**Dokumentace – Dáma na neobvyklé šachovnici**

**Stručné zadání**

Hra Dáma na netradiční válcovité šachovnici v prostředí Pygame. Uživatel má možnost si vybrat ze tří možností, buď hru dvou hráčů, uživatel proti počítači, nebo počítač proti počítači. Počítač využívá heuristiku pro výběr tahů. Herní logika zajišťuje platnost tahů, vybírání figurek a vyhodnocení vítěze.

**Přesné zadání**

Cílem programu je vytvořit funkční aplikaci dámy, kde:

* Hrací plocha má kruhové propojení ve vodorovném směru (sloupce se opakují).
  + Lze si ji představit jako klasickou šachovnici ohnutou a spojenou do válce tak, že podél horního kruhu jsou rozestaveny jedny figurky a podél dolního druhé. Tyto dvě kružnice, horní a dolní pak určují, kde se protivníkovy figurky mění na dámu.
  + V programu je zobrazena jako osm soustředných mezikruží.
* Každá figura se pohybuje podle standardních pravidel dámy.
  + Každá barva má 8 figurek rozestavěných po tmavých polích ve dvou řadách. Šachovnice má 8x8 políček.
  + Pěšci se pohybují pouze dopředu, diagonálně o 1 pole. Přeskočit soupeřovu figuru lze jen tehdy, pokud je za ní volné místo.
  + Dáma vznikne, když pěšec dojde na poslední řadu šachovnice. Dáma se může pohybovat i zpět po diagonále.
  + Pokud je možné přeskočení, hráč ho musí provést.
  + Hráči se střídají v tazích, začíná bílý.Hra končí, pokud jeden hráč ztratí všechny figurky, nebo nemá možný tah.
* Ovládání probíhá myší.
  + Nejdříve hráč kliknutím vybere figurku a potom místo kam se má přesunout.
  + Pokud hráč klikne špatně, druhé kliknutí se „vyčerpá“ a až třetí kliknutí se bere jako nové vybrání figurky.
  + Při pokusu o neplatný tah program vypíše chybovou zprávu. Tah se neprovede a hráč hraje znovu.AI hráč vybírá tah na základě jednoduchého hodnocení.

**Zvolený algoritmus**

* **Lidský hráč**: vybírá tah kliknutím – první klik vybere figuru, druhý klik určí cílové pole.
* **AI hráč**: zohledňuje několik kritérií:
  + - Materiál – počet pěšců a dam (dáma má vyšší váhu).
    - Mobilita – počet dostupných tahů.
    - Povinné braní a hrozby – zda má hráč možnost skoku nebo zda jeho figurky hrozí okamžitým odebráním.
    - Tlak na proměnu – bonus pro pěšce blízké poslední řadě.
    - Centrum a formace – bonus za umístění figurek ve středu a za diagonální souhru.
    - Na rozdíl od klasického minimaxu jde o rychlou heuristiku. AI hodnotí všechny možné tahy, každý krátce simuluje a vybere tah s nejlepším skóre

**Algoritmus hry (pravidla a tahy)**

Hra probíhá na válcové šachovnici, kde levý a pravý okraj desky jsou propojené.

1. **Inicializace hry**
   * Načte se hrací deska, rozloží se figurky na začáteční pozice. Začíná bílá.
2. **Tah hráče**
   * Program čeká na vstup: buď kliknutí myší (člověk), nebo volbu algoritmu (AI).
   * Vybere se figurka a cílové pole.
   * Program ověří, zda je tah podle pravidel platný:
     + pohyb jen diagonálně, nelze hrát soupeřovou figurkou,
     + může-li brát, musí, pohybuje se jen po tmavých políčkách,

**Provedení tahu**

* + Figurka se přesune, případně se odstraní soupeřova figurka.
  + Pokud figurka dojde na poslední řadu, stane se dámou.

1. **Kontrola stavu hry**

* Pokud některý hráč ztratil všechny figurky, hra končí.
* Pokud hráč nemá žádný tah, prohrává.
* Jinak se střídají hráči a hra pokračuje.

**Diskuse výběru algoritmu**

Nejdříve jsem přemýšlela o implementaci **minimax algoritmu s prohledáváním do hloubky** pro silnější AI. Nakonec jsem zvolila jednodušší řešení založené na heuristickém hodnocení. Kód je přehlednější a rychlejší odezva hry. Minimax by byl chytřejší a hrál by optimálně, ale kvůli velkému množství stavů by trval velmi dlouho a potřeboval by hodně optimalizací.

**Struktura programu**

Projekt je rozdělen do více souborů:

* **piece.py**
  + Třída Piece reprezentuje jednu figurku.
  + Uchovává barvu, typ (pěšec/dáma) a definuje možné směry kroků a skoků.
* **board.py**
  + Třída Board reprezentuje šachovnici.
  + Obsahuje matici polí a aktuální rozložení figurek.
  + Ověřuje platnost tahů a implementuje logiku přeskočení.
  + Umí zkontrolovat konec hry a počet figurek.
* **game.py**
  + Třída Game obsahuje hlavní herní logiku.
  + Střídání hráčů, zpracování tahů, povýšení pěšce na dámu.
  + Ošetření pravidel (povinné braní, zákaz tahu zpět).
  + Správa herních zpráv.
* **player.py**
  + Třída Player (abstraktní).
  + HumanPlayer – hráč ovládaný myší.
  + AIPlayer – počítačový hráč, využívající heuristické hodnocení.
* **heuristics.py**
  + Obsahuje hodnoticí funkci AI.
  + Kritéria: materiál, mobilita, povinné braní, tlak na proměnu, kontrola centra a formací.
* **visual.py**
  + Grafické vykreslování pomocí **pygame**.
  + run\_gui\_game(p1, p2) spouští hru mezi dvěma hráči.
  + Funkce start\_menu() umožňuje výběr režimu (hráč vs hráč, hráč vs AI, AI vs AI).
  + Zobrazuje zprávy o hře, aktuálního hráče a obrazovku „Game Over“.
* **main.py**
  + Hlavní soubor, který propojuje všechny moduly.
  + Spustí startovní menu, vytvoří hráče a hru a spustí GUI.
* **tests.py**
  + Obsahuje unit testy pro hlavní herní pravidla.
  + Testuje např. inicializaci, možnost tahu, povýšení na dámu, vícenásobné skoky nebo konec hry.

**Reprezentace vstupů a výstupů**

* Vstupy
  + Kliknutí myší pro lidské hráče (výběr figurky a cílového pole).
  + Vnitřně generované tahy pro AI.
* Výstupy
  + Herní plocha zobrazuje aktuální stav figurek.
  + Ve spodní části se vypisují chybové hlášky (např. neplatný tah) a tahy, které AI zvažuje.
  + Po vítězství hráče se zobrazí obrazovka „Game Over“ s barvou vítěze.

**Průběh vývoje**

1. Vytvoření základní šachovnice a pohybu figur.
2. Implementace kontrol platnosti tahů a střídání hráčů.
3. Přidání AI podle heuristické hodnoticí funkce.
4. Dokončení grafického zobrazení válcové šachovnice.
5. Přidání unit testů pro ověření pravidel hry.

**Co nebylo implementováno**

* Pokročilá AI založená na minimaxu.
* Ukládání a načítání rozehrané hry.
* Více variant pravidel (hra zůstává u základní verze).

**Závěrečný povzdech**

Bylo zajímavé řešit netradiční kruhovou šachovnici, ladění AI zabralo více času, než jsem čekala. Dokonce i více než grafika a logika samotná. Testování krajních případů a zda věci fungují jak mají mě bavilo, zároveň mě fascinuje, že v programu přepíšu pár čísel a místo šachocvice 8\*8 mám 6\*6, nebo jedním # nemám 8 figurek ale čtyři.