1. Úvod

Tento šachový program je aplikace pro WPF vytvořená v jazyce C#. Umožňuje dvěma hráčům zapojit se do klasické šachové hry. Hra uplatňuje všechna standardní šachová pravidla, včetně různých speciálních tahů, jako je en passant, rošáda a další. Implementuje také pravidla pro mat, pat, trojnásobné opakování a pravidlo 50 tahů, aby zajistil realistický zážitek ze hry šachu.

Aplikace poskytuje grafické uživatelské rozhraní (GUI), ve kterém si uživatelé mohou kliknutím na šachové figury zobrazit možné legální tahy, čímž je zajištěno, že nemohou být provedeny žádné nelegální tahy.

2. Požadavky

- Operační systém: Windows 7 nebo novější
- Vývojový rámec: .NET Framework
- IDE: Visual Studio (pro vývoj a sestavování)
- Knihovny: Program využívá standardní knihovny jazyka C# a grafické rozhraní WPF.

3. Instalace

1. Naklonujte projekt.
2. Otevřete projekt ve Visual Studiu.
3. Sestavte projekt (Stisknutím Ctrl + Shift + B).
4. Nastavte startup projekt na ChessUI, popřípadě MinMax (pro testy).
5. Spusťte projekt (Ctrl + F5)

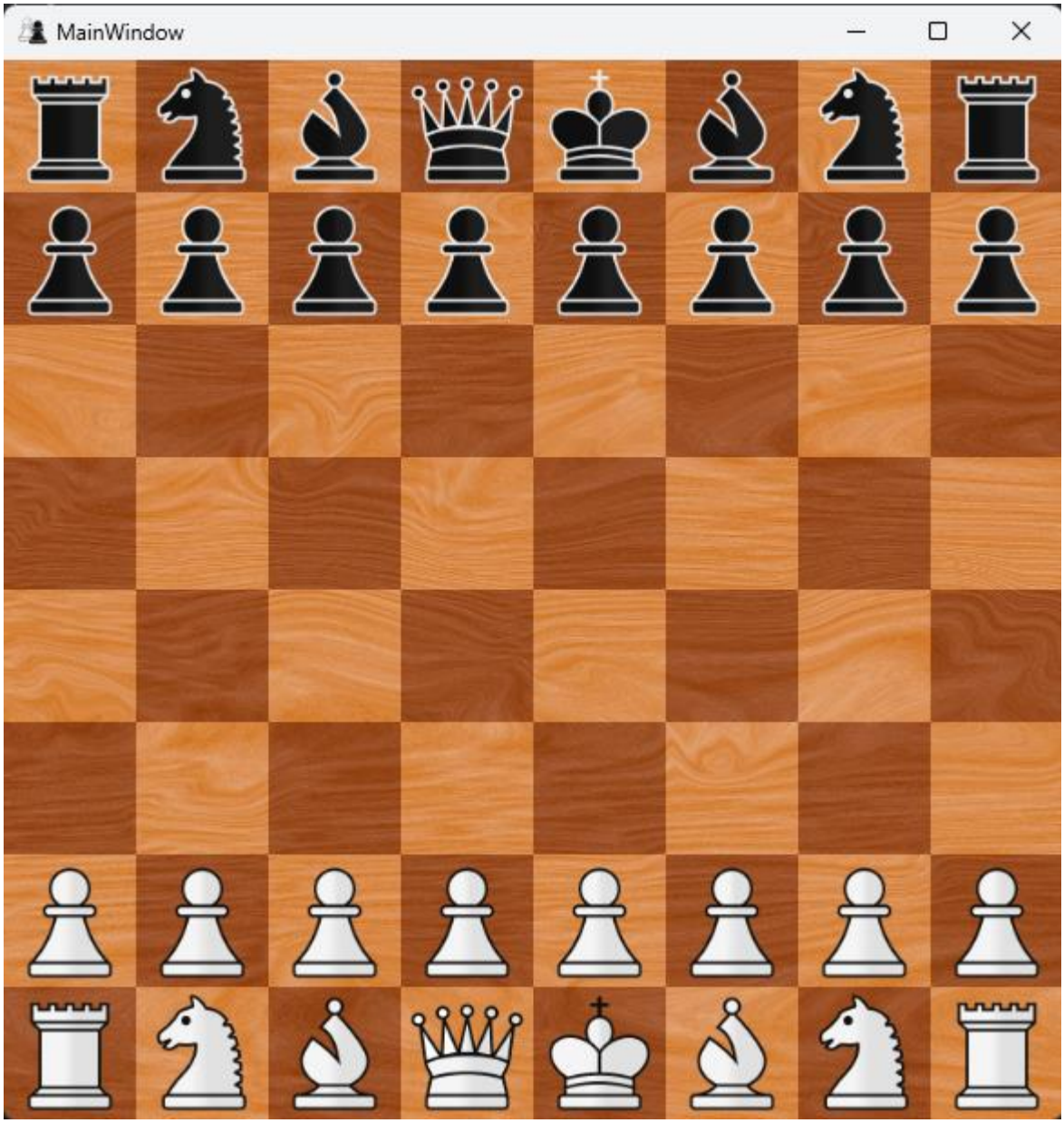
4. Architektura programu

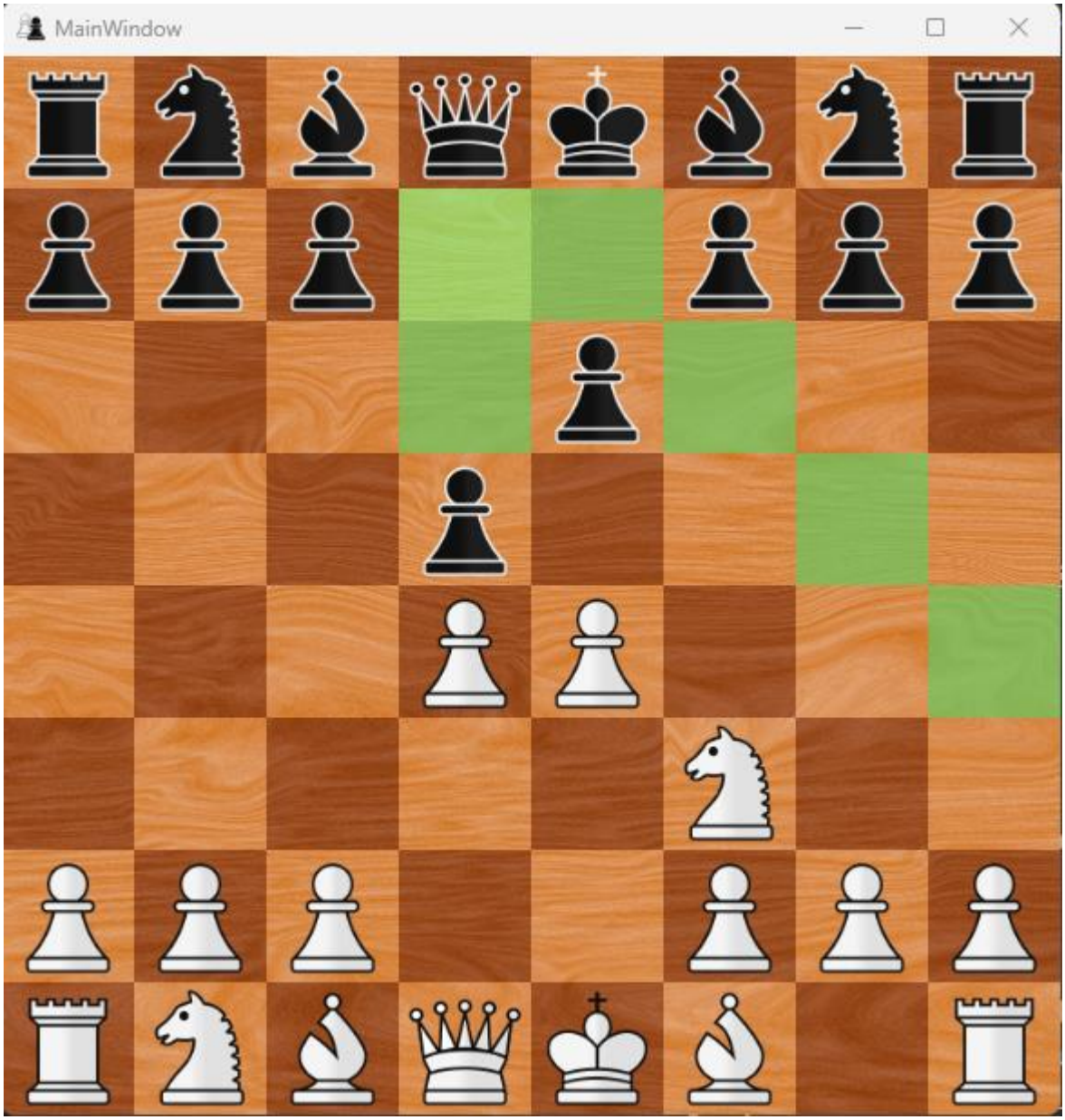
Program je strukturován do několika komponent, které se starají o uživatelské rozhraní šachovnice, pohyb figur a ověřování pravidel. Základními komponentami jsou:

4.1 Uživatelské rozhraní (ChessUI)

Uživatelské rozhraní je vytvořeno pomocí WPF. Obsahuje grafické znázornění šachovnice, figur a tlačítek pro ovládání hry. Když uživatel klikne na figurku, na šachovnici se zvýrazní možné legální tahy.

- Reprezentace šachovnice
- Reprezentace figur: Každá figurka je zobrazena pomocí příslušné grafiky/ikony.
- Interakce: Uživatelé interagují s figurkami pomocí kliknutí myší.


MainWindow


MainWindow

4.2 Logika hry (ChessLogic)

Hra se řídí oficiálními pravidly šachu a zajišťuje správnou tahovou hru. Logika je rozdělena do několika částí:

- Řízení tahů: Po každém platném tahu se střídavě střídá bílý a černý hráč.
- Ověřování platnosti tahů: Program kontroluje legální tahy.
- Detekce konce hry: Po každém tahu kontroluje, zda nedošlo k matu, patu a dalším scénářům.

5. Herní funkce

5.1Pohyb figurky

Program umožňuje pohyb všech standardních šachových figur a vynucuje si jejich jedinečné vzorce pohybu:

- Pěšci: Pěšci se mohou pohybovat o jedno pole vpřed, o dvě pole v prvním tahu a brát diagonálně.
- Věže: Mohou se pohybovat horizontálně nebo vertikálně.
- Koně: Pohybují se ve tvaru písmene „L“.
- Střelec: Pohybují se diagonálně.
- Dámy: Mohou se pohybovat o libovolný počet polí v libovolném směru (kombinace pohybu věží a střelců).
- Králové: Pohybují se o jedno pole libovolným směrem.

5.2 Ověřování tahů

- Legální tahy: Program kontroluje, zda vybraná figura může provést platný tah na základě stavu hry.
- Zajišťuje, aby tahy nezanechaly hráčova krále v šachu.
- Zvýraznění: Po výběru figurky jsou všechny platné tahy vizuálně zvýrazněny.

5.3 Speciální tahy

- Rošáda: Program zkontroluje, zda je možná rošáda, a zajistí, aby se král a věž nepohnuli a aby král nebyl v šachu.
- En Passant: Tento speciální tah pěšce je realizován a kontrolován, když se pěšci posunou o dvě pole dopředu ze své výchozí pozice.
- Povýšení pěšce: Když pěšec dosáhne poslední pozice, program umožní povýšení na dámu, věž, střelce nebo koně.

- Dvojitý tah pěšcem: Pěšci se mohou ze své výchozí pozice posunout o dvě pole vpřed, pokud jim v tom nebrání žádná figura.

5.4 Podmínky konce hry

Program po každém tahu kontroluje několik podmínek konce hry:

- Šach mat: Když je hráčův král v šachu a žádný legální tah nemůže zabránit matu.
- Pat: Když hráč nemá žádný legální tah, ale není v šachu.
- Trojnásobné opakování: Pokud se během partie třikrát vyskytne stejná pozice, je vyhlášena remíza.
- Pravidlo 50 tahů: Pokud je provedeno 50 po sobě jdoucích tahů bez tahu pěšcem nebo bez braní, partie končí remízou.

6. Dekompozice programu

6.1 ChessLogic

- Board se stará o reprezentaci šachovnice, používá přitom všechny třídy figurek. Pomocí funkce Initial() se vytvoří šachovnice se začínajícím rozmístěním. Dále ve třídě jsou funkce IsInCheck(), která hlídá, jestli je daný hráč v šachu, funkce FindPiece(), která najde danou figurku na šachovnici, a funkce jako IsUnmovedKingAndRook() a IsKingKnightVKing(), které hlídají, jestli hráči mohou použít dané pohyby jako třeba rošádu.
- GameState se stará o daný stav hry. Ukládá v sobě šachovnici se současným stavem hry, dělá dané pohyby na šachovnici pomocí MakeMove(move), hlídá stav hry pomocí funkce CheckForGameOver(), která vrátí, jestli hra stále běží, nastal šachmat nebo další možnosti.
- Counting počítá figurky na danné šachovnici.
- Direction definuje pohyby na šachovnici (jako NORTH, SOUTH, EAST, WEST) a definuje operace mezi nimi (+, *).
- EndReason definuje všechny možné konce hry.
- MoveType definuje všechny možné typy pohybů. Nejdříve je tam abstraktní třída Move, podle které se formují všechny ostatní. NormalMove je pohyb, který jde definovat pouze pomocí Directions (střelec, královna, věž). A pak jsou tam zbylé speciální pohyby jako rošáda, En passant atd.
- PieceType definuje všechny možné figurky na šachovnici. Dále máme třídu na kažnou možnou figurku (král, královna, pěšec atd.).
- Player definuje barvu a opponenta daného hráče.
- Position definuje pozici na šachovnici a operace mezi nimi.
- StateString generuje string pro stav dané šachovnice.

6.2 Chess UI

- MainWindow vytváří grafická políčka pro kažné pole na šachovnici ve funcki InitializeBoard(). Maluje na obrazovku figurky z dané šachovnice ve funkci DrawBoard() a stará se o input od uživatele.
- ChessCursors načte dva typy kurzoru (pro bílého a černého hráče).
- Images načítá obrázky figurek a pomocí funkce GetImage() vrací obrázek pro danou figurku.

6.3 MiniMaxAlgorithm

- FindBestMove je funkce, která vrátí buď nejlepší možný pohyb, a nebo null, pokud není možné rozhodnout o kvalitě pohybu z 5 příštích tahů.
- MiniMax je klasická implementace minimax algoritmu, která jde maximálně do hloubky 5
- EvaluateGameState vezme GameState a vrátí správné ohodnocení podle výsledku hry.