Programátorská dokumentace

Petra Emmerová

1 Technické údaje

Aplikace je typu Windows Form application, napsaná v jazyce C#, pod .NET frameworkem 4.5.2. Samotné řešení se skládá ze dvou projektů, hlavní aplikace pojmenované *GeneralBoardGame*, a vedlejší konzolové aplikace *GameTester* určené pro testování algoritmů umělé inteligence ve hrách proti sobě. V grafickém rozhraní jsou použity figurky dostupné na wikimedia commons pod public domain licencí.

V první části se zaměřím na samotnou aplikaci pro hraní her, *GeneralBoardGame*.

Tato aplikace je postavena podle návrhového vzoru *Model-View-Controller*, popis tříd proto rozdělím do těchto tří oblastí.

1.1 View

View obsahuje soubor tříd starající se o vzhled interface a získávaní uživatelského vstupu.

- **MainGameWindow** obsahuje samotný design hlavního okna aplikace. Tento vzhled je složen z elementů Button, Label a ComboBox, u nichž také jejich třída zajišťuje viditelnost. Např. po uživatelově kliknutí na tlačítko *Šachy* tato třída skryje úvodní obrazovku a zobrazí uživateli panel dávající mu na výběr další parametry hry.

  Dále je zde uložena důležitá metoda *DrawChessboard*, jež po svém zavolání vykreslí herní šachovnici a vloží na ni figury. Tato šachovnice je bitmapová, kde jednotlivá políčka jsou přímo vkreslena do okna aplikace, a herní pole, s nimiž hráč může interagovat, jsou reprezentovány pomocí třídy PictureBox.

  Na šachovnici jsou rozestavěny dvě vrstvy PictureBox objektů. První vrstva, tvořena PictureBox objekty s průhlednou barvou, zajišťuje interaktivitu s uživatelem na každém poli. Tyto objekty jsou uloženy ve dvojrozměrném poli *pictureBoxes*, logicky stejně jako šachovnice.

  Na každém objektu PictureBox může být poté položen další, který představuje příslušnou figuru. Tyto jsou uloženy v dvojrozměrném poli *pictureBoxes*, kopírujícím herní šachovnici.

  Poslední důležitá metoda v této třídě je *AddPieceToBoard*, která na vstupu dostane souřadnice a číslo figury a na dané souřadnice umístí nový objekt PictureBox se zadanou figurkou.

- **CustomGameInit** se stará o tvorbu vlastní hry za pomocí interface uživatelem. Od uživatele zjistí příslušné hodnoty jako rozměry šachovnice a typ hry a poté v metodě *DrawCustomChessboard* vykreslí prázdnou šachovnici a zobrazí menu s výběrem figur.

  Po vybrání figury a kliknutí na volné políčko se zavolá metoda *GetPieceNumberFromUserAndAdd*, jež nejprve pomocí názvu figury v objektu ComboBox *CustomGameChooseCombobox* zjistí, o kterou figuru se jedná, a poté pomocí metody *AddPieceToGame* zajistí její přidání do hry.

  Poté, co uživatel klikne na tlačítko *Hrát*, se zavolá metoda *NewGameButton_Click*, v níž se nejprve zkontroluje, zda uživatelem vytvořená hra splňuje požadavky validní hry, tedy zda hra typu šachy má na obou stranách šachového krále, hra typu šogi má na obou stranách šogi krále, a zda je na každé straně alespoň jedna figura. Pokud tomu tak není, je na daný problém uživatel upozorněn, v opačném případě začíná hra.

- **BoardMovement** se stará o vizualizaci pohybu figurek na šachovnici, a čte uživatelův vstup ohledně šachovnice.

  Po kliknutí na políčko šachovnice se zavolá metoda *MoveGamePiece*, jež se dle momentálních parametrů rozhodne, co se právě chystá hráč udělat. Může přidávat figuru v rámci tvoření nové hry, přidávat figuru v rámci šogi hry, vybrat figuru se kterou chce udělat tah, vybrat pole na níž chce přesunout vybranou figuru, nebo se také může stát, že klikl na prázdné pole či soupeřovu figuru, aniž by na něj mohl přesunout figuru svoji. V každém případě vykoná tato metoda příslušnou změnu a v případě validního tahu v singleplayer módu dá signál pro spuštění algoritmu na výběr dalšího tahu.

  Pokud se figura přesune na políčko, na kterém se změní v jinou, tato třída se o to také postará. Změna figury může nastat v případě:

  - Pokud dojde figura pěšáka na druhou stranu šachovnice, uživateli se zobrazí okno **PawnChange**, v němž si může vybrat z dámy, koně, střelce a věže, za kterou figuru se má pěšák vyměnit. V umělé inteligenci se vždy figura povýší na dámu.
  - Pokud kámen dámy dojde na druhou stranu šachovnice, automaticky se mění v dámu.
  - Pokud šogi figura dojde na třetí poslední políčko či dále, může být povýšena. Ve hře umělé inteligence je figura povýšena automaticky, hráči se ale po vstupu na dané políčko zobrazí vyskakovací okno **Propagation**, kde se rozhodne, zda chce figuru povýšit.

  Po provedení tahu se zavolá metoda *EndGame*, kontrolující, zda nenastal konec hry podle pravidel jednotlivých typů her. Dále v šachu kontroluje, zda není ohrožen král. Pokud se tomu tak stalo, aplikace to oznámí textem na *GameStateLabel* a hra se nastaví jako skončená.

- **ShogiAddPiece** se stará o přidání figury do hry typu šogi. Její funkce jsou rozděleny mezi spodního a horního hráče, neboť ke každému patří jiný ComboBox a

jiný set figur. *ChooseShogiButtonBottom_Click* a *ChooseShogiButtonUpper_Click* se zavolá poté, co uživatel klikne na tlačítko *Přidat* u příslušného ComboBox objektu, a nastaví proměnnou *AddBottomShogiPiece* či *AddUpperShogiPiece* na *true*, což v příštím kliknutí na šachovnici zajistí přidání zvolené figury. Metoda *AddUpperShogi* nebo *AddBottomShogi* je zavolána po zvolení prázdného pole na šachovnici, načež je na něj vložena daná figura.

Pokud je vkládaná figura šogi pěšák, probíhá kontrola sloupce, do něhož je vkládán, neboť dle pravidel nesmí být v jednom sloupci dva pěšáci stejné strany. V případě, že je tato podmínka nesplněna, se zobrazí okno *ShogiPawnProblem*, informující hráče o tomto problému, a zruší se přidávání figury na plochu.

1.2 Model

Model v sobě uchovává logickou reprezentaci hry, její momentální stav, figury a tahy.

- **Board** reprezentuje šachovnici, na níž se odehrává hra. Obsahuje statické dvojrozměrné pole *board* typu *Pieces*, které představuje logickou reprezentaci momentální partie.

  Metoda *AddPiece* zajistí přidání figury do logické reprezentace na základě jejího číselného označení - vytvoří nový objekt *Pieces* a vloží jej na požadovanou souřadnici v poli.

  MakeCustomPiece vytvoří objekt figury definované hráčem. Tato metoda je volána z *AddPiece* v případě potřeby vytvořit novou figuru.

- **Gameclass** obsahuje popis momentální partie. Obsahuje tři objekty enum: *GameType*, *PlayerType* a *AlgorithmType*, spolu se statickými proměnami daného typu *gameType*, *whiteGameType*, *blackGameType*, *playerType*, *playType*, *whitePlayType*, *blackPlayType*, a *algorithmType*, uchovávající typ konce hry pro příslušnou stranu, typ hry příslušné strany, počet hráčů a typ algoritmu během dané partie.

  Dále tato třída obsahuje metody, jež hlídají, zda hra skončila, zda je král v šachu či zda v momentální hře dámy musíme vzít nepřítelovu figuru.

- **LoadGame** se stará o načítání hry ze souboru. Po stisknutí tlačítka *Načíst hru* se zavolá metoda *LoadGameButton_Click*, jež načte vybraný JSON soubor, vytvoří z něj objekt *CustomGame* a nastaví příslušné parametry. V případě chyby v načítaném souboru ukáže vyskakovací okno s popisem chyby.

  Při definování nové figury bez váhy používá metodu *GetPieceValue*, jež dle tahů, které může figura dle své definice provádět, vygeneruje hodnotu figury pro použití v minimaxovém algoritmu.

- **Moves** slouží pro generování a uložení tahů figur. Obsahuje čtyři statické objekty typu List pro uložení souřadnic, *start_x, start_y,* kde se uchová startovní pole figury, a *final_x, final_y,* pro uložení koncového pole.

  Pro uschování souřadnic mimo tyto globální seznamy slouží třída *CoordinatesCopy*, do které se pomocí metody *MakeCopyEmptyCoordinates* dají přesunout aktuální souřadnice tahů.

Dále třída obsahuje generování jednotlivých tahů. Každý jednotlivý tah je implementován jako jedna metoda. Na vstupu dostane tah souřadnice figury, pro níž tah generujeme, a šachovnici s figurami. Poté se do statických seznamů pro uložení souřadnic vloží hodnoty reprezentující daný tah.

Dále třída obsahuje metody pro hromadnou práci s vygenerovanými tahy: *GetCount* vrátí počet vygenerovaných tahů, *RemoveAt* smaže z listu souřadnic tah na daném indexu, *Delete* smaže z listu souřadnic zadanou hodnotu, *ReplaceAt* nahradí souřadnice na daném indexu za zadané číslo a *EmptyCoordinates* vyprázdní seznamy souřadnic.

- **Pieces** obsahuje popis objektů figur. Obsahuje třídu *Pieces*, od níž všechny figury dědí, a tudíž získávají její parametry.
  - **Name** určuje jméno figury. Toto jméno se poté objeví v ComboBox elementu, když je tato figura během hry šogi sebrána.
  - **Value** je integer určující hodnotu figury pro účely evaluace minimaxového algoritmu.
  - **Moved** u figury krále určuje, zda se již pohnul. Pokud ano, nemůžeme s ním provést rošádu.
  - **isWhite** je boolean udávající barvu figury.
  - **GenerateMoves** je metoda, po jejímž zavolání se do seznamů ve třídě *Moves* vygenerují tahy dané figury. Každá figura v této metodě volá svůj seznam elementárních tahů, z nichž se pak utvoří celý její set pohybů.
  - **Number** je integer označující číselný kód figury, viz 2.5. Na tento parametr odkazují metody *GetNumber* a *SetNumber*.

  Dále je tu třída **DefinedPiece**, popisující uživatelem definovanou figuru. Tyto figury jsou uložené v listu *DefinedPieces*.
- **PiecesNumbers** obsahuje několik slovníků umožňující překlad z textových názvů figurek na číselnou reprezentaci a zpátky. Dále také obsahuje metodu *updatePiece*, přidávající do všech těchto slovníků novou, uživatelem definovanou figuru, a několik metod rozpoznávající z číselného kódu, co je daná figura zač.

1.3 Controller

Controller se stará o změny v modelu podle vstupů od uživatele získaného přes třídy View.

- **Generating** zajišťuje správné generování tahů na šachovnici. Hlavní metoda této třídy je *Generate*, která po zavolání na určité figuře pro ni vygeneruje všechny možné tahy. Navíc má možnost zkontrolovat, zda nějaký z těchto tahů nepovede k šachu ve hře šachy, a případně tyto tahy vyřadí. Stejně tak pohlídá, aby ve hře dámy byla sebrána oponentova figura pokud je to možné tím, že nepovolí žádné jiné tahy než ty, které končí na oponentově figuře.

  Třída obsahuje statický boolean *WhitePlays*, signalizující, která strana v momentální partii hraje.

- **MoveController** zajišťuje dělání tahů na šachovnici. Metoda *ApplyMove* podle svých vstupních parametrů provede tah a zkontroluje případné eventuality a speciální tahy. Mezi ně patří:

  - Sebrání oponentovy figury kamenem z dámy. Běžný tah funguje tak, že figura na cílovém políčku je sebrána, u této figury je však oponentův kámen přeskočen. Metoda tedy hlídá, zda podobný tah nenastal, a případně odejme sebraný kámen.
  - Stejně funguje i braní mimochodem v šachu, kdy se pěšec přesune na jiné pole, než ze kterého je nakonec vyhozena figura. Metoda kontroluje, zda momentální tah nebere jinou figuru mimochodem.
  - Pokud se figura díky svému tahu změní na jinou, je na šachovnici nahrazena. U hry umělé inteligence se tak děje u povýšení šogi figur a výměna pěšce a kamene za dámu, pokud dojdou na druhou stranu šachovnice. U hráče se automaticky nahrazuje pouze kámen za dámu.
  - Pokud je momentální tah rošáda, musíme kromě krále posunout i věž.

  Metoda *ReapplyMove* je používána v minimaxovém prohledávání herního stromu. Abychom nemuseli kopírovat celou šachovnici, tahy provádíme na hlavní herní desce, a posléze je pomocí této metody vracíme zpět.

  Dále třída obsahuje statické parametry sloužící k signalizaci povahy daného tahu a metody na získávání informací o specifikacích daného tahu.

- **Minimax** zajišťuje implementaci minimaxového algoritmu s alfa-beta ořezáváním.

  Metoda volaná pro získání dalšího tahu je *GetNextMove*, jež nejprve vygeneruje všechny validní tahy z momentálního stavu hry, včetně přidávání figur, pokud to daná hra povoluje, načež z každého rozvede strom hry požadované hloubky. V případě, že typ hry je šogi a jsou dostupné figury pro přidávání na hrací desku, se za validní tahy počítá i přidání figury a herní strom se rozvede i z těchto tahů.

  Strom vytváří metoda *OneStep*, v níž je implementováno i alfa-beta ořezávání. Vezme momentální šachovnici a pro stranu, která je na řadě, vygeneruje všechny možné tahy. Poté každý tah provede pomocí metody *ApplyMove* a rekurzivně se zavolá se změněným booleovským parametrem *isMaxing* a s integer parametrem *depth* sníženým o jedna. Poté použije metodu *ReapplyMove*, aby šachovnici vrátila do původního stavu. Díky tomu si ve svém běhu nemusí vytvářet nové šachovnice, stačí jí si jen pamatovat několik parametrů. Po dosažení *depth* rovné 0 se zavolá metoda *EvaluateChessboard*, jež vyhodnotí úspěšnost daného tahu a vrátí jeho ohodnocení, které se načež propaguje do kořene. Ohodnocení figur pro minimaxový výpočet je v následující tabulce.

Figura	Hodnota pro minimaxový algoritmus
Král	900
Královna	500
Věž	50
Kůň	30
Střelec	30
Pěšec	10
Kámen	10
Dáma	90
Shogi král	900
Shogi věž	50
Povýšená shogi věž	60
Shogi střelec	30
Povýšený shogi střelec	40
Zlatý generál	30
Stříbrný generál	30
Povýšený stříbrný generál	70
Shogi kůň	30
Povýšený shogi kůň	40
Kopiník	30
Povýšený kopiník	40
Shogi pěšák	10
Povýšený shogi pěšák	20

Je běžné, že se několik tahů ohodnotí stejně, pak se náhodně vybere jeden z nejvýše hodnocených tahů.

- **MonteCarlo** je implementace Monte Carlo Tree Search algoritmu pro hru dvou hráčů.

  Stejně jako ve třídě *Minimax*, i zde metoda *GetNextMove* po svém zavolání vrátí nejlepší možný tah dle Monte Carlo Tree Search algoritmu. Metoda vytvoří kořen stromu, na němž posléze zavolá metodu *MonteCarloRoot*, jež prochází čtyřmi fázemi tohoto algoritmu, dokud nevyprší čas. Poté metoda *BestChild* vybere nejvýhodnější tah a vrátí jeho polohu v listech třídy *Moves*.

1.4 GameTester

V solution je zařazena i konzolová aplikace *GameTester*, určená pro simulaci her, kdy proti sobě hrají dva zvolené algoritmy na zadané šachovnici.

Po spuštění aplikace se nejprve zvolí typ hry zadáním cesty k souboru hry. Poté se aplikace zeptá, jaký algoritmus hledání nejlepšího tahu má být na které straně použit, jeho parametry, počet opakování a zda se má průběh hry vizualizovat. Nakonec proběhne simulace, na jejímž konci jsou oznámeny výsledky všech her.

Co se implementace týče, GameTester se dělí na dvě třídy, *Program* a *Game*.

- **Program** načte vstup od uživatele a nastaví parametry hry. Poté spustí simulace.

- **Game** obsahuje metody potřebné k hraní hry v *GameTesteru*. Nejprve se zavolá *CreateChessBoard* na vytvoření logické reprezentace šachovnice. Poté se při hraní hry využívá metody *MakeMove* pro nalezení a vykonání dalšího tahu. K vizualizaci momentálního stavu hry slouží metoda *DrawBoard*, jež používá symboly ze slovníku *getSymbol*.

2 Další možná rozšíření

Aplikace je implementována tak, aby se do ní daly jednoduše přidat rozšíření. Her na šachovnici s odlišnými, ještě neimplementovanými prvky je celá řada a všechny je dát do jednoho programu je nemožné. Ve vytvářeném softwaru je tedy prostor na rozšíření a níže jsou popsány některé možnosti.

2.1 Přidání speciálního pohybu figury

Ač již je možné pomocí načítání hry ze souboru přidat vlastní figuru, stále je poněkud omezena co se pohybu týče. Pro přidání vlastního pohybu je potřeba upravit třídu *Moves*, konkrétně přidat metodu přijímající parametry (integer, integer, Pieces[,]) a vracející typ bool. Pro každý postup na jiné pole se musí vložit příslušné hodnoty do listů *start_x, start_y, final_x a final_y*. Pro zjištění, zda je tah z daného místa validní (tedy zda např. nepřesahuje hranice šachovnice) se volá metoda *ValidMove*, jež v případě možného tahu vrátí true. Tato metoda navíc pomocí booleovské proměnné *EnemyMet* hlídá, zda jsme již narazili na nepřítele, a pokud ano, změní její hodnotu na *true* a neoznačí další tahy jako validní, dokud tuto proměnou nenastavíme na *false*.

Pokud dále chceme definovaný pohyb používat u figurek definovaných načtením souboru, daný tah přidáme do slovníku *getMoveFromNumber* (nebo i *getMoveName* pro použití jmenného označení pohybu) ve třídě *PiecesNumbers*.

2.2 Změna figury šachového či šogi krále

Momentálně je konec her s typem *šachy* a *shogi* svázán se specifickými figurkami, ale není těžké změnit kód programu tak, aby se stejná pravidla vztahovala na figurky s jiným pohybem a obrázkem.

Docílíme toho přepsáním následujících funkcí:

- Pro změnu šachového krále je potřeba přepsat metody *IsKing, IsWhiteKing* a *IsBlackKing* v třídě *PiecesNumbers*, konkrétně upravíme hodnotu na základě které vracíme *true*.

- Stejný postup je i pro šogi krále. Pro jeho nahrazení jinou figurou v této třídě změníme metody *IsShogiKing, IsUpperShogiKing* a *IsBottomShogiKing*.

2.3 Úprava vzhledu šachovnice

V programu se vykresluje základní černobílá šachovnice. Pro úpravu jejích barev, velikostí polí atp. je potřeba změnit metodu *DrawChessboard* ve třídě *MainGameWindow*, je ale třeba dát pozor na velikost okna, jež se přizpůsobuje šířce a výšce šachovnice.

2.4 Možnost definice vlastní propagace figury

Momentálně se v programu dají během hry propagovat pouze klasické šogi figury. Pro přidání dalších figur je třeba upravit pole *canPropagate* ve třídě *PiecesNumbers*, a to přidáním čísla figury, jež se má propagovat a zajištěním, že figura o číslo vyšší je ta, na kterou se má propagovat.

Tato změna se dá udělat pouze v kódu, nebo i přidáním dalších parametrů implementovat do her načítaných ze souboru.

2.5 Seznam figur, jejich čísla a zkratky

Číslo	Zkratka	Název
0	CWK	Bílý král
1	CWQ	Bílá královna
2	CWR	Bílá věž
3	CWN	Bílý kůň
4	CWB	Bílý střelec
5	CWP	Bílý pěšec
6	DWP	Bílý kámen
7	DWQ	Bílá dáma
8	SWK	Spodní shogi král
9	SWR	Spodní shogi věž
10	SWR+	Povýšená spodní shogi věž
11	SWB	Spodní shogi střelec
12	SWB+	Povýšený spodní shogi střelec
13	SWG	Spodní zlatý generál
14	SWS	Spodní stříbrný generál
15	SWS+	Povýšený spodní stříbrný generál
16	SWN	Spodní shogi kůň
17	SWN+	Povýšený spodní shogi kůň
18	SWL	Spodní kopiník
19	SWL+	Povýšený spodní kopiník
20	SWP	Spodní shogi pěšák
21	SWP+	Povýšený spodní shogi pěšák
22	CBK	Černý král
23	CBQ	Černá královna
24	CBR	Černá věž
25	CBN	Černý kůň
26	CBB	Černý střelec
27	CBP	Černý pěšec
28	DBP	Černý kámen
29	DBQ	Černá dáma
30	SBK	Vrchní shogi král
31	SBR	Vrchní shogi věž
32	SBR+	Povýšená vrchní shogi věž
33	SBB	Vrchní shogi střelec
34	SBB+	Povýšený vrchní shogi střelec
35	SBG	Vrchní zlatý generál
36	SBS	Vrchní stříbrný generál
37	SBS+	Povýšený vrchní stříbrný generál
38	SBN	Vrchní shogi kůň
39	SBN+	Povýšený vrchní shogi kůň
40	SBL	Vrchní kopiník
41	SBL+	Povýšený vrchní kopiník
42	SBP	Vrchní shogi pěšák
43	SBP+	Povýšený vrchní shogi pěšák