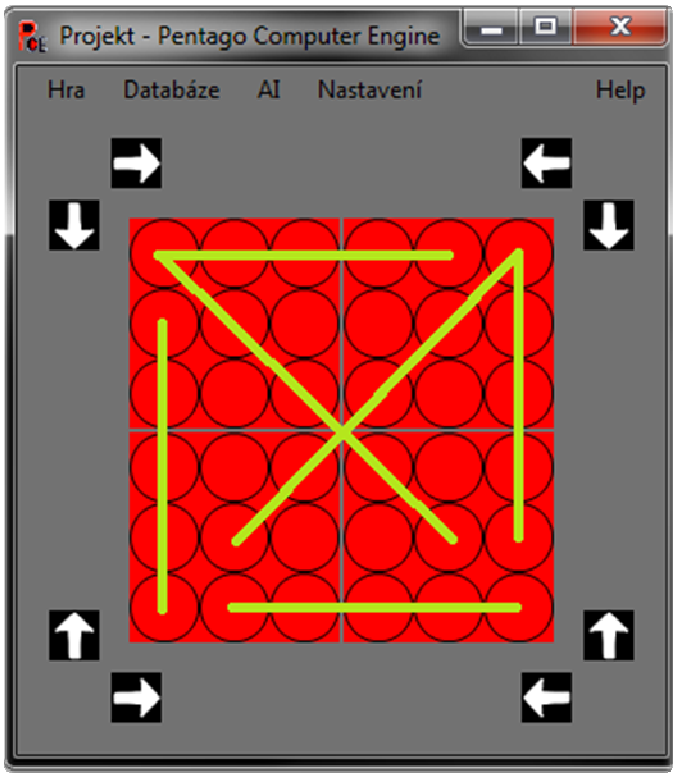
Tvorba AI

Implementace pravidel

Pro korektní funkci AI musela být implementována pravidla hry, obvzláště remíza: Pokud hráč vytvoří pětku až rotací, tak se změní hodnota proměnné HracCekaNaVyhru na hodnotu hráče a v dalším tahu, pokud hráč neudělá pětici položením kuličky, je voláno hlášení výhry hráče s pěticí a zastavena hra. Dále pak je volána remíza při zaplnění herní plochy.

Hodnotící funkce

Při každém tahu počítače jsou hodnoceny všechny pozice, do kterých může počítač táhnout. Každá pozice je hodnocena určitým číslem hodnotícím danou pozici z hlediska výhodnosti. Hodnocení se skládá z hodnocení pětek. Vezmou se všechny možné pětice, které se mohou na hrací ploše naskládat. Každá taková pětice je ohodnocena nezávisle. Celé hrací pole je ohodnoceno součtem ohodnocení všech pětic, které jím procházejí.



Obrázek 18 - Ukázka pětic – celkem je těchto pětic 32.

Každá pětice je ohodnocena podle toho, kolik je v ní značek 1. nebo 2. hráče. Čím víc tam má jeden hráč zabraných polí, tím větší hodnotu pětice má (pokud je pětice hráče na tahu je hodnocení kladné, pokud protihráče je záporné). Pokud jsou v pětici současně značky obou hráčů, má hodnotu 0, protože z ní už vítězná pětice být nemůže. Pokud je prázdná, má malou hodnotu.

Hodnoty pětic zaplněných jedním hráčem jsou konstanty. Postupně se exponenciálně zvyšuje jejich hodnota tak, aby nedošlo k tomu, že 2 trojky jsou víc než 1 čtyřka a podobně.

Implementace algoritmů

Úspěšně byly implementovány algoritmy Naivní, Minimax, Alfabeta. Nejzajímavější je implementace Alfabety:

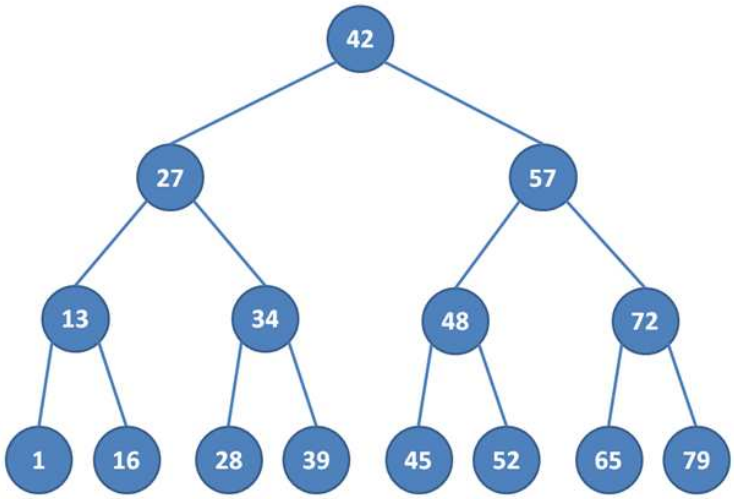
```
Function AlfaBetaAI_NejlepsiTah()

        ->Zjisti remízu, nebo výhru ->dál nepočítej
        ->Genreruj možné tahy
        ->Redukuj možné tahy

        alfa = MinusNekonecno

        ->Pro všechny tahy:
              AI_ZahrajTah()
              cena = -AlfaBeta_Rekurze(alfa, beta)
              AI_OdHrajTah()
              If (cena > alfa) Then
                    alfa = cena
                    ->indexNejlepsihoTah

    Return Tahy(indexNejlepsiho)

End Function


Function AlfaBeta_Rekurze()

        ->Zjisti remízu, nebo výhru ->dál nepočítej

        If (Hloubka <= 0) Then Return HodnotaPozice

        ->Genreruj možné tahy
        ->Redukuj možné tahy

    ->Pro všechny tahy:
              AI_ZahrajTah()
              cena = -AlfaBeta_Rekurze(alfa, beta)
              AI_OdHrajTah()
              If (cena > Alfa) Then
                    Alfa = cena
                    If (cena >= Beta) Then
                    'Zde je právě to ořezání.
                    Return Beta
                    End If
              End If

    Return Alfa

End Function
```

Provedené optimalizace

- Větvená podmínka If_Then místo Select_Case

Anebo použití binárního vyhledávacího stromu. Použitím stromu se zkracuje čas pro vyhodnocení mnohonásobné podmínky. V podmínce typu select_case je přiřazena všem možnostem

stejná priorita -> jsou testovány popořadě (to znamená, že položky na konci listu podmnínek jsou vyhodnoceny za dlouho). Binární strom zkracuje čas k nalezení správné varianty.



Obrázek 19 – Binární strom

Například tento strom je koncipován tak, že nalezení jakéhokoliv prvku trvá maximálně 4 kroky, kdežto při použití select_case by to trvalo až 15 kroků.

- Vyřazení zápisu výsledků při výpočtech

  Přehnaně časté volání vykreslování mělo negativní důsledky na výkon.

- Použití proměnných co nejmenších typů – byte, short, boolean

Šetřit se má všude, i když jde jen o pár kBytů. Tato optimalizace nemá vliv na výkon, ale na paměťovou náročnost.

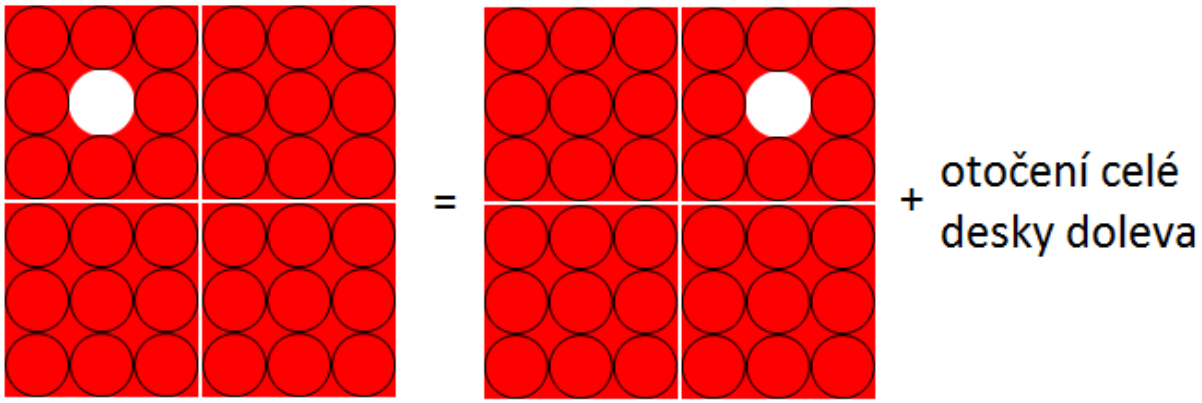
- Inteligentní používání zanořených cyklů s podmínkami

  Sjednocování cyklů, provádění podmínek ve vnějších cyklech.

- Refaktorizací kódu

  Dle standardů jazyka Visual Basic vydaných Microsoftem.

- Brute-force redukce při generaci tahů pro alfabetu a minimax



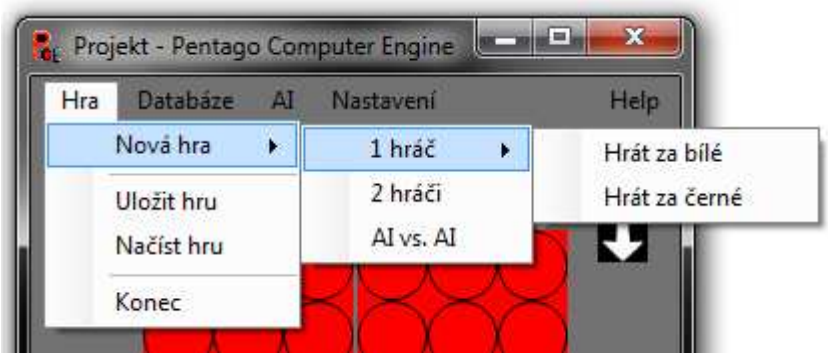
Obrázek 20 – Redukce symetrií

Tato redukce využívá symetrie desek (->nejlepší výsledky ze začátku hry, po pár tazích je téměř nepoužitelná). Redukční algoritmus zkoumá, jestli 2 tahy po provedení nejsou náhodou stejné. Na začátku hry zredukuje významně větvící faktor z 288 na pouhých 9.

Funkce programu

Mezi funkce programu zatím patří:

- hra dvou hráčů na jednom počítači
- hra jednoho hráče proti počítači
- manuální analýza pozice počítačem



Obrázek 21 – Funkce programu

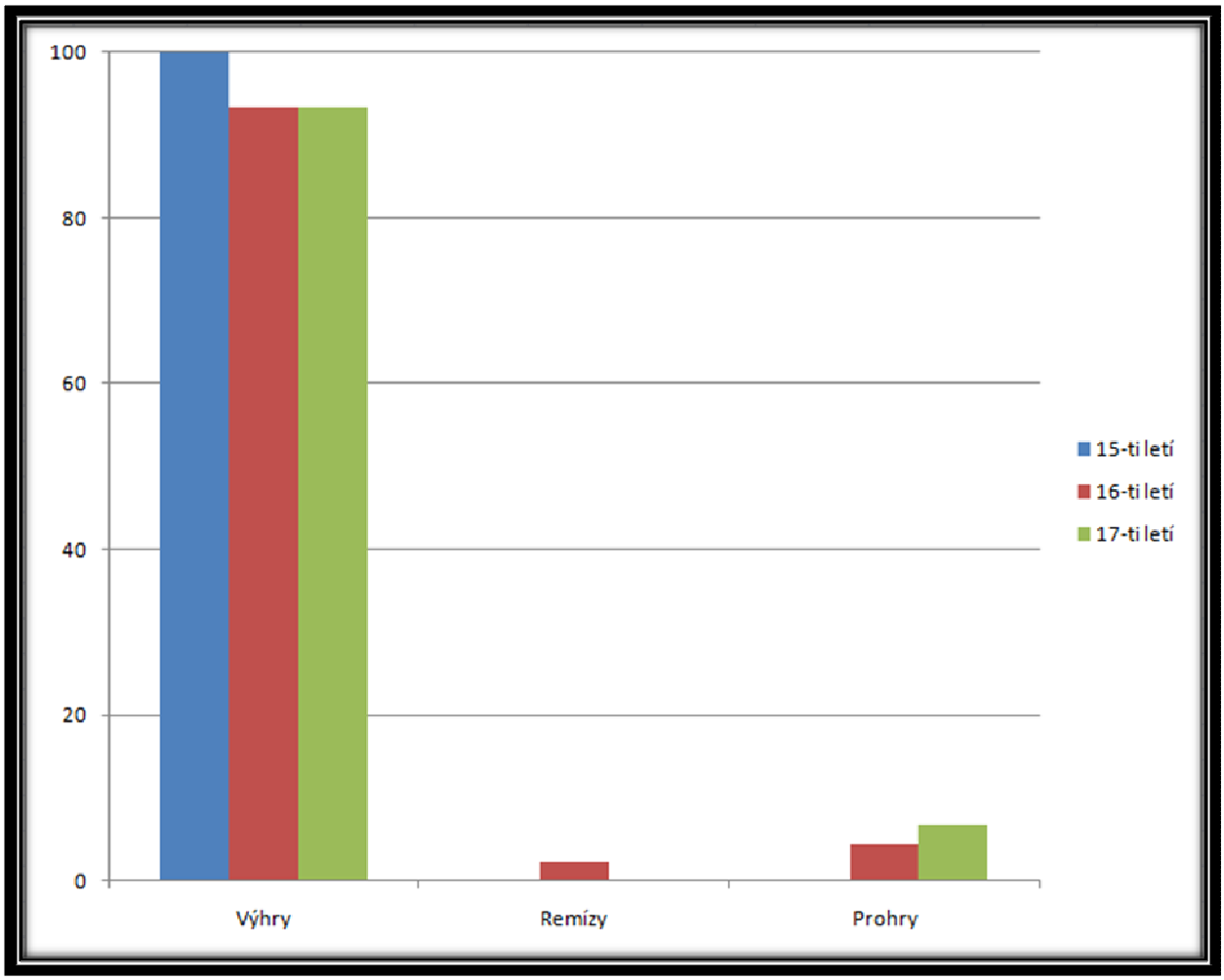
Testování proti studentům

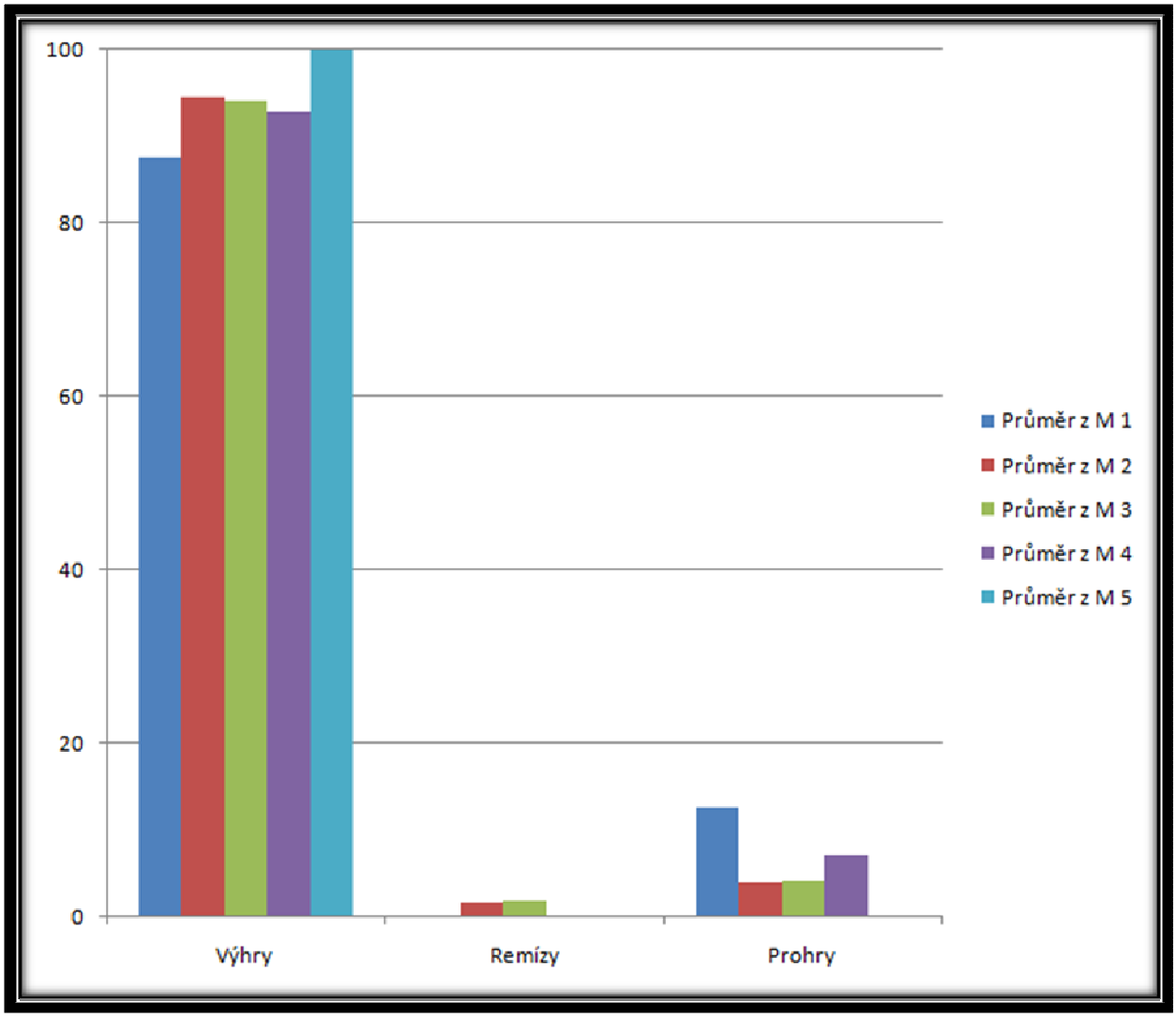
Program byl testován proti studentům našeho gymnázia.

Výsledky verze k 20.3.2010 proti alfabetě hloubka=3 jsou:

Hry celkem	Výhry uživatelů	Remízy uživatelů	Prohry uživatelů
419	393	5	21

Obrázek 22 - Graf závisloti výsledku hry na věku (%)





Obrázek 23 - Graf závislosti výsledku hry na průměrné známce z matematiky (%)

Výsledky práce

Plán vylepšení

Program není ani zdaleka dokončen, chystám mnoho vylepšení, mezi která patří:

- databáze her (-> částečně učící se algoritmus)

Poskytuje základ pro učící se umělou inteligenci a byla by také dobrá pro uživatele, který by měl zájem analyzovat si odehrané hry.

- počítačová analýza pozice (->hash funkce pro ukládání tahů, prohlubování propočtu)

Umožňovala by uživateli analyzovat pozici nekonečně dlouhou dobu (-> zlepšování výsledků)

- síťová hra

Šlo by o umožnění hry 2 hráčů proti sobě po internetu nebo v síti LAN prostřednictvím protokolu TCP/IP.

- implementace algoritmu Negascout

Algoritmus pro prořezávání stromu Negascout by měl být až o 10% rychlejší (pro šachy -> pro Pentago by měl být výkonnostní rozdíl větší, jelikož má větší větvící faktor) než Alfabeta.

- mutlithreading

Více jader -> větší výkon, jediný prostředek jak znatelně zvýšit výkon optimalizovaného programu.

- databáze zahájení a koncovek

Klíčový prvek pro vylepšení úrovně umělé inteligence.

Závěr

Podařilo se mi vytvořit program, který je přesvědčivou digitální kopií hry Pentago. Obsahuje intuitivní grafické rozhraní se základy uživatelského přispůsobení a umělou inteligenci.

Umělá inteligence sice zatím není tak dobrá, jak bych si představoval, ale je zde vidět že od počátku vývoje se AI značně zlepšila a to jak do kvality, tak do rychlosti. Z počátečních 500 pozičních analýz za sekundu je nyní rychlost kolem 400.000 pozičních analýz za sekundu.

Při kódování programu jsem se snažil dodržovat standardy daného jazyka a používat optimalizace, jak jen to šlo. Ukázalo se, že na otimalizacích značně závisí rychlost programu a optimalizace jsou jedinou cestou dalšího zvyšování výkonu.

Ve stálém vylepšování programu a hlavně umělé inteligence budu i nadále pokračovat, protože tato tematika mě zajímá a vidím zde (na poli umělé inteligence) potenciál hry Pentago.

Zdroje informací

[1] Programování ve VB.NET [online] – Dostupný z www <http://www.vbnet.cz/>.

[2] Jan Němec: Šachové myšlení [online] – Dostupný z www < http://www.linuxsoft.cz/>.

[3] Niklaus Wirth: Algoritmy a štruktúry údajov [s.l.]: Alfa 1988 1.vydání [MDT 519.254, 519.681]

[4] Ludovít Molnár: Programovanie v jazyku pascal [s.l.]: Alfa 1989 2.vydání [ISBN 80-05-00118-5]

[5] Ján Hanák: Visual Basic 2005 pro pokročilé [s.l.]: Zoner Press 2006 1.vydání [ISBN 80-86815-52-8]

[6] Miroslav Novák: Algoritmus piškvorek [online] – Dostupný z www <http://www.miroslavnovak.com/>

[7] Pentago: Oficiální pravidla hry [s.l.]

[8] Robin A. Reynolds-Haertle: OOP VB.NET/C#.NET – Krok za krokem [s.l.]: Mobil Media 2002 1.vydání [ISBN 80-865993-25-8]

Seznam obrázků

Seznam příloh