GYMNASIUM JANA KEPLERA

Parléřova 2/118, 169 00 Praha 6

Digitální Šógi

Maturitní práce

Autor: Daniel Žampach

Třída: R8.A

Školní rok: 2020/2021

Předmět: Informatika

Vedoucí práce: Šimon Schierreich

Praha, 2021

GYMNASIUM JANA KEPLERA
Kabinet informatiky

ZADÁNÍ MATURITNÍ PRÁCE

Student:	**Daniel Žampach**
Třída:	**R8.A**
Školní rok:	**2020/2021**
Platnost zadání:	**30. 9. 2021**
Vedoucí práce:	**Šimon Schierreich**
Název práce:	**Digitání šógi**

Pokyny pro vypracování:

Cílem této práce je vytvořit digitální verzi tradiční japonské deskové hry šógi. Tento program by měl umožnit dvěma hráčům zahrát si partii na jednom počítači. Součástí budou grafické rozhraní, souhrn pravidel a jejich vysvětlení, časovač a možnost upravení partií.

Doporučená literatura:

[1] IIDA, Hiroyuki, Makoto SAKUTA a Jeff ROLLASON. Computer shogi. Artificial Intelligence. 2002, 134(1-2), 121-144. ISSN 0004-3702. Dostupné z: https://doi.org/10.1016/S0004-3702(01)00157-6.

[2] MARTIN, Robert C. Design Principles and Design Patterns. www.objectmentor.com, 2000. Dostupné z: https://fi.ort.edu.uy/innovaportal/file/2032/1/design_principles.pdf.

[3] FOWLER, Martin. Patterns of enterprise application architecture. Boston: Addison-Wesley Professional, 2003. ISBN 978-0321127426.

URL repozitáře:

https://github.com/danielz9999/Shogi

vedoucí práce *student*

V Praze dne 29. 10. 2020

Prohlášení

Prohlašuji, že jsem svou práci vypracoval samostatně a použil jsem pouze prameny a literaturu uvedené v seznamu bibliografických záznamů. Nemám žádné námitky proti zpřístupňování této práce v souladu se zákonem č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších předpisů.

V Praze dne 7. dubna 2021 Daniel Žampach

Poděkování

Chtěl bych poděkovat svému vedoucímu práce Šimonu Schierreichovi, svým spolužákům a rodině.

Abstrakt

V této práci se zabývám vytvořením digitální verze japonské stolní hry šógi v programovacím jazyce java. Chtěl jsem aby hra byla pro dva hráče a hratelná na lokálně na počítačích. Při práci jsem vytvořil vlastní grafiku herního plánu a figur a implementoval pravidla hry šógi.

Klíčová slova

Šógi, java, interakce

Abstract

In this project I dealt with creating a digital version of the japanese board game Shogi in the progamming language Java. I wanted the game to be for the players and playable locally on PCs. During the project I created my own graphical interface of the board and pieces and I implemented the rules of the game Shogi.

Keywords

Shogi, java, interaction

Obsah

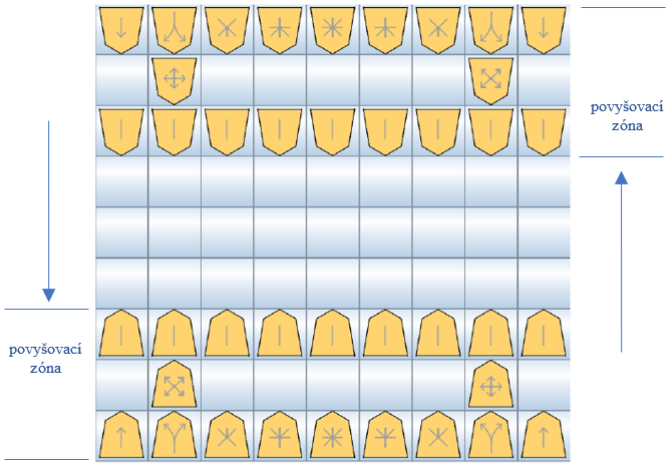
1. Teoretická část

1.1 Úvod a cíl

Jako někdo, kdo se zajímá o šachy, mi hra šógi vždy přišla fascinující. Když jsem se však rozhlížel po online verzích, které by šly stáhnout a hrát lokálně, nebyl jsem s žádnou úplně spokojen. A tak jsem se rozhodl, že si jako maturitní projekt vytvořím svou vlastní, lokálně hratelnou verzi.

Šógi je hra podobná šachům, z toho také pochází její přezdívka "japonské šachy". Zároveň se v některých aspektech liší a proto by bylo dobré si krátce projít a vysvětlit její pravidla.

Šógi je hra pro dva hráče, která se hraje na šachovnici, o rozměrech devět na devět políček. Hráči se střídají po jednom tahu. Ve svém tahu může hráč buď pohnout svou figurkou, nebo vrátit na plán, jako svou vlastní, figurku, kterou sebral soupeři. Cílem hry je sebrat soupeři krále. Základní postavení figurek na začátku hry je toto



Hračovy figruky jsou na začátku hry uspořádány ve třech řadách a posledních třech řadách na opačné straně hracího plánu tvoří tzv. povyšovací zónu.

Když jdeme ode středu směrem k dolů, nejprve narazíme na řadu pěšců.

Pěšec (angl. pawn), se dékáže pohybovat pouze o jedno políčko dopředu. Pokud dosáhne povyšovací zóny, může se povýšit na zlatého generála.

O řadu níže máme dvě figurky, vlevo střelce, a vpravo věž.

Střelec (angl. bishop) se dokáže pohybovat neomezeně po diagonálách. Nemůže však skákat přes

figurky. Pokud dosáhne povyšovací zóny, může se změnit na povýšeného střelce

Povýšený střelec se pak může posunout i na všechna políčka kolem sebe.

Věž (angl. rook), se dokáže pohybovat neomezeně vertikálně a horizontálně, ale také nemůže přeskakovat přes figurky. Když dosáhne povyšovací zóny, může být vylepšen na povýšenou věž.

Ta se může pohybovat i na čtyři políčka diagonálně kolem ní.

Poslední řada je pak na hracím plánu osově symetrická, zleva do prava obsahuje:

Kopiník (angl. lancer) se může pohybovat nekonečně dopředu, nemůže však skákat přes figurky. Pokud dosáhne povyšovací zóny, může se povýšit na zlatého generála.

Jezdec (angl. Knight) se může pohybovat o dvě dopředu a o jednu do buď leva nebo prava. Jezdec je jediná figurka, která může skákat přes ostatní figurky. Pokud dosáhne povyšovací zóny, může se povýšit na zlatého generála.

Stříbrný generál (angl. Silver general, zkráceně pak pouze silver), se může pohybovat o jednu po diagonálách, a nebo o jednu dopředu. Pokud dosáhne povyšovací zóny, může se povýšit na zlatého generála.

Zlatý generál (angl. Golden general, zkráceně pak gold) se může pohybovat na všchena políčka kolem sebe, krom těch dvou diagonálně za ním. Pokud dosáhne povyšovací zóny, nemůže se povýšit na žádnou figurku.

Král (angl. King), je nejdůležitější figurka ve hře. Když hráč ztratí krále hra končí. Král se může pohybovat na všech osm sousedících políček. [3]

Braní figurek funguje identické jako v šachách. Když figurka dojde na políčko obsazené nepřátelskou figurkou, ona nepřátelská figurka je sebrána a odstraněna z herního plánu. Jediný rozdíl je, že v Šógi je sebraná figurka ještě přidána do seznamu figurek sebraných danou stranou. To nás přivádi k:

Vracení figurek, hráči můžou místo pohybu svůj tah strávit vrácením jedné figurky na hrací plán. Tato (původně nepřátelská) figurka se teď chová identicky jako ostatní spojenecké figurky. Toto vracení má určitá pravidla:

- Figurky se vrací nepovýšené

- Figurky musí být položeny na políčko, ze kterého se mohou pohnout. Kopiníci a pěšáci nemohou být vloženi do poslední řady, jezdec nemůže být vložen do posledních dvou (je zde myšleno omezení hrací deskou, ne omezení sousedícími figurkami).

- Pěšec nemůže být vložen do sloupce, který již obsahuje přátelského pěšce

- Pěšec vložením nemůže dát šachmat

Pokud jsou tato pravidla splněna, může být jakákoliv sebraná figurka vrácena na jakékoliv pole.

Již zmíněné **povyšování figurek** je další klíčovou složkou Šógi. Pokud figurka po pohybu skončí v povyšovací zóně, tedy posledních třech řadách na druhém konci hrací desky, má možnost tuto figurku povýšit. Povýšené figurky se nemohou vrátit do nepovýšeného stádia, a pokud nejsou sebrány zůstávají povýšeny do konce hry.

Pokud se stejné uskupení figurek na šachovnici opakuje čtyřikrát, hra končí remízou.

Pokud je král uveden v šach, hráč není povinnen to svému soupeři zmínit. Hráč, jehož král je sebrán, hru okamžitě prohrává.

Cílem mé práce bylo vytvořit počítačový program, ve kterém si mohou dva hráči proti sobě zahrát hru šógi podle výše popsaných pravidel

2. Implementace

2.1 Kostra

Jako programovací jazyk, ve kterém bych měl zpracovat svou práci, mi byla doporučena **Java**. Její objektově orientované programování by mělo snadno modelování reality figurek, políček a hrací plochy. Java mi také vyhovovala, protože její zápis je velmi podrobný a kód v javě se velmi dobře čte. Proto jsem zvolil Javu jako programovací jazyk pro svou práci.

Kostrou této práce je hrací plán, **šachovnice 9x9.** Hrací plán se zkládá z 81 políček, kde každé políčko má svoje vlastní tlačítko, které se na něm zobrazuje. Každé políčko si drží svou figurku (nebo popřípadě drží absenci figurky), ona figurka si pamatuje na jakých souřadnicích se nachází. A hrací plocha pak drží všechna políčka.

Toto řešení mi přišlo jako nejrozumnější, protože bere jedno políčko jako základní stavební jednotku, programovací objekt celé hry. Je totiž jasné, že je potřeba rozlišovat jednotlivá políčka, protože jinak nejdou logicky implementovat pravidla hry. Zároveň je potřeba aby každé políčko reagovalo jinak, když se na něj klikne, reagovalo na specifické situace. Každá situace mi přiřazený vlastní typ tlačítka.

Dalším důležitým stavebním kamenem je abstraktní **Piece** třída. Z ní pak dědí vlastnosti všechny třídy specifických figurek (pěšec, věž atd.) a zároveň je díky polymorfismu abstraktní třída využívána ve všech funkcích a metodách do kterých dosazuji různé třídy figurek. Bez tohoto polymorfismu by nebylo možné projekt v rozumném měřítku vypracovat. Polymorfismus v podstatě osmkrát zmenšuje počet potřebných použitých v programu metod. Důležitá u této třídy je také funkce getMoves, kterou si přepisuje každý její potomek. Tato funkce vypíše pozice, na které se daná figurka z daného políčka může podle implementovaných pravidel přesunout.

Klíčovým prvkem celého programu je pak **mouse listener** tlačítek, který určuje co se stane, když se na tlačítko klikne. Kliknutí levým tlačítkem figurku označí, a když je další kliknutí na jedno z možných políček získaných z getMoves dané figurky, pak se tam figurka přesune. Tato implementace mi přišla jako nejjednodušší. Přesun figurky na dvě kliknutí je snažší na provedení než, dejme tomu, táhání figurek po šachovnici. Takto je vždy jasné, co je začáteční a co je konečné políčko. Zároveň je to intuitivnější než na klávesnici zapisovat v textové podobě, kam se má figurka pohnout. Pravé tlačítko pak slouží k vracení figurek. Zde jsem využil JComboBox, protože opět se mi to jevilo jako nejjednodušší způsob, jak umožnit hráči výběr figurky.

Během vývoje jsem uvažoval nad použitím grafického rozhraní někoho jiného. Nakonec jsem se však rozhodl vytvořit si vlastní. Grafické .png obrázky jsem stvořil v programu paint.net a poté jsem je aplikoval na jednotlivá tlačítka na šachovnici.

2.2 Problémy a jejich řešení

Vývoj této práce se nedá popsat jako lineární. Většinou se postup zasekl na jednom velkém problému, který zastavil celý proces. V této sekci bych chtěl tyto zásadní problémy popsat.

Již zmíněný **mouse listener** by prvním velkým problémem. Kámen úrazu spočívá v tom, že po tlačítku reagovalo nějak když je nějaká figurka označena, a jinak když žádná označená není. Řešení se zdálo být jednoduché - nastavit proměnnou, která si pamatuje, zda je figurka označena. Bohužel se mi neustále nedařilo implementovat tuto proměnnou správně, pokaždé vracela špatnou hodnotu. Problém spočíval v tom, že jsem onu proměnnou deklaroval přímo ve listeneru. Takže konečným (a poměrně elegantním) řešením bylo, deklarovat si pole souřadnic moveList, který si šachovnice vždy drží. Když se pak klikne na figurku, která se může na nějaké jiné políčko pohnout, dosadí se souřadnice všech takovýchto políček do pole moveList. Pak podle toho zda je moveList prázdný nebo ne, jde snadno odvodit, zda je figurka označena.

Dalším problémem byla moje nezkušenost. Na začátku projektu jsem neměl příliš dobré chápání teorie objektově orientovaného programování. Pojmům jako **Objektová hierarchie a zapouzdření** jsem proto nevěnoval velkou pozornost. To však mělo své následky. Během své druhé konzultace jsem si nevěděl rady, jak přistoupit k určitým proměnným a zároveň jsem některé měnil z míst, kde se měnit neměly. Na konzultaci jsem byl informován, že tyto problémy jsou více méně následky mé předchozí nedbalosti. Na doporučení jsem tedy založil projekt znovu a přepsal jsem drtivou většinu původního kódu. Nová objektová hierarchie a zapouzdření velké části atributů vedly k tomu, že bylo najednou jasnější, kde co je a co může být kdy zpřístupněno. Tento "Restart"je také důvod proč se jméno Shogi2 objevuje v některých souborech - je to název onoho druhého projektu.

Posledním problémem, který jsem řešil, je**Časovač**. Časovač je funkce, která měla být součástí již od začátku. I tak to byla jedna z posledních funkcí přidaných do hry. Byly zde dva hlavní problémy. Jak měřit skutečný čas a jak umožnit více procesům paralelní průběh. První problém jsem vyřešil pomocí metody System.currentTimeMillis(), která vypíše momentální čas počítače v milisekundách.[4] Když si tuto hodnotu uložím a později ji odečtu od nové instance sebe samé, dostanu čas, který hráčům uplynul. Druhý problém jsem pak vyřešil pomocí zavedení nového vlákna specificky pro časovač.[2]

Některé věci se mi však implementovat nepovedlo. Jednou z nich je pravidlo, že pěšec nemůže být vložen tak aby dal šach mat. Problém zde byl, že funkce, která pěšce vrací, musí nejenom zjistit vzájemnou polohu pěšce a nepřátelského krále, ale také jaké nepřátelské figurky krále obklopují, ale především musela tyto věci kontrolovat proti všem možným pohybům všech přátelských figurek. Nakonec se mi toto pravidlo povedlo implementovat částečně. Momentálně je pravidlo implementováno tak, že nelze položit pěšáka před nepřátelského krále.

3. Technická dokumentace

Na spuštění programu je potřeba java. Verze 1.8 by měla stačit, ale samozřejmě čím novější, tím lepší.

Nejjednodušší metoda spuštění je, že se složka "images"a jar soubor "Shogi2.JAR"přemístí do samostatné složky, a poté se spustí soubor Shogi2.JAR. Ideálně by to mělo jít přes dvojité kliknutí, ale občas je nutné otevřít ho skrz příkazovou řádku. Musí se samozřejmě navigovat do složky, kde jsou soubory uloženy, a pak provést java -jar Shogi2.JAR. Jde také samozřejmě zkompilovat .java třídy, ať už pomocí příkazu javac v příkazové lince či skrz nějaké java prostředí, a poté spustit nově vzniklou Graphics.class skrz příkazovou řádku pomocí: java Graphics. Všechny soubory a složka by měly zůstat celou dobu v jedné složce.

Po spuštění se objeví menu, ve kterém jde buď hru zapnout nebo program vypnout. Po spuštění vyskočí několik Y/N otázek týkajících se nastavení hry, případně bude potřeba zadat čas do časovače. Poté se spustí dvě okna. Šachovnice se objeví uprostřed obrazovky a je plně funkční. V levém horním rohu se pak objeví časovač, ten zároveň také ukazuje, kdo je zrovna na tahu. Pokud byl časovač vypnut, okno ukazuje pouze, kdo je na tahu, a připomíná, jak má hráč interagovat se šachovnicí.

Závěr

S prací jsem celkově spokojen. Jako někdo kdo před začátkem této práce neuměl programovat, jsem šťastný, že se mi povedlo projekt dotáhnout až do konce. Myslím, že zadání je splněno dostatečně, jediné co chybi jsou okrajová pravidla, která sice nejsou implementovaná úplně správně, ale zároveň to, jak jsou provedeny teď, hru příliš neovlivňuje. Zde bych také měl zmínit stránku Codeacademy[1], ze které jsem se naučil základy programování. Dělal jsem sice nějaké menší kurzy i předtím, ale toto byl můj první důležitý projekt, a tak jsem se naučil nejenom jak programovat, ale také jak tyto velké problémy strukturovat, jak takovýto dlouhodobý vývoj probíhá a kam se mám obrátit když něco v tomto oboru nechápu.

Seznam použité literatury

[] *Codecademy*. URL: https://www.codecademy.com/. (citováno: 07.04.2021).

[] *How to make a threads in java*. URL: https://www.w3schools.com/java/java_threads.asp. (citováno: 07.04.2021).

[] *Pravidla šógi*. URL: https://www.shogi.cz/pravidla. (citováno: 07.04.2021).

[Agi] AgilePro. *How to make a simple timer*. URL: https://stackoverflow.com/questions/10820033/make-a-simple-timer-in-java/14323134. (citováno: 07.04.2021).