UNIVERZITA PALACKÉHO V OLOMOUCI
KATEDRA INFORMATIKY

ROČNÍKOVÝ PROJEKT

Hra 16 vojáků

Michal Kandr
Aplikovaná informatika, II. ročník

červen 2008

Abstrakt

Popis projektu jehož cílem je implementace aplikace deskové hry 16 vojáků. Program je vytvořen v jazyce Java. Dokument obsahuje programátorskou dokumentaci s popisem struktury aplikace, jednotlivých tříd a jejich hlavních metod.

Obsah

1. Požadavky na projekt

- korektní implementace pravidel hry (nemožnost provést tah odporující pravidlům, správné ukončení hry, a podobně)
- algoritmy pro herní strategii, nastavitelná obtížnost hry v adekvátním rozsahu
- možnost hry dvou lidí, člověka proti počítači, a počítače proti počítači
- možnost nastavit a kdykoliv změnit obtížnost i v průběhu hry
- možnost kdykoliv zaměnit počítačového a lidského hráče, nebo černého a bílého hráče (bez ohledu na to, je-li hráčem člověk nebo počítač, změna i v průběhu hry)
- nápověda nejlepšího tahu
- ukládání a načítání (ukončených, rozehraných) partií
- undo/redo tahů do libovolné úrovně
- prohlížení historie tahů (přehledné zobrazení provedených tahů)
- zpětné přehrání partie po jejím dokončení (replay) s možností zastavení přehrávání, pohybu v historii stavů hry a opětovného rozběhnutí hry ze zvoleného stavu. Při zastavení přehrávání možnost začít novou hru nebo otevřít uloženou hru.
- robustnost (program musí reagovat správně na nesprávné uživatelské vstupy, zejména ovládání, vadný formát souboru apod., aplikace nesmí havarovat)
- vestavěná nápověda
- grafické uživatelské rozhraní (GUI) se zpracované podle standardů
- program ve spustitelné formě, je-li to pro zprovoznění aplikace nutné pak také instalátor
- kompletní zdrojové kódy programu včetně dalších částí nutných pro sestavení aplikace
- programátorská dokumentace k projektu vytvořená pomocí závazného stylu (PDF verze a zdrojová verze). Obsahuje zejména popis struktury kódu, algoritmy hry včetně herní strategie, postup pro sestavení aplikace. Dokumentace nemusí obsahovat uživatelskou příručku.

2. Architektura aplikace

Aplikace je členěna do dvou nezávislých celků jádra a uživatelského rozhraní. Jádro obsahuje veškerou aplikační logiku, takže je samo o sobě po algoritmické stránce plně funkční implementací hry. Grafické rozhraní představuje zobrazovací logiku aplikace a zajišťuje komunikaci s uživatelem.

Na straně jádra aplikace zajišťuje komunikaci s okolním světem (v tomto případě s uživatelským rozhraním) třída Manager, na straně uživatelského rozhraní komunikaci s jádrem realizuje třída UIAdapter.

Jádro je na uživatelském rozhraní zcela nezávislé a rozhraní tak může být libovolně změněno aniž by se změna nějak projevila v jádře, stačí na třídu UIAdapter navázat jinou implementaci uživatelského rozhraní se kterou bude komunikovat. Totéž samozřejmě platí i pro jádro aplikace.

Aplikace běží standardně ve dvou vláknech jedno vlákno se stará o GUI, ve druhém běží samotná hra. To umožňuje zachovat rychlou reakci GUI i při náročných výpočtech prováděných počítačovým hráčem a umožňuje lepší oddělení jádra od GUI. Pro přehrávání historie hry (funkce replay) je krátkodobě použito ještě třetí vlákno.

3. Jádro aplikace

Třídy tvořící jádro aplikace a jejich hlavní metody. Metody triviální, které pouze delegují funkčnost do příslušných jiných tříd, a metody pomocné s minoritním významem pro hlavní architekturu aplikace vynechávám lze dohledat ve zdrojovém kódu a nevyžadují explicitní vysvětlování.

3.1. Třída Manager

Třída Manager je hlavní třídou jádra a zajišťuje funkčnost aplikace reprezentuje samotnou hru a řídí práci ostatních tříd jádra. Manager realizuje hru, implementuje rozhraní pro komunikaci s vnějším světem, jsou v něm definovány všechny v aplikaci používané konstanty a synchronizují se přes něj jednotlivá vlákna.

3.1.1. Metoda Run

Nejdůležitější metoda třídy Manager, která realizuje samotnou hru. Hra je realizována cyklem, ve kterém se nejdříve zjistí který hráč (černý nebo bílý) je na tahu, od třídy Arbiter se zjistí jaké tahy může hráč provést, kolekce takto získaných povolených tahů se předá příslušnému hráči, který vrátí index tahu jež si vybral a tento tah se provede.

Cyklus je možné dočasně pozastavit nastavením synchronizační konstanty STOP_MAIN na hodnotu true, což způsobí uspání cyklu až do doby, kdy hodnota konstanty opět nenabude hodnoty false. Toho využívají například funkce manipulující s historií hry nebo měnící nastavení hráčů. Není-li cyklus zastaven, probíhá dokud není konec hry, pak se zastaví sám a čeká dokud mu nějaká jiná metoda znovu nepovolí běh.

Třída Manager implementuje rozhraní Runnable (proto se také tato metoda jmenuje Run), takže tento cyklus a vše co je z něj voláno běží v samostatném threadu nezávislém na uživatelském rozhraní.

3.1.2. Metoda Replay

Realizuje přehrání historie tahů a je spouštěna v samostatném threadu, tzn. nezávisle na threadu ve kterém běží samotná hra realizovaná metodou Run.

Před zahájením činnosti samozřejmě ukončí vybírání tahů v třídách hráčů a pozastaví běh hlavní smyčky hry. Pak již zcela triviálně přetočí historii tahů na začátek a přehrává tahy.

Její činnost může být ukončena nastavením konstanty STOP na true, stejně jako činnost hráčů při výběru tahů.

3.2. Třída PlayingArea

Tato třída reprezentuje stav hry uchovává aktuální stav hrací desky, historii provedených tahů a implementuje operace pro manipulaci s nimi.

3.2.1. Reprezentace hrací desky

Deska je reprezentována dvourozměrným polem čísel, kde je hodnotou konkrétního prvku rozlišeno, zda na dané pozici je bílý kámen, černý kámen, volná pozice nebo jde o místo mimo hrací plochu.

3.2.2. Reprezentace tahu

Tah reprezentuje třírozměrné pole čísel. První rozměr tvoří dvourozměrná pole reprezentující jednotlivé základní instrukce tahu. Libovolný tah lze rozložit na posloupnost tří typů instrukcí odeber kámen, přidej kámen a změň hráče který je na tahu.

Instrukce tahu je tvořena dvěmi dvouprvkovými poli v prvním je typ instrukce a barva kamene pro který se to má provést, v druhém pak souřadnice x a y na kterých se má operace provést (u instrukce pro změnu hráče je pole logicky prázdné).

3.2.3. Reprezentace historie

Při takto zavedené reprezentaci tahu je implementace historie zřejmá kolekce tahů v pořadí v jakém byly provedeny a proměnná s indexem tahu který byl proveden naposledy (kvůli posunu v historii).

3.2.4. Metoda Move

Zahraje předaný tah projde popořadě jeho jednotlivé instrukce a pomocí příslušných metod (addPiece, deletePiece a setOnMove) je provede.

3.2.5. Metoda ReverseMove

Pracuje stejně jako Move, jen u každé instrukce provede instrukci přesně opačnou.

3.2.6. Metody forward a backward

Realizují přesuny v historii tahů forvard zahraje další tah v historii následující po posledním zahraném tahu, je-li nějaký, a backward vrátí poslední zahraný tah, je-li nějaký.

3.2.7. Metoda clearNextMoves

Smaže vše v historii co následuje za aktuálním tahem nezbytné v případě že se uživatel vrátí o několik tahů zpět a začne hrát od této pozice.

3.3. Třída Arbiter

Reprezentuje ve hře rozhodčího, jeho úkolem je zjisti jaké tahy může hráč určité barvy provést při konkrétním stavu na hrací desce a zjišťovat zda není konec hry.

3.3.1. Metoda getValidMoves

Realizuje hlavní funkčnost Arbitera pro zadanou pozici najde všechny pozice kam z ní může hráč který je na tahu přesunout svůj kámen při aktuálním stavu na desce.

Metoda si nejdříve zjistí na jaké pozice je možné z požadované pozice jít v případě že je deska prázdná (tj. všechny sousední pozice ke kterým vede čára z aktuální) tato informace je uložena v poli VALID_MOVES které obsahuje sousední pozice pro horní levou čtvrtinu desky (deska je symetrická, takže pro zbytek desky lze jednoduše dopočítat).

Takto nalezené pozice projde a je-li pozice volná, přidá příslušný tah do kolekce povolených tahů. Je-li na pozici soupeřův kámen a aktuální hledání není hledáním druhého tahu v bočních trojúhelnících desky (viz. další odstavec), ověří zda další pozice v přímém směru (patří-li taková pozice do desky a lze-li na ni jít) je volná a pokud ano, přidá do kolekce povolených tahů přeskočení tohoto kamene a zavolá pomocnou metodu getJumps (popis viz dále).

Pokud je počáteční pozice v některém z bočních trojúhelníků, znamená to že hráč může táhnout o dvě pozice. Projdou se všechny nalezené tahy a jde-li o tah o jednu pozici, zavolá se metoda getValidMoves pro cílovou pozici a do seznamu tahů se přidají všechny tahy z původní pozice do pozic na které lze jít z koncové pozice aktuálního tahu.

3.3.2. Metoda getJumps

Pomocná metoda metody getValidMoves, která provede předaný tah, zavolá metodu getValidMoves pro koncovou pozici provedeného tahu, tah vrátí zpět a nakonec vrátí kolekci tahů vzniklých prodloužením předaného tahu o všechny tahy vrácené metodou getValidMoves. Takto jsou nalezeny všechny vícenásobné přeskoky které může hráč potenciálně provést

3.3.3. Metoda getAllValidMoves

Vrátí kolekci všech tahů které může provést hráč který je zrovna na tahu. Projde všechny pozice na desce a je-li na pozici kámen hráče které je na tahu, zavolá pro tuto pozici metodu getValidMoves a přidá do kolekce kterou bude vracet tahy touto metodou vrácené.

3.3.4. Metoda isGameOver

Zjistí zda je konec hry.

3.3.5. Metoda getState

Vrátí konstantu reprezentující aktuální stav hry hra probíhá, vyhrál bílý, vyhrál černý nebo remíza.

3.4. Hráči interface Player

Hráče reprezentuje interface Player, který předepisuje že každý hráč musí mít metodu getMove, která dostane seznam povolených tahů a vrátí index tahu který hráč zvolil.

3.5. Třída HumanPlayer

Reprezentuje lidského hráče a výběr tahu zajišťuje uživatelské rozhraní, se kterým třída komunikuje přes třídu UIAdapter.

3.5.1. Metoda getMove

Požádá UIAdapter o tah, přičemž mu předá seznam povolených tahů, a čeká dokud tah není k dispozici.

3.6. Třída ComputerPlayer

Reprezentuje počítačového hráče a tah vybírá pomocí algoritmu minimax.

3.6.1. Metoda getMove

Projde všechny tahy ze kterých může vybírat, každý tah provede a požádá třídu MiniMax (viz dále) o hodnocení vzniklého tahu. V případě že tah je opačný k poslednímu tahu který hráč udělal, sníží mu hodnocení o polovinu, aby nebyl opakovaně přesouván jeden kámen mezi dvěmi stejnými pozicemi. Bez takové úpravy ohodnocení by docházelo k situacím, kdy počítačový hráč opakovaně přesouvá jeden kámen mezi dvěmi pozicemi dokud mu protihráč nedá příležitost sebrat jeho kámen (tzn. při hře dvou počítačů se oba zacyklí). Z tahů se vybere tah s nejlepším hodnocením.

3.7. Třída MiniMax

Realizuje rozhodovací logiku počítačového hráče pomocí algoritmu minimax.

3.7.1. Metoda getRate

Pro aktuální stav hrací desky zjistí jeho výhodnost aplikací algoritmu minimax do požadované hloubky a ohodnocení pomocí předané ohodnocovaní funkce Rate.

3.8. Třída Rate

Zajišťuje ohodnocení aktuálního stavu hrací desky.

3.8.1. Metoda getRate

Dostane aktuální hrací desku a barvu hráče z jehož pohledu má stav desky ohodnotit a vrátí výhodnost stavu pro tohoto hráče.

Základem hodnocení je poměr mezi počtem kamenů hráče a jeho soupeře čím více kamenů hráč má a čím méně jich má jeho soupeř, tím lépe. Druhým kritériem, které má však menší váhu než první, je celkový počet kamenů na desce čím méně, tím lépe. Kdyby druhé kritérium nebylo použito, počítačový hráč by neochotně dělal výměny pokud sebere soupeřův kámen a ten ho následně sebere jemu, poměr mezi počtem kamenů hráče a soupeře se nezmění, tudíž všechny tahy budou rovnocenné. V praxi je však lepší, obzvláště na začátku, výměnu udělat a pročistit plochu. Jinak by hraní proti takovému hráči asi žádného uživatele nenadchlo, neboť počítač by jeho kameny nepřeskakoval a jen by si bránil svoje, tzn. neudělal-li by protihráč chybu, takto mírumilovně by se pokračovalo do nekonečna.

Posledním kritériem, tentokráte s nejmenší vahou, je průměrná vzdálenost kamenů hráče od kamenů protihráče. čím blíže jsou, tím lépe. To zajišťuje, že se počítačový hráč bude přibližovat soupeři a útočit na něj, jinak by byl pořád na své straně a nepřibližoval by se. Krom toho by mu příliš dlouho trvala koncovka, ve které je třeba pronásledovat poslední protihráčovy kameny.

Výsledná strategie je pak použitelná pro všechny fáze hry začátek, průběh i konec a jiných ohodnocovacích funkcí není zapotřebí.

3.9. Třída Export

Zajišťuje uložení stavu hry do souboru ve formátu XML.

3.9.1. Metoda Save

Uloží historie tahů a nastavení hráčů do specifikovaného souboru.

3.9.2. Metoda Load

Načte ze souboru historii tahů a nastavení hráčů, ověří platnost soboru pomocí kontrolního hashe a je-li platný vynuluje hrací desku, provede na ní všechny načtené tahy, následně vrátí tolik tahů kolik má být vráceno (pozice v historii tahů je určena parametrem u posledního zahraného a nevráceného tahu) a nastaví hráče podle údajů v souboru.

4. Uživatelské rozhraní

Popis části s uživatelským rozhraním bude spíše stručnější, neb na něm není nic moc zajímavého většinu tvoří kód grafiky naklikané ve vývojovém prostředí.

4.1. Třída UIAdapter

Zajišťuje komunikaci mezi jádrem a uživatelským rozhraním. Implementuje metody které jsou k tomu potřeba a sama defacto nic zajímavého nedělá, jen přeposílá požadavky a deleguje svoji funkčnost jinam.

4.2. Třída Gui

Obsahuje samotnou grafiku a obslužné metody starající se o obsluhu událostí a komunikaci s uživatelem. Grafika je implementována v základní grafické knihovně Javy knihovně Swing.

Většina metod jsou vcelku triviální ovladače událostí, které jen volají třídu UIAdapter a mění grafické rozhraní, tudíž nemá cenu je zde popisovat. Za zmínku stojí pouze veřejné metody které volá UIAdapter a ovladač kliknutí na hrací plochu (areaClick), kterážto metoda je na rozdíl od ostatních složitější a komentář si zasluhuje.

4.2.1. Metoda wantMove

Metodu zavolá třída lidský hráč když chce požádat uživatele o vybrání tahu který chce zahrát. Parametrem je kolekce povolených tahů ze kterých lze vybírat. Metoda nastaví interní proměnné třídy tak, aby ovladač kliknutí na plochu umožňoval vybrat tah.

4.2.2. Metoda dontWantMove

Zavolá ji lidský hráč v okamžiku kdy chce dát najevo že výběr tahu je ukončen buď když si přečetl číslo tahu který uživatel vybral, nebo když byl jeho běh přerušen. Metoda vynuluje doposud vybraný tah a proměnné s výběrem tahu související a nastaví interní proměnné tak, aby nešlo vybírat tahy.

4.2.3. Metoda getMove

Metodu volá lidský hráč když se chce podívat, jestli uživatel již vybral nějaký tah, nebo ne. Pokud uživatel ukončil výběr tahu vrací číslo vybraného tahu, nebo -1 v případě opačném.

4.2.4. Metoda endOfGame

Signalizuje uživateli že je konec hry.

4.2.5. Metoda paintArea

Zobrazí plochu předanou jako pole integerů místo plochy aktuálně zobrazené (updatuje stav plochy).

4.2.6. Metoda paintMoveHistory

Zobrazí uživateli historii tahů předanou jako kolekce tahů v chronologickém pořadí. Aktuálně zobrazenou historii smaže a nahradí historií předanou.

4.2.7. Metoda areaClick

Ovladač události kliknutí na hrací plochu. Je-li hra pozastavena, spustí ji a pokud není nastaveno že uživatel může vybírat tahy (zavoláním metody wantMove), nic dalšího neudělá. Následně ověří zda uživatel klikl na nějaký kámen a zda je jeho barva shodná s barvou hráče který je na tahu a pro kterého se má vybrat tah.

Pokud uživatel provedl dvojklik na koncovou pozici vybraného tahu (případně pozici jejímž vybráním bude doposud vybraný tah ukončen), tento tah se provede (tah se označí jako ukončený a pošle se signál třídě hráče).

Klikl-li jen jednou na pozici která je součástí vybrané posloupnosti pozic (ať již tvoří kompletní tah nebo jen jeho část), tato pozice se odznačí stejně tak jako všechny pozice v posloupnosti následující (označené po aktuální).

Jinak se projdou všechny povolené tahy a pokud již vybrána posloupnost pozic spolu s pozicí na kterou se kliklo tvoří začátek posloupnosti tvořící některý z povolených tahů, pozice se označí a odpovídající tah se vybere. Pokud je vybrána posloupnost a pozice na kterou se kliklo je pozicí na níž lze jít kamenem kterým chce hráč hrát (tzn. je to začátek jiného tahu stejným kamenem), původní posloupnost se smaže a nahradí se tahem na tuto pozici. A konečně pokud hráč klikne na jiný kámen kterým může hrát, vybraná posloupnost se zruší a označí se kámen na který se kliklo. To samé se stane pokud není žádný kámen vybrán.

4.3. Třída GraphicArea

Grafická komponenta reprezentující hrací desku. Na pozadí zobrazuje obrázek hrací desky a obsahuje pole pozic na desce reprezentovaných instancemi třídy Piece, jehož struktura je obdobou struktury pole reprezentujícího hrací desku ve třídě PlayingArea.

Třída se umí nakreslit, nastavit barvu některé z pozic na desce, zjistit zda určitá pozice obsahuje bod s konkrétními souřadnicemi a upravit barvu bodů tak, aby stav odpovídal konkrétnímu stavu hrací desky.

4.4. Třída Piece

Grafická komponenta reprezentující pozici na hrací ploše. Má barvu odpovídající stavům pozice (prázdná, obsazená bílým kamenem, obsazená černým kamenem a označená) kterou lze nastavit a získat, umí se nakreslit v podobě kolečka s vzhledem odpovídajícím určitému stavu a umí říct zda určitý bod je uvnitř ní.