PŘÍRODOVĚDECKÁ FAKULTA UNIVERZITY PALACKÉHO
KATEDRA INFORMATIKY

PROJEKTOVÝ SEMINÁŘ

Osetinská dáma

Červenec 2017

Aleš Trunda
Aplikovaná informatika, II. ročník

Abstrakt

Cílem projektu bylo navrhnout a implementovat desktop aplikaci pro deskovou hru *Osetinská dáma*. Důraz byl kladen na to, aby aplikaci bylo možné pohodlně ovládat pomocí grafického rozhraní a aby aplikace nabízela i možnost počítačového hráče.

Obsah

1. Úvod

Aplikace byla vyhotovena v jazyce *C#* za použití běžně dostupných knihoven. Jako vývojové prostředí bylo použito *Visual Studio 2017*, toto prostředí je doporučeno pro sestavení aplikace. Uživatelské rozhraní bylo vytvořeno pomocí *Windows Forms.*

Vstupní bod aplikace je soubor *Program.cs*, je velice triviální - vytvoří instance hlavních tříd *GameRules*, *Desk* a *Engine* a následně je předá hlavnímu oknu aplikace *FormMain*, pomocí kterého se poté řídí běh celé aplikace. Toto okno pomocí grafického rozhraní řeší vstup a výstup pro uživatele a dále podle akcí uživatele vhodně pracuje s ostatními třídami.

2. Třídy

2.1 Desk

Třída reprezentující šachovnici, tedy obsahuje především seznam všech políček a seznamy políček, kde má který hráč figury. Jedná se o holé ztvárnění, třída neobsahuje žádnou logiku, například při provedení tahu se nekontroluje, zda tah odpovídá pravidlům, nebo zda tah provedl hráč, který je na tahu apod. Aby bylo možné šachovnici použít pro různé hry, ani není vhodné, aby šachovnice sama o sobě měla přehled o hře a mohla rozhodovat o tom, co je a není z pohledu hry proveditelné.

Z metod, které zde najdeme se jedná především o: přidávání a rušení polí, nastavení konkrétního pole, provedení a zpětné zrušení tahu (třída *Move*).

2.2 Engine

Ztvárňuje počítačového hráče, respektive logiku nalezení nejvýhodnějšího tahu v aktuální pozici. Z metod zmiňme ztvárnění **minimaxu** *minimax* a dále *getBestMove*, což je ta stěžejní metoda vracející nejvýhodnější tah.

Aby bylo možné mít robustní *Engine* nezávislý na konkrétní hře, je třeba mít pravidla hry určující ohodnocení pozice mimo třídu *Engine*. Tedy *Engine* nezná pravidla hry, řeší pouze vhodné zanořování minimaxu. V každém zanoření zkoumá u třídy *GameRules*, jediné třídy, která zná pravidla hry, zda aktuální pozice znamená konec hry, v tom případě sám nastavuje extrémní ohodnocení pozice. V opačném případě postupně pokračuje až na poslední úroveň zanoření a pak o ohodnocení pozice žádá opět *GameRules*, kde se rozhodne, jak je aktuální pozice pro zadaného hráče výhodná.

Dále stojí za zmínku, že vzhledem k faktu, že v jedné pozici obyčejně existuje několik vhodných tahů, je implementována možnost náhodného výběru jednoho z těchto nejlepších nalezených tahů. Důvodem je, aby logika počítačového hráče nebyla stereotypní.

Samotný postup pro určení nejlepšího tahu najdete v kapitole 4.

2.3 GameRules

Zajišťuje všechna pravidla hry, tedy který hráč je na tahu, jakým směrem lze pohybovat figurami, jaké je základní rozestavení figur na začátku hry, jaký tah je proveditelný apod. Ostatní komponenty získávají z této třídy potřebné informace o pravidlech hry a bez další logiky je slepě interpretují. K vytvoření nové podobné hry by tedy mělo stačit upravit pouze tuto jednu třídu.

Ve třídě najdeme několik proměnných určujících velikost hrací plochy, směry, který lze pohybovat figurami, počáteční rozestavení figur a podobně.

Následují metody, z těch hlavních se jedná o: určení následujícího hráče na tahu, nalezení všech možných tahů, rozhodnutí o konci hry a ohodnocení aktuální pozice.

2.4 GameVar

Sada konstant pro přehlednější a čistší práci s předem definovanými proměnnými.

2.5 Move

Jednoduchá třída reprezentující tah na šachovnici, rozlišujeme počáteční a koncové pole, dále skupinu polí, přes která se během tahu projde (v případě násobného skoku) a skupinu polí, která se při tahu odstraňují.

2.6 XMLHandler

Zajišťuje práci s XML soubory, které slouží pro ukládání a opětovné načítání hry, stěžejní metody jsou tedy *saveXML* a *loadXML*.

2.6.1 Uložení souboru

Ukládají se tato data: jméno hráče, zda hráče ovládá člověk nebo počítač a množina všech provedených tahů. Při načtení souboru se tyto tahy provádějí ze startovní pozice, což umožňuje ukládat pouze základní informace o hře. Například který hráč je na tahu, nebo historii tahů není třeba ukládat, provedením tahů tyto informace jednoduše opětovně získáme.

Nad rámec zvoleného řešení lze uvažovat i o ukládání času hry, případně o komplexnějším zpracování historie tahů, kdy v případě, že se například 3 tahy vrátí (*UNDO*) a následně se hra uloží, měli bychom po načtení hry možnost opět se o tyto 3 tahy posunout dopředu (*REDO*).

2.6.2 Načtení souboru

Při načítání dat se provádějí tahy ze startovní pozice, což je velice výhodná kontrola správnosti uložených dat. Nicméně tento přístup má nevýhodu v tom, že nelze načíst libovolnou pozici, pouze pozice, které mohou vzniknout z počátečního rozestavení figur. Ale možnost libovolného rozestavení figur ani nebyla uvažována při vývoji aplikace.

3. Práce s vlákny

Grafické okno s uživatelským rozhraním běží v jednom procesu, který ve smyčce kontroluje akce uživatele a vhodně na ně reaguje. Pokud ale obsluha akce vyžaduje delší čas na zpracování, po tuto dobu grafické rozhraní nereaguje na akce uživatele, takzvaně "zamrzá". V této aplikaci může být nalezení nejvýhodnějšího tahu výpočetně velice náročné, proto je třeba tuto operaci provádět ve vlastním vlákně, aby nedocházelo k blokování uživatelského rozhraní.

Dlouhý čas pro tah počítačového hráče tedy znamená problém, ovšem stejně tak i velice krátký - při hře počítač proti počítači by celá partie mohla trvat pouze pár sekund, pozorovatel by neměl čas prohlédnout si souboj. V tomto případě chceme zajistit, aby tah počítačového hráče trval nějaký minimální čas, ale samozřejmě aby to nebylo pasivní čekání, ale aby výpočet a čekání probíhalo zároveň. V tomto bodě se už tedy dostáváme na potřebu tří samostatných vláken a jejich synchronizace.

3.1 Synchronizace

Hlavní vlákno s uživatelským rozhraním je v tomto případě relativně pasivní, jen čeká až bude k dispozici tah a případně na žádost uživatele zastavuje vlákna.

Vlákna pro výpočet a čekání jsou podstatně zajímavější - obě musí běžet zároveň na pozadí a musí být nějak synchronizované, protože tah se bude provádět až poté, co skončí obě vlákna. A samozřejmě nelze vědět, které vlákno skončí dříve.

Řešit to lze jednoduchým příznakem určujícím, zda vlákno skončilo či nikoli. Tedy jakmile vlákno skončí, zkontroluje se, zda skončilo i to druhé, pokud ne čeká se dál, pokud ano předá se řízení hlavnímu vláknu a provede se tah. Aby nedošlo k uváznutí, přístup vláken ke sdíleným proměnným je řešen **zámkem**.

4. Výpočet pro nalezení nejlepšího tahu

Výpočet je postaven na metodě minimax, výchozí zanoření je 2, což umožňuje prakticky okamžitý tah počítačového hráče. Během výpočtu se hrací deska nekopíruje, je stále jedna a pouze se podle potřeby mění, což podstatně snižuje výpočetní nároky. Konkrétně se na desce provede zkoumaný tah, rekurzivně se zavolá minimax s nižším zanořením a po skončení se tah na desce zase vrátí zpátky.

Jakmile se dojde na poslední úroveň zanoření ohodnotí se pozice, z těchto ohodnocení se na závěr vybere to nejvýhodnější s tím, že i každé zanoření má na vliv na toto ohodnocení tak, aby se například v případě nalezení více tahů vedoucích k vítězství, vybrala ta nejrychlejší varianta.

Dále je možné uvažovat i rozšíření o **alfa/beta prořezávání**, nicméně tato varianta nebyla implementována, sice snižuje výpočetní nároky, ale zároveň také snižuje úroveň počítačového hráče - vzhledem k faktu, že skákání figur je povinné, oběť figury, která se z počátku jeví jako nevýhodná, může v následujících krocích znamenat zisk. U alfa/beta prořezávání by tato oběť byla pravděpodobně ihned zamítnuta jako nevýhodný tah.

4.1 Ohodnocení pozice

Nejprve zmiňme, že pozice znamenající konec hry je již předně kontrolována v Engine, tedy samotné ohodnocení pozice hodnotí vyloženě pouze postavení figur na šachovnici.

Základní ohodnocení pozice je stanoveno na základě rozdílu v počtu figur zkoumaného hráče a soupeře. Všechny figury mají stejnou hodnotu a ani fakt, že figura dojde na konec šachovnice apod. nehraje roli.

Jako další kritérium pro hodnocení se již nabízí pouze samotná pozice figur s tím, že pozice figury je pouze okrajové hodnocení, rozdíl v počtu figur tedy bude mít podstatně vyšší váhu než jejich pozice.

U pozice figury chceme docílit toho, aby mohla skončit soupeřovy figury, a zároveň aby sama nemohla být skočená, nabízejí se tyto základní varianty:

4.1.1 pozice figury na kraji desky

Umístění figury na okraj desky soupeřovi podstatně snižuje možnosti skočení, v případě, že se figura nachází přímo v rohu nemůže být skočená vůbec. Na druhou stranu ale figuře omezujeme rozsah pohybu a také jí omezujeme možnosti, aby sama skončila. Každopádně se to jeví jako zajímavý mechanismus, který by mohl vést ke zlepšení logiky počítačového hráče.

Mechanismus byl zkoumán na pár desítkách her, výsledky neprokázaly, že by tento mechanismus znatelně zlepšoval sílu počítačového hráče. Nicméně ve hře je implementován, ve výchozím stavu je vypnutý, zapíná se automaticky při vyšší obtížnosti počítačového hráče.

4.1.2 pozice figury ve středu desky

V této variantě sice figura může jednoduše skočit protihráčovy figury, ale stejně jednoduše může být i sama skočená, tedy nejeví se, že by varianta mohla přinášet nějakou výhodu.

4.1.3 pozice figury v okolí vlastních figur

Shromažďování vlastních figur v jednom místě sice soupeřovi snižuje možnosti skočení, ovšem na druhou stranu nabízí soupeřovi možnosti násobných skoků, což může mít fatální dopad na logiku počítačového hráče. Pro vyvození nějakých závěrů by ale bylo třeba provést testy.

4.2 Závěr

Sílu počítačového hráče nejvíce ovlivňuje počet zanoření v metodě minimax, samozřejmě vyšší zanoření podstatně zvyšuje výpočetní náročnost, tedy je vhodné volit co nejvyšší zanoření, které je z pohledu složitosti výpočtu akceptovatelné.

Mechanismy pracující s pozicí figury se nejeví jako příliš účinné.

5. Vyskakovací okna

Všechna okna mimo *Game Rules* přebírají od hlavního okna řízení, tedy uživatel musí provést nějakou akci, aby se okno zavřelo, až poté je možné pokračovat ve hře v hlavním okně.

Okno *Game Rules* je záměrně otevíráno bez vazby na hlavní okno, aby uživatel mohl hrát hru a zároveň mít otevřené toto okno s pravidly ke hře.

5.1 About

Obsahuje informace o aplikaci, otevírá se z hlavního menu.

5.2 Main

Hlavní okno aplikace, otevřeno se spuštěním aplikace.

5.3 Game Over

Souhrn o ukončené hře, otevírá se automaticky po ukončení hry.

5.4 Game Rules

Informace o hře a pravidlech, otevírá se z menu nebo stiskem F1 jako okno s nápovědou.

5.5 New Game

Základní nastavení nové hry, otevírá se z hlavního menu.

5.6 Settings

Obsahuje kompletní nastavení hry, otevírá se z hlavního menu.

5.7 Systémová okna

Jsou to okna jejichž podobu a interakci zajišťuje operační systém, nejsou obsaženy v aplikaci.

5.7.1 Výběr souboru

Jedná o okna pro výběr soubory pro funkcionalitu uložení a načtení hry ze souboru.

5.7.2 Dialogy

Pro interakci s uživatelem jsou použity různé dialogy, které vyžadují akci uživatele, aby uživatel potvrdil, že je s touto informací seznámen. Je to například dialog informující, že otevíraný soubor nelze načíst protože je poškozen.