Obsah obrázku logo

Popis byl vytvořen automaticky

Šachy aplikace

Dokumentace projektu

Nguyen Xuan Tiep IT 4.BI

Leden–Březen 2025

Střední soukromá odborná škola 1.KŠPA v Litoměřicích

obor Informační technologie-IT zaměřený na Programování a Tvorbu webových stránek

# Zadání:

Cílem projektu je vytvořit šachovou aplikaci v jazyce C#, která umožní hraní šachu proti druhému hráči na stejném počítači. Každý hráč bude mít určený čas, který si vyberou hráči. Aplikace bude dodržovat standardní šachová pravidla, včetně speciálních tahů jako je rošáda, en pasant a povýšení pěšce. Při opakování tahů nebo 50 tahů bez zachycení šachové figurky určí remízu. Uživatelské rozhraní bude jednoduché a přehledné šachovnice v levé časti s polem pro historii pohybů, tlačítka pro uložení hry a časomíry. Další bude funkce pro zpětný tah, který jak název napovídá vrátí hru o jeden tah zpět. Pohyb figurek bude pomocí myší, prvním stiskem vyberete figurku a druhým ji položíte na další pozici. Při vybraní šachové figurky budou zvýrazněné platné tahy. Konec hry bude při šach matu nebo opakování 3 stejných tahů za sebou.

## Dílčí úkoly:

* Vytvoření šachovnice
* Vytvoření figurek
* Implementace tahů jednotlivých figurek
* Nastavení pravidel
* Detekce legálních tahů
* Řešení speciálních tahů
* Detekce šachu a šach-matu
* Vytvoření tlačítka pro uložení hry a zpětný tah
* Vytvoření časomíry
* Pole pro historie zobrazení tahů
* Vytvoření startovního okna a okno k ohlášení konce hry

# Čestné prohlášení

Prohlašuji, že jsem maturitní práci na téma „ŠACHY V JAZYCE C#“ zpracoval sám. Veškeré prameny a zdroje informací, které jsem použil k sepsání této práce, byly citovány v poznámkách pod čarou a jsou uvedeny v seznamu použitých pramenů a literatury.

Obsah

Dne ………………………………… Podpis studenta…………………………….

[Zadání: 1](#_Toc193797718)

[Dílčí úkoly: 1](#_Toc193797719)

[Čestné prohlášení 2](#_Toc193797720)

[Úvod 5](#_Toc193797721)

[Popis a v čem pracuji 6](#_Toc193797722)

[Programovací jazyk 6](#_Toc193797723)

[Visual Studio 6](#_Toc193797724)

[PROCREATE 6](#_Toc193797725)

[Cílová skupina aplikace 6](#_Toc193797726)

[Jak se hrají šachy 7](#_Toc193797727)

[Jak jsem pracoval diagramech 8](#_Toc193797728)

[Use case a aktivity diagram 8](#_Toc193797729)

[Use case diagram šachy 8](#_Toc193797730)

[Aktivity diagram Chess bot 9](#_Toc193797731)

[Aktivity diagram šachy 10](#_Toc193797732)

[Postup práce 11](#_Toc193797733)

[Vytvoření hracího pole 11](#_Toc193797734)

[Vytvoření figurek 11](#_Toc193797735)

[Uložení figurek do programu 12](#_Toc193797736)

[Jednoduchá funkce pro pohyb figurek 12](#_Toc193797737)

[Výpočet tahů 13](#_Toc193797738)

[Zakomponování vypočítaných tahů do pohybu 13](#_Toc193797739)

[Detekce figurek v cestě 14](#_Toc193797740)

[Kreslení figurek na šachovnici 15](#_Toc193797741)

[Sebrání figurek 15](#_Toc193797742)

[Vizualizace tahů 15](#_Toc193797743)

[Třída pro vykonání tahů 16](#_Toc193797744)

[Speciální tahy 16](#_Toc193797745)

[Zachycení pěšákem 16](#_Toc193797746)

[Rošáda 17](#_Toc193797747)

[En pasant 19](#_Toc193797748)

[Povýšení 20](#_Toc193797749)

[Vymazání tahů, co nechají svého krále v šachu 21](#_Toc193797750)

[Detekce konce hry 22](#_Toc193797751)

[Výpis tahů do listu 24](#_Toc193797752)

[Výpis sebraných figurek 26](#_Toc193797753)

[Metoda nová hra 26](#_Toc193797754)

[Počítačový oponent 27](#_Toc193797755)

[První verze 27](#_Toc193797756)

[Druhá verze 28](#_Toc193797757)

[Třetí verze 30](#_Toc193797758)

[Předefinované tahy – čtvrtá verze 35](#_Toc193797759)

[Uložení/načtení her 36](#_Toc193797760)

[Závěr 39](#_Toc193797761)

[Citace 40](#_Toc193797762)

# Úvod

V dnešní době je téměř každý seznámen s hrou šachy, nebo o ní alespoň slyšel. K jejich popularizaci přispěl trend z roku 2020, kdy se šachy staly populárními mezi širší veřejností na internetu. Téma projektu jsem si vybral, protože jsem byl jedním z lidí, kteří v té době začali hrát šachy, a zajímalo mě, jak se šachy programují. Další věcí, která mě zajímala, bylo, jak fungují šachové stroje (Chess Engine) a jak mohou nahlížet několik stovek tahů dopředu během krátké doby.

Cílem projektu bylo, jak jsem již zmínil, zjistit, jak fungují šachové stroje, a zároveň pochopit strukturu objektově orientovaného programování ve větším projektu. Samozřejmě mě zajímala i logika hodnocení tahů a detekce šachmatu.

K práci na projektu jsem použil Visual Studio a C# jako součást .NET Framework. Další aplikací, kterou jsem využil, byl Procreate.

Tato dokumentace popisuje, na čem jsem pracoval, a jak jsem postupoval při vývoji aplikace. Je rozdělena na teoretickou a praktickou část, přičemž každá z nich dále rozděluje jednotlivé části projektu.

Hlavní funkcí aplikace je možnost hrát šachy proti sobě nebo proti počítači. Hru lze uložit a později znovu načíst, pokud nelze dále pokračovat. Po šachmatu nebo remíze se hra resetuje.

Teoretická část

# Popis a v čem pracuji

## Programovací jazyk

Pro svůj projekt jsem si vybral jazyk C#, protože to byl jediný jazyk, který jsem ovládal kromě JavaScriptu, a ten bych zde stejně nepoužil.[[1]](#_Citace)

C# je vysokoúrovňový programovací jazyk od firmy Microsoft. Vysokoúrovňový programovací jazyk je řazení programovacího jazyka podle toho, jak moc se podobá lidské řeči a tím je jednodušší s ním pracovat další jazyk ve stejné kategorii je třeba Python.

## Visual Studio

Pro svoje vývojové prostředí jsem si vybral Visual Studio, protože jsem s ním měl nejvíce zkušeností a je jednodušší s ním pracovat v C#.[[2]](#_Citace)

Microsoft Visual Studio je vývojové prostředí (IDE) od Microsoftu. Může být použito pro vývoj konzolových aplikací a aplikací s grafickým rozhraním.

Visual Studio obsahuje editor kódu podporující IntelliSense a refaktorování. Integrovaný debugger pro tvorbu aplikací s GUI.

Visual Studio podporuje jazyky prostřednictvím jazykových služeb, což umožňuje, aby editor kódu a debugger podporoval jakýkoliv programovací jazyk. Mezi vestavěné jazyky patří C/C++/C#.

## PROCREATE

Tato aplikace je rastrový grafický editor nebo jinak aplikace pro digitální kreslení a animaci. Je to vlastně jako malování ve Windows akorát s více nástroji. V této aplikaci jsem vytvořil obrázky figurek pro mojí aplikaci.[[3]](#_Citace)

# Cílová skupina aplikace

Jak vyplývá z názvu, aplikace je cílená na lidi, co by si rádi zahráli šachy ve svém volném času proti kamarádovi nebo jednoduchému počítači.

# Jak se hrají šachy

Šachy jsou strategická desková hra pro dva hráče, která se hraje na šachovnici o rozměru 8×8 polí. Každý hráč začíná s 16 figurami: jedním králem, jednou dámou, dvěma věžemi, dvěma střelci, dvěma jezdci a osmi pěšci. Hráči se střídají v tazích, přičemž každá figura má přesně daný způsob pohybu. Cílem hry je dát soupeřovu králi mat – tedy ho napadnout tak, aby se nemohl bránit žádným legálním tahem. Hra může skončit i remízou, například pokud dojde k patové situaci nebo se třikrát zopakuje stejná pozice na šachovnici.

Pěšci se pohybují pouze dopředu o jedno pole, v prvním tahu mohou o dvě, a berou diagonálně. Jezdci se pohybují do tvaru „L“ a jsou jediní, kteří mohou přeskakovat jiné figury. Střelci se pohybují diagonálně libovolně daleko, věže po sloupcích a řadách, dáma může kombinovat pohyb věže i střelce a král se posouvá o jedno pole libovolným směrem. Existují i speciální tahy – rošáda, při které se král a věž v jednom tahu vymění, braní mimochodem při pohybu pěšců a proměna pěšce, když dojde na poslední řadu soupeře.

Šachy jsou hrou logiky a strategie, kde je klíčové předvídat soupeřovy tahy a plánovat několik tahů dopředu. Mistrovské zvládnutí šachu vyžaduje nejen talent, ale také pravidelný trénink a zkušenosti.

Praktická část

# Jak jsem pracoval diagramech

Na diagramech jsem pracoval v aplikaci drawio. Kostru diagramů jsem vzal z této dokumentace a projektu.[[13][14][15]](#_Citace)

## Use case a aktivity diagram

### Use case diagram šachy

Tento diagram znázorňuje základní funkce šachové aplikace. Uživatel má možnost uložit aktuální stav hry, aby ji mohl později pokračovat, nebo načíst dříve uloženou partii. Pro začátek nové hry slouží funkce začít hru, která inicializuje šachovnici a figury do výchozího stavu. Pokud hráč potřebuje ukončit aktuální partii, může vypnutím aplikace, nebo ji restartovat pro rychlý začátek nového zápasu. Během hry může hráč provádět tahy manuálně, nebo může nechat vypočítat tah počítače, pokud hraje proti AI. Aplikace také umí řešit koncové stavy hry. V případě, že partie skončí nerozhodně, například patem nebo opakováním pozic, označí se jako remíza. Pokud jeden z hráčů dosáhne vítězné pozice, aplikace určí výhru nebo prohru podle toho, kdo byl matován nebo se vzdal.

Obsah obrázku kruh, text, snímek obrazovky, měsíc

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 1 - Use case diagram šachy

### Aktivity diagram Chess bot

Tento diagram znázorňuje proces rozhodování šachového bota (AI) během hry. Bot nejprve zkontroluje seznam předdefinovaných her, aby zjistil, zda existuje sekvence tahů odpovídající aktuální situaci na šachovnici. Pokud ano, vybere náhodně jednu z těchto her a provede další tah podle této sekvence. Pokud ne, přepne se na použití Minmax algoritmu, který simuluje možné tahy a protitahy do určité hloubky, aby našel optimální tah. Během, tohoto procesu bot kontroluje, zda byl překročen časový limit pro výpočet tahu. Pokud ano, provede Quiescence search, což je minmax, ale prochází všechny tahy, kde může sebrat figuru. Poté provede nejlepší dosud nalezený tah. [[7][10]](#_Citace)

Obsah obrázku text, snímek obrazovky, Písmo, diagram

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 2 - Aktivity diagram Chess bota

### Aktivity diagram šachy

Tento diagram znázorňuje proces hraní šachové partie, a to jak pro lidského hráče, tak pro počítačového oponenta. Hráč nejprve vybere figurku, se kterou chce provést tah, a systém zkontroluje, zda je figurka správné barvy. Pokud ano, vygeneruje všechny platné tahy pro tuto figurku. Hráč pak vybere cílové pole, kam chce figurku přesunout, a systém zkontroluje, zda je tento tah platný a zda na cílovém poli není přátelská figurka. Pokud jsou všechny podmínky splněny, figurka je přesunuta.

Poté systém zkontroluje, zda hra skončila (šachmat, remíza atd.). Pokud ne, hra pokračuje a tah je předán druhému hráči (lidskému nebo počítačovému). Pokud hraje počítač, provede svůj tah pomocí předdefinovaných algoritmů, jako je Minmax, a poté opět přepne strany. Tento cyklus se opakuje, dokud hra nedospěje ke konci.

Obsah obrázku text, snímek obrazovky, Písmo, diagram

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 3 - aktivity diagram šach

# Postup práce

## Vytvoření hracího pole

Začal jsem vytvořením projektu ve Visual studiu a použil jsem .NET framework template. Prvně jsme si myslel, že si nakreslím hrací pole a pozice figurek bude podle mřížky, kterou bych dal na hrací pole. Nedokázal jsem to však tuto myšlenku přenést do funkčního kódu. Nakonec jsem se rozhodl udělat hrací pole z tlačítek a kvůli své lenosti jsem jejich vytvoření udělat v kódu. To mi pomohlo potom s rozložením aplikace, když bych potřeboval změnit velikosti poliček v hracím poli. Pro lepší vzhled políček jsem dal tlačítka na placatého vzhled. V kódu je to implementováno dvěma FOR cykly, jeden pro sloupce a druhý pro řádky. Pozice tlačítek se vypočítává z nuly a pak se přičte velikost tlačítka krát kolikáté tlačítko tvoříme. Barva se potom rozhoduje podle pozice tlačítka XY, což se přičte a podle výsledku se vybere barva pro sudé nebo liché číslo. Potom se přidá do tlačítka klik metoda pro pozdější hýbání s figurkami. Nakonec se přidá do formu a do listu tlačítek pro její pozdější úpravu. List jsme později musel změnit na 2D pole pro generování tahů.

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obsah obrázku šachová figurka, šachy, černobílá, silueta

Obsah vygenerovaný umělou inteligencí může být nesprávný.Obrázek 4 - kód generace šachovnice

## Vytvoření figurek

Figurky jsou udělané jako PNG. Inspiraci jsem si vzal z klasických figurek, co jsou na stránce Chess.com. Figurky jsou více kostkaté a jejich “hlavy“ levitují nad jejich těly. Použil jsem k tomu program s názvem Procreate, což je rastrový grafický editor pro digitální malbu. Raději jsem požil tento program než malování ve Windows, nabízelo mi to více nástrojů ke tvorbě.[[4][3]](#_Citace)

Pěšák

### Uložení figurek do programu

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.Figurky jsem potom uložil jako materiál v programu, aby nebral figurky podle cesta k souboru. Vytvořil jsem si pak novou třídu pojmenovanou třída figurky. V té jsem si udělal slovník pro přístup k figurkám. Každá figurka pak má klíč podle kódu figurky. Jako poslední jsem tam přidal funkci, co vrací obrázek ze slovníku podle kódu figurky.

Obrázek 5 - slovník všech figurek a jejich vrácení

## Jednoduchá funkce pro pohyb figurek

První jsem si chtěl napsat funkci pro pohyb figurek kdy si vyberete figurku a dalším kliknutím ji položit. Při kliknutí na pole s figurkou se daný kód figurky uloží do proměnné „vybraná figurka“. S tou se pak pracuje při dalším kliknutím prázdného políčka, na které se položí. Položení je v kódu jako úprava dříve zmíněného pole s tlačítky. Souřadnice se zjišťují pomocí Indexof Listu tlačítek. Souřadnice figurky se při vybrání uloží do proměnných, aby bylo možné při položení smazat figurku na správných souřadnicích. Figurka se teda položí na pole, co se kliklo. Figurky lze, ale položit kam chcete na hracím poli. Ke konci jsem přidal jednotlivé metody pro řešení tahů.

#### Přidání checkru, kdo je na řadě

Po tom, co jsem si vyzkoušel, jestli kód pro pohyb funguje správně jsem přidal IF příkazy. Ty měli na starosti, že černý mohl hýbat jenom s černými figurkami a naopak. Udělal jsem to pomocí bool proměnné, která reprezentovala, kdo byl na řadě. To se potom společně s kódem figurky zkontrolovalo a boolem, jestli to je správná figurka. Černý mohl hýbat jenom s figurkami, co měla kód figurky malým písmem a bílý jenom s figurkami, co měla kód velkým písmem. Nakonec se při položení figurky změní bool naopak.



Obrázek 6 - podmínka pro bílé figurky



Obrázek 7 - podmínka pro černé figurky

Obsah obrázku text, Písmo, snímek obrazovky, design

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 8 - změní kdo je na řadě

## Výpočet tahů

Obsah obrázku text, snímek obrazovky, software, displej

Obsah vygenerovaný umělou inteligencí může být nesprávný.Jako další, jsem začal pracovat na funkci výpočtu tahů figurek. Jako první jsem implementoval jednoduché odčítání nebo sčítání z pozice figurky. Měl jsem totiž list tlačítek a ne pole. Například máme bílého pěšáka a chceme ho posunout o jednu dopředu, v kódu by byl výpočet aktuálního čísla pole +-8, protože je hrací pole 8x8. Takhle jsem postupoval stejně u ostatních figurek. Všechny až na pěšáka, protože se hýbou podle strany, na které jsou, jsou tahy stejné na obou stranách ve výpočtu. Bohužel jsem zjistil, že tento způsob přestane fungovat když, začnu řešit horizontální tahy tak pak vidím tahy jinde, než by měli být. Tak jsem místo listu tlačítek tak jsem použil dvou rozměrné pole pro lepší manipulaci tahů. To funguje pomocí dvou os, vertikální a horizontální, to se potom odčítá nebo sčítá podle směru tahu. Takže je pěšák na pozici [1,0] a chce se posunout o jeden do předu, tah se vypočítá součtem [1,0] a [+1,0]. což vyjde [2,0] nová pozice. Přidal jsem i limit, aby tahy nebyly mimo pole. Funkce potom vrací list všech tahů konkrétní figurky, která byla vybrána.

Obrázek 9 - generace tahů

### Zakomponování vypočítaných tahů do pohybu

Upravil jsem funkci na přemisťování figurek tak aby se dali jen na ty políčka, která jsou správná pro ni. Část kódu pro vybírání figurky je pořád ten samý jenom jsem upravil kód pro její položení. To funguje tak, že se načtou legální tahy z a projdou se všechny tahy v listě a jestli se rovná tah s vybraným políčkem tak se pokračuje jako normálně.

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 10 - upravený kód pro položení

### Detekce figurek v cestě

Obsah obrázku text, Písmo, snímek obrazovky, řada/pruh

Obsah vygenerovaný umělou inteligencí může být nesprávný.Většina figurek nemohou jít přes ostatní figurky, až na koně, tak jsem upravil generaci tahů, aby bral v potaz, jestli je směr tahu blokován.

Obrázek 11 - přeskakovaní na nový směr

Kód prochází každý směr tahů (nahoru, dolu doleva, doprava) a každý tah v tom směru, když narazí na pole s figurkou přidá souřadnice předchozích polí ve směru a přeskočí na další směr tahů. Pokud je pole obsazené nepřátelskou figurkou tak se přidá i to obsazené do listu pro tahy kde ji můžete sebrat. Přidal jsem i pole pro reprezentaci obsazených polí jako 1 a prázdné 0. Přeskakování je přes přiklad v „strany Pohybu“ kde se vytelí aktuální číslo v listu čtrnácti a přičte se jedna, čímž nám vyjde, jaká se teď prochází a přičte se čtrnáct pro přeskočení na další směr, ke konci se ještě odečte dva, aby nám index nevyjel z velikosti listu na konci. Všechny výsledky jsou v celých číslech.

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 12 - přidání platných tahů do listu tahů

## Kreslení figurek na šachovnici



Obrázek 13 - kreslení figurky krále na pole 7,4

Do této doby jsem jenom vykresloval figurky ručně jak jde vidět na obrázku. Potřeboval jsem jej na testy tahů jednotlivých figurek. Pro vykreslení všech figurek na šachovnici jsem si vytvořil samostatnou funkci, která přímá textový řetězec jako pozice figurek.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Funkce tedy příjme textový řetězec a aktuální šachovnici. Řetězec je ve formátu kód figurky a její pozice seřazeno řádek a sloupec. 

Obrázek - resetování šachovnice a stavu

Obrázek 15 – textový řetězec šachovnice

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Na začátku šachovnici vyčistí, aby byla prázdná, totéž udělá i stavu pole. Poté se zpracuje textový řetězec. Textový řetězec prochází pomocí přes FOR cyklus a přeskakuje v inkrementech tří. Ze řetězce se kód figurky podle aktuálního indexu cyklu a na dalších dvou pozicích se vezmou souřadnice. Souřadnice se musí převést z charakteru do čísel pomocí odečtení ASCII charakteru nula, aby se převedli správně. A na šachovnici se pomocí kódu figurky a souřadnic vykreslí figurka. Nakonec se na stejné pozici ve stavu pole změní číslo na jedna, aby reprezentovala obsazené políčko.

Obrázek - cyklus kreslící figurky na šachovnici

## Sebrání figurek

Jako další jsem přidal funkci, která se zaobírá zachycením/ sebráním figurek oponenta. V kódu je podmínka, která když je políčko obsazené zkontroluje, jestli to je figurka oponenta. Z pole se vezme tag a zjistí se, pokud je velkým nebo malým písmem spolu s kým je na řadě. Pokud tyto podmínku pustí tak se položí figurka. Je to zde aby se zabránilo sebrání vlastní figurky. Položení figurek jsem dal do vlastní metody pro přehlednosti. Snímek z kódu jsem si zapomněl uložit, proto ho zde nemám.

## Vizualizace tahů

Do aplikace jsem přidal funkci, která při vybrání figurky změní barvy platných tahů. Vygenerují se platné tahy pro danou figurku a přes cyklus se projedou všechny tahy a změní se barva jejich políček na červenou. Po položení nebo zrušení výběru se spustí metoda která, změní barvy políček zpátky.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 17 - vizualizace platných tahů

Obsah obrázku text, Písmo, software, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 18 - resetování barev šachovnice

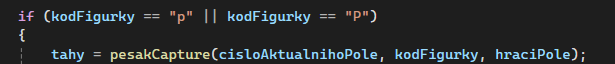
## Třída pro vykonání tahů

Pro metody, které vykonají tah jsem dal do vlastní třídy pro lépe upravitelný kód. Třída se jmenuje realizace pohybu a obsahuje metodu pro klasické položení figurek a později i metody pro speciální tahy.

## Speciální tahy

### Zachycení pěšákem

Sebrání pěšákem je speciální, protože nemůže normálně sebrat figurku před ní, pouze figurky na vedlejších předních políčkách. Přidal jsem do generace pohybů tahy pro sebrání. Tahy byly uloženy do slovníku tahů pod klíčem „p3“. samotnou generaci těchto tahů jsem přidal do vlastní metody. Metoda se spustí při generaci tahů pro pěšáky. A přidá tah do listu jenom pokud na této pozici je figurka protivníka. Nakonec se vrátí tento list do hlavní metody a generují se ostatní tahy.



Obrázek 19 - spuštění metody pro sebrání pěšákem

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 20 - generace, kontrola a přidání tahu

### Rošáda

#### Zjištění, pokud se král nebo věž již neposunula

Rošáda není možná pokud král nebo věž již vykonali tah. Rošádu nelze také pokud jsou v cestě mezi nimi figurky.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Vytvořil jsem si boolean pro černého a bílého krále, pokud se od začátku hry nevykonali pohyb. Další jsem si vytvořil string pole pro zaznamenávání pohybu věží. Při každém pohybu se spustí metoda pro detekci pohybu těchto figurek. Te změní booleany králů na pravda podle kódu krále. Pokud byla figurka věž zjistí se pole, na kterém byly před pohybem a zaznamená se do adekvátního indexu „RookMoves“ pole.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 22 - detekce, pokud byl král posunut

Obrázek 21 - detekce, pokud byla věž posunuta

#### Zjištění, pokud v cestě nepřekáží figurka

Jak jsem zmiňoval ve předešlé části, rošáda není možná pokud královi a věží nepřekáží figurky.

Při pokusu o provedení rošády se zkontroluje, pokud v mezi dvěma figurkami není překážka. V kódu se vyhodnotí, na jakou stranu bude rošáda podle toho, pokud je číslo slupce krále je větší, než sloupec věže v tom případě to je na stranu královny a pokud menší tak strana krále.

Kód projde políčka mezi králem a věží a jestli narazí na obsazené políčko vrátí nepravdu. Projde-li políčka, aniž by narazil na obsazené tak vrátí pravda,

Obsah obrázku text, snímek obrazovky, Písmo

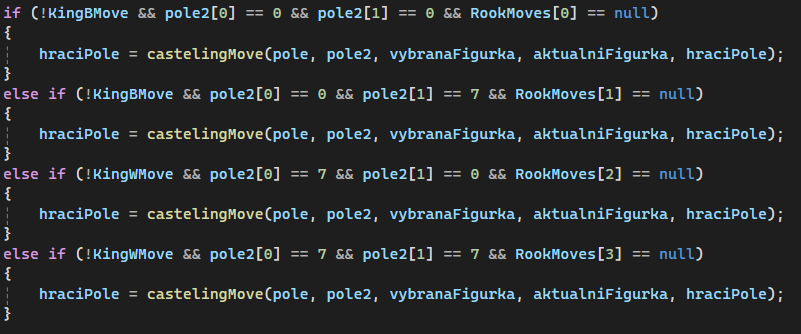
Obsah vygenerovaný umělou inteligencí může být nesprávný.Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

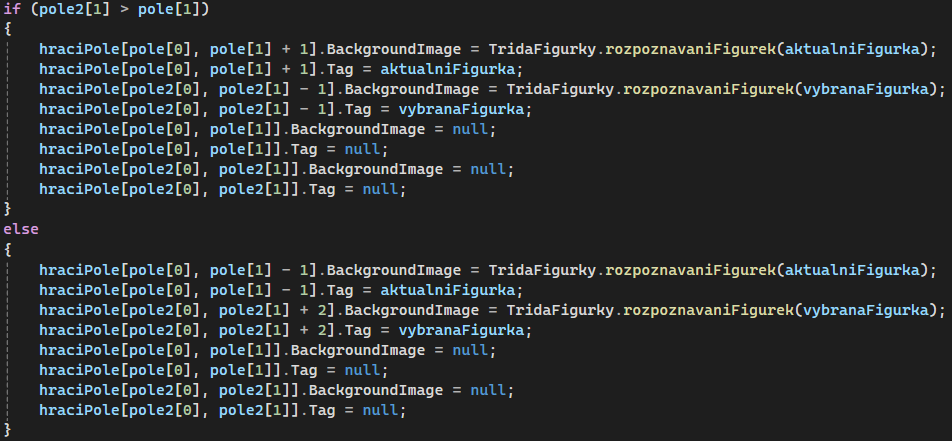
Obrázek 23 - kontrola překážek (strana královny)

Obrázek 24 - kontrola překážek (strana krále)

#### Vykonání rošády

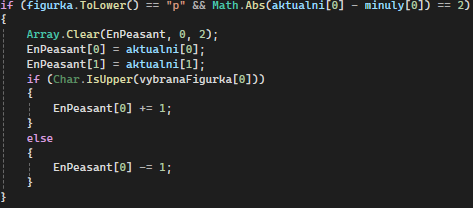
Pokud se splní všechny podmínky pro rošádu projde ještě poslední podmínkou ta zkontroluje, pokud jsou figurky na správných místech. Poté se vykoná rošáda. 

Obrázek 25- kontrola správných souřadnic figurek



Obrázek 26 - vykonání rošády

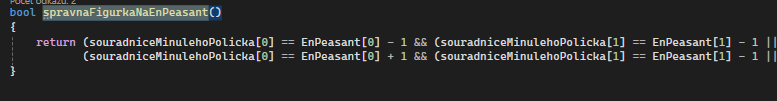
### En pasant

 en pasant je tah kdy pěšák může sebrat pěšáka oponenta, když se posune za něj. Nepřátelský pěšák se však musel minulý tah posunout o dvě pole. Pěšáka lze vzít pouze svým pěšákem a musí být hned na vedlejším poli. en pasant nelze vykonat, pokud byl předminulý tah (Černý posune pěšáka o 2, ale na dalším tahu ho bílý nesebere nebo nemůže, potom en pasant stane nevalidním).

Obrázek 27 - detekce posunutí pěšáka o dvě pole

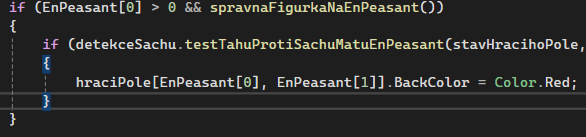
Pro en pasant jsem si vytvořil pole pro uložení souřadnic daného pěšáka. Při tahu, kde se pěšák posune o dvě pole se aktuální souřadnice figurky uloží do en pasant pole. Na konci se poté odečte/přičte jedna, aby byla souřadnice za pěšákem.

Obrázek 28 - kontrola správného pěšáka pro en pasant (1/2)





Obrázek 29 - kontrola správného pěšáka pro en pasant (2/2)

Pokud se hned na dalším tahu vybere pěšák, zkontroluje se, pokud je správná figurka pro vykonání en pasant. Pokud je pěšák správná figurka tak se zobrazí en pasant tah. Při položení pěšáka na en pasant se znovu zkontroluje, pokud je en pasant pole větší než 0(en pasant na poli 0 nejde, tak se používá jako null). Poté se zkontroluje, pokud je figurka správná na en pasant a položí se pomocí metody speciální en pasant.

Obrázek 30 - vybarvení políčka en pasant

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 31 - Kontrola en pasant před položením.

Obsah obrázku text, Písmo, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 32 - položení en pasant

### Povýšení

Obsah obrázku text, kreslené

Obsah vygenerovaný umělou inteligencí může být nesprávný.Povýšení je tah kdy se pěšák dostane na druhý okraj šachovnice, což mu umožní změnit se na jinou figurku (až na krále nebo pěšáka).

Obrázek 33 - okénko povýšení

Na vybrání povýšené figurky jsem si vytvořil nový okénko, kde jsou čtyři tlačítka. Na každé tlačítko jsem dal obrázky možných figurek. Když se klikne na některé tlačítko vezme se kód dané figurky a položí se na místo povýšeného pěšáka. Pokud si však nic nevyberete a zavřete okno vybere se automaticky královna. Form se vždy při položení pěšáka otevře, pokud je řádek aktuální souřadnice 0 nebo 7 (okraje šachovnice). Ještě se zkontroluje, pokud je pěšák na správném okraji (bílý pěšák se musí být na řádek 0 a černý a 7 řádek)

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Obrázek 34 - Klik metoda pro tlačítka.

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 35 - Vybrání královny v případě, kdy není vybraná figurka.



Obrázek 36 - kontrola správné strany šachovnice pro povýšení.

Obsah obrázku text, Písmo, snímek obrazovky, Grafika

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 37 - Otevření povyšovacího okna.

## Vymazání tahů, co nechají svého krále v šachu

V šachách nemůžete provézt tah, který vašeho krále nechá v šachu.

V aplikaci je třída kde se vždy po vygenerování tahů pro vybranou figurku otestují všechny tahy, jestli nenechají jejich krále v šachu.  V oddělené metodě se vezmou vygenerované tahy a otestují se, pokud je podmínka pravda tak se přidá tah do listu, který se pak vrací pro platné tahy figurky. Udělá se kopie šachovnice, na které se provede tah a zkontroluje se, pokud je král v šachu a vrací pravdu nebo nepravdu.

Kontrola krále v šachu je provedena tak, že se projdou všechny figurky oponenta a podívá se, pokud mají tah na krále. Krále souřadnice se hledají projížděním celé šachovnice a hledaní políčka s kódem krále.

Pro en pasant a rošádu, jsem musel udělat od dělenou metodu, protože potřebují jiné parametry pro vykonání tahu.

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 38 - zkoušení tahu

## Detekce konce hry

Hra může skončit tím, že hráč nemá platné tahy (remíza), hráč nemá platné tahy a jeho král je v šachu (vyhraje jeden z hráčů) nebo hráč nemá dostatek figurek (remíza).

První metoda kontroluje, pokud má hráč nějaké plané tahy. To je provedeno tak, že se projdou všechny figurky a jejich tahy se přidají do listu. Pokud je na konci list prázdný tak se vrací pravda.

Další metoda je při nedostatku materiálu, ten zase projede celou šachovnicí a spočítá figurky jako král, královna, věž atd. Na konci se pak podle předem určených podmínek vrací pravda pokud je nedostatek figurek nebo nepravda při opaku.

V hlavním kódu jsem pak při každém položení figurky zkontroluje, pokud není, již konec hry. Metoda pro počet platných tahů se použije samotná pro remízu nebo společné s metodou pro kontrolu krále v šachu se zase může jednat o výhru jednoho z hráčů. Potom metoda pro nedostatek materiálů je samostatné podmínka.

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 39 - vyhodnocení počtu platných tahů

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 40 - sčítání materiálu 1/3

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 41 - sčítání materiálu 2/3

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 42 - sčítání materiálu 3/3

## Výpis tahů do listu

Při každém tahu se vždy zaznamená do listu historie.

Vytvořil jsem si list Box, který se přidá do formu při vytvoření šachovnice a jeho pozice je vázána na velikost šachovnice. Nachází se na pravé straně. Vzor mám podle šachová notace z wikipedie.[[5]](#_Citace)

Kód pro výpis tahů je v samostatné třidě. K vypíšu vždy dojde při položení figurky. Notace je rozdělena na tři části pro lepší úpravu. Vytvořil jsem si i slovník pro správnou notaci, protože souřadnice, které používám jsou jen číslem. První se v hlavní metodě upraví souřadnice, aby seděli notaci. Poté se podívá, jestli byla sebrána figurka což zjistí sečtením figurek před a po vykonání tahu. Pokud byla sebrána přidá se do notace „x“ jako sebraní figurky poté se přidá upravená souřadnice. Jestli jste provedly rošádu tak se notace vymaže a nahradí se speciální notací „O-O“ nebo „O-O-O“ podle její strany. Pokud jste sebrali figurku přes en pasant tak se úplně na konec přidá „e.p.“ nebo při povýšení se na konec přidá „=> {povýšená figurka}“. Nakonec pokud je král oponenta po tahu v šachu přidá se na úplný konec znak „+“.

Poté jsem přidal list, do kterého se ukládá aktuální šachovnice přeložená do textového řetězce. Pro funkci, kdy stisknete na předchozí tah tak vám to ukáže, jak vypadala šachovnice po jejím provedení.

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 43 - slovník správných notací

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 44 - změna souřadnic na správnou notaci

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 45 - metoda citace 1/3

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 46 - metoda citace 2/3

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 47 - metoda citace 3/3

## Výpis sebraných figurek

Pro výpis sebraných figurek jsem vytvořil datagridview kam se při sebrání figurek objeví obrázek sebrané figurky. Na detekci sebraných figurek jsem použil tu samou metodu jako předešlé funkci. Figurka se přidá tak, že se udělá nový image kolem ten se upraví na velikost z toho pixelů a dá se do ní obrázek sebrané figurky nakonec se potom přidá do listu černého nebo bílého hráče.

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 48 - výpis sebraných figurek

## Metoda nová hra

Pokud hra skončí tak jsem přidal metodu, kde všechny proměnné a šachovnice vyresetují pro novou hru.

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 49 - reset hry na novou

## Počítačový oponent

### První verze

Začal jsem pracovat na počítačovém oponentu, který hraje proti vám. Je implementován tak, že se vezme náhodná figurka a její platné tahy. Tahy se poté seřadí podle důležitosti. Seřazeny jsou následovně záchytné tahy, tahy, kde je král oponenta v šachu a ostatní tahy. Pokud jsou záchytné tahy prázdné přejde se na další tahy. Pokud se vybere tah, kdy se pěšák dostane na druhou stranu šachovnice, automaticky se povýší na královnu.

Na konci se potom vrací kód vybrané figurky a souřadnice pro další zpracování (výpis tahů nebo sebraná figurka).

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 50 - náhodný počítač 1/4

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 51 - náhodný počítač 2/4

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 52 - náhodný počítač 3/4



Obrázek 53 - náhodný počítač 4/4

### Druhá verze

Ve druhé verzi jsem přidal evaluační třídu, která vyhodnotí skóre aktuální pozice šachovnice. Evaluace funguje tak, že se sečtou všechny typy a vynásobí jednotlivé druhy jejich daným skórem např. pěšák je 250 bodů. Nakonec se přičte skóre jednotlivých pozic figurek, což znamená, že se pěšák nerovná 250 bodů, ale 255, protože je uprostřed šachovnice. To samé se udělá i s oponentovými figurkami a odečtou se jejich skóre, aby vyšlo celkové skóre pozice.

Figurku jsi pořád vybere náhodně, její tahy provede na kopii šachovnice a vyhodnotí její pozici. Tah, kde je pozice s nevyšším skóre se vybere a zahraje na šachovnici.[[6]](#_Citace)

Obsah obrázku text, snímek obrazovky, Písmo, design

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 54 - hodnota figurek a přiklad bonusu souřadnic pěšáka

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 55 - prohození tabulek pro černéhoObsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 56 - metoda evaluace 1/2

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 57 - metoda evaluace 2/2

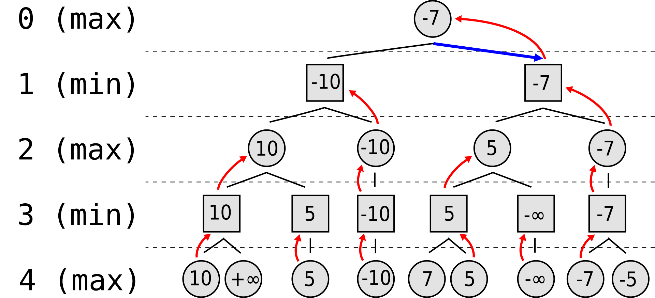
### Třetí verze

Ve třetí verzi jsem upravil počítač, aby vzal každý tah zahrál na kopii šachovnice a vyhodnotil skóre za pomocí algoritmu MinMax a poté vrátí tah. Poté vybere tah s nejlepším skórem a zahraje ho na opravdové šachovnici. [[7]](#_Citace)Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 58 - procházení všech dostupných tahu a hodnocení

#### MinMax

Minmax je algoritmus používaný pro hraní strategických her mezi dvěma a více hráči. Slouží k nalezení nejlepšího možného tahu v dané pozici. Funguje tak, že postupně simuluje všechny možné tahy a po každém tahu předpokládá, že protivník zahraje svůj nejlepší možný tah. Tento proces se opakuje, dokud se nedosáhne určité hloubky nebo konečného stavu hry. Poté se vyhodnotí skóre pozice z pohledu protivníka – čím lepší je jeho skóre, tím horší je tah pro nás. Na základě těchto výsledků se zpětně vybere nejlepší tah pro počítač. Algoritmus není náročný na paměť, ale může být výpočetně náročný kvůli velkému množství možných tahů, které je třeba analyzovat. [[7]](#_Citace)

Obrázek 59 - vizualizace minmax algoritmu

Obsah obrázku text, snímek obrazovky, software, Multimediální software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 60 - minmax algoritmus 1/2

Obsah obrázku text, snímek obrazovky, software

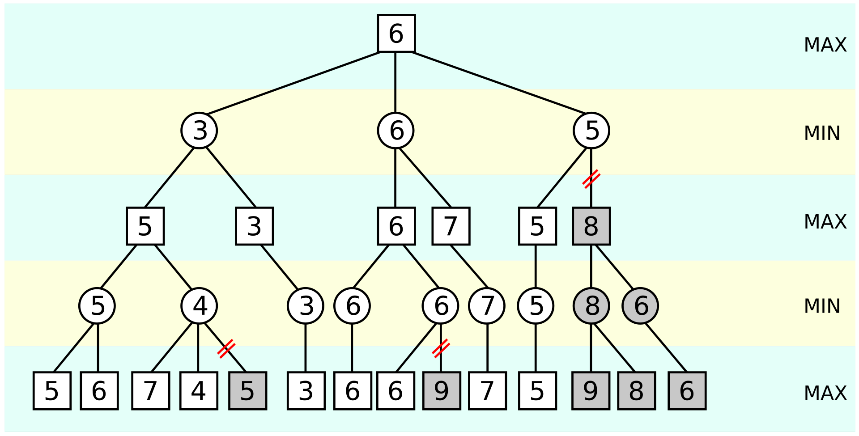
Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 61 - minmax algoritmus 2/2

#### Alfa-Beta ořezávání

Přidal jsem do algoritmu MinMax alfa-beta ořezávání, což by mělo zkrátit čas vyhodnocování obětováním teoretických lepších výsledků.

Princip alfa-beta ořezávání spočívá v tom, že během simulace tahů sledujeme dvě hodnoty – alfa a beta. Alfa představuje nejlepší známé skóre pro max hráče (tedy počítač), zatímco beta reprezentuje nejlepší známé skóre pro min hráče (tedy protivníka). Jakmile se při prozkoumávání některé větve stromu zjistí, že je zaručené, že daná větev nebude mít lepší výsledek než už prozkoumané tahy, algoritmus tuto větev ořízne a nepokračuje v její analýze. Tím se šetří výpočetní čas, protože se vyhodnotí pouze relevantní tahy. [[8]](#_Citace)



Obrázek 62 - vizualizace minmax s alfa-beta ořezáváním

Obsah obrázku text, Písmo, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 63 - alfa-beta ořezávání v kódu

#### Transpoziční stoly

Je velmi jednoduché vylepšení Alfa-Beta ořezávání. Funguje to tak, že při každém vyhodnocení pozice se uloží. Pokud MinMax narazí na pozici uloženou v transpozičního stolu, tak nemusíme vyhodnocovat znova tuto pozici a můžeme ušetřit čas tím, že jenom vrátíme její uložené skóre.[[9]](#_Citace)

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 64 - uložení nové nebo vrácení existující pozice

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 65 - Generování unikátního hash pro pozici

#### Seřazení tahů podle důležitosti

Seřazení tahů slouží ke zvýšení šance střihu v Alfa-Beta ořezávání tím, že lepší tahy jako sebrání figurky se zařadí na první místa v listu.

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 66 - srovnání tahů

#### Quiescence hledání

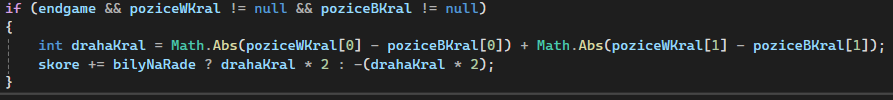
Předtím MinMax na konci vždy vracel evaluaci šachovnice, ale to nemusí vracet správné hodnocení tak jsem přidal quiescence hledání. Quiescence hledání je jenom složitější název pro algoritmus, který funguje stejně jako minmax jenom prochází jen tahy kde lze sebrat figurku. Tento algoritmus však nepotřebuje danou hloubku prochází tahy, dokud nejsou další. Tento algoritmus nám pomáhá s přesnějším vyhodnocením stavu šachovnice.[[10]](#_Citace)

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 67 - Quiescence hledání

#### Upravení evaluace šachovnice

Evaluaci jsem upravil, tak aby ke konci hry byl král nucen být blízko nepřátelského krále pro šachmat. A přidal bonusové body za více platných tahů.

Obrázek 68 - bonusové body za menší vzdálenost mezi králi

Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 69 - detekce end game

### Předefinované tahy – čtvrtá verze

Jako poslední úpravu jsem přidal menší list odehraných her ze stránky „The Week in Chess“. Počítač si prvních pět tahů vybere z listu, pokud se najde hra, která obsahuje všechny minulé tahy. List je přeložen z PGN (portable game notation) do prvních 10 tahů. Každý řádek je jedna hra. [[11]](#_Citace)

Při startu hry se načte ze souboru list her a uloží se do slovníku. Když je počítač na řadě projde všechny hry ve slovníku a projde, pokud obsahují všechny minulé tahy je důležité, že jsou tahy i ve stejném pořadí. Pokud se najde hra, která tyto podmínky splňuje uloží se do listu vybraných her. Potom se vybere náhodně jedna hra z vybraných her a vrátí její klíč. V metodě počítačového hráče je podmínka, pokud se najde klíč hry tak se projdou všechny figurky, které se posunout na další tah v listu. Vybere se potom nejlépe hodnocený tah a zahraje ho. Pokud se však nenajde předdefinovaná hra tak se přejde na minmax algoritmus.

Obsah obrázku text, snímek obrazovky, Písmo, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 70 - načtení her

Obsah obrázku text, snímek obrazovky, software, displej

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 71 - zjištění stejných her

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 72 - odehrání dalšího tahu

## Uložení/načtení her

Jako poslední funkci, na které jsem pracoval bylo načten nebo uložení hry.

Hra a její důležité parametry jako en pasant, rošáda nebo kdo je neřadě se uloží do XML souboru. Jednotlivé parametry jsou rozděleny do jednotlivých kategorií pro lepší orientaci při načítání. Kam si chcete uložit soubor se dá vybrat pomocí průzkumníka souborů, který se při uložení spustí. To samé se stane i při načítání hry. Informace ze XML souboru se čtou podle jednotlivých kategorií.

Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 73 - načtení hry 1/3

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 74 - načtení hry 2/3

Obsah obrázku text, Písmo, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 75 - načtení hry 3/3

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 76 - uložení hry 1/2

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obrázek 77 - uložení hry 2/2

# Závěr

Během vývoje bylo nutné řešit řadu výzev, od implementace základních pravidel šachu až po optimalizaci výpočetního výkonu při simulaci tahů počítačového oponenta. Klíčovým prvkem byla implementace algoritmů MinMax a Alfa-Beta ořezávání, které umožňují počítači hrát na slušné úrovni. Tyto algoritmy byly vylepšeny pomocí transpozičních tabulek a quiescence search, což zvýšilo přesnost výpočtů. Projekt přinesl cenné zkušenosti v oblasti objektově orientovaného programování, práce s grafikou a optimalizace kódu, a zároveň poskytl hlubší pochopení šachových pravidel a logiky, která stojí za šachovými stroji.

Aplikace má potenciál pro další rozšíření, jako je hra přes síť, která by umožnila hráčům hrát proti sobě online, nebo implementace složitějších AI oponentů. Dalším možným vylepšením by byla analýza her, která by hráčům pomohla zlepšit jejich strategii a pochopit chyby. Přidání podpory pro různé šachové varianty, jako je šach 960 nebo rychlý šach, by také rozšířilo možnosti hry.

Projekt byl úspěšně dokončen a splnil všechny stanovené cíle. Vývoj této aplikace přinesl cenné zkušenosti a otevřel dveře pro další vylepšení a rozšíření funkcionality.

# Citace

1. *MICROSOFT. C Sharp. Online. Wikipedie. 2024. Dostupné z:*[*https://cs.wikipedia.org/wiki/C\_Sharp*](https://cs.wikipedia.org/wiki/C_Sharp)*. [cit. 2025-03-08].*
2. *MICROSOFT. Microsoft Visual Studio. Online. Wikipedie. 2025. Dostupné z:*[*https://cs.wikipedia.org/wiki/Microsoft\_Visual\_Studio*](https://cs.wikipedia.org/wiki/Microsoft_Visual_Studio)*. [cit. 2025-03-08].*
3. *SAVAGE INTERACTIVE. Procreate (software). Online. Wikipedie. 2025. Dostupné z:*[*https://en.wikipedia.org/wiki/Procreate\_(software)*](https://en.wikipedia.org/wiki/Procreate_(software))*. [cit. 2025-03-08].*
4. *CHESS.COM, LLC. Chess. Online. Chess.com. 2007. Dostupné z:*[*https://www.chess.com/*](https://www.chess.com/)*. [cit. 2025-03-08].*
5. WIKIPEDIA. *Šachová notace*. Online. 2024. Dostupné z: <https://en.wikipedia.org/wiki/Algebraic_notation_(chess)>. [cit. 2025-03-15].
6. CHESSPROGRAMMING WIKI. *Evaluace pozice*. Online. 2024. Dostupné z: <https://www.chessprogramming.org/Evaluation>. [cit. 2025-03-15].
7. WIKIPEDIA. *MinMax algoritmus*. Online. 2023. Dostupné z: <https://cs.wikipedia.org/wiki/Minimax_(algoritmus)>. [cit. 2025-03-15].

CHESSPROGRAMMING WIKI. *Negamax*. Online. 2024. Dostupné z: <https://www.chessprogramming.org/Negamax>. [cit. 2025-03-15].

1. WIKIPEDIA. *Alfa-beta ořezávání*. Online. 2023. Dostupné z: [https://cs.wikipedia.org/wiki/Alfa-beta](https://cs.wikipedia.org/wiki/Alfa-beta_o%C5%99ez%C3%A1v%C3%A1n%C3%AD). [cit. 2025-03-15].
2. CHESSPROGRAMMING WIKI. *Transpoziční stoly*. Online. 2024. Dostupné z: <https://www.chessprogramming.org/Transposition_Table#How_it_works>. [cit. 2025-03-15].
3. WIKIPEDIE. *Quiescence search*. Online. 2024. Dostupné z: <https://en.wikipedia.org/wiki/Quiescence_search>. [cit. 2025-03-15].
4. THE WEEK IN CHESS. *TWIC Archive*. Online. 1994, 2025-03-10. Dostupné z: <https://theweekinchess.com/twic>. [cit. 2025-03-15].
5. CITACE PRO. *Citace*. Online. 2015, 2025. Dostupné z: [https://www.citacepro.com](https://www.citacepro.com/dok/200hqFsIVDAyQYEH?kontrola=1). [cit. 2025-03-15].
6. DRAWIO. *Diagramy*. Online. ---. Dostupné z: <https://www.drawio.com/>. [cit. 2025-03-15].
7. LUCIDCHART. *UML Activity Diagram*. Online. 2025. Dostupné z: <https://www.lucidchart.com/pages/uml-activity-diagram>. [cit. 2025-03-15].
8. LUCIDCHART. *UML Use Case Diagram*. Online. 2025. Dostupné z: <https://www.lucidchart.com/pages/uml-use-case-diagram>. [cit. 2025-03-15].