Dokumentace k projektu Gobango

Jan Sedlák a Aleš Dujíček

31. března 2010

Obsah

1 Úvod

Piškvorky je jedna z nejznámějších her, jejíž kořeny můžeme najít ve starověkém Japonsku. Již od pradávna se mnozí snaží najít co nejlepší výherní taktiku, programování umělé inteligence (AI) pro piškvorky patří po programování AI pro šachy mezi nejoblíbenější úlohy.

Gobango je aplikace pro hraní piškvorek, která obsahuje myší ovládané grafické prostředí pro hru proti jedné ze dvou AI, hru inteligencí proti sobě navzájem nebo hru člověka proti člověku. Program je napsán v jazyce C a využívá multiplatformní knihovny SDL a SDL_gfx, takže jej lze bez obtíží používat v systémech MS Windows, GNU/Linux a dalších.

2 O hře piškvorky

2.1 Pravidla

Hra piškvorky je stolní logická hra pro dva hráče. Hraje se na čtverečkované jednobarevné šachovnice. Každý hráč má své tokeny (většinou křížky a kolečka). Cílem hry je umístit své tokeny na šachovnici tak, aby vodorovně, svisle či diagonálně vznikla neporušená přímá řada pěti hráčových tokenů. Hráči se v umisťování tokenů střídají. Povoleno je samozřejmě umisťování tokenů pouze na volná místa. První, kdo vytvoří neporušenou řadu pěti tokenů, vyhrává.

Existuje samozřejmě několik taktik hry, rozumné je používat vlastní tokeny k blokování soupeřových řad. Častá je také snaha o vytvoření několika konkrétních kombinací, které hráče dostávají do výhodnější pozice.

2.2 Historie

Dnešní podoba piškvorek (angl. Tic-Tac-Toe či Five-in-a-row) se nejspíše vyvinula z oblíbené japonské hry Gomoku. V japonském Gomoku nejsou tokeny rozlišeny tvarem, ale barvou. Kladou se na křížení čar tvořících šachovnici. Oproti dnešním piškvorkám se dále liší pravidlem, zakazujícím vyhrát pomocí řady delší jak pět tokenů. Její kořeny sahají až k tradiční japonské hře Go, se kterou Gomoku sdílí nejen herní šachovnici (tzv. Goban), ale také některé charakteristiky.

V dnešních dobách vznikají snahy o vytvoření co nejlepšího algoritmu pro hraní piškvorek. Holandský programátor L. Victor Allis naprogramoval algoritmus, který na šachovnici 15*15 zajistí začínajícímu hráči výhru. Algoritmus však bohužel nebyl nikdy zveřejněn. Existují jednoduché matematické důkazy o tom, že na neomezené herní ploše musí existovat neprohrávající strategie pro začínajícího hráče.

3 Obecné algoritmy AI

Základem následujících algoritmů je existence nějaké ohodnocovací funkce, která dokáže každému možnému tahu přiřadit hodnotu, která vystihuje, jak je tah v daný okamžik výhodný. Liší se způsobem výběru nejlepšího tahu.

3.1 Prohledávání do šířky - hloubky

Jedná se o nejjednodušší algoritmus. Pracuje tak, že prochází všechny možné následující tahy a vždy si pamatuje ten doposud nejlepší. Jestliže následně nalezne ještě lepší, předchozí zahodí a nový si ponechá. Projde-li všechny tahy, oznámí tah, který si zapamatoval jako nejlepší.

Nevýhodou algoritmu je, že herní situaci hodnotí takovou, jak ji v daném okamžiku vidí a nezohledňuje následující možný vývoj, což může vést ke špatným rozhodnutím a následné prohře. Výhodou je pak jednoduchost a nenáročnost na systémové prostředky.

3.2 Minimax

Narozdíl od prohledávání do šířky - hloubky nevyhledává minimax nejcennější tah jen z pohledu aktuální situace, ale posuzuje tahy i z hlediska možné reakce soupeře. Při každém simulativně odehraném tahu pak vyzkouší odehrát i tah soupeře a zjistit tak úspěšnost svého tahu odehraného prvně. Takto zkouší rozehrát hru sám se sebou až do stanovené hloubky. Principem algoritmu je tedy procházení stromu hry a minimalizace maximálních možných ztrát. Algoritmus tedy hledá takovou větev, ve které je jeho zisk vždy co nejvyšší a soupeřovy reakce co nejnižší.

Nevýhodou algoritmu je vysoká časová náročnost, zvláště u úloh, kde dochází k velkému větvení. Výhodou pak volba výhodnějších tahů.

3.3 Alfa-beta ořezávání

Jedná se o optimalizaci algoritmu minimax. Oproti minimaxu je algoritmus vylepšen o rozhodování, zda-li má ještě smysl pokračovat v propočtech až do stanovené hloubky. 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. Pokud během propočtu narazíme na variantu, která je horší než alfa, víme, že to není hlavní varianta s nejlepšími tahy za oba hráče, protože máme lepší tah. Vyjde-li varianta lépe než beta, může se ji zase vyhnout soupeř a zahrát tah, který je lepší pro něj.

Výhodou oproti minimaxu je nižší časová náročnost, je tedy možné ve stejném čase vést výpočet, do větší hloubky.

4 Vlastní řešení

Ve funkci `main` se nejdříve ověří, zda uživatel nezadal parametr pro výpis nápovědy, v takovém případě dojde k jejímu vypsání a ukončení programu.

Poté se inicializuje SDL pro práci s grafikou a vytvoří se okno aplikace, ve kterém funkce `drawMenu` zobrazí nabídku, kde se rozhodne, jací hráči budou hrát, a jsou nastaveny ukazatele `player1` a `player2` na příslušné funkce. Nádledně se losuje, která z nich začíná hru. Funkce hráče vrací strukturu `TCoord` obsahující souřadnice svého tahu. Pokus o hru mimo herní pole nebo na obsazené pole je chybová návratová hodnota herní funkce a hra je ukončena. V cyklu se pak hráči střídají a funkce `checkWin` ověřuje, zda poslední tah nebyl vítězný, v tom případě je hra ukončena výhrou daného hráče. Hra je ukončena remízou v okamžiku, kdy je zaplněno celé herní pole.

Po skončení hry je volána funkce `gameEnd`, kde se podle volby hráče, zda se má sputit nová hra, nebo ukončit program.

4.1 Ovládání programu Gobango

Program Gobango má jednoduché GUI ovládané myší a klávesnicí. V příkazové řádce reaguje jen na parametry `-h` a `--help`, které způsobí výpis nápovědy a ukončené programu.

Po spuštění programu se objeví okno akplikace, ve kterém bude zobrazena prázdná herní šachovnice a dole ve stavovém řádku nabídka `[F1] AIxAI, [F2] AI1, [F3] AI2, [F4] human x human`. Výběrem z nabídky se určí, kdo bude hrát. Pořadí, kdo začíná a kdo hraje druhý, se na začátku každé hry náhodně losuje. Stisknutí příslušné klávesy vyvolá požadovanou akci:

- **F1** hra umělé inteligence proti umělé inteligenci.
- **F2** hra hráče proti první (jednodušší) AI.
- **F3** hra proti druhé (těžší) AI.
- **F4** hra hráče proti hráči.

Při hře AI proti AI se po automatickém odehrání hra ukončí. Během hry není možné program ukončit, je potřeba vyčkat, až dohrají.

Při hře hráče umisťuje hráč své tokeny stiskem levého tlačítka myši na příslušném místě. Pokud je ve hře na řadě hráč, může stisknout klávesy ‘q’ pro ukončení hry.

Po ukončení hry se zobrazí ve stavovém řádku nápis oznamující, jak hra skončila a nabídka `[q]uit [r]estart [m]enu`.

- **q** ukončení programu
- **r** start nové hry stejných hráčů

- **m** návrat do úvodního menu a výr nového typu hry

Oznámení 'hra byla ukoncena' znamená, že hru ukončil uživatel, nebo nastala nějaké chyba a hra nemohla pokračovat dále.

4.2 Pod pokličkou AI1

Naše první umělá inteligence funguje na opravdu jednoduchých principech. Pro vlastní potřebu jsme ji pokřtili na metodu „Franta hraje piškvorky". Jedná se o specifickou odnož metody zvané „Algoritmus prohledávání do šířky - hloubky". Umělá inteligence obsahuje funkci, která zjišťuje pro zadaný token posloupnost o zadané délce. Na začátku svého tahu zkopíruje celé herní pole, poté na každou volnou pozici dá svůj znak a zkontroluje, jestli se nejedná o výherní posloupnost (tím by vytvořila pětici a vyhrála).

Pokud neexistuje žádné takové místo, které by vedlo k výherní posloupnosti, zopakuje totéž, nicméně s nepřítelovým tokenem. Dosazuje nepřítelův token a zkouší, jestli by nepřítel nemohl vyhrát. Pokud ano, zabrání výhře dosazením vlastního tokenu. Algoritmus se opakuje znovu, nejdříve pro zabránění nepřítelových čtveřic, poté trojic a pak se už inteligence soustředí na tvoření co nejdelší vlastní posloupnosti. Jedná se částečně o brute-force metodu. Při hře AI1 nedochází k ohodnocování tahů.

AI1 je kratší a jednodušší inteligence, při soupeření s naší druhou inteligenci většinou prohrává. Mezi její výhody však patří „pozornost" - AI1 nemůže udělat chybu špatným ohodnocením políčka díky špatně nastavené prioritě (v konečném důsledku nevznikne situace kdy není jasné, proč AI1 položilo token na pozici na kterou ho položilo). Jedná se také o transparentnější AI.

4.3 Pod pokličkou AI2

Druhá umělá inteligence, kterou jsme vytvořili, pracuje na principu ohodnocování tahů. Pro každé volné políčko vyjádří číselně, jak by byl tah na něj výhodný, a to pro křížky a kolečka zvlášť.

Jak je tah výhodný neovlivňuje jen velikost řady znaků, která by takovýmto tahem vznikla, ale také to, zda je v daném směru dostatek místa pro celou výherní řadu, zda je řada znaků přerušená mezerou a zda je řada z jedné nebo obou stran blokovaná okrajem pole nebo soupeřovým znakem. Všechny tyto informace je pak potřeba převést na jediné číslo. Pro tento účel slouží vzorec $z \cdot (z - m) \cdot (z + 24 \cdot v)$, kde z je počet znaků, které už jsou v řadě, m je počet mezer, které řada znaků obsahuje, a v je počet volných konců řady znaků. Pokud není dostatečný prostor pro výherní posloupnost, pak je ještě číslo násobeno nulou. Vyjímku z tohoto vzorce má situace, kdy je počet znaků o jedna menší než počet výherních a současně není řada přerušena žádnou mezerou. V tomto případě je výhodnost tahu nastavena vysokou konstatou tak, aby tam AI za každých okolností hrála. Tato výjimka

je z toho důvodu, že snížení priority tahu tam, kde jsou znaky z obou stran blokované, je příliš vysoké, a přestože je takový tah určitě výherní, AI by tam nehrála.

Výhodnost tahu ale nezávisí jen na znacích v jednom směru, ale na celkovém postavení v poli. Je výhodnější hrát tak, aby vznikly dvě trojice znaků, než jen jedna, proto jsou priority tahů ze všech čtyř směrů nakonec sečteny, tento výsledný součet konečně udává, jak je tah výhodný.

Pozice všech tahů s nejvyšší prioritou jsou potom uchovávány v zásobníku, ze kterého je nakonec jeden tah náhodně vybrán. Nejlepší tahy pro kolečka a křížky jsou uchovávány v oddělených zásobnících, to proto, aby se podle hodnoty priority tahu dalo rozhodnout, zda už je nutné bránit soupeřově tahu, nebo je lepší volit útok. Toto rozhodování však zatím není nijak dokonalé, zatím se jen zvýhodňuje útok v situaci, kdy jsou si hodnoty nejlepších tahů rovny.

Tento postup vyhledávání se ukázal jako výhodnější, při hře proti předchozí AI ve většině případů vyhrává a nabízí mnohé další výhody, kterou je třeba možnost procházet možnosti několik tahů dopředu a vyhledávat, jakým tahem by AI později získala výhodu. Toto již ale není součástí řešení.

5 Závěr

Program Gobango nabízí uživateli možnost změřit své síly v hraní piškvorek. Přestože ani jedna z AI neprochází herní možnosti do hloubky a ohodnocuje jen aktuální situaci bez ohledu na možný vývoj v příštích tazích, zvláště druhá inteligence si vede obstojně i proti skutečným hráčům, kteří tuto možnost mají. Jejich nespornou výhodou je fakt, že se AI nestane, že by nějakou situaci přehlédla. Nevýhodou pak zůstává, že pokud hráč vypozoruje neoptimální chování AI v nějaké situaci, bude se vždy snažit ji před takovou situaci postavit a využít její slabinu k získání výhody.

Program Gobango byl testován na operačních systémech Microsoft Windows XP, Windows 7 a linuxových distribucích Arch Linux, Debian, Fedora a Ubuntu.

Použité zdroje

[1] Wikipedia
`http://cs.wikipedia.org/wiki/Minimax_(algoritmus)`
`http://cs.wikipedia.org/wiki/Alfa-beta_ořezávání`

[2] Jan Němec: *Šachové myšlení*
`http://www.linuxsoft.cz/article_list.php?id_kategory=241`

[3] Tomáš Novosád: *Problematika implementace umělé inteligence v herních programech*
`http://www.node.cz/download/XUIMIN_R06631_novosadtomas_sempr.pdf`