UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Programová realizace jednoduché strategické hry

Květoslav Čáp

Bakalářská práce
2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Květoslav ČÁP**
Osobní číslo:	**I07874**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Programová realizace jednoduché strategické hry**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce je realizace jednoduché počítačové strategické hry na principu minimaxu. Součástí práce je i vhodná volba pravidel definujících vlastní hru a vytvoření jednoduchého rozhraní pro komunikaci s uživatelem. Teoretická část bude obsahovat popis metod, které mají vztah k řešenému problému. Implementační část bude obsahovat výsledný algoritmus a zpracování výsledků, popř. demonstraci.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Mařík V., Štěpánková O., Lažanský J. Umělá inteligence 1. Academia, Praha, 1993.

Vedoucí bakalářské práce: **doc. Ing. Jan Cvejn, Ph.D.**
Katedra řízení procesů

Datum zadání bakalářské práce: **15. ledna 2010**

Termín odevzdání bakalářské práce: **14. května 2010**

Univerzita Pardubice Česká republika -S-

L.S.

prof. Ing. Simeon Karamazov, Dr.
děkan

Ing. Lukáš Čegan, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2010

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 13.8.2010 Květoslav Čáp

Poděkování

Na tomto místě bych chtěl poděkoval vedoucímu mé bakalářské práce, panu doc. Ing. Janu Cvejnovi Ph.D.. Díky mu patří za věcné připomínky k této práci, ale také za snahu mi pomoci a jeho trpělivost, kterou se mnou měl.

Anotace

Účelem bakalářské práce je vytvoření jednoduché strategické hry na počítači, využívající prostředků umělé inteligence, včetně vlastních pravidel. Jádrem řešení je algoritmus minimax. Vytvořený program umožňuje nastavit základní parametry a realizovat hru člověk/počítač a počítač/počítač. Byly provedeny kroky k urychlení algoritmu a byl experimentálně potvrzen jejich vliv.

Klíčová slova

Minimax, OpenGL, Hry, Algoritmy

Title

Program realization of a simple strategy game

Annotation

The goal of this is work is creating a simple computer strategy game, using artificial intelligence methods, including own rules. Algorithm minimax is a kernel of the game. Created program enables to set basic parameters and to realize the games human/computer and computer/computer. Steps for speeding-up the algorithm were made and their effect was verified experimentally.

Keywords

Minimax, OpenGL, Games, Algorithms

Obsah

Seznam obrázků

Seznam tabulek

Seznam grafů

1 Úvod

S nástupem výpočetní techniky nastal rozvoj umělé inteligence. Lidé se pokoušejí vložit inteligenci do stroje a naučit tak nějakým způsobem stroje přemýšlet. Vytvořit komplexní umělou inteligenci srovnatelnou s lidskou je velmi obtížné, proto existují různé metody a modely pro řešení úloh. Do jedné z těchto metod patří i prohledávání stavového prostoru.

Hra je určitá činnost, která má definovaná pravidla, hráče a určitý cíl. Hry se hrají v dnešní době většinou pro zábavu, ale mohou také sloužit pro modelování problémů reálného světa a navrhnutí strategie pro nalezení vhodného řešení.

Nejjednodušší hry je sice možné řešit prohledáváním stavového prostoru neinformovanými metodami. U většiny her, jako jsou například šachy, dáma nebo go, by ale neuspěl ani nejvýkonnější počítač na světě bez využití znalostí. Tato znalost o stavu bývá nejčastěji vyjádřena pomocí statické ohodnocovací funkce a pomáhá k nalezení nejslibnější cesty.

Účelem této práce je navrhnout a realizovat hru na počítači s umělou inteligencí využívající metodu minimax. Další částí je navržení a definice vlastních pravidel.

2 Hledaní strategií metodami umělé inteligence

2.1 Stavový prostor

Množina všech stavů prostředí se nazývá stavový prostor řešení. Stavový prostor může být buď konečný, nebo nekonečný. Konečným stavovým prostorem je takový stavový prostor, který má konečné množství stavů. Pokud je stavový prostor nekonečný, ale lze přiřadit přirozené číslo ke každému stavu, potom je daný stavový prostor spočitatelný. To u nekonečného stavového prostoru není možné, protože stavové proměnné v něm jsou reálná čísla.

Metody, jejichž účelem je procházení stavů určitého řešeného problému s cílem najít požadovaný stav se nazývají prohledávání stavového prostoru. Existuje řada různých metod, pomocí nichž lze stavový prostor prohledávat. Tyto metody se dají hodnotit podle třech základních kriterií, a to podle časové náročnosti – vyjadřuje minimální nebo průměrný čas, nebo počet prozkoumaných jevů, který je potřeba k vyřešení daného problému. Druhým kriteriem je prostorová složitost vyjadřující počet stavů, které jsou současně uloženy v paměti. Posledním důležitým kriteriem je kvalita výsledků obsahující informaci o tom, jestli je dané řešení optimální. Nelze určit, která metoda je nejlepší, různé metody mají své výhody i nevýhody.

Tyto metody lze rozdělit na informované a neinformované. Neinformované metody jsou takové, které nemají žádné informace o stavovém prostoru, pomocí nichž by se snáze dostaly k cíli.

2.2 Informované metody prohledávání stavového prostoru

K informovaným metodám jsou řazeny takové metody, které mají hodnotící funkci umožňující každému stavu přiřadit ohodnocení. Tato informace pomáhá odhadnout, jak daleko od současného stavu je možné nalézt požadované řešení. Proto budou expandovány stavy s nejlepším ohodnocením a zabrání se v prohledávání cest, které nevedou k nalezení řešení. Mezi informované metody prohledávání stavového prostoru patří gradientní algoritmus, algoritmus uspořádaného prohledávání, algoritmus A*, algoritmus větví a mezí a další.

Gradientní algoritmus expanduje uzel, který byl hodnotící funkcí vyhodnocen jako nejlepší a vyhodnocují se jeho potomci. Potomek s nejlepším ohodnocením je dále rozvíjen. V paměti je uchováván pouze expandovaný uzel. Prohledávání je ukončeno, když aktuální stav má lepší ohodnocení než jeho následovníci.

Algoritmus uspořádaného prohledávání je gradientní algoritmus rozšířený o paměť. Expandují se stavy s nejmenším ohodnocením. U uzlů je evidován název, hodnota a rodič.

Algoritmus A* se řadí mezi grafové vyhledávací algoritmy. Jeho úkolem je nalézt nejkratší cestu mezi dvěma uzly v ohodnoceném grafu. Algoritmus funguje tak, že prohledá všechny sousedy počátečního uzlu. Pro každý sousední uzel vypočítá pomocí heuristické funkce hodnotu. Vypočtené hodnoty jsou potom seřazeny podle velikosti. Následně je vybrán a rozvíjen uzel s nejmenší hodnotou.

Algoritmus větví a mezí je algoritmus se stejnou cenou. Po nalezení cílového stavu si zapamatuje jeho hodnotu a pokračuje dále v prohledávání. Na konci prohledávání je nalezena optimální cesta.

2.3 Neinformované metody prohledávání stavového prostoru

Neinformované metody jsou založeny na prohledávání stavového prostoru hrubou silou. To znamená, že jsou systematicky procházeny všechny možné stavy. Nevýhodou těchto metod je jejich neefektivnost. Základní rozdělení neinformovaných metod je slepé prohledávání do šířky a slepé prohledávání do hloubky. Obě metody mají různé výhody a nevýhody. Určitým kompromisem mezi prohledáváním do hloubky a prohledáváním do šířky je algoritmus iterativního prohlubování, který v sobě kombinuje obě tyto metody.

Hlavní výhodou u metody prohledávání do šířky je nalezení nejkratší možné cesty od počátečního stavu ke stavu cílovému. Naopak nevýhodou při velké hloubce prohledávání je značná paměťová náročnost. Výhodou metody prohledávání do hloubky jsou malé nároky na paměť. Nalezená cesta od počátečního stavu k cílovému nemusí být nejkratší.

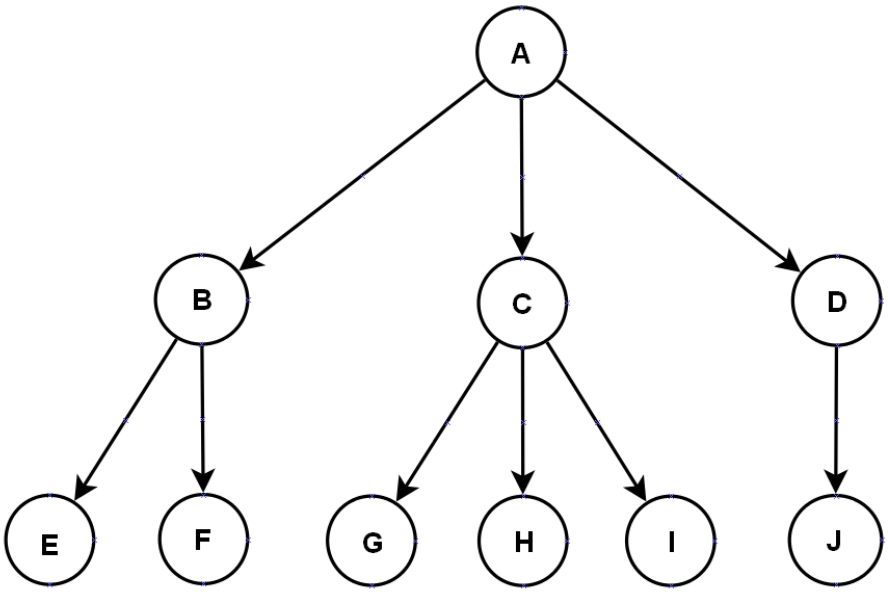
Neinformované metody je možné realizovat pomocí rekurzivní funkce s dobře navrženými vstupními parametry a vhodnou návratovou hodnotou. Existuje možnost realizace bez rekurze a to s využitím datových struktur jako jsou fronta, zásobník a seznam.

Slepé prohledávání do šířky

Algoritmus prohledávání do šířky

1. Počáteční stav vložen do fronty.
2. Z fronty je odebrán jeden stav a ten je expandován.
3. Pokud je potomek expandovaného stavu shodný s cílovým, řešení bylo nalezeno. Konec prohledávání.
4. Potomci expandovaného stavu jsou přidáni do fronty.
5. Pokud je fronta prázdná, řešení neexistuje. Konec prohledávání.
6. Pokračování krokem 2.

Pro vysvětlení algoritmu je použit stavový prostor podle obrázku 1. Přechod mezi stavy je zastoupen orientovanou šipkou. K uchování neexpandovaných stavů slouží fronta F, která je na začátku prázdná. Vrchol A je počáteční stav a vrchol G cílový.



Obrázek 1 – Pořadí zkoumaných uzlů při prohledávání do šířky

Výpočet je prováděn následovně:

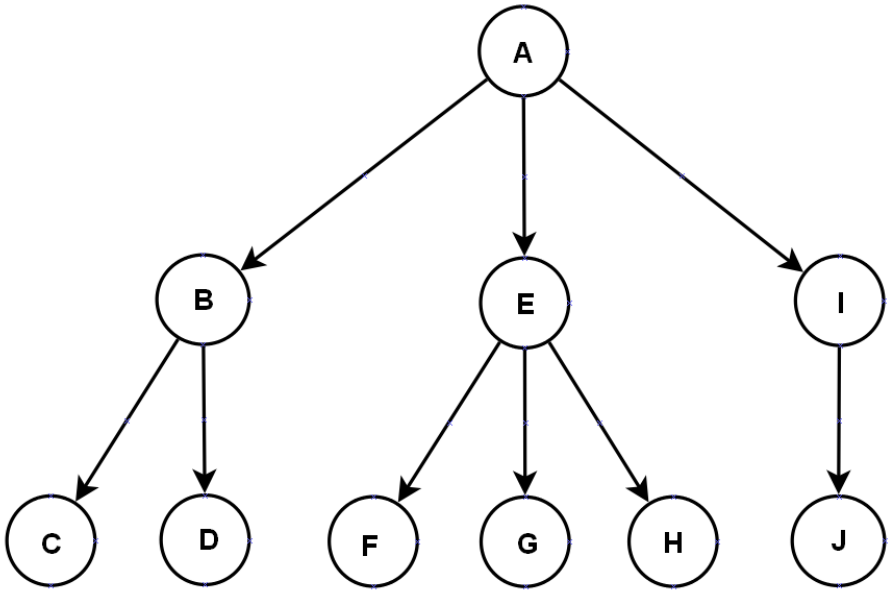
Nejprve je aktuální stav vložen do fronty F(A). Následně je první prvek z fronty odebrán a expandován. Zkoumaný stav A má tři potomky - B, C a D. Potomci jsou zapsáni do fronty F(B, C, D). V dalším kroku je odebrán z fronty prvek B a jeho potomci E a F přidáni do F(C, D, E, F). Dále je z fronty odebrán prvek C. Protože jeho potomkem je stav cílový, je nalezeno řešení a ukončeno prohledávání.

Slepé prohledávání do hloubky

Algoritmus prohledávání do hloubky

1. Počáteční stav vložen do zásobníku.
2. Ze zásobníku je odebrán jeden stav a ten je expandován.
3. Pokud je potomek expandovaného stavu shodný s cílovým, řešení bylo nalezeno. Konec prohledávání.
4. Potomci expandovaného stavu jsou přidáni do zásobníku.
5. Pokud je zásobník prázdný, řešení neexistuje. Konec prohledávání.
6. Pokračování krokem 2.

Pro vysvětlení algoritmu je použit stavový prostor podle obrázku 2. Přechod mezi stavy je zastoupen orientovanou šipkou. K uchování neexpandovaných stavů slouží zásobník Z, který je na začátku prázdný. Vrchol A je počáteční stav a vrchol G cílový.



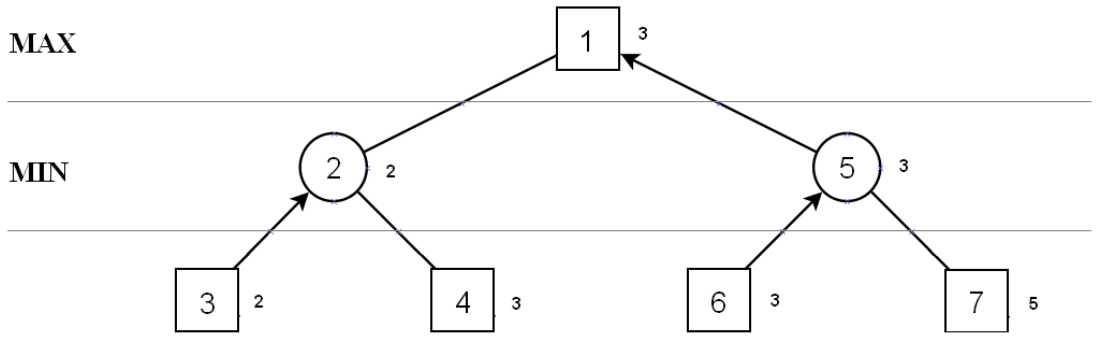
Obrázek 2 – Pořadí zkoumaných uzlů při prohledávání do hloubky

Na začátku je aktuální stav A vložen do zásobníku Z(A). Potom je tento prvek odebrán a expandován. Jeho tři potomci B, E a I jsou přidáni do Z(B, E, I). V dalším kroku je odebrán prvek B a jeho dva následovníci C a D zapsáni do zásobníku Z(C, D, E, I). Protože prvek C nemá potomka, bude ze zásobníku pouze odebrán. To samé platí i pro prvek D. Následně je ze zásobníku odebrán a expandován prvek E. Protože jeden z jeho potomků je stavem cílovým, je nalezeno řešení a ukončeno prohledávání.

2.4 Metoda minimax

Metoda minimax je realizací algoritmu prohledávání do hloubky s omezením hloubky prohledávání. Výchozí stav je expandován do zvolené hloubky. Každému stavu je přiřazena hodnota. Neexpandovaným stavům je vypočítána pomocí statické ohodnocovací funkce. U expandovaných stavů je vybrána nejlepší hodnota potomka a ta je pak přiřazena. Nejlepší hodnota je závislá na tom, kdo je na tahu. Pokud je na tahu hráč, vybírá se z vrácených hodnot maximum. Když je na tahu soupeř, vybírá se minimum.

Pro vysvětlení algoritmu je použit graf podle obrázku 3. Pozice jsou znázorněny jako vrcholy. Přechod z jedné pozice do druhé je vyobrazen hranou. Čtvercem je označena pozice, kdy je na tahu hráč, kruhem, kdy je na tahu soupeř. Vrchol 1 zobrazuje výchozí pozici. Z pozice 1 tedy může hráč táhnout do pozice 2 a 5. Z pozice 2 pak může soupeř přejít do pozice 3 a 4. Výchozí pozice je rozvinuta do hloubky 2.



Obrázek 3 – Minimax

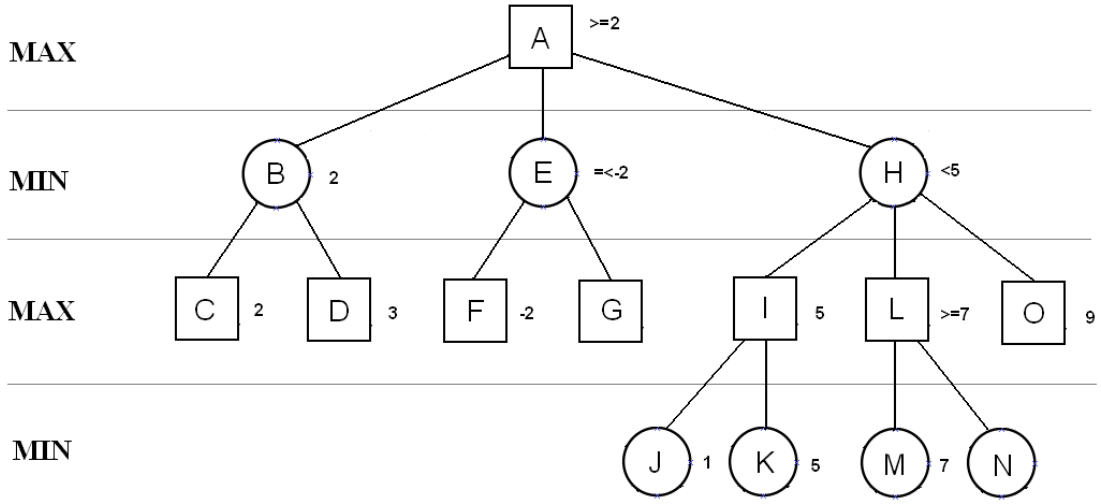
Z aktuální pozice 1 jsou za hráče postupně generovány a následně v paměti zahrány všechny možné tahy. Nejprve je vykonán tah, který vede do pozice 2. V této pozici počítač vygeneruje možná pokračování za soupeře a opět v paměti provede první z tahů (pozice 3). Protože pozice 3 odpovídá stanovené hloubce, další tahy za hráče již nebudou generovány. Dojde pouze k zavolání ohodnocovací funkce a vypočítání hodnoty pozice. Ta je vrácena a přiřazena o úroveň výše (do pozice 2). Následuje tah zpět, tím dojde přesunu z pozice 3 do pozice 2. Pokračuje tah do pozice 4, kde opět proběhne jen výpočet hodnoty pozice. Vrátí se hodnota a následně i tah z pozice 4 do pozice 2. Vrácená hodnota je porovnána s aktuální hodnotou. Lepší hodnota je nastavena a opět předána o úroveň výše, tedy do pozice 1. Vrácením tahu zpět následuje přesun z pozice 2 do pozice 1. Teprve nyní je zahrán tah do pozice 5. V této pozici počítač generuje a provede první z možných tahů soupeře (pozice 6). Vypočítaná hodnota z pozice 6 je vrácena a nastavena do pozice 5. Po vrácení tahu a přechodu zpět do pozice 5, je zahrán tah vedoucí do pozice 7. Potom je vypočítaná hodnota pozice 7. Následně je vrácen tah i tato hodnota do pozice 5. Pokračuje vybrání lepší hodnoty pozice a vrácení této hodnoty společně s tahem do pozice 1. V pozici jedna je opět vybrána pozice s nejlepším ohodnocením.

2.5 Prořezávání alfa–beta

Tato metoda vychází z klasického minimaxu, který je metodou prohledávání do hloubky. Je využívána technika větví a mezí, při které jsou vyřazena řešení, která jsou horší než doposud nalezená. K vyřazení řešení slouží mezní hodnota uzlu.

Technika větví a mezí aplikovaná na hru dvou hráčů se nazývá prořezávání alfa–beta. Každý hráč má vlastní mezní hodnotu. Hodnota alfa představuje dolní mez ohodnocení uzlu, při kterém je na tahu hráč. Hodnota beta představuje horní mez ohodnocení uzlu, při kterém je na tahu soupeř.

Pro vysvětlení algoritmu je použit graf podle obrázku 4. Značení je stejné jako u obrázku 3.



Obrázek 4 – Prořezávání alfa–beta

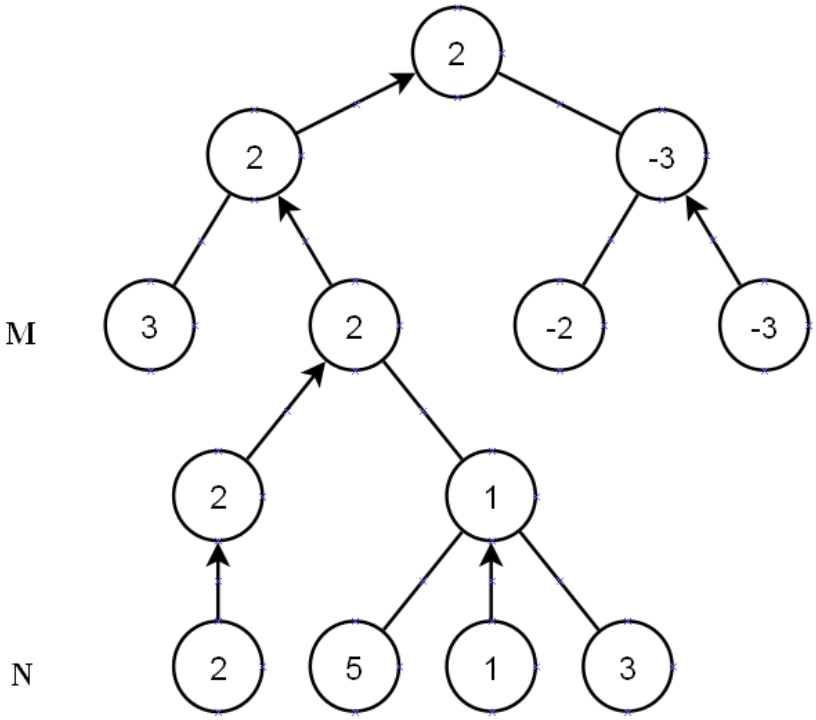
Po ohodnocení pozice B hodnotou 2, je zaručeno, že hodnota pozice A bude nabývat také minimálně 2. To je dáno tím, že hráč si zvolí maximum (alfa=2). Jakmile je ohodnocena pozice F, která je -2 (je menší než alfa), není třeba zjišťovat hodnotu pozice G, protože soupeř v pozici E si určitě vybere minimum (je proveden alfa řez).

Jakmile je určena hodnota pozice I, která nabývá hodnoty 5, je jisté, že hodnota pozice H bude také maximálně 5, protože soupeř zvolí minimum (beta=5). Po zjištění hodnoty pozice M, která je 7 (je větší než beta), není třeba dále zkoumat pozici N, protože soupeř zvolí výhodnější tah (je proveden beta řez).

Při prořezávání alfa–beta jsou vynechány celé části podstromu řešení a je tak urychleno a zefektivněno prohledávání.

2.6 Technika druhotného prohledávání

Tato metoda je postavena na prohledávání do určité hloubky M. Po ukončení prohledávání je vybrán stav s nejlepším ohodnocením. Následně je tento stav expandován do hloubky N. Po tomto kontrolním expandování je jisté, že výsledná cesta nenarazí ani po M+N tazích na nějaký problém. Druhotné prohledávání, které začíná na m-té úrovni, je podstatně rychlejší a efektivnější, než kdyby se prohledával celý strom o hloubce M+N.

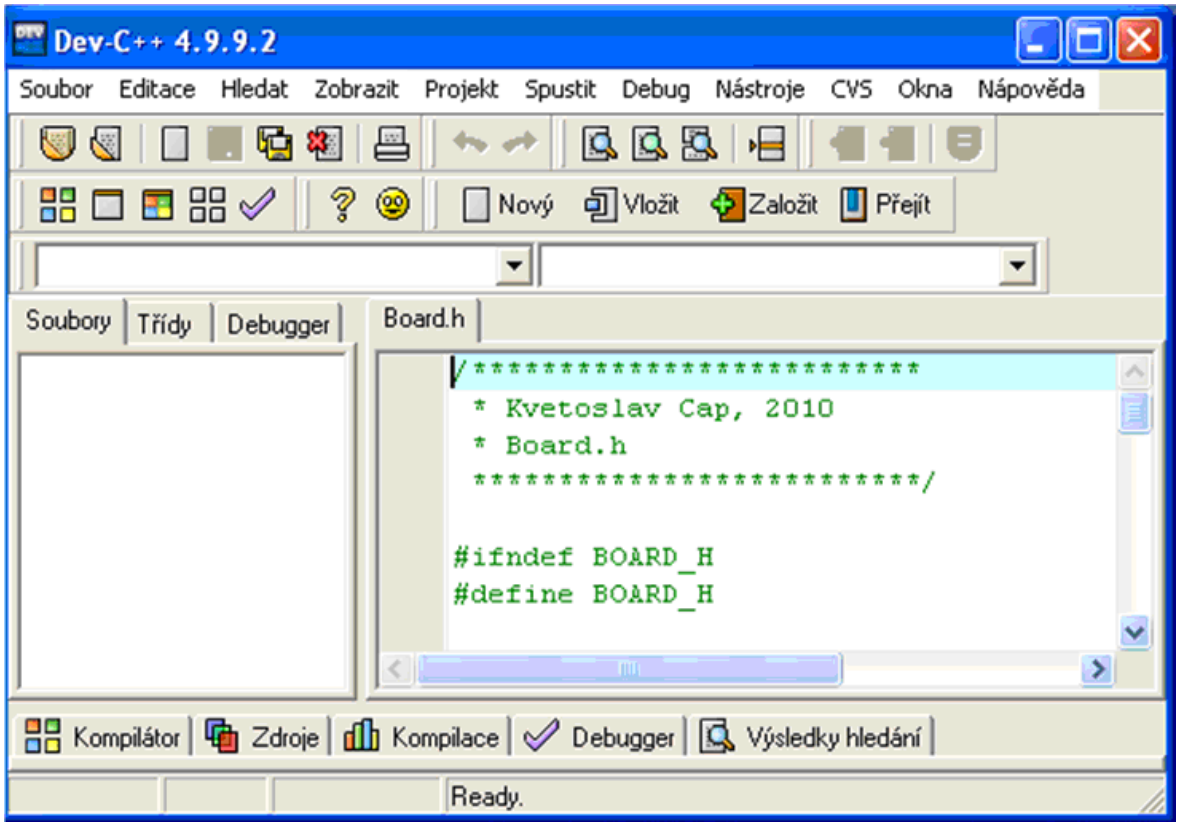


Obrázek 5 – Druhotné prohledávání do hloubky M+N

3 Realizace hry

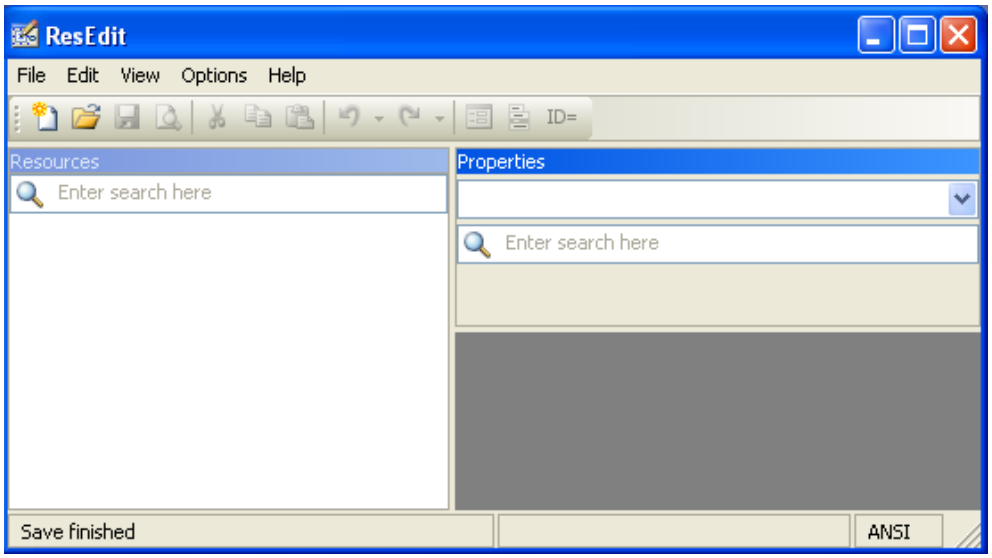
3.1 Popis prostředí a programů

Hra byla realizována podle zadání v programovacím jazyce C++. Pro vývoj bylo zvoleno vývojové prostředí Dev–C++ ve verzi 4.9.9.2. Dev–C++ lze volně stáhnout z Internetu a je šířeno pod GNU licencí. Tento nástroj je vhodný spíše pro vývoj menších projektů. Editor kódu podporuje barevné zobrazení syntaxe jazyka C++. Součástí programu je překladač GCC. Vývojové prostředí Dev–C++ je zobrazeno na obrázku 6.



Obrázek 6 – Vývojové prostředí Dev-C++

Protože Dev–C++ neobsahuje ressource editor, jako mají konkurenční komerční produkty, byl při vytváření menu využit program ResEdit ve verzi 1.4.13. Tuto aplikaci je také možné volně stáhnout z Internetu.



Obrázek 7 – Editor prostředků ResEdit

Při vývoji byla použita OpenGL šablona, která je součástí Dev–C++. Hra tedy využívá grafickou knihovnu OpenGL.

OpenGL, celým názvem Open Graphics Library, je grafické rozhraní, které je mezi hardwarem a softwarovou aplikací. Tato knihovna umožňuje vytvářet 2D a 3D aplikace s využitím akcelerované grafiky. Je často využívaná pro tvorbu počítačových her. Obsahuje velké množství struktur a funkcí. Tato knihovna byla vyvinuta v roce 1992 firmou Silicon Graphics International Corporation.

OpenGL je knihovna vhodná pro začínající programátory v oblasti počítačové grafiky. Možnosti tvorby jsou téměř neomezené.

3.2 Realizace algoritmu minimax

```
01 /*************************
02  * Value_position
03  * vrati hodnotu pozice
04  *************************/
05
06 float TComputer::Value_position(TBoard* Board)
07 {
08   return (Board->Get_quantity_blue()-Board->Get_quantity_red());
09 }
```

Důležitou částí minimaxu je statická ohodnocovací funkce. Tato metoda provádí rychlou analýzu pozice. Z řádku 8 je patrné, že návratovou hodnotou tvoří rozdíl mezi poli obsazenými modrým a červeným hráčem. Když je vrácená kladná hodnota, je pozice výhodnější pro modrého hráče, je–li záporná, pro hráče červeného.

```
01 /*************************
02 * Minimax
03 * hra pocitace
04 *************************/
05
06 float TComputer::Minimax(int &U, int &V, TBoard* Board, int Depth)
07 {
08   int A, B;
09   float best_value, value=0;
10   if((Depth<=0)||(Board->Get_quantity_free()==0))
11   return Value_position(Board);
12   else
13   {
14     if(Board->Get_who_play()==BLUE) best_value=-100;
15     else best_value=100;
16     EPlayer Memory[5];
17     for(int i=0; i<Board->Get_u(); i++)
18     {
19       for(int j=0; j<Board->Get_v(); j++)
20       {
21         if(Board->Get_data(i, j)==FREE)
22         {
23           Board->Save_position(i, j, Memory);
24           Board->Make_move(i, j);
25           quantity_moves++;
26           value=Minimax(A, B, Board, Depth-1);
27           Board->Change_move();
28           Board->Restore_position(i, j, Memory);
29           if(((Board->Get_who_play()==BLUE) && (value>best_value)) ||
30           ((Board->Get_who_play()==RED) && (value<best_value)))
31           {
32             best_value=value;
33             U=i;
34             V=j;
35           }
36         }
37       }
38     }
39     return best_value;
40   }
41 }
```

Metoda minimax má čtyři vstupní parametry. První dva slouží k uložení souřadnic nejlepšího nalezeného tahu. Dalšími parametry jsou pozice a hloubka výpočtu. Na řádku 9 jsou definovány lokální proměnné value a best_value. V nich je uložená aktuální a doposud nejlepší nalezená hodnota pozice. Pokud je hloubka výpočtu rovna nule, je zavolána ohodnocovací funkce a hodnota vrácena minimaxem. Pak se určí, který hráč je na tahu a podle toho je nastavena do proměnné best_value hodnota. Na řádku 16 je deklarována lokální paměť pro jeden tah. Pomocí for cyklů jsou postupně generovány všechny tahy. Tah má souřadnice [i, j].

Na řádku 23 je pozice uložena do paměti. Pak je vykonán vygenerovaný tah. Na řádku 26 dochází k rekurzivní volaní metody minimax. Je předána pozice s nově zahraným tahem a novou hloubkou výpočtu. Kdyby hloubka výpočtu nebyla snížená, došlo by k zacyklení. Vrácená hodnota pozice je uložena do proměnné value. Následuje vrácení tahu. Na řádku 29 dochází k porovnání hodnot zahrané a doposud nejlepší nalezené pozice. Když je zahraná pozice lepší než doposud nalezená, dojde k aktualizaci proměnné best_value. Souřadnice zvoleného tahu jsou uloženy do odkazů U a V.

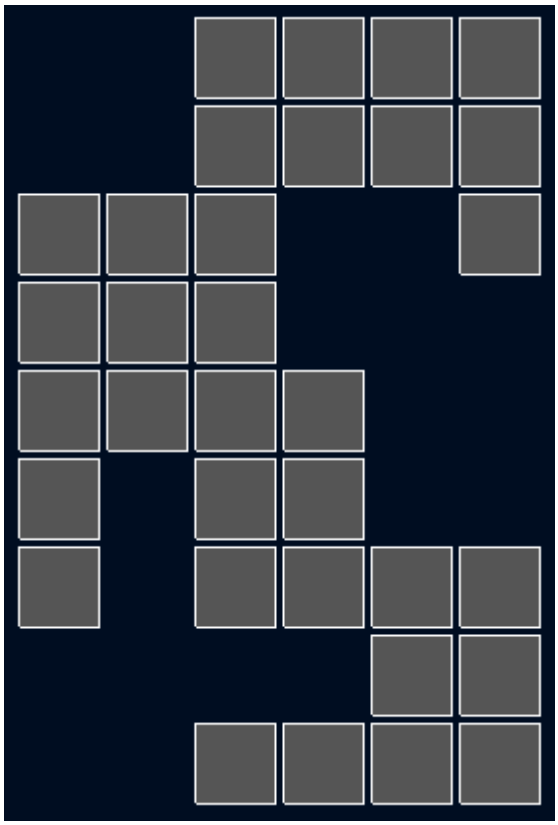
3.3 Pravidla hry

Na začátku vývoje bylo potřeba navrhnout vlastní hru, která by byla zajímavá a měla smysl. To se ukázalo jako poměrně nelehký problém. Byl zvolen deskový typ hry bez bližšího upřesnění velikosti desky, s tím, že hráči budou postupně obsazovat volná pole na této desce. Inspirací při návrhu hry byli go, šachy a piškvorky. Mezi testované varianty patřil například návrh s různým ohodnocením polí na desce. To se však ukázalo pro člověka jako velmi nepřehledné. Po simulacích na papíře vznikl návrh hry s poměrně jednoduchými pravidly.

Pravidla

Hra je určena pro dva hráče. Herní deska se skládá z polí. Pole jsou na začátku prázdná. Hráč může táhnout pouze na neobsazené pole. Pokud tak učiní, zabere toto pole a ještě po jednom poli na všechny strany ve vertikálním a horizontálním směru. Hráči se po tahu střídají. Hra končí, když na herní desce už není žádné volné pole. Vítězem je hráč, který obsadil nejvíce polí.

Aby se hra stala zajímavější pro člověka, byla rozšířena o pole, které nepůjde žádným hráčem obsadit. Díky tomu je možné vytvářet mapy s různými tvary. Příklad nepravidelné mapy je zobrazen na obrázku 9.



Obrázek 8 – Nepravidelná mapa

4 Experimentální část

Hra využívající minimax byla testována na herní desce o rozměrech 5x5 a 7x7 polí. Před experimentem byl stanoven mezní čas pro počítač na dvacet sekund pro jeden tah. Pro malou hloubku prohledávaní se choval minimax velmi rychle. Čas byl také ovlivněn velikostí herní desky. Z grafů je patrné, že s rostoucí hloubkou prohledávání roste čas potřebný k vyhodnocení pozice exponenciálně.

Tabulka 1 – Doba hledání řešení

	Herní deska 5x5		Herní deska 7x7	
Hloubka	**minimax ***	**alfa–beta***	**minimax***	**alfa–beta***
0	0,000	0,000	0,000	0,000
1	0,000	0,000	0,000	0,000
2	0,016	0,016	0,078	0,015
3	0,062	0,078	1,500	0,109
4	0,718	0,172	47,390	0,578
5	6,188	0,312	1436,984	2,953
6	41,172	0,594		19,078
7	195,483	1,281		81,359
8		1,546		457,047
9		2,437		

*Čas v sekundách

Tabulka 2 – Velikost prohledávaného stavového prostoru

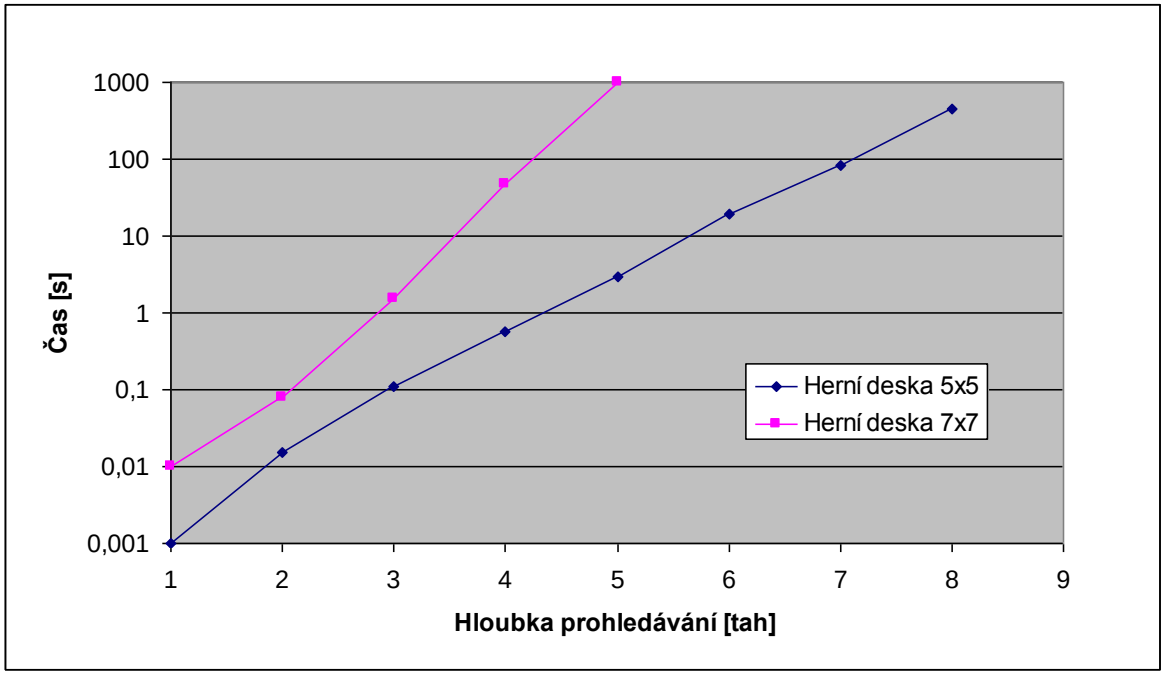
	Herní deska 5x5		Herní deska 7x7	
Hloubka	**minimax***	**alfa–beta***	**minimax***	**alfa–beta***
0	25	25	49	49
1	545	285	2233	1057
2	9389	2172	90373	15540
3	129965	10126	3295429	117768
4	1418885	63888	107776112	1005738
5	11966885	139046	3143644672	5375828
6	76468808	577667		37838688
7	365039040	990863		162370496
8		1984847		936541696
9		2520945		

*Počet zkoumaných tahů

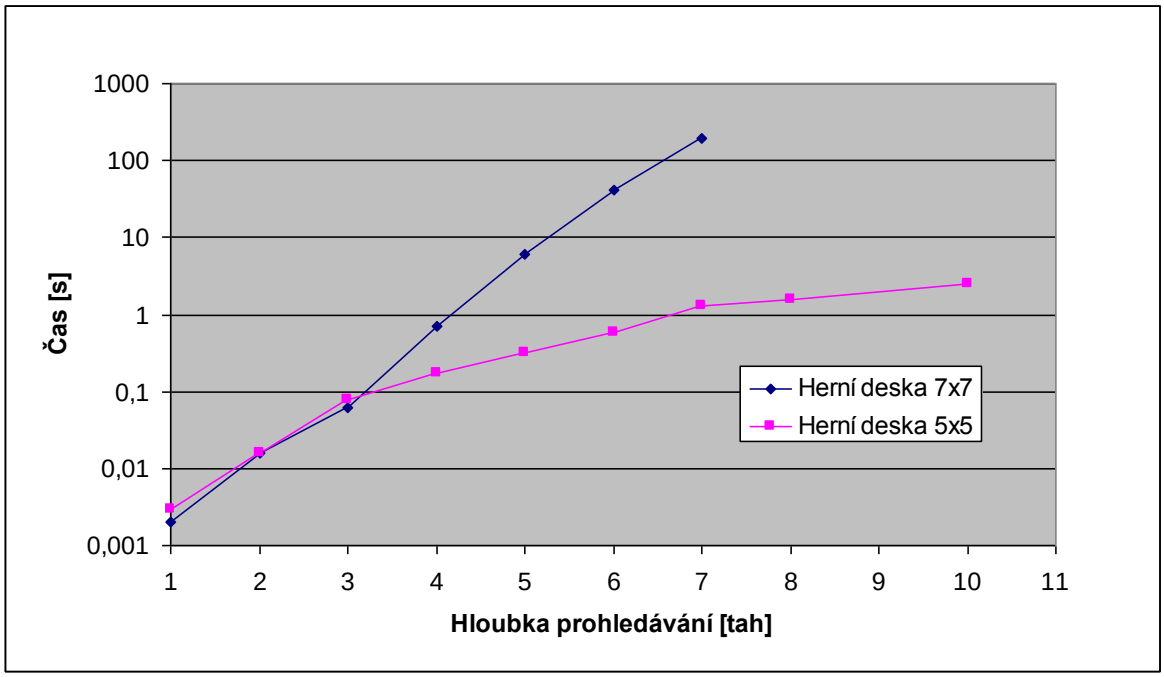
Experimenty bylo ověřeno, že pro velkou hloubku je metoda minimax zcela nepoužitelná. Proto byla do hry přidána metoda prořezávání alfa–beta. V tabulce 1 jsou uloženy časy jednotlivých metod pro různou hloubku prohledávání a různou velikost herní desky.

V tabulce 2 je uloženo množství procházených tahů pro obě metody. Pokud se toto množství vzájemně porovná, je jasné, že prořezávání alfa–beta nalezne řešení mnohem rychleji než minimax. Proto tuto metodu lze použít k prohledávání do větší hloubky a získání lepších výsledků.

Graf 1 – Doba hledání řešení v závislosti na hloubce metodou minimax



Graf 2 – Doba hledání řešení v závislosti na hloubce metodou prořezávání alfa–beta



5 Závěr

Práce se zabývala návrhem a realizací jednoduché strategické hry. V teoretické části práce byly popsány metody, které se vztahují k řešenému problému. V implementační části byla popsána realizace hry. Součástí této práce je vlastní návrh pravidel, využívající prostředků umělé inteligence. Umělá inteligence byla realizována pomocí metody minimax. Navíc byla vylepšena o metodu prořezávání alfa beta.

Výsledkem vznikl program s jednoduchým grafickým rozhraním, který umožňuje nastavit základní parametry hry, jako je výběr hráčů a nastavení algoritmů. Vytvořený program tedy umožňuje realizovat hru člověk/člověk, člověk/počítač a počítač/počítač.

V této práci byly experimentálně porovnávány metody minimax a prořezávání alfa–beta. Porovnání časů výpočtů prokázalo teoretický předpoklad, že metoda prořezávání alfa–beta je v hledání řešení lepší než–li minimax.

6 Literatura

NĚMEC, Jan. 2006. Šachové myšlení (2) - minimax [Online] 15. 3 2006. [Citace: 13. 5 2010.] http://www.linuxsoft.cz/article.php?id_article=1141.

NĚMEC, Jan. 2006. Šachové myšlení (3) - alfabeta [Online] 12. 5 2006. [Citace: 13. 5 2010.] http://www.linuxsoft.cz/article.php?id_article=1190.

NĚMEC, Jan. 2006. Šachové myšlení (5) – kaskádová metoda, prohlubování [Online] 9. 8 2006. [Citace: 13. 5 2010.] http://www.linuxsoft.cz/article.php?id_article=1288.

MAŘÍK, Vladimír, et al. *Umělá inteligence 1.* 1. vyd. Praha: Academia, 1993. 264 s. ISBN 80-200-0496-3.

MAŘÍK, Vladimír, et al. *Umělá inteligence 2.* 1. vyd. Praha: Academia, 1997. 373 s. ISBN 80-200-0504-8.

CHALUPA, RADEK. *1001 tipů a triků pro Visual C++* 1. vyd. Brno: Computer Press, 2003. 434 s. ISBN 80-7226-842-2.

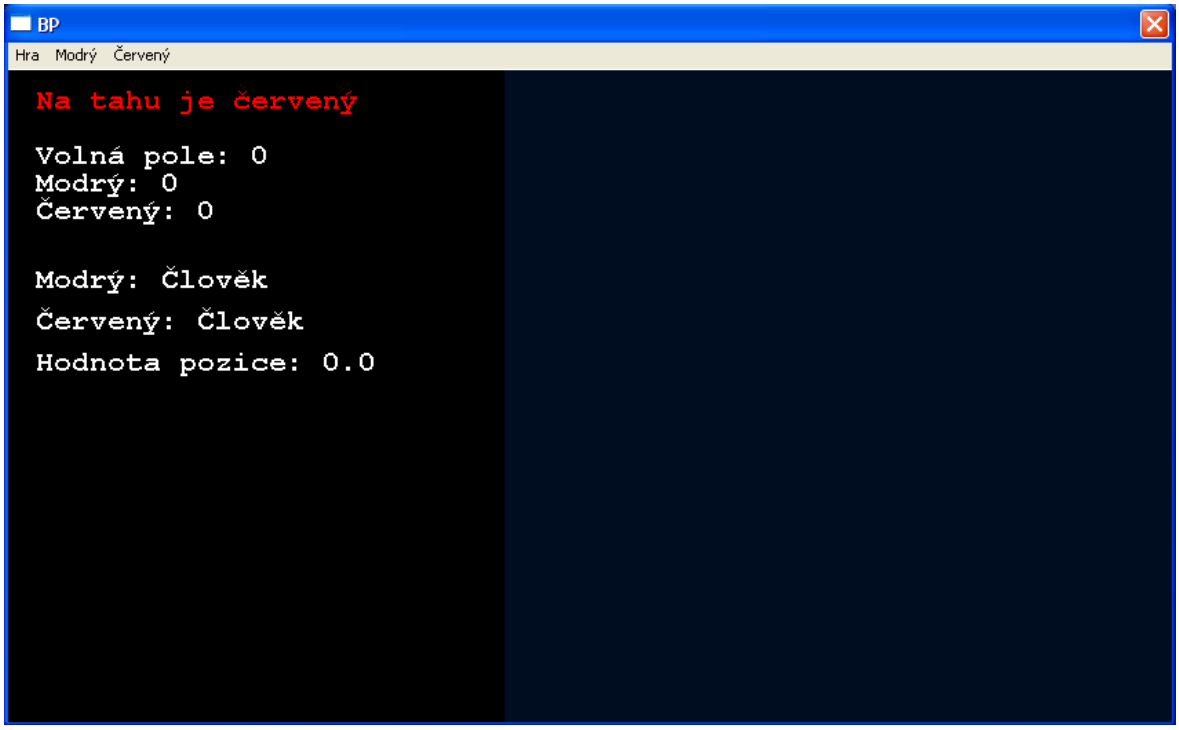
Příloha – A Uživatelská příručka

Instalace a spuštění

1. Vložte CD do mechaniky
2. V adresáři Exe spusťte soubor Bp.exe

Ovládání a nastavení

Program se ovládá myší. Po spuštění se objeví okno, které má v horní části menu. V tomto menu se nastavuje hra. Obsazení pole na herní desce je provedeno levým kliknutím tlačítka myši na dané pole. V levé části okna jsou umístěny informace o hře.



Obrázek 9 – Program po spuštění

Menu

Hra

- Nová – spustí novou hru;
- Načíst – načte herní desku ze souboru;
- Start/Stop – zapne a vypne provádění výpočtů;
- Změna tahu – změní hráče, který je na tahu;
- Konec – ukončí program;

Modrý

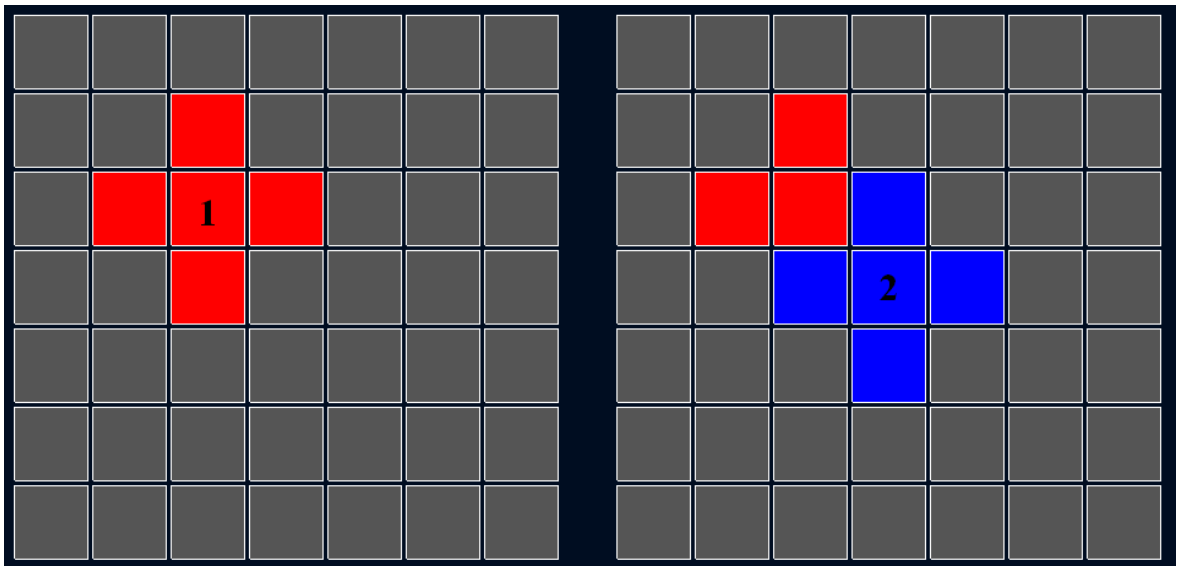
- Člověk – za modrého hráče bude hrát člověk;
- Počítač – za modrého hráče bude hrát počítač;
- Algoritmus – nastaví počítači vybraný algoritmus;
- Hloubka prohledávání – nastaví množství počítaných tahů dopředu;

Červený

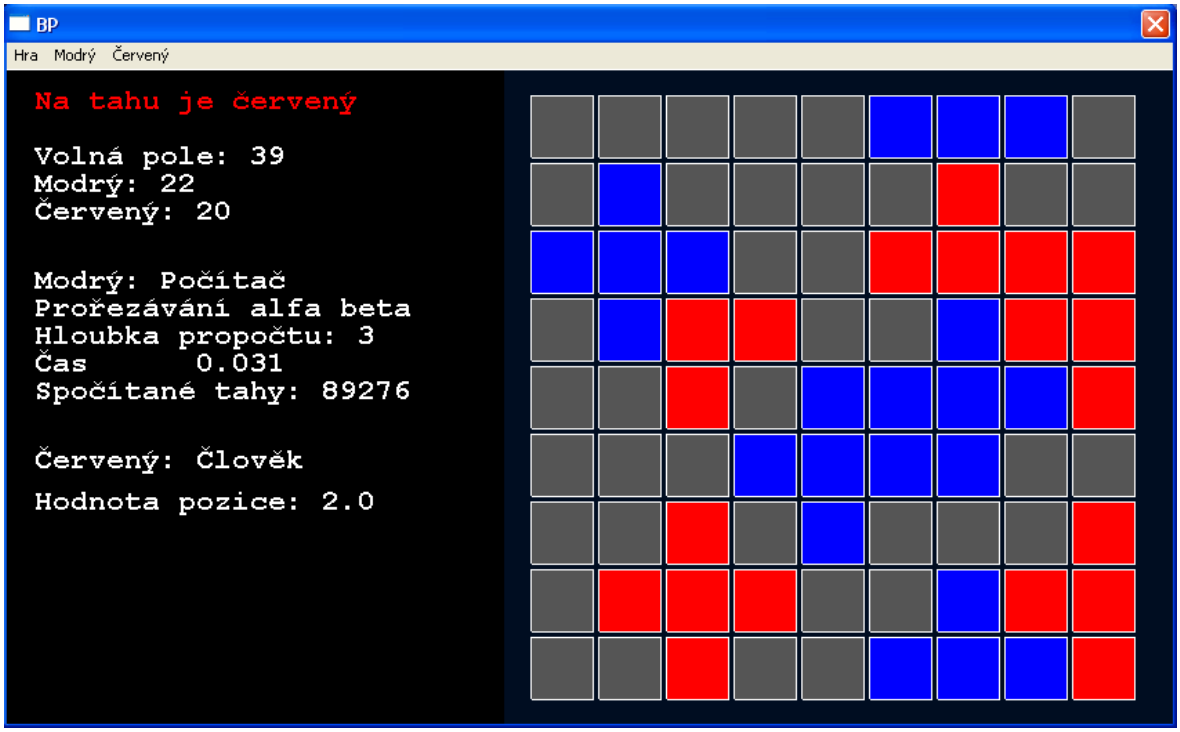
- Člověk – za červeného hráče bude hrát člověk;
- Počítač – za červeného hráče bude hrát počítač;
- Algoritmus – nastaví počítači vybraný algoritmus;
- Hloubka prohledávání – nastaví množství počítaných tahů dopředu;

Pravidla hry

Herní deska se skládá z polí. Pole jsou na začátku prázdná. Hráč může táhnout pouze na neobsazené pole. Pokud tak učiní, zabere toto pole a ještě po jednom poli na všechny strany ve vertikálním a horizontálním směru. Hráči se po tahu střídají. Hra končí, když na herní desce už není žádné volné pole. Vítězem je hráč, který obsadil nejvíce polí.



Obrázek 10 – Ukázka obsazování polí na desce



Obrázek 11 – Ukázka hry

Příloha – B Obsah přiloženého CD

Obrázek 12 – Obsah přiloženého CD