

Střední průmyslová škola Třebíč

Maturitní práce

Pexeso

Profilová část maturitní zkoušky

Studijní obor: Informační technologie

Třída: ITA4

Školní rok: 2024/2025 Jakub Černý

Zadání práce

ABSTRAKT

Tématem této maturitní práce je vytvoření známe deskové hry pexeso. Hlavní funkce projektu je nabídnout zážitek prostřednictvím jednoduchého uživatelského rozhraní. Cílem hry je najít dvojici stejných obrázků. Práce popisuje proces tvorby této hry pomocí programovacího jazyku C# a systému GitHubu.

KLÍČOVÁ SLOVA

maturitní práce, pexeso, C#, GitHub, hra

ABSTRACT

The topic of this graduation thesis is the creation of well-known board game Pexeso. The main function of the game is to provide the users with an enjoyable simple user experience. The goal of the game is to find pairs of matching images. The work describes the development proces of the project using C# programming language and GitHub.

KEYWORDS

graduation thesis, pairs, memory, C#, GitHub, game

PODĚKOVÁNÍ

V Třebíči dne 9. 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 9. března 2025

podpis autora

Obsah

[Úvod 7](#_Toc191976808)

[1 Teoretická část 8](#_Toc191976809)

[1.1 Vývoj a vliv deskových her 8](#_Toc191976810)

[1.2 Benefity deskových her 9](#_Toc191976811)

[1.3 Historie pexesa 11](#_Toc191976812)

[1.4 Pravidla hry pexeso 12](#_Toc191976813)

[1.5 Téma mého pexesa 12](#_Toc191976814)

[1.6 Microsoft 13](#_Toc191976815)

[1.7 Visual Studio 14](#_Toc191976816)

[1.8 .NET Framework 14](#_Toc191976817)

[1.9 C# 15](#_Toc191976818)

[1.10 Windows Forms 15](#_Toc191976819)

[1.11 GitHub a Git 16](#_Toc191976820)

[2 Praktická část 17](#_Toc191976821)

[2.1 Třídy 17](#_Toc191976822)

[2.2 Třída GameBoard 18](#_Toc191976823)

[2.2.1 Konstruktor herního pole 18](#_Toc191976824)

[2.2.2 LoadImages 18](#_Toc191976825)

[2.2.3 SetupLayout 19](#_Toc191976826)

[2.2.4 CreateCardLabel 19](#_Toc191976827)

[2.2.5 PlaceCards 20](#_Toc191976828)

[2.2.6 InitializeBoard 21](#_Toc191976829)

[2.2.7 GetCardImage 21](#_Toc191976830)

[2.2.8 FlipCardFront 21](#_Toc191976831)

[2.2.9 GetBackImage 21](#_Toc191976832)

[2.2.10 FlipCardBack 21](#_Toc191976833)

[2.3 Třída GameLogic 21](#_Toc191976834)

[2.3.1 Konstruktor logiky hry 22](#_Toc191976835)

[2.3.2 Enum GameState 22](#_Toc191976836)

[2.3.3 OnCardClicked 22](#_Toc191976837)

[2.3.4 OnTimerTick 24](#_Toc191976838)

[2.3.5 GetRight 24](#_Toc191976839)

[2.3.6 ComputerTurn 24](#_Toc191976840)

[2.3.7 WinnerCheck 26](#_Toc191976841)

[2.4 Třída GameScoreManager 26](#_Toc191976842)

[2.4.1 Add, Get, Set 26](#_Toc191976843)

[2.4.2 GetSortedScores, EndScore 27](#_Toc191976844)

[2.5 Třída SoundManager 27](#_Toc191976845)

[2.5.1 Konstruktor 27](#_Toc191976846)

[2.5.2 LoadAudio 27](#_Toc191976847)

[2.5.3 PlayAudio 28](#_Toc191976848)

[2.5.4 StopAudio 28](#_Toc191976849)

[2.6 Psaní úvodu 29](#_Toc191976850)

[2.7 Struktura odstavců 29](#_Toc191976851)

[2.8 Obrázky, tabulky a rovnice 30](#_Toc191976852)

[2.9 Řazení a struktura kapitol 32](#_Toc191976853)

[2.10 Závěr 32](#_Toc191976854)

[2.11 Seznam použitých zdrojů 32](#_Toc191976855)

[2.12 Seznam použitých symbolů a zkratek 32](#_Toc191976856)

[2.13 Seznamy použitých obrázků a tabulek 33](#_Toc191976857)

[2.14 Seznam příloh 33](#_Toc191976858)

[Závěr 34](#_Toc191976859)

[Seznam použitých zdrojů 35](#_Toc191976860)

[Seznam použitých symbolů a zkratek 37](#_Toc191976861)

[Seznam obrázků 38](#_Toc191976862)

[Seznam tabulek 39](#_Toc191976863)

[Seznam příloh 40](#_Toc191976864)

Úvod

Už od mladého věku jsem byl nadšeným hráčem deskových her, které jsem hrával se staršími členy mé rodiny. A přesto, že mě vždycky nechávali vyhrát, abych se nevztekal, tak mi tato lásko k deskovým hrám zůstala pořád.

Od svých deseti let jsem vyměnil deskové hry a knížky za počítač, kde se zrodila moje láska k programování. Nejdřív začala ve formě videoher a poté se změnila, když jsem nastoupil na SPŠT, kde jsem byl vystaven programování.

Z programování mě nejvíce zaujal vývoj počítačových her, z důvodu nostalgické lásky k nim. Takových her nemám mnoho, ale jejich vývoj mě neskutečně bavil a přinesl mi spoustu nových a cenných zkušeností. S třemi spolužáky jsme založili amatérské herní studio v rámci řízení projektů, kde jsme vytvořili dvě hry. Já jsem se díky této velice cenné zkušenosti naučil používat programování v praxi mimo teorii ve škole.

Zadáni práce bylo ve stejném programovacím jazyce jako jsme používali ve vývoji našich menších her. Přesně z těchto několika důvodů jsem si vybral tuto maturitní práci.

Hra pexeso bude tvořena za pomocí programovacího jazyka C# v prostředí Visual Studio a verzovacího prostředí GitHub.

# Teoretická část

Tato část maturitní práce je určena k představení použitých technologiích a pojmů. Také se v ní nachází stručný popis deskových her, pexesa, jeho pravidel a zvoleného témata.

## Vývoj a vliv deskových her

Trh s deskovými hrami má dnes největší zisk za celou dobu svojí existence. [] Může tomu vděčit za dnešní možnost vše přetvořit do digitální podoby. Dnes každý nápad nebo vylepšení je možno předělat do aplikace nebo webové stránky a díky tomu je výběr mezi hrami tak neskutečně moc velký, že můžeme hrát každý den novou hru. Mezi hrami můžeme vybírat z několika žánrů, ze starých klasik anebo třeba podle počtu hráčů. Obrovskou výhodou online světa je, že nepotřebujeme být se spoluhráči v jedné místnosti, ale můžeme hrát na dálku, klidně i přes půl světa. Nemusíme spoluhráče dokonce znát, můžeme prostě a jednoduše kliknout na tlačítko a na hrací ploše se objeví neznámý hráči nebo umělá inteligence.

Ale ne vždycky to tomu tak bylo, naši dávní historičtí předci neměli takovou škálu výběru a informací. Vystačili si s jednoduchými deskovými hrami, které sami po domácku vyrobili. Každá významná i nevýznamná historická civilizace měla svoje hry, ať už byly originální nebo převzaté z jiné kultury. Spoustu se jich dochovalo v hrobkách anebo v záznamech a můžeme, díky tomu do dávné historie nahlédnout.

Úplně nejstarší předchůdce deskové hry, o které alespoň víme, je starší než samotné písmo. Archeologové našli předchůdce hrací kostky na pohřebišti v jihovýchodním Turecku. Toto pohřebiště je přibližně z roku pět tisíc let před Kristem, což se zdá neuvěřitelné. Na místě se ocitlo asi padesát vyřezaných a namalovaných kamínků, které sloužili jako kousky hry. Originální hrací kostky, byly ze dřeva, kostí a z kamení. Podobné nálezy se vyskytují po celém středním východě, což naznačuje, že deskové hry jako takové, se začaly hrát přímo tady. [1]

Deskové hry hráli i starověcí Egypťané, a první záznamy o jejich populární hře, byly datovány do roku 3500 př.n.l. Hra jménem Senet, připomíná předchůdce šachů. Cílem hry je přejít kamennými figurkami herní pole a vyhnout se jeho nástrahám. Obyvatelé dávného Egypta brali hru nejen jako zábavu, ale i jako spirituální a náboženský nástroj, který simuloval cestu do světa mrtvých. Hráč se takto mohl vyzkoušet ovlivnit svoje poslední momenty posmrtného života. Senet byl součástí pohřební výbavy mnoha faraonů a byl nalezen v několika slavných hrobkách. []

https://www.cestyarcheologie.cz/single-post/hry-v-prubehu-veku-4-senet

Populární hra Go, kterou pravidelně stále hraje několik desítek miliónů hráčů, převážně v Asii, odkud hra také pochází. Její původ se odhaduje být asi 3000 let starý. Hra byla v Japonsku tak vážená, že v sedmnáctém století vláda založila a podporovala čtyři školy, určené k výuce této hry. Hráč Go byl v této době uznáván jako povolání. Hra byla také populární v Číně a Koreji. Hra se rozšířila do celého světa kvůli druhé světové válce.[2]

Šachy, nejkóničtější a nejznámější desková hra na světě pochází z šestého století našeho letopočtu. Hra byla nejdříve hraná v Indii, poté se se díky vlivu muslimů roznesla do západního světa. Kolem patnáctém století se v Itálii a Španělsku zrodili pravidla hry, tak jak je známe. Začaly se psát knihy o pravidlech a umění šachu. Moderní turnaje a soutěže začaly v devatenáctém století, kdy se také odehrál úplně první světový turnaj, který je pořád pravidelně pořádán. Později vznikla mezinárodní organizace a začala spravovat světové turnaje a další soutěže.

Od začátku roku dva tisíce začaly být populární počítačové analýzy, které naprosto změnili přistup k hře. Hráči mohli zdokonalovat svoje taktiky a získat tím výhodu nad protihráči. Online prostředí šachů je dnes také velice populární, milióny hráčů po celém světě hraje každý den. [3]

## Benefity deskových her

Z historie deskových her lze vidět, že byly důležitou součástí životů našich předků. Byla to jedna z mála možností jejich omezené zábavy. Proto byli kreativní a po celém světě se vymýšleli různé druhy. Hry můžeme dělit do různých žánrů, například podle tématu, materiálu anebo počtu hráčů.

Drtivá většina deskových her se hraje ve více hráčích, samozřejmě existuje několik výjimek. Je to úžasný způsob, jak můžeme strávit večer se svými blízkými, malými sourozenci, partnerem anebo s celou rodinou. Hry dokážou být úžasným rozptýlení a zábavou.

Častá nutná spolupráce, která je vyžadována kvůli vítězství, je skvělým způsobem, jak si ji procvičit do zbytku života. Díky spolupráci v týmu, dokážeme posílit naši schopnost komunikovat s ostatními členy týmu. Komunikaci je základ každého dobrého týmu, pochopení a dostatečné vysvětlovaní, aby ostatní členové týmu pochopili týmovou strategii, je velikou výhodou, která se hodí do profesního a každodenního života.

Deskové večery jsou také velice dobrý způsob, jak se spřátelit s někým nový nebo si naopak zocelit staré přátelství anebo rodinné pouto. Kvůli spolupráci v týmech se vztahy mezi hráči sblíží. Mezi hracími koly se často vede společenská řeč, nikdy nevíte, na které zajímavé téma si najdete někoho se stejným názorem.

Nutnost pamatovat si komplexní pravidla, svoje a protihráčovi tahy a cíl hry, je velice náročné na paměť. Některé deskové hry, jsou výrazně soustředěné na procvičení paměti. Lepší paměť v životě je velikou výhodou, převážně když příště zapomenete, kde jste nechali ležet klíče.

Komplexnost a náročnost pravidel má ještě jednu velikou výhodu. Hráč musí přemýšlet a vymýšlet nové strategie, díky kterým přelstí ostatní protihráče a samotnou hru. Toto strategické a kritické myšlení zapojuje části mozku, které je nutné pravidelně procvičit. Takové myšlení je velikou výhodou, jak v profesním, tak každodenním životě.

Různorodost konečného výsledku na konci hry, drtivá výhra až i suverénní prohra je cenou lekcí pro naše emoce. Udržet kontrolu nad svými emocemi, při prohře je velikou lekcí převážně pro soutěživé povahy a malé děti, které chtějí za každou cenu vyhrát.[4]

Nátlak z každodenního života je velice stresující věc, který čeká každého. Deskové hry jsou vědecky potvrzený způsob, jak se uvolnit od tohoto stresu. Hra vyžaduje vaši plnou soustředěnost, protože pokud budete myšlenkami jinde, určitě nevyhrajete. Proto když myšlenky zapojíme plně na hru, vytvoří to takové uvolnění od zbytku světa a našich problémů. Úspěšná výhra vytvoří pocit úspěchu a ten dokáže zlepšit náladu a snížit náš stres. [] https://bandpassdesign.com/blogs/news/benefits-of-board-games?srsltid=AfmBOoob1w8gXQnjEmzTnUycI-i5v-mG7k9KFTV2P1m61fPBpzRpwjYh.

Velice zajímavých benefitem deskových her je, že dokáže zabránit nebo alespoň zpomalit spoustu vážných nemocích, jako jsou Alzheimerova choroba a demence. Zapojení mozku při hře, trénuje části mozku, které mají na starosti kognitivní funkce. Tyto funkce se zvyšujícím věkem zhoršují a je potřeba je trénovat. [5]

## Historie pexesa

Desková hra pexeso je už populární v Česku a na Slovensku už více než padesát let. Jejímu vzniku vděčíme už zesnulému Zdeňkovi Princovi, který údajně dostal nápad na sestrojení této legendární hry přímo v katedrále sv. Víta na Pražském hradě.

Autor se v této době podílel na tvoření mozaiky Kristova křtu, na které spolupracoval s mnoha malíři a sochaři. V tomto kreativním prostředí se mu dostalo i spoustu času na jeho projekt, který měl nejdříve nést název po jeho projektu – obrázková mozaika. [6]

Zdeněk Princ nastoupil do nakladatelství Pressfoto, kde mu byla vystavena nabídka vytvoření hry, kterou měl zvětšit výnosy a činnost nakladatelství. Nakladatelství se nápad obrázkové mozaiky složené ze třiceti dvou párů obrázků líbil, ale nelíbil se mu název. Zdeněk tedy zkrátil název Pekelně se soustřeď pomocí akronymů, na námi už známe Pexeso, se kterým už vedení nakladatelství bylo už spokojené. Autor se inspiroval československou televizí, kde se spojení Pekelně se soustřeď objevil jako název televizní soutěže.

První vydání deskové hry mělo mezi lidmi obrovský úspěch. První vydání mělo ilustraci hrdinů z filmů o Vinnetouovi, slavného náčelníka Apačů. Kombinace jednoduchých pravidel a ilustrace slavných hrdinů určitě pomohla slávě a popularitě hry mezi všemi věkovými skupinami. Hra se brzy stala nutnou součástí každé domácnosti. [6]

## Pravidla hry pexeso

Hra pexeso je převážně doporučena pro dva hráče, ale může ji člověk hrát sám nebo třeba až se šesti hráči. Proto je hra skvělá aktivita pro volný čas s přáteli, rodinnou anebo dětmi.

Hrací pole tvoří sudý počet párů karet, se stejným obrázkem, základní počet karet bývá třicet dva, ale variací je více. Hráč, který je na řadě otočí dvě karty lícem nahoru podle jeho volby. Pokud jsou karty stejné, karty si odebere k sobě a dostává bod. Pokud karty nejsou stejné, karty otáčí zpátky lícem dolu. Hráč s nejvíce body získává titul výherce.

## Téma mého pexesa

Téma pexesa, kterého můžete zakoupit v hračkářství nebo v jakémkoliv jiném obchodě se velice liší. Témat je spousty, přes barvy, pohádkové bytosti a spoustu mnoho. Tématem toho pexesa je ovoce.

Návyky, na které si zvykneme v dětství, nám v dospělosti setrvávají, a to platí i o stravování. V dnešní době je velice těžké se stravovat správně a vyhýbat se nezdravému jídlu.

Ovoce patří do kategorie zdravého jídla a obsahuje spoustu vitamínů, vody a vlákniny, které v dnešní průměrné stravě chybí. Ovoce taky pomáhá s hubnutím a se předstupem obezity, kvůli jeho malému počtu kalorií. [] https://nutritionsource.hsph.harvard.edu/what-should-you-eat/vegetables-and-fruits/

Spoustu z lidí má problém dodržovat doporučenou dávku denního příjmu ovoce, což je přibližně 400 gramů. [] https://www.ferpotravina.cz/clanky/doporucena-denni-davka-ovoce-a-zeleniny

Existuje studie, kterou provedli nizozemští vědci, která zjistila, že děti po hraní paměťové hry s ovocem, drasticky zvýšili svůj příjem a chuť k němu. .[7]

Přesně proto, tématem této hry je ovoce. Možná někomu zlepší aspoň trochu stravovací návyky.

## Microsoft

Microsoft Corporation je technologicky zaměřená americká společnost, která se zabývá vývojem softwaru, služeb a hardwaru. Právě technologie, které jsou využity ve vypracování této práce, jsou vyvinuty touto společností.

Společnost byla založena dvěma kamarády ze stření školy, světoznámým Bill Gatesem a Paul Allenem. Oba byli zarytí fanoušci počítačů už na střední škole, kde nechodili na vyučovací hodiny, jen aby mohli strávit více času v počítačové místnosti. Dokonce se nabourali do počítačového systému, za tento čin překvapivě nebyli nějak potrestání, ale bylo jim výměnou za jeho zdokonalení nabídnuto více na času v počítačové učebně. Už tehdy založili svoji první malou společnost Traf-O-Data, která se zabývala počítáním dopravy. [8]

V roce 1975 idea osobního počítače v každé domácnosti byla pouhým snem. Tento sen se pomalým tempem stával skutečností. Společnost MIT vydala jeden z prvních mikropočítačů Altair 8800, který byl dostupný pro osobní použití. Ovšem Gates a Allen si tohoto počítače byli vědomi, ale mysleli o něm své. Mysleli si totiž, že počítači chyběl kvalitní software. []

<https://5minutehistory.com/the-history-of-microsoft/>

Domluvili se s výrobci mikropočítače a po dobu osmi týdnů vyvíjeli svoji implementaci programovacího jazyka BASIC. Tento vývoj vedl ke spolupráci se společností. Tento veliký úspěch je přesvědčil, aby skončili se studováním na univerzitě a odstěhovali se do Nového Mexika, kde sídlila firma MIT. Zde oficiálně založili, dnes už světoznámou, firmu Microsoft. Na její název přišli ze spojení slov mikropočítač a software.[]

https://marketing4ecommerce.net/en/history-of-microsoft/

Od té doby společnost vyvinula nespočet důležitých technologií a produktů, které používají milióny lidí každý den na celém světě. Nejvýznamnější produktem je jejich operační systém Windows, který běží na sedmdesáti procentech všech počítačů na celém světě. []https://gs.statcounter.com/os-market-share/desktop/worldwide

## Visual Studio

Visual studio je ve vlastnictví společnosti Microsoft, která ho vyvinula a do dnešního dne i spravuje a aktualizuje. Patří k nejoblíbenějším vývojovým prostředím mezi programátory, a to díky své přehlednosti a variabilitě. V tomto prostředí lze vyvíjet spoustu různých druhu projektů jako jsou webové aplikace, konzolové aplikace, desktopové aplikace a spoustu mnoho dalších. Vývojové prostředí je možno používat na operačních systémech Windows a macOS. [9]

Ani možnost volby programovacího jazyka není omezená, studio podporuje oblíbené jazyky pro back-end i kompletní vývojáře. Mezi ně patří HTML, CSS, JavaScript, JSON, LESS, SASS, PHP, Python nebo jazyk C# s technologií ASP.NET. []

https://visualstudio.microsoft.com/cs/vs/features/web/languages/

Přesný počet podporovaných jazyků je třicet šest. []

<https://www.oxtrys.com/top-programming-languages-used-in-visual-studio>

Visual studio se dělí na tři edice. Při zpracování této maturitní práce byl použita verze Comunnity. Tato edice byla vydaná v roce 2014 a její největší výhodou je, že je zdarma, což u ostatních edic neplatí. Pro individuální vývojáře není nijak omezená, ale pro společnosti s výším počtem zaměstnanců a příjmů má určité omezení. Její hlavní funkcí je zprostředkovávání přistup k několika tisícům knihoven, rozšíření a plnou podporu k mnoha populárním jazykům. []

https://pangea.ai/resources/visual-studio-community-professional-or-enterprise

## .NET Framework

.NET Framework, vyvinut a spravován společností Microsoft, je softwarová běhová platforma určená pro budovaní webových, desktopových a mobilních aplikacích. Platformu tvoří dvě hlavní časti. Common Lanugage Runtime, který spravuje paměť a stará se o spuštěné aplikace. Další částí je knihovna tříd, který vývojářům poskytují spoustu opakovatelného kódu, které mohou zakomponovat do svých aplikací. []

https://learn.microsoft.com/cs-cz/dotnet/framework/get-started/#net-framework-for-developers

Největší výhodou .NET Frameworku je jeho podpora pro velikou škálu možností z programovacích jazyků. Vývojář si může vybrat mezi programovacími jazyky, tak aby jeho volba nejlépe seděla danému problému, který se snaží vyřešit. Nesejde na jeho volbě, ale stále bude moci využít stejné funkce a nástroje, které podporuje .NET Framework.

Výběr typů projektů není také velmi omezen, vývojář si může vybrat mezi různými typy, což je velikou výhodou pro vývojáře, které pracují na různých projektech. Framework také obsahuje podporu prvků, které zlepšují bezpečnost, výkon a spolehlivost aplikací. Jelikož je .NET Framework vyvíjen společností Microsoft, je navržen tak, aby byl kompatibilní s dalšími nástroji od společnosti. Jsou to nástroje jako SQL Server, SharePoint, Office.[10]

https://www.geeksforgeeks.org/introduction-to-net-framework/

## C#

C# je moderní programovací jazyk z rodiny jazyků C, což znamená je podobný ostatních jazyků z rodiny, jako jsou C++ a Java. Byl vyvinut společností Microsoft v roce 2000 a běží na platformě .NET Framework. C# je jedna z nejlepších voleb ze všech programovacích jazyků na celém světě, což také odpovídá jeho popularitě. Je ideální na vývoj Windows aplikací, vývoj her za pomocí Unity, mobilních aplikacích, webových aplikací a služeb. Od verze vydaní verze .NET Core, C# aplikace lze spustit na operačních systémech macOS a Linux. [11]

## Windows Forms

Windows formuláře neboli „WinForms“, jsou součástí knihovny pro tvorbu jednoduchého uživatelského rozhraní převážně pro stolní počítače. Windows formuláře poskytují přístup k velkému množství jednoduchých prvků rozhraní, které mohou vývojáři používat. Prvky jsou vytvořeny tak, aby vývojář jejich vzhled a chovaní mohl nastavit velice jednoduše pomocí vlastností a událostí, Visual Studio poskytuje pro tuto knihovnu vizuálního návrháře, který vývoj zlehčuje ještě více kvůli své jednoduchosti a uživatelsky přívětivému rozhraní. Mezi základní prvky jsou například tlačítka, textová okna, list boxy a mnoho dalších. [12]

## GitHub a Git

Git je verzovací systém, který zaznamenává změny v složkách. Je převážně používán při práci v týmu, a to zejména vývojářském. Umožňuje více vývojářům pracovat na stejných souborech najednou. A to tím způsobem, že si vývojář udělá pull, kopii z hlavního uložiště, kterému se říká master branch, na své lokální uložiště. Následně provede změny na svém lokálním uložišti a ty vloží do hlavního, tomu se říká commit. Díky Gitu je velice jednoduché změny provádět a případně vrátit.

https://about.gitlab.com/topics/version-control/what-is-git-version-control/

GitHub je cloudová platforma a její hlavní funkcí je sdílení a ukládaní projektů. Projekty se ukládají v repositářích, což umožňuje několik výhod. Lze si tímto způsobem vytvořit portfolio, a předvést ostatním svoji práci. Velice jednoduše se orientuje a spravuje kód. Další vývojáři mohou zhodnotit projekty ostatních a navrhnout nápady na zlepšení. A nejdůležitější výhodou je, práce na společném projektu, bez obavy zničení práce ostatních.[13]

# Praktická část

Cílem části této dokumentace je přiblížit praktickou strukturu a implementaci hry Pexesa. Projekt je rozdělen do několika částí, a to kvůli přehlednosti a omezení duplicitě kódu. Samotný kód je rozdělen do osmi tříd a čtyř formulářů. Velikou výhodou toho rozdělení je, že při nefunkčnosti, opravě či optimalizaci jedné částí kódu bude zbytek projektu fungovat.

Obsah obrázku text, snímek obrazovky, software, Počítačová ikona

Popis byl vytvořen automaticky

## Třídy

Každá třída nese název podle toho, co k formulářům přidává. Například třída, která je správcem herního pole, nese jméno spojené s ním. Třídy rozšiřují formuláře o svůj kód a zároveň zanechávají čistotu kódu. Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

## Třída GameBoard

Třída GameBoard má na starosti inicializaci herního a jeho otáčení karet za průběhy hry.

### Konstruktor herního pole

Díky konstruktoru se předávají dané proměnné, která třídy potřebuje. Konstruktor si vyžaduje celočíselný počet karet, který si hráč vybere v nastavení hry. Toto číslo určuje velikost hracího pole a jeho množství karet. Další vyžádanou proměnnou je TableLayoutPanel, který je součástí knihovny Windows Formulářů. Je to prvek, který jeho obsah uchovává v řádcích a sloupcích. Jeho výhodou je, že se může počet mřížek dynamicky měnit a není pevně daný. []<https://learn.microsoft.com/cs-cz/dotnet/desktop/winforms/controls/tablelayoutpanel-control-windows-forms?view=netframeworkdesktop-4.8>.

### LoadImages

LoadImages je prostředníkem mezi herním polem a soubory hry s obrázky.

Vytvoří se nový List obrázků, do jenž se budou ukládat obrázky karet. Nastaví se proměnná, která určuje id obrázku karty otočené lícem dolů na mínus jedna. Hra pomocí třídy Assembly projede všechny soubory, které jsou uloženy v předem staticky nastavené složce projektu.

Vybere jenom soubory s příponou png, což jsou obrázky karet. Každý obrázek se otevře pomocí proudu dat Stream. Pokud se čtení proudu povedlo a obsahuje správný soubor, tak se zkontroluje, zda obrázek není zadní strana karty. Pokud je, tak se obrázek uloží do globální proměnné backImage. Zdali je tomu naopak, tak se uloží do seznamu obrázků.

Zároveň, při ukládání do seznamu, se uloží číslo do dalšího seznamu s názvem cardImagesIds. Tento seznam obsahuje čísla, která představují odkaz na položky v seznamu obrázků. Každý obrázek má svoje identifikační číslo. V logice hry se bude pracovat právě s tímto listem čísel, kvůli rychlejšímu běhu kódu než při verzi s obrázky. Metoda je ošetřena podmínkou, zda je počet obrázků menší než osmnáct, což je maximální počet obrázků. Podmínka zjistí, zda se načetly všechny obrázky.

Obsah obrázku ovoce, snímek obrazovky, Multimediální software, Grafický software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### SetupLayout

Tato metoda nastavuje už zmíněný mřížkovaný prvek TableLayoutPanel, který představuje herní pole. Nastaví se počet sloupců a řádků na převzatý počet karet z konstruktoru třídy. Následně se vymaže veškerý obsah a styly pole, což umožní cyklem for nastylovat prvek. Každý sloupec a řádek dostane procentuální velikost ze celkové velikosti panelu. Například pokud má mřížka 4 sloupce, tak každý sloupec bude mít 25% celkové velikosti pole.

### CreateCardLabel

Cílem této metody je vytvořit novou kartu neboli prvek Label. Metoda přijímá parametr celého čísla tagValue, ten se uloží do vlastnosti prvku jménem Tag. Vlastnost umožňuje velice jednoduše rozpoznat karty od sebe, každá má totiž Tag jiný.

Obrázek Labelu se ze začátku hry nastaví na obrázek zadní strany karty, což vytváří efekt otočení všech karet na začátku hry. Také se určí jeho velikost, styly a vyplnění celého prostoru mřížky. Připojí se událost Click, což znamená, že hráč kliknutím na kartu spustí část v této události. Událost vyvolává asynchronního delegáta CardClicked, který svoji implementaci má ve třídě s herní logikou. Následně metoda vrací vytvořenou kartu.

### PlaceCards

Metoda PlaceCards rozmístí vytvořené prvky Labelů po hracím poli. Pracuje se s parametrem isLoading, který značí, zda se načítá už uložené hra, nebo se tvoří nová. Podle hodnoty v parametru se rozmístí karty dvěma různými způsoby.

V obou případech se nejdříve inicializuje nový seznam hiddenLabels, který je pro uložení hodnoty Tagů karet, která ještě nebyly otočeny. S tímto seznamem se pracuje v herní logice.

Pokud se načítá už uložená hra, tak se cyklem for projedou všechny mřížky hracího pole a do každého se vytvoří nová karta s celočíselnou hodnotou Tag nula. Následně se karta přidá do hracího pole a její Tag se uloží do seznamu neotočených karet.

V případě, že se načítá hra nová, tak je logika odlišná. Nejdříve se vytvoří nový pomocný List icons pro uložení celých čísel. Následně cyklem for, ve kterém je deklarována proměnná na celé číslo nula, projedeme polovinu všech vytvořených mřížek v hracím poli. Při každé mřížce přidáme do seznamu dvě hodnoty. Hodnotu celého čísla zvětšenou o jedničku a znovu tuto hodnotu, ale vynásobenou číslem 100.

Hodnoty v seznamu reprezentují hodnoty, které budou uložené v Tagu karet. Na celém hracím poli budou vždy dvě karty, které k sobě patří a tvoří pár. Pár se pozná podle toho, že právě jedna karta v Tagu bude mít číslo a druhá jeho násobek 100.

Čísla obou karet nemůžou být stejná, protože by to narušovalo zbytek logiky hry. Zároveň číslo sto je nejmenší možné číslo k násobení, protože se počet karet pohybuje v desítkách.

Pomocí třídy Random a cyklu while, se hodnoty v seznamu přeházejí a vytvoří se tak efekt zamíchání. Cyklus for vytvoří ze zamíchaného seznamu pomocí metody CreateCardLabel nové karty, která přidá do hracího pole a jejich hodnotu Tag do seznamu neotočených karet.

### InitializeBoard

Funkcí této metody, je zavolání ostatních metod inicializace herního pole. Místo toho, aby ve hlavním formuláři hry byly volány všechny metody samostatně, tak se zavolá tato metoda, která vyvolá ostatní. Jsou to LoadImages, SetupLayout a PlaceCards s parametrem boolean IsLoading. Také nastavuje správné odsazení mřížky pole.

### GetCardImage

Vrácenou hodnotou této metody je její obrázek. Volá se v případě, že je potřeba otočit kartu obrázkem nahoru. Pracuje s celočíselnou hodnotou, která reprezentuje identifikační číslo obrázku. Pokud je větší než 100, je číslem 100 vyděleno. Metoda je ošetřena podmínkou, zdali je větší nebo rovno nula a zároveň menší, než počet obrázků. Podmínka zajišťuje, že vstup je validní. Metoda následně vrátí obrázek ze seznamu, který odpovídá číslu.

### FlipCardFront

Při otočení karty, je potřeba zobrazit její obrázek. To je funkcí této metody, která pracuje s prvekem Label. Zkontroluje se, zda uložená hodnota v Tagu je celé číslo a poté je nastaven obrázek karty pomocí metody GetCardImage. Které se hodnota předá.

### GetBackImage

Vracenou hodnotou této metody je obrázek zadní strany.

### FlipCardBack

Metoda je volána při špatném přiřazení dvou karet a potřebě karty otočit zpět na zadní stranu. Pomocí GetBackImage se nastaví obrázek prvku Label.

## Třída GameLogic

Třída GameLogic spravuje veškerou herní logiku projektu. Zpracovává tah hráče, počítače a zároveň zjišťuje vítěze. V této třídě jsou velice důležité Listy flippedLabels a hiddenLabels, na kterých stojí veškerá logika hry počítače. V těchto seznamech jsou uloženy celá čísla, která představují identifikační číslo obrázků.

### Konstruktor logiky hry

Konstruktor nastavuje dané proměnné, se kterými třída pracuje. Tyto proměnné spravují herní prostředí a pravidla. Jsou to objekt gameBoard, který odkazuje na správu herní plochy. Instance správce skóre, který nastavuje a aktualizuje skóre hráčů. Konstruktor také vyžaduje počet hráčů a karet, zdali je zapnuta hra s počítačem, jeho případná obtížnost a nastavení zvuku. Nastavení zvuku spravuje objekt soundManager. Inicializuje se také slovník alreadyFlipped, který zaznamenává pomocí datového typu boolean a jejich id, jaké karty byly otočeny. Při začátku hry jsou hodnoty ve slovníku nastaveny na false.

### Enum GameState

Enum určuje různé stavy hry. Každý stav hry určuje specifickou fázi tahu, což zjednodušuje herní logiku.

Při čekání na tah, je enum nastaven na Idle. Otočení jedné z karet změní stav na OneCardFlipped. Správné kliknutí na druhou kartu hráčem nastaví enum na Processing, při tomto stavu je herní pole zablokováno, aby se nenarušila herní logika. Stav ProcessingForComputer zablokovává hrací pole pro hráče při tahu počítače, aby hráč nemohl pokazit jeho tah.

### OnCardClicked

OnCardClicked je asynchronní událost, která zajišťuje správu kliknutí hráče na karty. Spravuje také logiku otáčení karet, porovnávání párů, aktualizaci skóre a stav enumu gameState. Událost pracuje se třemi Labely. ClickedLabel, což je karta, na kterou hráč klikl. A s Labely first a second, kterým se při správném kliknutí přiřadí kliknutý Label.

Při kliknutí na hrací pole je nejdříve ošetřen vstup. Ten je ošetřen několika podmínkami. Vstup je zamítnut v několika situacích. Pokud gameState je nastaven na Processing nebo ProcessingForComputer, neboli se zpracovává jiný tah. Zdali má kliknutá karta hodnotu null. Nebo v případě, že hodnota Tagu kliknuté karty je nastavena na backImageId. Také když gameState je nastaven na stav otočené jedné karty a kliknutá karta je rovna otočené kartě.

Hráč klikl na kartu a vstup je validní. Zdali je gameState nastaven na Idle, znamená to, že otočená karta je první v tomto tahu. GameState se nastaví na oneCardFlipped a začne zpracovávání logiky.

Do pomocné proměnné first se uloží clickLabel. Objekt gameBoard otočí metodou FlipCardFront kartu obrázkem vzhůru. V případě zapnutého zvuku soundManager zavolá metodu PlayFlipCardSound. Následně se pomocí indexu Labelu a seznamu alreadyFlipped zjistí, zda karta byla už otočena. Pokud nebyla, přidá se její hodnota Tag do seznamu již otočených karet flippedLabels. Hodnota Tagu se také odebere ze seznamu ještě neotočených karet hiddenLabels. Metoda se asynchronně pozastaví jednu vteřinu, pro vytvoření dramatického efektu otáčení karty a vrátí se.

V případě, že gameState je nastaven na OneCardFlipped a hráč klikne na další kartu, začne zpracovávání logiky druhé karty. GameState se nastaví na Processing a pomocné proměnné se přiřadí kliknutá karta. Následuje stejná logika, jako při otočení první karty.

Zjištění, zda otočených karet tvoří pár stejných karet se provádí porovnáváním jejich hodnot v Tagů. Pokud jeden z Tagů vynásobený 100 je roven druhému, karty tvoří pár.

Jsou-li zapnuté zvuky, jejich správce soundManager spustí pomocí metody PlayMatchedCorrect zvuk správně přiřazených karet. Správce skóre zvedne hráči, který byl na řade skóre o 1 bod díky AddScore metodě a akci ScoreUpdated. Z Listu flippedLabels se odeberou Tagy správně otočených karet, aby nenarušovaly zbytek hry. Do obou Tagu Labelů se nastaví hodnota backImageId, tímto jsou karty mimo hru a už s nimi nejde hrát. Pomocné proměnné se vynulují, aby byly připravené na další tah. Zkontroluje se, zda hra nemá vítěze, zavoláním metody WinnerCheck a gameState se nastaví na stav Idle.

V případě, že karty netvoří stejný pár je na řadě jiná logika. Jsou-li zvuky zapnuté spustí soundManager zvuk metodou PlayMatchedWrong. Metoda se asynchronně pozastaví pro vizuální efekt na jednu vteřinu. Proměnná currentPlayer, která udává hráče, který je na řadě přepne svoji hodnotu na dalšího hráče. Spustí se akce ScoreUpdated a metoda OnTimerTick.

### OnTimerTick

Metoda OnTimerTick spravuje otáčení, vynulování proměnných a přepínání hráčů zároveň s vteřinou odezvou. Po zkontrolování platnosti Tagů v podmínce, se karty otočí zadní stranou nahoru. To provede objekt gameBoard za pomocí metody FlipCardBack. Následně se vynulují pomocné proměnné first a second, aby byly připravené pro další tah. GameState se nastaví na Idle, ze stejného důvodu. Pokud je zapnuta hra s počítačem a v pořadí je hráč druhý, je na řade počítač. Tah počítače se zpracovává v metodě ComputerTurn.

### GetRight

Tah počítače je závislý na procentuální šanci, tato šance je předem určena hráčem, který si ji v nastavení hry vybral. Metoda obsahuje switch, který vrací tuto šanci v podobě celého čísla. Celé číslo představuje procentuální šanci, že počítač nebude otáčet karty náhodně, ale pokusí se najít už otočený pár.

Hráč si vybral mezi možnostmi 1,2 a 3, které představují dané obtížnosti. Pokud hráč zvolil možnost 1, vrátí mu switch celé číslo 30. Možnost 2 představuje číslo 60 a možnost 3 představuje možnost číslo 100.

### ComputerTurn

Asynchronní metoda ComputerTurn zajištuje logiku tahu počítače, pracuje s celými čísly uloženy v Tagu Labelů. Nepracuje přímo s obrázky v Labelech, protože řešení s čísly je daleko rychlejší a optimalizovanější.

Tah začíná nastavením gameState na ProcessingForComputer, kvůli zablokování interakce hráče s hracím polem. Poté se asynchronně zastaví na jednu vteřinu pro vytvoření efektu, že počítač „přemýšlí“.

Následně se inicializují dvě proměnné indexFirstLabel a indexSecondLabel. Do těchto proměnných se uloží hodnota Tagu vybraných karet. Také se určí, zda počítač nevybere karty náhodně, to díky proměnné chance, která uchovává datový typ bool. Do chance se uloží hodnota z podmínky, která zjišťuje, zda hodnota z metody GetRight je větší nebo rovno náhodně vygenerovanému číslu pomocí třídy Random z intervalu 0 až 100.

Pokud chance je pravda, tak počítač zkusí najít pomocí dvou for cyklů dvě čísla, které tvoří pár karet. Počítač při srovnávání prochází pouze List už otočených karet flippedLabels. Zdali najde pár uložení si jejich hodnoty do pomocných proměnných a cyklus předběžně ukončí pomocnou proměnnou.

V případě, že pár stejných karet nenašel, nebo chance nebyla pravda, vybere první index z ještě neotočených karet hiddenLabels za pomocí třídy Random. Také ji odebere z tohoto listu a přidá do již otočených karet.

V situaci, že chance je pravda a první číslo už bylo náhodně vybráno, projede počítač znova cyklem for List už otočených karet. Najde-li kartu, která tvoří pár, zapamatuje si ji do druhé pomocné proměnné indexSecondLabel.

Nastane-li, že chance není pravda a bylo vybráno jenom první číslo, tak se za pomocí třídy Random vybere druhé. To se následovně odebere z Listu neotočených karet a přidá se do Listu otočených. V případě, že v listu neotočených karet není už žádná hodnota, vybere se náhodná hodnota z listu už otočených.

Následně se asynchronně metoda pozastaví pro dramatický efekt výběru karet. Inicializují se dvě nove pomocné proměnné firstLabel a secondLabel, datového typu Label. Cyklem foreach se zkontrolují všechny Labely uvnitř TableLayoutPanelu. Pokud hodnota Tagu uvnitř Labelu je rovna hodnota v pomocné proměnné indexFirstLabel nebo indexSecondLabel, tak se Label přiřadí do proměnné firstLabel nebo SecondLabel.

Objekt gameBoard otočí metodou FlipCardFront obrázkem vzhůru a pokud je hra se zvuky, tak jejich správce zavolá metodu PlayFlipCardSound. Mezi otočením obou karet je asynchronní pozastavení jedna vteřina pro dramatický efekt výběru.

Zjištění, zda karty tvoří stejný pár probíhá porovnáváním pomocných proměnných celých čísel, a ne samotných Labelů. Pokud jedno číslo je rovno druhému násobenému 100, je to pár stejných karet.

V případě, že karty tvoří stejný pár, odeberou se z jejich hodnoty Listu již otočených karet flippedLabels. Ve hře se zvukem se zavolá metoda PlayMatchedCorrect. Správce skóre přidá počítači jeden bod a zavolá akci ScoreUpdated. Vybraným Labelům se nastaví do hodnoty Tag backImageId, aby na ně hráč nemohl už kliknout a byly mimo hru. Následně se zavolá metoda WinnerCheck, aby zkontrolovala vítěze, poté hraje počítač znovu.

Pokud karty netvoří pár stejných karet, tak se díky objektu gameBoard a metodě FlipCardBack otočí zpátky zadní stranou nahoru. V hře se zvuky se zavolá metoda PlayMatchedWrong. Proměnná currentPlayer, která udává, jaký hráč je na řadě, přepne svoji hodnotu na dalšího hráče. GameState se přepne na Idel, aby mohl hrát další hráč.

### WinnerCheck

Po správném otočení dvou karet se vždy zavolá tato metoda, která zjistí, zda není na hracím poli už vše správně otočeno. To zjistí za pomocí součtu počtu karet v Listu neotočených i otočených. Pokud toto číslo je rovno nule, hra skončí vyvoláním metody ze správce skóre EndScore a akce GameEnded.

## Třída GameScoreManager

Třída GameScoreManager spravuje, i aktualizuje skóre v průběhu hry a vypisuje konečné skóre. Zajištuje, aby správný hráč dostal body, které si zaslouží.

Třída pracuje s dvěma poli, pole jmen hráčů a jejich skóre. Na každém indexu v poli jen je jeden hráč a na stejném indexu v druhém poli je jeho skóre. Jelikož známe předem počet hráčů je pro jejich uložení rychlejší využít pole, než List či jiné kolekce.

### Add, Get a Set

V třídě existuje několik metod, které nastavují nebo vrací skóre nebo jméno. Pracují s indexem, který odkazuje na hráče v poli jmen a skóre. Jsou ošetřeny podmínkou, zda index je validní. Například AddScore přidává hráči na určitém indexu v poli skóre 1 bod. Nemá smysl všechny zde vypisovat, jelikož jsou si téměř totožné.

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### GetSortedScores a EndScore

Tyto dvě metody zaručují v čase, kdy na herním poli jsou správně otočené všechny karty, výstupní okno s finálním skóre. GetSortedScores skóre a jména správně přiřadí a seřadí od největšího počtu bodů. EndScore seřazená jména vypíše pomocí MessageBoxu.

Výsledné okno je důležité, proto aby hráči věděli svůj konečný počet bodů.

## Třída SoundManager

Třída SoungManager slouží jako správce zvukových efektů ve hře. Načitá zvuky ze souborů hry a má na starosti jejich přehrávání a uvolnění po skončení hry.

### Konstruktor

Konstruktor pracuje s parametrem bool isSound, který určuje, zda si hráč přeje mít zvuky povolené. Pokud ano, tak inicializuje přehrávač zvuku (WaveOutEvent) a slovník audioStreams, ve kterých zaznamenává uložené zvuky. Následně se zavolá metoda LoadAudio, která načte zvuky ze souborů hry.

### LoadAudio

Metoda LoadAudio prochází soubory hry, konktrétně složku Root, která obsahuje zvukové efekty s příponou mp3. Správně nalezené soubory uloží do slovníku jako paměťový proud (MemoryStream) s klíčem, jenž je název souboru. Procházení souboru je ošetřeno výjimkou.

### PlayAudio

Metoda pracuje s názvem zvuku, která následně přehraje. Nejdříve pomocí metody StopAudio zastaví případně už přehrávající zvuk a poté Mp3FileReaderem načte paměťový proud ze slovníku. Načtený zvuk přehraje pomocí přehrávače WaveOutEvent.

### StopAudio a Dispose

StopAudio zastaví přehrávání a uvolní data spojené se čtením zvuků. To provede metodou Dispose, která zastaví zvuky a vyčistí kompletně všechny proudy.

### Play Metody

Pro usnadnění volání zvuků v logice hry, jsou vytvořeny tři metody, PlayFlipCardSound, PlayMatchedCorrect, PlayMatchedWrong. Uvnitř nich je volána metoda PlaySound, která přehraje staticky název zvuku, který ladí s názvem metody.

## Třída GameSave

Třída GameSave obsahuje všechny důležité proměnné a informace o stavu hry, kterou chce hráč uložit. Atribut Serializable usnadňuje jednoduché uložení do libovolného soubory a jeho následné otevření (Deserializace).

Obsah obrázku text, snímek obrazovky, software

Obsah vygenerovaný umělou inteligencí může být nesprávný.

## Třída GameSaveManager

Statická třída GameSaveManager má dva hlavní účely, uložení instanci třídy GameSave do binárního souboru a jeho následné načtení do hry. Je zvolené uložení do binárního souboru, protože běžný uživatel bude mít problém data upravit, jelikož jsou pro něj nečitelné. Všechny uložené soubory v tomto projektu používají právě binární soubory, právě z tohoto důvodu.

### SaveGame a LoadGame

Metoda SaveGame pracuje s herními daty z rozehrané hry a s cestou k souboru, kam chce hráč hru uložit. Třída FileStream s módem Create využije BinaryFormatter a do zvolené lokace převede do binárních dat herní data.

Metoda LoadGame dělá přesný opak. Na zvolené lokaci pomocí třídy FileStream s módem Open a využitím BinaryFormatter deserializuje binární data a převede je na použitelná herní data. Obě metody jsou ošetřeny výjimkou v případě, že uživatel soubory poškodil. Všechna logika s binárními soubory je v tomto projektu ošetřena přesně z toho důvodu.

## Třída ScoreData

Třída ScoreData obsahuje všechny důležitá data o hráči, který dohrál hru. Mezi data patří jeho jméno, počet výher, počet proher, počet nalezených karet a celkový počet karet v jeho hrách. Tyto informace budou uloženy a použity do tabulky skóre. Třída obsahuje atribut Serializable.

## Třída GameScoreSaveManager

Statická třída GameScoreSaveManager spravuje data instance třídy ScoreData díky binární serializaci. Může je uložit, načíst a mazat.

Metoda SaveScoreData pracuje s Listem datového typu ScoreData. Uloží data z Listu do souboru pomocí třídy FileStream a BinaryFormatteru. Data jsou uložena v binární podobě.

Metoda LoadScoreData je přesným opakem a převede data z binární podoby do Listu, tak aby byly použitelné v další implementaci.

Metoda ClearScoreData zkontroluje, zda binární soubor s daty existuje a případně ho vymaže za pomocí třídy File metody Delete.

## Formuláře

Formuláře představují uživatelské rozhraní, jejích cílem je dosáhnout jednoduchou orientaci ve hře, ale s přívětivým zážitkem. Proto jsou části hry, které jsou na to vhodné, rozděleny do více nemodálních formulářů. Takové části jsou čtyři. Každá část má svůj designer soubor a třídu formuláře. Designer soubor tvoří vzhled formuláře, ten byl tvořen za pomocí Drag and Drop způsobu (Grafické rozhraní návrháře), nikoliv za čistého kódování. Třida formuláře dědí z třídy Form, zde se určuje logika, která řídí, co formulář děla a jak reaguje.

## StartingMenu

Formulář StartingMenu je první okno, které hráč při spuštění uvidí. Slouží jako křižovatka mezi ostatními částmi hry. Mezi nimi se přechazí za pomocí tlačítek.

Obsah obrázku text, ovoce, jahoda, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Obsahuje proměnné, které mění nastavení nově spuštěné hry za pomocí tlačítka Nová hra. Tyto proměnné se dají změnit v nastavení. V tomto formuláři jsou nastaveny na počáteční hodnotu, to aby hráč nemusel vždy měnit. Obsah obrázku text, snímek obrazovky, Písmo

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Z formuláře lze také otevřít uloženou hru z datového souboru. Hráč soubor vybere pomocí OpenFileDialogu a hru načte do nového formuláře NewGame pomocí třídy GameSaveManager a její metody LoadGame. Tato metoda soubor otevře a převede z datového souboru do použitelných dat. Zbylou logiku načtení hry obsahuje formulář NewGame.

## GameSettings

Formulář GameSettings pracuje s proměnnými, které předává hlavnímu menu. Hráč může zde měnit nastavení samotné hry a následně se může pomocí tlačítka vrátit zpět.

Obsah obrázku text, ovoce, snímek obrazovky, jahoda

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Platný počet hráčů je ošetřen za pomocí podmínky. Pokud počet hráčů není platný, tlačítko nepustí hráče do menu a vyhodí upozornění v podobě MessageBoxu.

Zvolení velikosti hracího pole a obtížnosti počítače jsem zvolil za pomocí RadioButtonů. Ty jsou odděleny do dvou skupin GroupBoxem, proto lze vybrat vždy jenom jednu možnost. Což je účelem, protože nelze mít více možností velikosti či obtížnosti naráz.

## NewGame

Ve formuláři NewGame se nachází hrací pole. Zde hráč „hraje“ samotnou hru. Obsah obrázku text, snímek obrazovky, kreslené, diagram

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### Konstruktor

Konstruktor formuláře nastavuje proměnné, které získává z hlavního menu. Vytvoří se nová instance třídy GameBoard, která metodou InitializeBoard nastaví herní pole. Pokud je hra nová a nenačítá se z uloženého souboru, tak se zavolá metoda GetNames. A to proto, protože v případě načítaní už uložené hry, jsou jména už zadaná.

Vzniknou nové instance tříd ScoreManager a GameLogic, které budou ve formuláři spravovat logiku hry a skóre.

Konstruktor také zajištuje kliknutí na karty. Když hráč klikne na kartu spustí se Event Handler. Pokud je prvek v Event Handleru kartou, zavolá se asynchronní metoda OnCardClicked ze třídy GameLogic. Metoda zpracuje logiku tahu hráče. Po zpracování metody se zavolá metoda ShowScore, která případně aktualizuje skóre.

Konstruktor také připojuje k událostem ze třídy GameLogic, ScoreUpdated a GameEnded, metody z formuláře NewGame. To zajistí správné oddělení a volání kódu, která pracuje s prvky Windows Formulářů.

### GetNames

Metoda GetNames získává jména hráčů, se kterými pracují ostatní třídy. Při zapnutí nové hry vyskočí na hráče formulář, kde může jména zadat. Pokud žádné jméno nezadá, nastaví se jak jeho pořadí. Například hráč, který je v pořadí třetí, bude mít jméno 3.

V hře, kde je jeden z hráčů počítač, je vždy jméno druhého hráče PC, protože na tomto místě v pořadí bude hrát.

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

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### ShowScore

Tato metoda spravuje zobrazování jméno hráče, jeho pořadí a skóre v StatusStripu, který se nachází pod hracím polem. Informace o hráči získává za pomocí třídy GameLogic a ScoreManager.

### SaveGame

Metoda SaveGame se spustí po kliknutí tlačítko, které představuje zelenou šipku ve ToolStripu. Pomocí SaveFileDialogu hráč vybere, kam chce rozehranou hru uložit, pokud hru uloží správně, začne zpracovávání uložení hry.

Do Listu celých čísel cardPositions se uloží všechny Tagy (identifikační čísla obrázků) Labelů (hracích karet) v TableLayoutu (hracím poli). Do proměnné indexFirstFlipped a indexSecondFlipped se uloží indexy otočených karet, které se ještě nestihly zpracovat. Hráč může totiž uložit hru v půlce tahu. Následně se vytvoří nová instance proměnné třídy GameSave, která obsahuje všechny důležitá data o hře. Obsah obrázku text, snímek obrazovky

Obsah vygenerovaný umělou inteligencí může být nesprávný.

Zavoláním metody SaveGame v třídě GameSaveManageru se data na vybrané místo uloží.

### LoadGame

LoadGame dělá přesný opak jako předchozí metoda. Při správném vybrání souboru hry OpenFileDialogem, začne logiku načítání uložené hry.

Vytvoří se nová instance třídy GameSave, do které se nahrají data pomocí třídy GameSaveManager a její metody LoadGame. Následuje předání dat z proměnné do aktuálního okna NewGame.

Důležité proměnné, které hráč může vybírat v nastavení se předají jako první. Instance ScoreManager nastaví správně jména a skóre hráčů. Správce herního pole třídy GameBoard zavolá inicializaci pole. Po správné určení velikosti TableLayoutPanelu (herního pole) se cyklem for nahrají z cardPositions správné Tagy (id obrázků) Labelů (karet).

Třída GameBoard obsahuje také proměnnou MatchedPairs, což je slovník. Tento slovník pracuje s klíčem datového typu integer a hodnotou stejného typu. Slouží k zapamatování indexu v hracím poli a Tagu (id obrázku) správně otočených karet. Při každém správném otočení se informace o kartách přiřadily do slovníku.

Cyklem foreach se projede každá informace o kartě ve slovníku, pokud jsou informace platné, do indexu v hracím poli se přiřadí správný Tag. Správce herního pole kartu otočí a Tag následně změní na backImageId (id zadní karty obrázku), proto aby na kartu nešlo kliknout.

V případě, že uložená proměnná IndexFirstFlipped a IndexSecondFlipped v sobě mají platnou hodnotu, tak se v hracím poli otočí karta na jejich indexu. To zajistí, že karty otočené v průběhu uložení zůstanou v tahu.

Nakonec se nastaví proměnná currentPlayer, flippedLabels, hiddenLabels a gameState. Zavolá se i metoda ShowScore, která aktualizuje StatusStrip pod hracím polem.

Celá metoda je ošetřena podmínkou, protože hráč mohl poměnit datový soubor.

### RestoreFromGameSave

Metoda obsahuje stejnou logiku jako LoadGame, akorát pracuje se souborem, který se vybere a předává z hlavního menu. Metody jsou oddělené právě proto, aby hráč nemusel začít hrát novou hru vždy, kdy chce otevřít už hru uloženou.

### EndGame

EndGame má na starost zpracování dat o hráčích a jejich následné předání do tabulky skóre. Metoda se volá na konci hry.

Cyklem for projde všechny hráče a najde toho, kdo má nejvíce (vítěz) a nejméně (poražený) bodů.

Do Listu datového typu třídy ScoreData nahraje všechny uložené informace o hráčích v datovém souboru. To provede třídou GameScoreSaveManager a metodou LoadScoreData. Následně projede opět všechny hráče, ze odehrané hry. Pokud v listu neexistuje záznam se stejným jménem, tak se vytvoří nová instance ScoreData a ta se přidá do Listu. V opačném případě se do Listu jenom přičtou hodnoty.

List se uloží do datového souboru třídou GameScoreSaveManager a její metodou SaveScoreData. Při správném uložení vyskočí potvrzení ve formě MessageBoxu. Na úplný konec této metody se otevře formulář Score, kde je tabulka se všemi záznamy hráčů. Formulář NewGame se zavře.

## Score

Formulář Score zobrazuje tabulku skóre, která obsahuje všechny informace o hráčích z datového souboru. Okno se zobrazí po každé dohrané hře, ale lze i na něj přistoupit z hlavního menu. Hlavní proměnnou, se kterou se pracuje je List s datovým typem ScoreData. Tento List uchovává veškeré informace, které se následně zobrazí hráči v DataGridView.

Obsah obrázku text, snímek obrazovky, diagram, design

Obsah vygenerovaný umělou inteligencí může být nesprávný.

### Konstruktor

Konstruktor tohoto formuláře volá metody, které inicializují DataGridView (tabulku) a ComboBox (výběr sloupců), načítají skóre do Listu a zobrazují ho do tabulky. Konstruktory existují dva, jeden uchovává proměnné pro novou hru, které si hráč vybral v nastavení. Uchovává je proto, aby když hráč přešel z menu do tabulky a zpět, tak aby o ně nepřišel. Druhý konstruktor slouží pro Score zobrazené z dokončené hry, ten proměnné uchovávat nemusí.

### InitializeDataGridView a InitializeComboBox

Metoda InitializeDataGridView inicializuje a nastavuje DataGridView. Nastavuje styly, sloupce a další vlastnosti.

InitializeComboBox nastavuje hodnoty položek na jména sloupců a počáteční vybraný index.

### LoadScoreData a DisplayData

LoadScoreData načítá za pomocí třídy GameScoreSaveManager a metody která do Listu předá data z datového souboru. Metoda je ošetřena výjimkou, v případě, zda soubor je nepoužitelný.

DisplayData vymaže veškeré řádky v tabulce a nahraje všechny informace z Listu.

### buttonFilter\_Click a buttonClear\_Click

Metoda buttonFilter\_Click se spustí po kliknutí na tlačítko Filtrovat. Pracuje se s textem z TextBoxu. Pokud je text prázdný, zobrazí se všechny data. V opačném případě se do pomocné proměnné přiřadí vybrána položka v ComboBoxu.

Vytvoří se také pomocný List datového typu ScoreData, do kterého se uloží všechny data z tabulky, které odpovídají filtrování. To se provede cyklem foreach, kterým se projdou všechny záznamy v původním Listu. V případě, že byly nalezena nějaká data, tak se zobrazí v tabulce místo původních metodou DisplayData.

Při kliknutí na tlačítko Vymazat Tabulku se zobrazí MessageBox, který se ptá na potvrzení, zdali chce hráč tabulku opravdu vymazat. Pokud jeho odpověď je ano, tak se zavolá metoda ClearScoreData ze třídy GameScoreSaveManager, která vymaže datový soubor. Následně se vymažou data v Listu a ten se zobrazí metodou DisplayData. Mazání je ošetřeno výjimkou.



Obr. . Obsah

Text obsahu se píše od jednotné svislice (se zřetelem k nejdelšímu číselnému označení).

## Psaní úvodu

Úvod může být osobitějšího rázu. První odstavec by měl obsahovat motivaci či důvod, který autora přiměl k volbě daného tématu s ohledem na přínos práce. V dalších odstavcích může být popsán obsah jednotlivých kapitol.

## Struktura odstavců

V případě, že věta vychází se spojkou na konci řádku, je spojka vždy přesunuta pomocí Shift + Enter na následující řádek. Hodnoty s jednotkami musí mít mezi číslem a označením jednotky jednu mezeru. To lze uskutečnit pomocí Shift + Ctrl + Mezerník (tzv. pevná mezera).

Text práce by měl být napsán v neutrální formě, tj. ve 3. osobě v trpném rodě. V textu by se neměli objevovat slangové výrazy, citově zabarvená slova ani podmiňovací způsob (s výjimkou je-li, uvažujeme-li apod.). Autor by se měl vyhnout přílišnému opakování slov či užívání nadbytečných výrazů.

Forma textu:

* v neurčité formě (bylo zjištěno, navrhuje se…);
* v 1.  osobě jednotného čísla (zjistil jsem, navrhuji …) – zvláště vhodné v kapitolách, které jsou vlastní prací autora.

## Obrázky, tabulky a rovnice

Popisek obrázku a tabulky se vkládá kliknutím pravého tlačítka myši na objekt a výběrem možnosti *Vložit titulek*. Následně je vybrán typ objektu (*Tab.* nebo *Obr.*) a jeho poloha (obrázky pod objekt, tabulky nad objekt). Styl popisku je Times New Roman 11 kurzíva uprostřed. Seznamy obrázků a tabulek na konci dokumentu jsou automaticky vygenerovány. Obrázky a tabulky mají vždy i slovní popis a rovnice jsou bez slovního popisu. Za titulek obrázku nepatří tečka. [3] Číslování obrázků, tabulek a rovnic je provedeno dle **hlavní kapitoly**, v níž se vyskytují a jejich pořadí v této kapitole. Nástroj *Vložit Titulek* čísluje popisky obrázků a tabulek automaticky. V případě že dojde k editaci, odstranění nebo přesunu již existujícího popisku je nutné dokument aktualizovat. Pro samotné vkládání rovnic je užit nástroj *Rovnice* v záložce *Vložení* (rovnice jsou vždy psány kurzívou).

|  |  |
| --- | --- |
|  |  |

Vzorec pro měření entropie. Rovnice jsou bez slovního popisu.



Obr. . Příklad umístění legendy obrázku

Titulek ke grafu a obrázku se píše pod objekt.

Tab. . Legenda k tabulce

|  |  |  |  |
| --- | --- | --- | --- |
| Jméno | Příjmení | Body | Známka |
| Petr | Novák | 25 | 4 |
| Karel | Kolář | 36 | 3 |
| Martin | Pokorný | 89 | 1 |

Titulek tabulky se píše nad tabulku.

Vložený objekt musí být vždy okomentován. Je nutné napsat před a za objekt alespoň jeden odstavec textu.

## Řazení a struktura kapitol

Z hlediska přehlednosti každá nová kapitola (Nadpis 1) začíná na novém listu. První kapitola bývá zaměřena na rešeršní část, tedy definice pojmů, vymezení studované oblasti apod. Druhá a následující kapitoly jsou zaměřeny na řešení samotného problému.

## Závěr

Závěr obsahuje stručné shrnutí získaných poznatků, uvedení dalších možných postupů či řešení, hodnocení dostupné odborné literatury, ze které bylo čerpáno. Závěr by měl obsahovat kritické porovnání záměru práce a dosažených výsledků, srovnání dosažených výsledků s dosud známými poznatky a popis odlišností od doposud známých skutečností. Závěr může naznačit praktické uplatnění výsledků práce.

## Seznam použitých zdrojů

Účelem této části je zaznamenání publikací, jež byly využity v maturitní práci.

Citace se řídí normou ČSN ISO 690 Bibliografické citace. [1] V případě doslovné citace je potřebné citovaný text graficky odlišit od ostatního textu, ohraničit apostrofem, případně ještě kurzívou. Odkaz na citovaný zdroj, jehož plná citace bude uvedena v soupisu literatury na konci práce, se uvede přímo za doslovně citovaný text.

Citování zdrojů se provádí v hranatých závorkách, používá se metoda číselných odkazů. Tento seznam je vždy seřazen dle výskytu citací v textu práce. Necitování použité literatury je považováno za plagiátorství.

## Seznam použitých symbolů a zkratek

Seznam symbolů a zkratech je vytvořen pomocí tabulky. Zkratky a názvy veličin jsou psány stylem *Normální* a jednotky jsou vytvořeny pomocí nástroje *rovnice*. Pro přidání řádku je potřeba kliknout pravým tlačítkem myši do posledního řádku tabulky a z nabídky *vložit* vybrat možnost *vložit pod*. Pro odstranění pak obdobně vybrat možnost *odstranit celý řádek*.

## Seznamy použitých obrázků a tabulek

Tyto seznamy se vytvářejí automaticky dle použitých stylů. Seznamy je potřeba aktualizovat kliknutím pravého tlačítka myši na první položku seznamu a následným výběrem možnosti aktualizovat pole (nebo kliknutím na první položku seznamu a stiskem F9).

## Seznam příloh

Seznam příloh je nutné vyplnit ručně.

Závěr

Vytvořená šablona maturitních prací obsahuje formální požadavky maturitních prací na SPŠT Třebíč. Jedná se zejména o upravené styly v dokumentu, podrobný popis jednotlivých částí maturitní práce a jejího obsahu, snadno editovatelné záhlaví a zápatí s automatickým číslováním stránek a propojení stylů se seznamy a obsahem.

Seznam použitých zdrojů

[1] RISTOV, Ivan. The Complete History of Board Games. Online. Dostupné z: https://boardgamesland.com/the-complete-history-of-board-games/. [cit. 2024-12-29].

[2] AUGUSTYN, Adam (ed.). Go. Online. Dostupné z: https://www.britannica.com/topic/go-game. [cit. 2024-12-29].

[3] History of Chess. Online. Dostupné z: https://www.houseofstaunton.com/history-ofchess?srsltid=AfmBOorR3RfV9habA13fuCJIqDSO\_u67aq6anH2SEMY6vxo1zWTXV6lH. [cit. 2024-12-29].

[4] Anne-Marie. The Benefits of Board Games: How Tabletop Games Improve Life Skills. Online. Dostupné z: https://bristoltutors.co.uk/news/the-benefits-of-board-games-how-tabletop-games-improve-life-skills/. [cit. 2024-12-30].

[5] What Are The Psychological Benefits Of Board Games for Adults & Children? Online. Dostupné z: https://bandpassdesign.com/blogs/news/benefits-of-board-games?srsltid=AfmBOoob1w8gXQnjEmzTnUycI-i5v-mG7k9KFTV2P1m61fPBpzRpwjYh. [cit. 2024-12-30].

[6] HLAVÁČ, Jakub, MF DNES. Pekelně se soustřeď. Autor pexesa přišel kvůli komunistům o slávu i peníze. Online. Dostupné z: https://www.idnes.cz/hobby/domov/pexeso-hra-vznik-autor-zdenek-princ.A191015\_151742\_hobby-domov\_mce. [cit. 2024-12-15].

[7] FOLKVORD, Frans; ANASTASIADOU, Dimitra Tatiana; ANSCHÜTZ, Doeschka. Memorizing fruit: The effect of a fruit memory-game on children's fruit intake. ScienceDirect. [online]. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2211335516301565. [cit. 2024-12-27].

[8] BELLIS, Mary. A Short History of Microsoft. Online. Dostupné z: https://www.thoughtco.com/microsoft-history-of-a-computing-giant-1991140. [cit. 2024-12-30].

[9] GeeksForGeeks. Introduction to Visual Studio. Online. Dostupné z: https://www.geeksforgeeks.org/introduction-to-visual-studio/. [cit. 2025-01-03].

[10] GeeksForGeeks. Introduction to .NET Framework. Online. Dostupné z: https://www.geeksforgeeks.org/introduction-to-net-framework/. [cit. 2025-01-03].

[11] GeeksForGeeks. C# Tutorial. Online. Dostupné z: https://www.geeksforgeeks.org/csharp-programming-language/?ref=lbp. [cit. 2025-01-03].

[12] GeeksForGeeks. Introduction to C# Windows Forms Applications. Online. Dostupné z: https://www.geeksforgeeks.org/introduction-to-c-sharp-windows-forms-applications/. [cit. 2025-01-03].

[13] GitHub. About GitHub and Git. Online. Dostupné z: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git. [cit. 2025-01-03].

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