

Střední průmyslová škola Třebíč

Maturitní práce

Piškvorky

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITA4

Školní rok: 2024/2025 Jakub Libor Fejta

Zadání práce

ABSTRAKT

Tématem maturitní práce je zpracování klasické hry „Piškvorky“ do digitální formy. Cíl práce je vytvořit funkční desktopovou aplikaci s možností hry jednoho hráče proti počítači nebo dvou hráčů proti sobě.

KLÍČOVÁ SLOVA

maturitní práce, piškvorky, hra

ABSTRACT

The topic of the graduation thesis is to digitalize the classic game „Piškvorky “. The goal of this work is to create a functional desktop application with the option of one player playing against a computer or two players playing against each other.

KEYWORDS

graduation thesis, five in a row, game

PODĚKOVÁNÍ

Děkuji Mgr. Andreje Odehnalové za cenné připomínky a rady, které mi poskytla při vypracování maturitní práce.

V Třebíči dne 19. března 2025 podpis autora

PROHLÁŠENÍ

Prohlašuji, že jsem tuto práci vypracoval/a samostatně a uvedl/a v ní všechny prameny, literaturu a ostatní zdroje, které jsem použil/a.

V Třebíči dne 19. března 2025

podpis autora

Obsah

[Úvod 7](#_Toc193280793)

[1 Teoretická část 8](#_Toc193280794)

[1.1 Programovací jazyk C# 8](#_Toc193280795)

[1.2 Microsoft Visual Studio 8](#_Toc193280796)

[1.3 Github 9](#_Toc193280797)

[1.4 .NET Framework 9](#_Toc193280798)

[1.5 Piškvorky 10](#_Toc193280799)

[1.6 Gomoku 10](#_Toc193280800)

[1.7 Umělá inteligence 10](#_Toc193280801)

[1.7.1 Pravidlově založená AI 10](#_Toc193280802)

[1.7.2 Strojové učení (Machine Learning) 11](#_Toc193280803)

[1.7.3 Hluboké učení (Deep Learning) 11](#_Toc193280804)

[1.7.4 Evoluční algoritmy 11](#_Toc193280805)

[1.7.5 Hybridní systémy 11](#_Toc193280806)

[1.7.6 Umělá inteligence v hrách se zaměřením na logické hry 12](#_Toc193280807)

[1.8 Historie logických her 13](#_Toc193280808)

[1.9 Dopad logických her na přemýšlení 14](#_Toc193280809)

[2 Praktická část 16](#_Toc193280810)

[2.1 Vytvoření projektu 16](#_Toc193280811)

[2.1 Hrací plocha 17](#_Toc193280812)

[2.2 Hrací symboly 17](#_Toc193280813)

[2.3 Konec hry 18](#_Toc193280814)

[2.4 Umělá inteligence 19](#_Toc193280815)

[2.4.1 Lehká obtížnost 19](#_Toc193280816)

[2.4.2 Střední obtížnost 20](#_Toc193280817)

[2.4.3 Těžká obtížnost 20](#_Toc193280818)

[2.5 Nastavení 20](#_Toc193280819)

[2.6 Historie nejlepších 21](#_Toc193280820)

[2.7 Uložení a nahrání hry 21](#_Toc193280821)

[2.8 Demo 22](#_Toc193280822)

[Závěr 24](#_Toc193280823)

[Seznam použitých zdrojů 25](#_Toc193280824)

[Seznam použitých symbolů a zkratek 27](#_Toc193280825)

[Seznam obrázků 28](#_Toc193280826)

[Seznam tabulek 29](#_Toc193280827)

[Seznam příloh 30](#_Toc193280828)

Úvod

Již dlouhodobě trávím svůj volný čas hraním hry Piškvorky, což zahrnuje i účast na různých turnajích. Byl jsem dokonce 5.nejlepší v kategorii jednotlivců soutěže pIsQworky 2023 i pIsQworky 2024 a nejlepší junior a nováček v turnaji Brnocup 2024. Už delší dobu jsem přemýšlel o vytvoření vlastní platformy, takže když jsem viděl maturitní téma „Piškvorky“ jako projekt v C#, rozhodl jsem se tento nápad realizovat.

Cílem projektu je tedy vytvořit platformu pro hraní populární hry „Piškvorky“ v digitální formě. Jedná se o logickou hru, která se hraje typicky ve dvou hráčích na čtverečkovaném papíru. Ve světě je hra známá pod názvem „Gomoku“ a hraje se na dřevěné desce s černými a bílými kameny. Hra má jediný cíl, tím je vytvořit nepřerušenou řadu 5 symbolů (5 a více u Piškvorek).

V platformě jde počet symbolů pro výhru změnit, stejně tak velikost hracího pole nebo i hrací symboly. Platforma obsahuje hru pro dva hráče na jednom zařízení nebo hru pro jednoho hráče proti počítači.

Aplikace je tvořena v Microsoft Visual Studio, což je „Nejkomplexnější integrované vývojové prostředí (IDE) pro vývojáře v .NET a C++ ve Windows.“ [1] Využívá se programovací jazyk C# a pro zálohování dat při tvorbě projektu se využívá platforma GitHub.

# Teoretická část

Práce se zabývá přetvořením hry Piškvorky do digitální formy. Používá programovací jazyk C#, se kterým pracuje v Microsoft Visual Studiu. Konkrétní Framework, ve kterém je projekt vytvořen, je .NET Framework. Pro zálohování projektu a správu verzí se využívá Github.

## Programovací jazyk C#

„Jazyk C# je nejoblíbenějším jazykem pro platformu .NET, bezplatné, multiplatformní open-sourcové vývojové prostředí. Programy jazyka C# se můžou spouštět na mnoha různých zařízeních, od zařízení Internetu věcí (IoT) až po cloud a všude mezi sebou. Můžete psát aplikace pro telefony, stolní počítače a přenosné počítače a servery.“[1]

Je to moderní objektově orientovaný programovací jazyk vyvinutý společností Microsoft jako součást platformy .NET. Svým designem navazuje na jazyky jako C a C++, ale přidává vyšší úroveň abstrakce a bezpečnost, což ho činí oblíbeným mezi vývojáři po celém světě. C# podporuje různé programovací paradigmy, jako je imperativní, deklarativní, funkcionální a objektově orientované programování, což umožňuje flexibilitu při řešení různorodých problémů.

Jazyk je známý svou silnou typovou bezpečností, což minimalizuje chyby způsobené nesprávným typovým převodem. Navíc díky implementaci garbage collectoru C# automaticky spravuje paměť, což zvyšuje stabilitu a výkon aplikací. Mezi hlavní výhody C# patří jeho rozsáhlá standardní knihovna a možnost snadné integrace s technologiemi Microsoftu, jako jsou Windows, Azure a další.

C# nachází uplatnění v široké škále aplikací – od desktopových a webových řešení po herní vývoj prostřednictvím nástroje Unity. Díky své univerzálnosti a snadné čitelnosti je ideálním jazykem jak pro začátečníky, tak pro pokročilé vývojáře.Microsoft Visual Studio

## Microsoft Visual Studio

Microsoft Visual Studio je robustní integrované vývojové prostředí (IDE), které nabízí komplexní nástroje pro vývoj softwaru. Je navrženo tak, aby vývojářům poskytovalo jednotné prostředí pro psaní kódu, ladění, testování a nasazení aplikací. Toto prostředí podporuje mnoho programovacích jazyků, jako jsou C#, C++, Python, JavaScript a další.

Jedním z největších přínosů Visual Studia je IntelliSense – pokročilý systém automatického doplňování kódu, který zvyšuje produktivitu a snižuje počet chyb při psaní. Visual Studio také nabízí výkonný ladicí program, který umožňuje sledovat běh aplikace v reálném čase, nastavovat zarážky a sledovat hodnoty proměnných.

Kromě toho Visual Studio podporuje širokou škálu rozšíření a pluginů, které umožňují přizpůsobit prostředí konkrétním potřebám vývojáře. Integrace s cloudovými službami, jako je Azure, a podpora pro vývoj multiplatformních aplikací dělá z Visual Studia univerzální nástroj pro moderní vývoj.[2]

## Github

GitHub je přední platforma pro správu verzí a týmovou spolupráci, která staví na populárním verzovacím systému Git. Umožňuje vývojářům sledovat změny v kódu, spolupracovat na projektech a snadno spravovat různé verze aplikací. GitHub poskytuje uživatelům možnost vytvářet veřejné i soukromé repozitáře, což ho činí ideálním nástrojem pro open-source projekty i firemní vývoj.

Platforma nabízí užitečné funkce, jako jsou pull requesty, které umožňují snadno integrovat změny do hlavní větve projektu, nebo GitHub Actions, které podporují automatizaci procesů, jako je testování a nasazení aplikací. GitHub také funguje jako sociální síť pro vývojáře, kde mohou sdílet své projekty, sledovat ostatní programátory a inspirovat se jejich prací.[3]

## .NET Framework

.NET Framework je softwarová platforma od společnosti Microsoft, která umožňuje vývoj aplikací na Windows. Obsahuje bohatou knihovnu tříd, které poskytují předdefinované funkce pro širokou škálu úkolů, jako je práce s databázemi, grafickým uživatelským rozhraním, sítěmi a dalšími.

Jednou z hlavních výhod .NET Frameworku je jeho modulární architektura a podpora více programovacích jazyků, jako jsou C#, VB.NET a F#. Framework zahrnuje Common Language Runtime (CLR), což je runtime prostředí, které spravuje běh aplikací, uvolňování paměti a bezpečnost. [4]

## Piškvorky

Hra „Piškvorky“ je klasická hra hraná dvěma hráči typicky na list čtverečkovaného papíru. Hráči se střídají v zakreslování symbolů do políček na papíru, přičemž cílem hry je jako první mít pět nebo více symbolů v políčkách vedle sebe, a to diagonálně, horizontálně či vertikálně.

Hra je v České republice velmi známá a typicky hrána studenty ve škole, často i v době vyučování. Z tohoto vznikla česká soutěž v piškvorkách nesoucí název „PišQworky“ pořádaná pod spolkem Student Cyber Games.[5][6][7]

## Gomoku

Gomoku (z japonštiny „poskládej pět“) je strategická desková hra podobná piškvorkám, známá po celém světě. Její pravidla jsou téměř totožná, ale obvykle se hraje na desce 15x15 políček a řada pro výhru musí být pouze pět symbolů. Hraje se na dřevěné desce zvané Goban s černými a bílými kameny.[7][8]

## Umělá inteligence

Umělá inteligence (AI) je obor informatiky, který se zabývá tvorbou systémů schopných vykazovat chování, které bychom za normálních okolností považovali za inteligentní, pokud by ho vykonával člověk. Cílem AI je umožnit strojům analyzovat data, rozhodovat se, řešit problémy a učit se na základě předchozích zkušeností. AI má široké uplatnění – od herního průmyslu přes zdravotnictví až po autonomní vozidla.[9]

Existuje několik přístupů k vytváření umělé inteligence, z nichž každý má své výhody i nevýhody:

### Pravidlově založená AI

Tento přístup je založen na sadě pevných pravidel, která určují, jak se má AI chovat v určitých situacích. Pravidlově založené systémy jsou snadno pochopitelné a relativně jednoduché na implementaci.

Výhody: Snadná laditelnost a předvídatelnost chování.

Nevýhody: Omezena složitost systému; špatně se přizpůsobuje dynamickým nebo neznámým situacím.

### Strojové učení (Machine Learning)

V tomto přístupu AI analyzuje data a hledá vzory, na jejichž základě se učí. Nejčastěji se využívají algoritmy, jako je regresní analýza, rozhodovací stromy nebo neuronové sítě.

Výhody: Schopnost se přizpůsobit a zlepšovat na základě nových dat; efektivní u složitých problémů.

Nevýhody: Vyžaduje velké množství dat a výpočetní výkon; výsledky nejsou vždy snadno interpretovatelné.

### Hluboké učení (Deep Learning)

Hluboké učení je podskupinou strojového učení, která využívá vícevrstvé neuronové sítě k analýze a interpretaci složitých dat. Tento přístup je klíčový pro rozpoznávání obrazu, řeč a další náročné úlohy.

Výhody: Vysoce přesné výsledky u složitých datových souborů; schopnost automaticky identifikovat důležité vlastnosti dat.

Nevýhody: Požadavek na obrovské množství dat a výpočetního výkonu; často obtížná interpretace rozhodnutí.

### Evoluční algoritmy

Tento přístup simuluje přirozenou evoluci, kdy AI iterativně hledá optimální řešení pomocí mechanismů, jako je mutace, křížení a selekce.

Výhody: Schopnost prozkoumat širokou škálu řešení; vhodné pro optimalizační problémy.

Nevýhody: Časově náročné; může skončit v lokálních minimech bez nalezení optimálního řešení.

### Hybridní systémy

Kombinace více metod (např. pravidlové systémy a strojové učení) může vést k vytvoření robustnější AI, která využívá výhod různých přístupů.

Výhody: Flexibilita; vhodné pro širokou škálu úkolů.

Nevýhody: Vyšší složitost implementace a údržby.

### Umělá inteligence v hrách se zaměřením na logické hry

Umělá inteligence (AI) hraje v logických hrách klíčovou roli při tvorbě náročných a realistických protivníků, kteří poskytují hráčům odpovídající výzvu. Logické hry, jako jsou šachy, Go nebo piškvorky, vyžadují od AI schopnost analyzovat obrovské množství možných tahů, předvídat strategie protivníka a rozhodovat se na základě omezených informací. AI v těchto hrách nejen zvyšuje herní zážitek, ale také ukazuje sílu a omezení různých přístupů k její implementaci.

Jedním z nejpoužívanějších přístupů v logických hrách je využití stromových vyhledávacích algoritmů, jako je Minimax. Tento algoritmus prochází herním stromem, analyzuje všechny možné tahy a předpovídá výsledky na základě soupeřových reakcí. Minimax se často kombinuje s alfa-beta ořezáváním, což umožňuje eliminovat zbytečné větve stromu a tím výrazně zvýšit efektivitu vyhledávání. Tento přístup je obzvláště účinný ve hrách s dokonalou informací, kde jsou všechny možné tahy a stavy hry předem známy, například v šachu nebo piškvorkách.

Další významnou technikou v AI pro logické hry je strojové učení, které umožňuje AI adaptovat své chování na základě předchozích herních zkušeností. Například v šachu se AI může učit z velkých množství historických her, analyzovat tahy nejlepších hráčů a vylepšovat své strategie. V piškvorkách lze podobný přístup použít k analýze vzorců hráčských tahů, což umožňuje AI lépe reagovat na jejich strategie a plánování.

Heuristiky hrají také klíčovou roli v rozhodovacích procesech AI. V logických hrách se heuristiky používají k rychlému vyhodnocení aktuálního stavu hry. Například v piškvorkách může AI hodnotit určité pozice jako „silnější“ nebo „slabší“ na základě počtu symbolů, které hráč nebo protivník dokázali umístit do potenciálně výherních řad.

Logické hry však kladou na AI i výrazná omezení. Kvůli jejich komplexní povaze, například v šachu nebo Go, počet možných tahů exponenciálně roste, což může výrazně zvýšit výpočetní náročnost. Například ve hře Go počet možných stavů přesahuje počet atomů ve vesmíru, což znemožňuje kompletní analýzu všech možností tradičními metodami. Proto je zde klíčové využití pokročilých algoritmů, jako je Monte Carlo Tree Search (MCTS), který zkoumá herní stavy na základě pravděpodobnosti.

AI v logických hrách se stala také testovací platformou pro pokročilé technologie, jako jsou neuronové sítě. Příkladem je systém AlphaGo, který pomocí hlubokého učení a MCTS dokázal porazit nejlepší světové hráče Go. Tento přístup by mohl být přizpůsoben i pro hry jako piškvorky, kde by AI mohla kombinovat různé strategie a učit se na základě analýzy vzorců tahů.

Celkově AI v logických hrách poskytuje nejen zábavu a výzvu pro hráče, ale také ukazuje, jak může technologie řešit složité problémy a simulovat lidské myšlení. Ačkoliv má AI své limity, pokroky v jejím vývoji slibují stále realističtější a náročnější herní zážitky. Tyto pokroky nejen obohacují herní svět, ale zároveň přispívají k pochopení toho, jak lidé přemýšlejí a rozhodují se v různých situacích.

## Historie logických her

Deskové hry mají bohatou historii sahající tisíce let zpět. Byly nalezeny v různých kulturách po celém světě a sloužily nejen k zábavě, ale také k výuce a sociální interakci.

* **Starověký Egypt**: Hra Senet, datovaná kolem roku 3500 př. n. l., je jednou z nejstarších známých deskových her.
* **Mezopotámie**: Královská hra z Uru pochází z období kolem 2600 př. n. l.
* **Asie**: Hra Go vznikla v Číně před více než 2500 lety a dodnes je populární.

Tyto hry odrážejí kulturní a sociální aspekty společností, ve kterých vznikly, a jejich studium poskytuje vhled do historie lidské civilizace.

## Dopad logických her na přemýšlení

Logické hry, jako jsou piškvorky, šachy nebo Go, představují nejen zábavnou formu trávení času, ale zároveň působí jako efektivní nástroj pro rozvoj kognitivních schopností a přemýšlení hráčů. Tyto hry vyžadují od hráčů vysokou míru strategického plánování, schopnost předvídat tahy soupeře a rychle reagovat na změny situace na herní ploše.

Hraní piškvorek například podporuje abstraktní myšlení, schopnost vizualizovat budoucí tahy a plánovat několik kroků dopředu. Jak uvedl Jakub Horák, vítěz soutěže PišQworky: „Je to skvělý trénink na paměť, představivost a abstraktní myšlení.“ Tato hra hráče nutí přemýšlet v širším kontextu, hodnotit možnosti a efektivně se rozhodovat, což jsou dovednosti přenositelné i do každodenního života.

Dále se při hraní logických her rozvíjí trpělivost a vytrvalost. Často se totiž stává, že hráč musí dlouho čekat na příležitost ke správnému tahu nebo musí čelit složité herní situaci, která vyžaduje klid a promyšlený přístup. Podle článku na MujRozhlas.cz „piškvorky rozvíjejí mozek ve všech směrech, vyžadují trpělivost, vytrvalost i logické myšlení.“ Tyto vlastnosti se přirozeně promítají do dalších aspektů života hráčů, například do jejich schopnosti řešit problémy nebo zvládat stresové situace.

Kromě toho mají logické hry pozitivní dopad na paměť. Hráči si musí pamatovat rozložení herní plochy, předchozí tahy soupeře i vlastní strategie, což posiluje jejich krátkodobou i dlouhodobou paměť. Schopnost vizualizovat herní situace a předvídat tahy soupeře podporuje nejen paměť, ale také představivost, která je klíčová při řešení složitých úloh nebo plánování.

I když vědecké studie zaměřené přímo na piškvorky jsou omezené, je zřejmé, že jejich pravidelné hraní může přinášet mnoho podobných benefitů jako hry typu šachy nebo Go. Piškvorky jsou jednoduché na pochopení, avšak dostatečně náročné na to, aby podporovaly kognitivní vývoj, zejména u mladších hráčů, kteří se teprve učí strategicky myslet. Tato kombinace jednoduchosti a strategické hloubky z nich činí jedinečný nástroj pro rozvoj logického myšlení.

Logické hry jako celek přispívají nejen ke kognitivnímu rozvoji, ale také k rozvoji osobnostnímu. Učí hráče dovednostem, které mohou využít nejen při hře, ale i v každodenním životě – od trpělivosti a vytrvalosti přes řešení problémů až po schopnost spolupracovat a respektovat protivníka. Tyto hry tak představují nenápadný, avšak velmi efektivní způsob, jak podporovat mentální a emocionální růst jednotlivců.

# Praktická část

Praktická část práce se zabývá samotným vývojem aplikace Piškvorky – od založení projektu a konfigurace prostředí až po implementaci herní logiky, uživatelského rozhraní a umělé inteligence. Aplikace je vyvíjena v programovacím jazyce C# v prostředí Microsoft Visual Studio a využívá .NET Framework. Díky tomu je plně kompatibilní s operačním systémem Windows a může běžet na široké škále zařízení – od stolních počítačů po notebooky a tablety se systémem Windows.

Kromě funkce pro dvě lokální zařízení (dva hráči na jednom počítači) aplikace nabízí i hru proti počítači, který disponuje třemi různými obtížnostmi. Při vytváření těchto obtížností byl kladen důraz na vyváženost mezi rychlostí výpočtu tahů a kvalitou strategie virtuálního soupeře. Pro hráče, kteří chtějí mít soutěžní náboj, je k dispozici také tabulka s názvem „Historie nejlepších“, v níž se ukládají data o deseti nejlepších výsledcích.

Velkou přidanou hodnotou je také možnost ukládání a načítání hry. Uživatel tak nemusí dohrát hru na jeden zátah, ale může aplikaci kdykoli zavřít a později se k rozehrané partii vrátit. Ovládání aplikace je podpořeno responzivní herní plochou – velikost hracího pole se dynamicky přizpůsobuje oknu aplikace a lze ji volitelně nastavit. Tím je zajištěno, že se Piškvorky pohodlně ovládají na různých rozlišeních obrazovek.

## 2.1 Vytvoření projektu

Vytvoření projektu probíhalo pomocí Microsoft Visual Studio, které je „Nejkomplexnější integrované vývojové prostředí (IDE) pro vývojáře v .NET a C++ ve Windows.“ [1] Při založení projektu byla zvolena šablona Windows Forms App (.NET Framework), díky níž je možné intuitivně navrhovat uživatelské rozhraní a zároveň využívat knihovny .NET.

Aplikace vzniká pod verzovacím systémem Git; repozitář je hostován na GitHubu. Tím je zajištěn přehled o vývoji a hlavně možnost jednoduchého zálohování.

Po vytvoření základního návrhu a struktury projektu byla implementována hrací plocha, ze které byl postupně stavěn celý projekt. Při práci na uživatelském rozhraní se vycházelo z principů pro Windows Forms, které umožňují jednoduché drag-and-drop umisťování ovládacích prvků (Button, Panel, Label, atd.).

## Hrací plocha

Hrací plocha představuje jádro celé hry. V klasických Piškvorkách je obvykle tvořena papírem se čtverečkovanou sítí, avšak v této digitální verzi se vykresluje jako vlastní komponenta s názvem PlayingBoard pomocí použití trídy Graphics a metody DrawLine.

Responzivní design: Rozměry hrací plochy lze dynamicky měnit podle velikosti hlavního okna. Uživatel si v nastavení může zvolit, kolik polí bude hrací plocha obsahovat (např. 15×15, 20×20 atp.). Hrací plocha se také sama přizpůsobuje velikosti formuláře.

Délka výherní řady: Standardně je požadováno spojení pěti symbolů (tzv. „pět v řadě“). Nicméně aplikace umožňuje tuto hodnotu libovolně změnit, takže lze hrát jak klasické Piškvorky (5 v řadě), tak například hru „Tic Tac Toe“ (kde se obvykle hraje na hrací ploše 3x3 a výherní řada se skládá ze tří symbolů).

Implementace hrací plochy sestává z:

**Vykreslení mřížky** – Každý řádek i sloupec je vykreslen tak, aby vznikaly čtvercové buňky.

Interakce s uživatelem – Při kliknutí myší do určitého pole je vyhodnoceno, kam uživatel klikl, a do příslušné buňky je vykreslen symbol hráče (např. křížek či kolečko).

## Hrací symboly

V piškvorkách rozlišujeme nejčastěji křížky (X) a kroužky (O). V této aplikaci jsou symboly reprezentovány grafickými objekty, které se kreslí podle souřadnic kliknutí. Uživatel si přitom může změnit typ symbolu. Může zvolit jiné tvary nebo barvy, pokud k tomu chce dodat hře osobní styl.

Z programátorského hlediska je symbol uložen jako Enumerace (enum Symbol { Free, Symbol1, Symbol2 }) a při vykreslování se pak využije metoda Graphics.DrawLine nebo Graphics.DrawEllipse. Tento způsob vykreslování byl zprvu používán, ale nakonec byl nahrazen Graphics.DrawSymbol, což umožňuje vykreslování libovolých symbolů a usnadňuje nastavování vlastních herních symbolů pro uživatele.

**Vykreslení symbolů**

## Konec hry

Ukončení hry v piškvorkách nastává ve chvíli, kdy:

Někdo propojí požadovaný počet symbolů (např. 5) v řadě, a to horizontálně, vertikálně nebo diagonálně. Tento počet však lze nastavit.

Hráč položí poslední možný symbol na hrací plochu (v případě, že je hrací pole omezené a dojde k zaplnění).

V rámci aplikace je tedy nutné po každém tahu:

Zkontrolovat nově položený symbol (souřadnice X, Y) a spočítat, kolik symbolů stejného typu se nachází směrem doleva/doprava nebo nahoru/dolů, případně v diagonálních směrech.

Pokud některý z těchto součtů dosáhne požadované délky (5 a více), hráč vyhrává.

Při zaplnění hrací plochy bez výherce se vyhodnotí remíza. Pro kontrolu remízy se používá vzorec[video – piškvorky], který nastaví proměnnou typu bool na true pokud jeden z hráčů již nikde nemůže vytvořit požadovaný počet symbolů pro výhru.

Po určení výsledku hra zobrazí hlášení o vítězi a pokud se jedná o hru proti počítači s těžkou obtížností je zapsán výsledek do Historie nejlepších.

Hráči mají volbu hrát větší počet her za sebou, přičemž vyhraje hráč s větším skóre po všech odehraných hrách. V případě více her jsou individuální hry nazývány partiemi. Po odehrání všech partií se opět vypíše hlášení o vítězi.

## Umělá inteligence

Obsah obrázku text, snímek obrazovky, software, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.Jednou z nejzajímavějších částí aplikace je implementace hry proti počítači. Každému hracímu poli je přiřazena hodnota skládající se ze součtu dvou hodnot určujících důležitost pole pro hráči i pro soupeře (hodnota hráče i soupeře je vynásobena koeficientem, který nám umožňuje měnit důležitost hráčových nebo soupeřových polí, díky čemuž můžeme nastavit agresivitu inteligence). Ze všech hracích polí je vybráno pole s maximální hodnotou. Pokud se jedná o první tah hry, bude počítač vždy hrát doprostřed, díky prvotní podmínce, která středovému poli dává základně malou nenulovou hodnotu.

Aplikace disponuje třemi úrovněmi obtížnosti, které všechny vychází z tohoto algoritmu:

### Lehká obtížnost

Počítač hraje většinou smysluplné tahy, avšak může zahrát i náhodně. V 80 % případů vybere ze 3 nejlepších svých tahů, které vyhodnocuje podle základního algoritmu. Ve zbylých 20 % nenabízí výrazné strategické varianty a hraje náhodně. Celkově tato obtížnost slouží spíše pro seznámení s hrou.

### Střední obtížnost

Algoritmus vyhodnocuje několik tahů dopředu i přestože nepoužívá Minimax s algoritmus. Počítač se pokouší bránit, pokud hrozí rychlá prohra, a zároveň útočí, pokud vidí možnost vytvořit čtyřku, která následně přeroste v pět v řadě.

### Těžká obtížnost

Rozšířená verze strategického algoritmu. Snaží se obsadit klíčové pozice na hrací ploše a blokovat potencionální výhry hráče. Vzhledem k omezené velikosti hrací desky a nutnosti držet plynulý chod aplikace se však volí kompromis mezi hloubkou analýzy a rychlostí odezvy.

Díky těmto variantám obtížností je aplikace vhodná pro široké spektrum uživatelů, od začátečníků až po pokročilé hráče. Všechny obtížnosti byly testovány proti lidským hráčům i proti sobě navzájem (střední proti těžké, lehká proti střední) a bylo ověřeno, že umělá inteligence reaguje různorodě a dle očekávání.

## Nastavení

Sekce nastavení je klíčovou součástí aplikace, která uživateli umožňuje přizpůsobit si hru podle vlastních preferencí. Kromě toho, že se v okně nastavení dají zvolit základní parametry (např. velikost hracího pole nebo počet symbolů nutných k výhře), může uživatel konfigurovat i:

Typ a barvu hracích symbolů (křížky a kolečka vs. jiné tvary),

Zda se jedná o hru proti počítači,

Obtížnost hry při hře proti počítači (lehká, střední, těžká),

Počet kol hry.

Z programátorského pohledu je sekce nastavení řešena jako dialogové okno (SettingsForm), kde uživatel volby potvrdí tlačítkem „Ok,“ Nastavení se vždy zapíše do třídy GameSettings, která se vždy při zavírání aplikace zapíše do datového souboru pro zachování posledního nastavení. Díky tomuto řešení může uživatel zachovat své nastavení i po zavření a opětovném otevření aplikace.

## Historie nejlepších

Aplikace obsahuje tabulku 10 nejlepších hráčů, která je motivací k neustálému zlepšování a dává hře další soutěžní rozměr. Záznamy se však uchovávají pouze z her proti počítači, aby se předešlo podvádění. Tato tabulka se nazývá „Historie nejlepších“ a uchovává:

Jméno hráče (či přezdívku),

Výsledné skóre,

Výhry,

Prohry,

Výhry v %,

Počet tahů k výhře

Datová struktura – Ukládání výsledků probíhá do samostatné třídy (např. BestScoreRecord), která má atributy.

Seřazení záznamů – Při přidání nového záznamu se tabulka setřídí podle v sestupném pořadí. Pokud je tabulka plná (10 záznamů) a nový záznam má vyšší bodový zisk než nejnižší v tabulce, je nejnižší odstraněn.

Způsob uložení – Jednodušší variantou je ukládání do textového souboru, v tomto případě do XML. Tento formát však není šifrovaný a umožňuje tak přepisování dat běžným uživatelem. K zabezpečení záznamů jsou data zapsána do souboru datového typu, který nejde běžně uživatelem zobrazit či upravit.

Uživatelům se tabulka zobrazuje v okně aplikace formou tabulky pomocí komponenty DataGridView, v níž lze proklikem nebo přehledným zobrazením sledovat, kdo momentálně vede žebříček.

## Uložení a nahrání hry

Při implementaci funkce uložení a nahrání hry v C# Windows Forms se často využívají zabudované dialogy pro práci se soubory – tedy OpenFileDialog (pro načítání) a SaveFileDialog (pro ukládání). Tyto dialogy poskytují komfortní uživatelské rozhraní pro volbu názvu i umístění souboru, aniž by bylo nutné ručně psát cestu na disku.

SaveFileDialog se v aplikaci vyvolá v okamžiku, kdy uživatel potřebuje uložit rozehranou partii. Ve vybraném dialogu zvolí adresář a název souboru, do nějž se zapíšou všechna důležitá data o stavu hry (konkrétně rozložení symbolů na herní ploše, aktuální hráč, velikost hracího pole a další herní nastavení).

OpenFileDialog umožňuje uživateli vybrat soubor, z něhož se načte dříve uložená hra. Aplikace následně zvolený soubor otevře, přečte z něj potřebné hodnoty a podle toho zrekonstruuje herní pole i všechny ostatní parametry hry. Uživatel tak může plynule pokračovat v rozehraném duelu.

Díky těmto dialogům je práce se soubory pro uživatele maximálně intuitivní, neboť se jedná o známé systémové okno, kde lze vybírat složky a soubory tak, jak je uživatel zvyklý z jiných programů. Volba formátu, do něhož se data ukládají, může být buď obyčejný textový (přípona .txt), strukturovaný formát (například .xml nebo .json), případně jakýkoli jiný, který vývojář uzná za vhodný. V souboru se kromě stavu herní plochy a nastavení obvykle ukládá i část nastavení aplikace, například obtížnost AI či velikost hracího pole.

Celé řešení se soustředí na to, aby byla chráněna data uživatele – aplikace kontroluje, zda byl soubor vybrán platně, a pokud dojde k chybě (poškozený soubor, chybný formát apod.), zobrazí uživateli chybové hlášení s vysvětlením. Důležité je také počítat s možnou nekompatibilitou verzí – pokud se aplikační logika rozšíří o nové funkce, musí být zajištěn mechanismus, jak se k datům z dřívějších verzí přistupuje.

Díky propojení SaveFileDialog a OpenFileDialog s uložením herního stavu do čitelného formátu získá uživatel přehled a volnost v tom, kam a jak často si hru ukládá. Zároveň se jedná o lehké a rychlé řešení, které nevyžaduje žádnou nadbytečnou infrastrukturu (např. databáze). Pro desktopovou aplikaci, jež má poskytnout jednoduchou možnost ukládání stavu, je tento přístup ideální.

## Demo

Demo mód v tomto projektu slouží k automatickému přehrání hry, ve které se dva počítačoví hráči střídají v tazích. Hráč sleduje hru bez nutnosti zasahovat. Tento režim demonstruje schopnosti umělé inteligence a zároveň umožňuje vidět různé strategie v průběhu hry.

Po kliknutí na tlačítko Demo v hlavním menu se zobrazí formulář s běžnou hrací plochou, na které se však po kliknutí uživatele odehraje hra dvou počítačů nastavených na lehkou obtížnost. Po každém tahu je nastavena krátká prodleva, aby byl průběh hry přirozený a vizuálně přehledný. Hra pokračuje automaticky, dokud nenastane výhra nebo remíza, po které se uživatel vrátí do menu.

Technicky je demo řízeno pomocí proměnné GameSettings.DemoMode, která se aktivuje při spuštění demo režimu. V průběhu hry je pak na základě této hodnoty automaticky prováděn tah AI pro oba hráče, aniž by se čekalo na vstup uživatele. Tahy jsou prováděny v metodě zpracovávající tahy hráčů, kde po tahu jednoho hráče následuje okamžitě tah druhého.

Hlavním cílem této funkce je demonstrace hry, ukázka možností umělé inteligence a vizuální prezentace pravidel hry.

Závěr

Aplikace disponuje hrou pro dva hráči i pro hráče proti počítače. V aplikaci lze nastavit všechny potřebné požadavky pro úplnou aplikaci. Za více souběžných výherních řad je ovšem počítán pouze jeden bod do stavového řádku, a tedy i do historie nejlepších, což nevyhovuje plně zadání, avšak se jedná o běžnou praktiku u podobných aplikací. Aplikaci je stále možné rozšiřovat a zlepšovat, což je také mým plánem do příštích měsíců.

Seznam použitých zdrojů

1. <https://learn.microsoft.com/cs-cz/dotnet/csharp/>
2. MICROSOFT. *Visual Studio: Integrované vývojové prostředí (IDE) a editor kódu pro vývojáře softwaru a týmy*. Online. ©2024. Dostupné z: <https://visualstudio.microsoft.com/cs/#vscode-section>. [cit. 2024-10-24].
3. <https://docs.github.com/en/get-started/start-your-journey/about-github-and-git>
4. <https://learn.microsoft.com/cs-cz/dotnet/framework/get-started/overview>
5. <https://pisqworky.cz/o-soutezi>
6. <https://cs.wikipedia.org/wiki/Piškvorky>
7. <http://www.piskvorky.cz/federace/o-piskvorkach-neboli-gomoku-2/>
8. <https://cs.wikipedia.org/wiki/Gomoku>
9. <https://www.ackee.cz/blog/umela-inteligence-v-pocitacovych-hrach-aneb-mysli-ta-hra-skutecne-sama>
10. <https://github.com/Mnaukal/piskvorky-minimax/blob/master/2017_O_Töpfer_Michal.pdf>
11. <https://www.scienceworld.cz/technologie/umela-inteligence-v-pocitacovych-hrach-1-2333/>
12. <https://www.inf.upol.cz/downloads/studium/PS/minimax.pdf>
13. <https://www.stoplusjednicka.cz/pod-kridly-stesteny-tisicileta-historie-deskovych-her>
14. <https://cs.wikipedia.org/wiki/Desková_hra>
15. <https://www.scienceworld.cz/clovek/co-dela-sachoveho-genia-inteligence-nebo-pamet-597/>
16. <https://ovsachy.webnode.cz/proc-hrat-sachy>
17. <https://vasekupony.cz/blog/4-duvody-proc-by-deti-mely-hrat-sachy>
18. <https://vychova-hrou.cz/jak-hry-ovlivnuji-psychicky-vyvoj-deti>
19. <https://pisqworky.cz/novinky/2723-piskvorky-hraju-uz-sest-let-je-to-skvely-trenink-na-pamet-rika-sampion-xo21-jakub-horak>
20. <https://www.mujrozhlas.cz/apetyt/hra-ktera-vyzaduje-trpelivost-vytrvalost-i-logicke-mysleni-piskvorky-rozviji-mozek-ve-vsech>

Seznam použitých symbolů a zkratek

|  |  |  |
| --- | --- | --- |
| Symbol | Veličina | Jednotka |
| S | Entropie |  |
| Q | Teplo |  |
| T | Termodynamická teplota |  |
| t | Čas |  |
|  |  |  |

Seznam obrázků

[Obr. 2.1 Obsah 10](#_Toc147493921)

[Obr. 2.2 Příklad umístění legendy obrázku 12](#_Toc147493922)

Seznam tabulek

[Tab. 2.1 Legenda k tabulce 12](#_Toc147493615)

Seznam příloh

Prázdná šablona maturitní práce