Programátorská dokumentace programu Damebot

Václav Stupka

2. února 2022

Obsah

1 Úvod

Damebot je program, který umožňuje hrát dámu proti počítači. Tento projekt vznikl jako zápočtový program pro předmět Programování pro pokročilé v zimním semestru prvního ročníku na Matematicko-fyzikální fakultě Univerzity Karlovy (obor Informatika). Jedná se o jednoduchou TypeScriptovou aplikaci spouštěnou ve webovém prohlížeči.

Cílem projektu bylo vytvořit program, který si bude moci uživatel jednoduše stáhnout a následně ho používat bez nutnosti složité a dlouho trvající instalace. Program by měl být také jednoduchý na správu a otevřený budoucímu rozšiřování. Samozřejmostí je také jednoduché ovládání a (přiměřeně) pěkné uživatelské rozhraní.

Tento dokument představuje programátorskou dokumentaci celého projektu. Dočtete se zde, které technologie jsem použil pro správu projektu, jak probíhá sestavování programu a také popis nejdůležitějších částí zdrojového kódu.

2 Použité technologie a knihovny

Pro sestavování a kompilaci programu jsem použil nástroje *TypeScript* a *Webpack* (včetně různých pluginů a loaderů). Jako verzovací systém jsem použil *Git* spolu se vzdáleným repozitářem *GitHubem* a pro získání potřebných programů a knihoven jsem použil balíčkovací systém *npm*.

Program jsem psal v programovacím jazyce TypeScript, za použití standardizovaných funkcí (poskytovaných webovým prohlížečem) a knihovny *JQuery*, která mi usnadnila práci se strukturou HTML. Uživatelské rozhraní je vytvořeno pomocí čistého HTML a CSS.

Jako vývojové prostředí pro psaní kódu jsem použil *Visual Studio*, grafiku (obrázky hracích kamenů) jsem vytvořil pomocí programu *Inkscape*, tuto dokumentaci píšu v *TexStudiu* a program jsem testoval v prohlížeči *Firefox*.

3 Získání zdrojového kódu

Zdrojový kód lze najít na adrese `https://github.com/basvas-jkj/dame_bot`. Vytvořit lokální repozitář s kopií zdrojového kódu lze pomocí Gitu příkazem `$git clone https://github.com/basvas-jkj/dame_bot.git`. Stažení potřebných modulů zajistí příkaz `$npm install`. Po dokončení těchto dvou příkazů budete mít vše, co je potřeba k vývoji a kompilaci programu Damebot.

4 Kompilace a sestavení projektu

Sestavení projektu lze iniciovat příkazem `$npm run debug`, případně `$npm run release`. Oba příkazy spouští program *Webpack*, který provede následující kroky:

- Překlad TypeScriptu do JavaScriptu (webpackový modul *ts-loader* a program *tsc*).

- Spojení přeložených souborů (a souboru *starting_position.json*) do souboru *dist/script.js* (samotný *Webpack*).
- Zkopírování ostatních souborů do složky *dist* (*Webpack* a jeho moduly *HtmlWebpackPlugin*, *css-loader* a *MiniCssExtractPlugin*).
- Připojení odkazů na soubory .css a .js do souboru *index.html* (*HtmlWebpackPlugin*).
- Odstranění přebytečných souborů ze složky *dist* (*CleanWebpackPlugin*).
- V případě spuštění příkazem `release` se provede taky minifikace, tedy odstranění přebytečných mezer a komentářů (*Webpack*, *CssMinimizerPlugin*)

Tento systém velice usnadňuje testování i finální sestavování release verze programu. Není nutné nic provádět ručně, stačí zadat jeden příkaz do příkazové řádky a všechny potřebné programy se spustí automaticky.

5 Stručný přehled souborů

index.html Kostra uživatelského rozhraní. (Samotná šachovnice je generována dynamicky.)

style.css Definice všech použitých stylů.

svg/ Složka obsahující obrázky hracích kamenů ve formátu SVG.

ts/main.ts Vstupní bod programu (při kompilaci i spuštění). Sám o sobě nic nedělá, pouze spouští funkce z jiných souborů.

ts/board.ts Zajišťuje přípravu šachovnice, interakci s uživatelem, poskytuje většinu funkcí pro práci se šachovnicí. Kontroluje také, zda je tah zahraný uživatelem platný (využívá k tomu funkce třídy `PIECE`).

ts/field.ts Obsahuje třídu `FIELD`, která reprezentuje jedno pole šachovnice.

ts/piece.ts Obsahuje třídu `PIECE`, která reprezentuje jeden dámový kámen. Třída také zajišťuje kontrolu, zda je tah zahraný tímto kamenem možný a zjišťuje, zda daný kámen může skákat

ts/move.ts Obsahuje třídu `MOVE`, která ukládá potřebné informace o nalezeném tahu.

ts/player.ts Obsahuje třídu `PLAYER`, která mj. zajišťuje střídání obou hráčů a vybírá tah, který bot zahraje.

ts/next.ts Hledá všechny tahy, které bot může vykonat a počítá jejich statické ohodnocení.

ts/starting_position.json Ukládá výchozí pozici ve formátu JSON.

6 Popis vybraných částí zdrojového kódu

V této části se pokusím dopodrobna popsat vybrané části zdrojového kódu. Omezuji se převážně na ty funkce, které obsahují složitější algoritmus.

V celém programu čísluji sloupce od 0 do 7 zleva doprava a řádky od 0 do 7 shora dolů (ačkoli běžně se číslují v obráceném pořadí). Ve zbytku dokumentace budu tento způsob číslování také používat.

6.1 průběh hráčova tahu

O řízení tahu hráče se starají funkce `black_square_clicked()`, `end_move()`, `abort_move()` a `check_move()` ze souboru *ts/board.ts*. Funkce `end_move()` vykoná kroky potřebné k ukončení tahu (odstraní vyhozené kameny). Funkce `abort()` vykoná kroky potřebné k přerušení tahu (vrácení kamene do původní pozice). Obě funkce nastaví globální proměnné do výchozího stavu.

Funkce `check_move()` vrací hodnotu `MOVE_TYPE.possible` v případě, že hráč chce zahrát možný tah (pouze v případě, že hráč nemůže vzít žádný soupeřův kámen). V případě, že hráč může vzít některý soupeřův kámen, nebo se již nachází uprostřed braní, a chce zahrát skok, který odpovídá pravidlům, vrátí `MOVE_TYPE.capturing`. V případě, že chce hráč zahrát libovolný nemožný tah (včetně opomenutého braní), vrátí hodnotu `MOVE_TYPE.impossible`.

Funkce `black_square_clicked()` je vyvolána, kdykoli hráč klikne na černé pole. V případě, že hráč klikne na svůj kámen ve chvíli, kdy je zrovna na tahu, označí ho. Kromě toho se nastaví proměnná `moving` na `true` a `previous_field` na právě kliknuté pole. Pokud již předtím zahájil tah jiným kamenem, je tento tah přerušen funkcí `abort_move()`.

Pokud hráč označil některý svůj kámen (`moving == true`) a opětovně klikne na prázdné černé pole, spustí se funkce `check_move()`, která tah zkontroluje. Pokud vrátila `MOVE_TYPE.impossible`, nic se nestane a hráč může zahrát jiný tah (pořád stejným kamenem). Pokud vrátila `MOVE_TYPE.possible`, vykoná se příslušný tah. Pokud může být posunutý kámen proměněn v dámu, stane se tak. Následně je tah ukončen funkcí `end_move()`.

Pokud vrátila `MOVE_TYPE.capturing`, provede se příslušný skok. Dále se zkontroluje, zda může příslušný kámen pokračovat dalším skokem. Pokud ano, nastaví se proměnná `actual_field` na kliknuté pole, proměnná `original_field` na původní pole, na kterém stál kámen před začátkem tahu (podle ní se pozná, kam má funkce `abort_move()` kámen vrátit, případně zda probíhá vícenásobný skok – pokud žádný neprobíhá, je nastavena na `null`). V případě, že další skok není možný, je tah ukončen (`end_move()`). Ještě předtím ale proběhne případná proměna obyčejného kamene v dámu.

Funkce `abort_move()` je dále volána, pokud hráč klikne na bílé pole nebo pokud myš opustí plochu šachovnice.

6.2 třída `PLAYER`

V této třídě stojí za zmínku statická funkce `switch_players()`. Ta prohodí hráče, který je zrovna na tahu (to je potřeba kvůli několika dalším statickým funkcím, které vždy pracují s tím hráčem, který je právě na tahu). Zde ji ale zmiňuji kvůli bloku `else` na konci funkce. Tento blok obsahuje upravený kód funkce `choose_move()`. Změna spočívá v tom, že zde nejde o vybrání nejlépe ohodnoceného tahu, ale zjištění, zda vůbec nějaký tah existuje. Ve chvíli, kdy je nějaký možný tah nalezen, funkce se ukončí. Pokud žádný nalezen není, program oznámí, že vyhrál protivník (v mojí verzi vždy Damebot).

Drobný komentář si dovolím uvést i pro funkci `play`. Tato funkce si nejprve nechá vrátit příští tah (viz `choose_move()`), poté počká 1,5 sekundy, označí vybraný tah na šachovnici, počká další 3 vteřiny, a teprve poté zvolený tah vykoná. Čekání jsem zde použil kvůli tomu, že pro většinu lidí není okamžitá odpověď programu příjemná. Někteří lidé by navíc ani nemuseli postřehnout, jaký tah Damebot zahrál. Zvolená doba čekání je poměrně nízká, při běžném hraní nebude omezovat (hráč zpravidla na vymyšlení vlastního tahu potřebuje více času).

Nejdůležitější funkcí, kterou v této dokumentaci zmíním, je funkce `choose_move()`. Tato funkce projde všechny hráčovy kameny, ke každé vrátí příslušný generátor ze souboru *ts/next.ts* (podle toho, jestli jde o obyčejný kámen či dámu a zda hráč může brát soupeřovy kameny či nikoli). Postupně projde všechny vygenerované tahy a z nich vybere ten tah, který je nejlépe ohodnocený (v případě braní tah s největší váhou vyhodnocených kamenů – obyčejný kámen má váhu 1, dáma váhu 3). Všechny nejlépe ohodnocené tahy jsou uloženy v poli `chosen_moves`. Z těchto tahů náhodně jeden vybere a vrátí. Pokud žádný takový tah není nalezen, program oznámí výhru protivníka (tedy uživatele) a funkce vrátí hodnotu `null`.

6.3 třída `PIECE`

`is_possible_move()`

Kontroluje, zda se tento kámen může posunout na pole určené parametry `next_row` a `next_column`. Vrátí `true`, pokud:

- Jde o bílý kámen, `next_row` je o jedna menší než stávající číslo řádku a `next_column` je o jedna větší či menší než stávající číslo sloupce.
- Jde o černý kámen, `next_row` je o jedna větší než stávající číslo řádku a `next_column` je o jedna větší či menší než stávající číslo sloupce.
- Jde o dámu libovolné barvy a rozdíl souřadnic nového pole je roven rozdílu stávajících souřadnic nebo součet souřadnic nového pole je roven součtu stávajících souřadnic a zároveň všechna pole mezi dámou a cílovým polem jsou volná.
- Není nutné kontrolovat prázdnost cílového pole, protože na obsazené pole není možné v tomto kontextu kliknout.

V ostatních případech vrací `false`.

`is_possible_capture()`

Kontroluje, zda se tento kámen může posunout na pole určené parametry `next_row` a `next_column`. Vrátí objekt typu `PIECE` reprezentující vyhozený kámen, pokud:

- Tento kámen je bílý, `next_row` je o dva menší než stávající číslo řádku a `next_column` je o dva větší či menší než stávající číslo sloupce.
- Jde o černý kámen, `next_row` je o dva větší než stávající číslo řádku a `next_column` je o dva větší či menší než stávající číslo sloupce.
- Jde o dámu libovolné barvy a rozdíl souřadnic nového pole je roven rozdílu stávajících souřadnic nebo součet souřadnic nového pole je roven součtu stávajících souřadnic a zároveň právě jedno pole je obsazené protivníkovým kamenem. Zda již byl nějaký kámen přeskočen, ukládá proměnná `has_captured`. Přeskočený kámen je uložen v proměnné `captured_piece`.

V ostatních případech vrací `null`, což znamená, že příslušný skok není možný. Stejný význam má vrácená hodnota `null` i v případě, že v jedné ze tří předchozích situacích přeskočené pole, na kterém se očekával protivníkův kámen bylo prázdné (funkce `get_piece()` vrátila `null`).

`is_threatened()`

Kontroluje, zda je tento kámen přímo ohrožen (pokud je kámen ohrožen nepřímo vícenásobným skokem, tato funkce to nezjistí). Pokud tento kámen stojí na krajním řádku či sloupci, vrátí `false`. Jinak projde postupně všechny čtyři úhlopříčné směry a v každém bude postupně procházet všechna pole, dokud nenarazí na první kámen. Jde-li o dámu opačné barvy a v opačném směru je první pole za tímto kamenem volné, vrátí `true`. Dále vrátí `true` v případě, že bezprostředně před tímto kamenem (ve směru odpovídající pohybu obyčejných kamenů tohoto hráče) úhlopříčně (vlevo či vpravo) nachází protivníkův kámen a v opačném směru je první pole za tímto kamenem volné. V ostatních případech vrátí `false`.

`can_man_capture()`

Vrátí `true` v případě, že bezprostředně před tímto kamenem (ve směru odpovídajícím pohybu obyčejných kamenů tohoto hráče) úhlopříčně (vlevo či vpravo) nachází protivníkův kámen a pole za tímto kamenem ve stejném směru je prázdné. V ostatních případech vrací `false`.

`can_king_capture()`

Postupně projde všechny čtyři úhlopříčné směry od této dámy. Pokud v libovolném směru najde kámen opačné barvy, za kterým se nachází volné pole, a který doposud nebyl vyhozen, vrací `true`. Pokud ve všech čtyřech směrech narazí na kámen stejné barvy, kámen opačné barvy, který již byl vyhozen (takový kámen se nachází v poli předaném parametrem `captured_pieces`), kámen opačné barvy, za kterým se nenachází volné pole, nebo na okraj šachovnice, vrátí `false`.

6.4 hledání příštího tahu

Některé funkce v souboru *ts/next.ts* jsou tzv. generátory. To znamená, že příslušná funkce hodnotu nevrací pomocí příkazu `return`, ale příkazem `yield`, přičemž tento příkaz neukončí běh funkce, pouze ho přeruší, ale při dalším pokusu o čtení hodnoty pokračuje. Definitivně je běh generátoru ukončen, pokud je dosaženo jeho konce.

`will_be_threatened()`

Chová se podobně jako stejnojmenná funkce ve třídě `PIECE`, ale pole, na kterém se původně nacházela dáma zadaná parametrem `piece`, považuje za

prázdné (a může být přes něj ohrožena protivníkovou dámou).

`can_king_capture()`

Chová se podobně jako stejnojmenná funkce ve třídě `PIECE`, ale prochází pouze jeden směr (zadaný parametry `row_direction` a `column_direction`) a pole, na kterém se původně dáma zadaná parametrem `piece`, považuje za prázdné.

`man_move()`

Pokud se kámen předaný parametrem `piece` může posunout doleva dopředu o jedno pole (příslušné pole je prázdné), ohodnotí tento tah a vrátí ho (příkaz `yield`).

Pokud se kámen předaný parametrem `piece` může posunout doprava dopředu o jedno pole (příslušné pole je prázdné), ohodnotí tento tah a vrátí ho (příkaz `yield`).

`king_move()`

Projde postupně všechny čtyři směry od dámy předané parametrem `piece`. Pokud je aktuálně testované pole prázdné, vygeneruje tah na toto pole, ohodnotí tah, vrátí ho (příkaz `yield`) a posune aktuální pole o jedna tím směrem, který je v této chvíli procházen.

`man_capture()`

V případě, že je pole ležící před kamenem předaným parametrem `piece` úhlopříčně vlevo obsazené protivníkovým kamenem a další pole v témže směru je prázdné, uloží toto pole do zásobníku `fields` a rekurzivně pokračuje dalším skokem. Příkazem `yield*` vrátí všechny tahy vrácené tímto rekurzivním voláním. Poté zopakuje stejné kroky, ale pro úhlopříčný směr dopředu doprava.

V případě, že ani jedním z těchto směrů nelze vykonat skok a zároveň zásobník `fields` má nenulovou délku, ohodnotí tah, který končí na aktuálním poli, a vrátí ho příkazem `yield`.

`king_caputure_in_direction()`

Pokusí se najít skok dámou předanou parametrem `piece`, který vede směrem určeným parametry `row_direction` a `column_direction`. Prochází postupně všechna pole od dámy v tomto směru. Pokud narazí na prázdné

pole, pokračuje dál. Pokud nalezne pole, na kterém se nachází kámen stejné barvy nebo kámen opačné barvy, za kterým se nenachází volné pole (v témže směru), tak ukončí hledání. Pokud nalezne kámen opačné barvy, za kterým se nachází volné pole, nastaví proměnnou `has_captured` na `true`. Hledání následně pokračuje.

Pokud narazí na prázdné pole a `has_captured == true`, pak zjistí, jestli je možné pokračovat dalším skokem v kolmém směru na původní směr pomocí funkce `can_king_capture()` (definované v tomto souboru). Pokud to je možné, pokračuje rekurzivně skokem v příslušném směru (v jednom nebo v obou). Výsledky tohoto rekurzivního volání jsou vrácen příkazem `yield*`. V opačném případě je tah, který končí na aktuálním poli, vložen do fronty `move_queue`.

Pokud platí `has_captured == true`, aktuální pole je obsazené kamenem opačné barvy a následující pole je volné, pak je celá fronta `move_queue` vyprázdněna (vzhledem k povinnosti brát na polích ve frontě tah skončit nemůže). Hledání následně pokračuje rekurzivně ve stejném směru z posledního volného pole. Výsledky tohoto rekurzivního volání jsou vráceny příkazem `yield*`.

Pokud nalezne pole, na kterém se nachází pole stejné barvy nebo kámen opačné barvy, za kterým se nenachází volné pole (v témže směru) a `has_captured == true`, tak se hledání ukončí.

Poté, co skončí hledání, jsou všechny tahy uložené v `move_queue` staticky ohodnoceny a vráceny pomocí příkazu `yield`.

6.5 výchozí pozice

Výchozí pozice hry je uložena v souboru *ts/starting_position.json* jako JavaScriptový objekt se třemi atributy:

player_on_move Označuje, který hráč je ve výchozí pozici na tahu (hodnoty „white“ a „black“).

white Obsahuje pole kamenů bílého hráče.

black Obsahuje pole kamenů černého hráče.

Jeden kámen je reprezentován objektem s atributy:

row Číslo řádku, na kterém se kámen ve výchozí pozici nachází (hodnoty 0–7).

column Číslo sloupce, na kterém se kámen ve výchozí pozici nachází (hodnoty 0–7).

type Ukládá barvu, a zda jde o obyčejný kámen či dámu (hodnoty typu *PIECE_TYPE*).

Aplikace každé změny tohoto souboru vyžaduje kompletní sestavení programu, nestačí pouze restartovat program v prohlížeči.

7 Možná budoucí rozšíření

Damebot je otevřený budoucímu rozšiřování. Snažil jsem se ho navrhnout tak, aby případné změny a vylepšování nevyžadovaly začít programovat úplně od začátku. Nejjednodušší úpravou by byla možnost vybrat uživateli, jestli chce hrát za bílé či černé kameny, případně zda budou hrát proti sobě dva hráči nebo dva boti. Tato možnost je součástí zdrojového kódu, ale nemá podporu v uživatelském rozhraní.

Další variantou by bylo vylepšit algoritmus pro hledání nejlepšího tahu. Mnou vytvořený algoritmus je velmi jednoduchý a často přehlíží výhodné tahy, místo nich pak zvolí tah, který sice nemusí vést k prohře, ale ani k výhře. Zajímavou alternativou by také byla možnost přidat více různých algoritmů (náhodné tahy, minimax, ...), hráč by potom měl možnost lépe porovnat své schopnosti s kamarády, případně porovnávat různé algoritmy mezi sebou.

Nabízí se také možnost vylepšit uživatelské rozhraní. To stávající je mnohem hezčí, než jsem původně očekával, ale přesto by bylo pěkné doplnit např. animaci přesunu kamenů nebo možnost vybrat si jeden z více různých motivů vzhledu.

Posledním nápadem, který zde uvedu, by bylo přidat možnost vybrat si pravidla, kterými se bude Damebot řídit. Dáma má nejvíce národních variant ze všech deskových her, ale Damebot podporuje pouze českou dámu. Jistě by tedy bylo zajímavé přidat třeba polskou nebo anglickou dámu. Tato změna by ale vyžadovala komplexní změnu zdrojového kódu, protože jsem s ní při návrhu nepočítal.