Návrh a realizace 2D počítačové hry

Bc. Radek Majar

Diplomová práce
2023

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav elektroniky a měření

Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Radek Majar**
Osobní číslo:	**A21145**
Studijní program:	**N1032A020003 Bezpečnostní technologie, systémy a management**
Specializace:	**Bezpečnostní management**
Forma studia:	**Prezenční**
Téma práce:	**Návrh a realizace 2D počítačové hry**
Téma práce anglicky:	**Design and Implementation of a 2D Computer Game**

Zásady pro vypracování

1. Zpracujte literární průzkum o programu Visual Studio a existujících her.
2. Navrhněte novou 2D hru, vytvořte grafický návrh hry.
3. Sestrojte strukturu programu 2D hry ve Visual Studiu.
4. Vytvořte vlastní databázi otázek pro vybraný předmět z oboru BTSM.
5. Uveďte popis tvorby 2D hry a použité funkce.

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

1. Visual Studio .NET: praktické programování krok za krokem, František Šíma, David Vilímek, Vydavatel: Grada Publishing a.s., 2006, ISBN: 8024714183, 9788024714189, počet stran 254.
2. Microsoft Visual C# 2010, John Sharp, 2016, ISBN 8025139492, 9788025139493, počet stran 696.
3. Beginning C# 7 Programming with Visual Studio 2017, Benjamin Perkins, Jacob Vibe Hammer, Jon D. Reid, Vydavatel: John Wiley & Sons, 2018, ISBN: 1119458722, 9781119458722, počet stran 912.
4. Začínáme s Microsoft Visual C# 2008, Karli Watson , Christian Nagel , Jacob Hammer Pedersen , Jon D. Reid , Morgan Skinner , Eric White, Vydavatel: John Wiley & Sons, 2008, ISBN: 0470381515, 9780470381519, počet stran 1307.
5. Začínáme s Visual C# 2012 Programování, Karli Watson , Jacob Vibe Hammer , Jon D. Reid , Morgan Skinner , Daniel Kemper , Christian Nagel, Vydavatel: John Wiley & Sons, 2012, ISBN 111833194X, 9781118331941, počet stran 912.
6. Visual Studio 2022 do hloubky: Prozkoumejte fantastické funkce Visual Studio 2022 – 2. vydání, Ocert J. du Preez, Vydavatel: Publikace BPB, 2022, ISBN 9355512457, 9789355512451, Počet stran: 260.

Vedoucí diplomové práce: **Ing. Karel Perůtka, Ph.D.**
Ústav řízení procesů

Datum zadání diplomové práce: **2. prosince 2022**
Termín odevzdání diplomové práce: **1. června 2023**

doc. Ing. Jiří Vojtěšek, Ph.D. v.r.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

Ing. Milan Navrátil, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 8. prosince 2022

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové/bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 24.05.2023

Radek Majar, v.r.
podpis diplomanta

ABSTRAKT

Diplomová práce se věnuje vývoji 2D počítačové hry v programu Visual studio.

Teoretická část popisuje prostředí programu Visual studio, programovací jazyk C# a jiné 2D počítačové hry vytvořené v jazyce C#.

Praktická část práce obsahuje popis vývoje 2D počítačové hry, která je přiložena na disku CD-ROM. Hra je popsána z pohledu programátora a hráče.

Klíčová slova: Visual studio, C#, Programování

ABSTRACT

The diploma thesis deals with the development of a 2D computer game in the Visual Studio program.

The theoretical part describes the Visual Studio environment, the C# programming language and other 2D computer games created in the C# language.

The practical part of the thesis contains a description of the development of a 2D computer game, which is included on the CD-ROM. The game is described from the point of view of the programmer and the player.

Keywords: Visual studio, C#, Programming

Tímto bych chtěl poděkovat vedoucímu práce, panu Ing. Karlu Perůtkovi, Ph.D., za odborné vedení, návrhy a rady během vypracování diplomové práce.

Prohlašuji, že odevzdaná verze bakalářské/diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Cílem této diplomové práce je vytvoření 2D počítačové hry ve vývojovém prostředí Visual Studio pomocí programovacího jazyka C#. Hra by měla dále sloužit k procvičení znalostí z předmětů, které se vyučují na studijním oboru Bezpečnostní technologie, systémy a management.

V první části diplomové práce je popsáno vývojové prostředí Visual Studio a programovací jazyk C#. V další části jsou popsány 2D počítačové hry vytvořené ve vývojovém prostředí Visual Studio v programovacím jazyce C#.

V praktické části je popsán cíl hry a její pravidla. Následně je hra popsána z pohledu programátora, včetně popisu tvorby grafiky, všech funkcí a ukázek kódu. V poslední části je hra popsána z pohledu hráče. Jsou zde uvedeny obrázky všech oken, které může hráč ve hře vidět.

I. TEORETICKÁ ČÁST

1 MICROSOFT VISUAL STUDIO

Microsoft Visual Studio je integrované vývojové prostředí (IDE) od společnosti Microsoft. Využívá se pro vytváření konzolových aplikací nebo pro aplikace s grafickým rozhraním. MS Visual Studio podporuje několik programovacích jazyků, jako je C++, C#, Visual Basic, Java, Python a další. Aktuálně je nejnovější verze Visual Studio 2022. [1]

1.1 Editor kódu

Editor kódu umožňuje programátorům psát a editovat zdrojový kód pro jejich aplikace. Tento nástroj nabízí spoustu funkcí, které usnadňují psaní a správu kódu. Díky osnově můžete zobrazit nebo skrýt různé bloky kódu. Editor umožňuje zobrazit jednotlivé části kódu pomocí funkcí *Přejít na, Přejít k definici* nebo *Najít všechny odkazy*. Další informace o kódu si můžeme zobrazit v IntelliSense, který obsahuje funkce jako je seznam členů, rychlé informace, informace o parametrech. Tyto funkce umožňují uživateli získat informace o kódu, sledovat parametry, které píše, a přidávat volání vlastností a metod pomocí klávesových zkratek. [2]

Užitečné funkce editoru kódu ve Visual Studiu

- **Zvýraznění syntaxe –** editor zvýrazňuje syntaxi kódu různými barvami, aby byl kód čitelnější a srozumitelnější
- **Automatické dokončování** – editor umožňuje automatické dokončování kódu, díky tomu je psaní kódu rychlejší a programátor dělá méně chyb
- **Chybové a varovné značky –** během psaní kódu editor ukazuje chyby v syntaxi červenou barvou, chyby v kompilátoru modrou barvou, zelenou barvou upozornění a fialovou zobrazuje jiné typy chyb. Pomocí rychlé akce je možné zobrazit navrhované opravy problémů.
- **Sledování změn** – na levé straně editoru jsou zobrazeny žlutou barvou řádky, na kterých byla provedena změna. Po uložení se řádky změní na zelenou.
- **Změna znaků na velká písmena** – pomocí klávesové zkratky CTRL+Shift+U se všechny znaky ve výběru změní na velká písmena.
- **Změna znaků na malá písmena** – pomocí klávesové zkratky CTRL+U se všechny znaky ve výběru změní na malá písmena.

- **Vložení znaků komentáře** – pomocí klávesové zkratky CTRL+K a následně klávesové zkratky CTRL+C vložíme na vybrané řádky znaky komentáře.
- **Zrušení znaků komentáře** – pomocí klávesových zkratek CTRL+K a CTRL+U odebereme znaky komentáře z vybraných řádků. [2]

1.2 Debugger

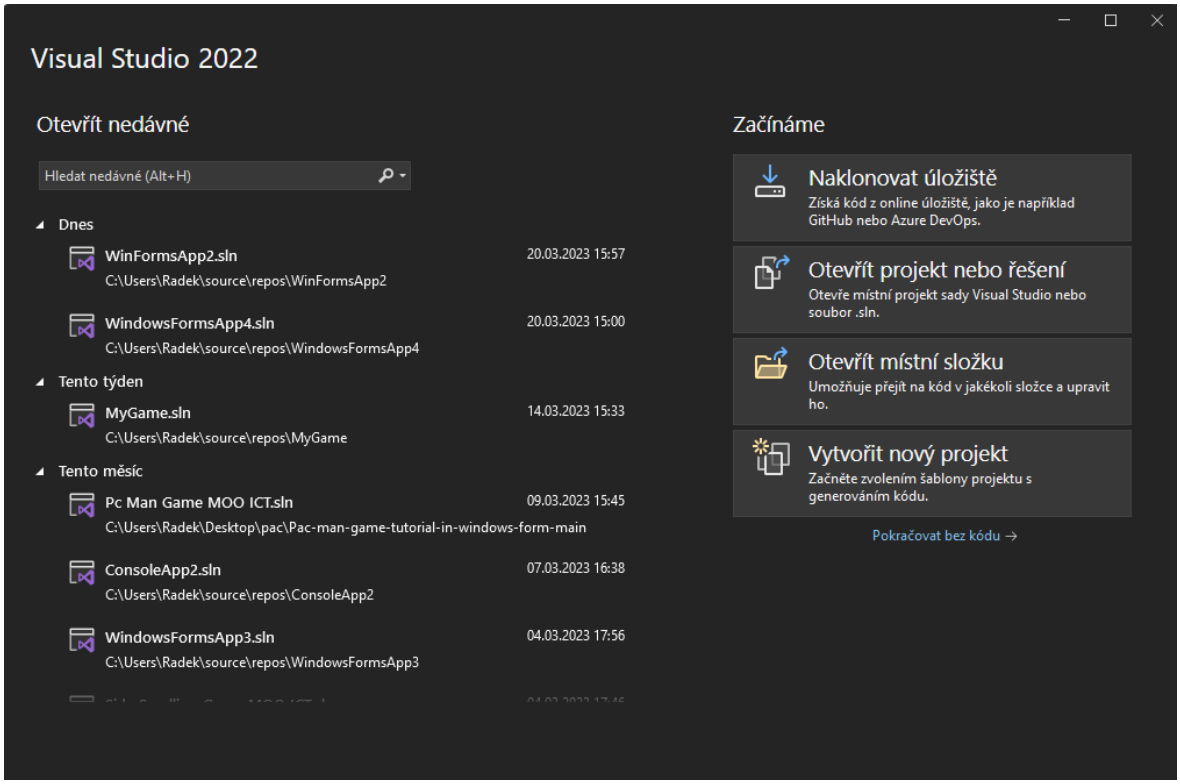
Debugger je nástroj, který umožňuje programátorům ladit svůj kód a nalézt chyby v programu. Díky debuggeru můžeme spustit kód krok po kroku a sledovat, co se děje v reálném čase. Díky tomu můžeme najít a opravit chyby v kódu a zvýšit tím stabilitu a spolehlivost aplikace. [3]

Debugger obsahuje průzkumník proměnných, díky němuž můžeme sledovat změny hodnoty proměnných v reálném čase. Pomocí nastavení zarážek v kódu můžeme zastavit spouštění aplikace na určitém místě a sledovat co se děje. Debugger obsahuje funkci *Edit and Continue*, takže můžeme upravovat kód během debuggingu. [3]

1.3 Vytvoření nového projektu

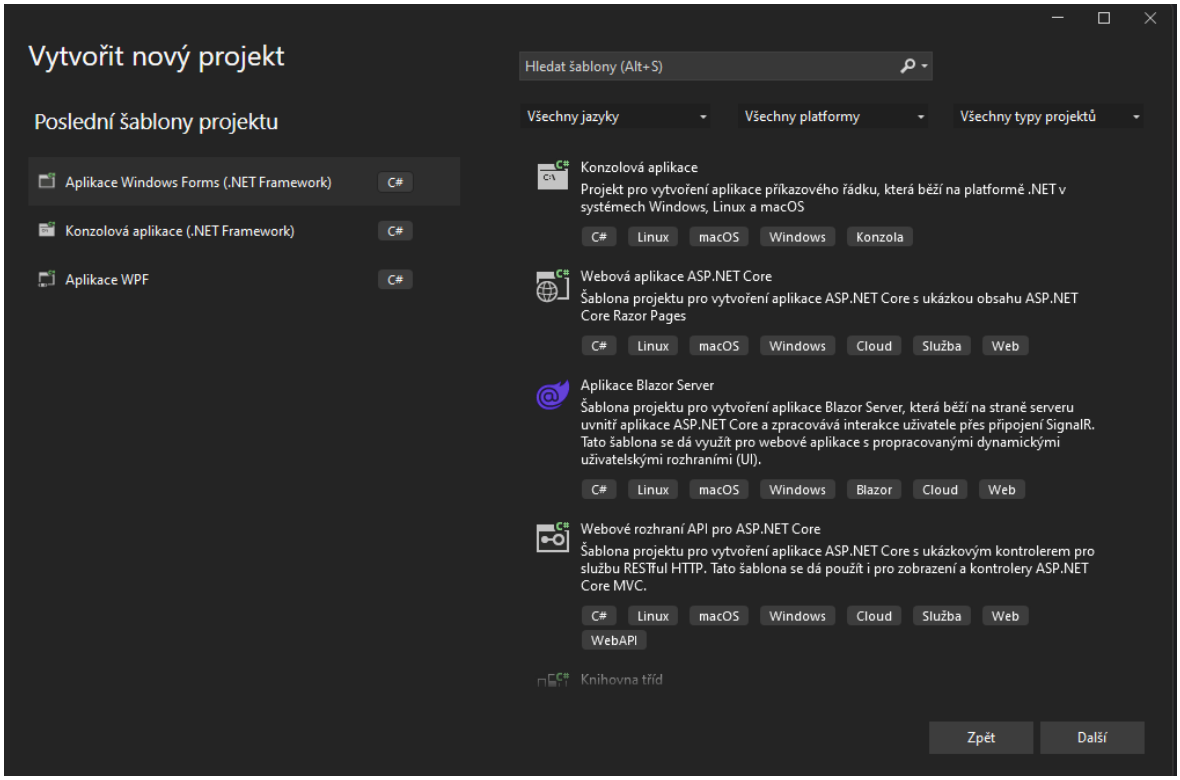
Po zapnutí MS Visual Studia se zobrazí okno s možnostmi, jak můžeme vidět na obrázku 1. V levé části okna se nachází nedávno otevřené projekty. V pravé části jsou možnosti:

- Naklonovat uložiště (získání kódu z online uložiště)
- Otevření projekt nebo řešení (otevření projektu, který je uložen na disku)
- Otevřít místní složku
- Vytvořit nový projekt.



Obrázek 1 Úvodní okno Visual Studia

Po kliknutí na možnost *Vytvořit nový projekt* se zobrazí okno pro vybrání šablony. Okno je zobrazeno na obrázku 2. V levé části okna je zobrazen seznam posledních použitých šablon. V pravé části okna si můžeme vyhledat šablonu, kterou budeme používat. V tomto případě vybereme šablonu *Aplikace Windows Forms (.NET Framework)* a budeme programovat v jazyce C#.

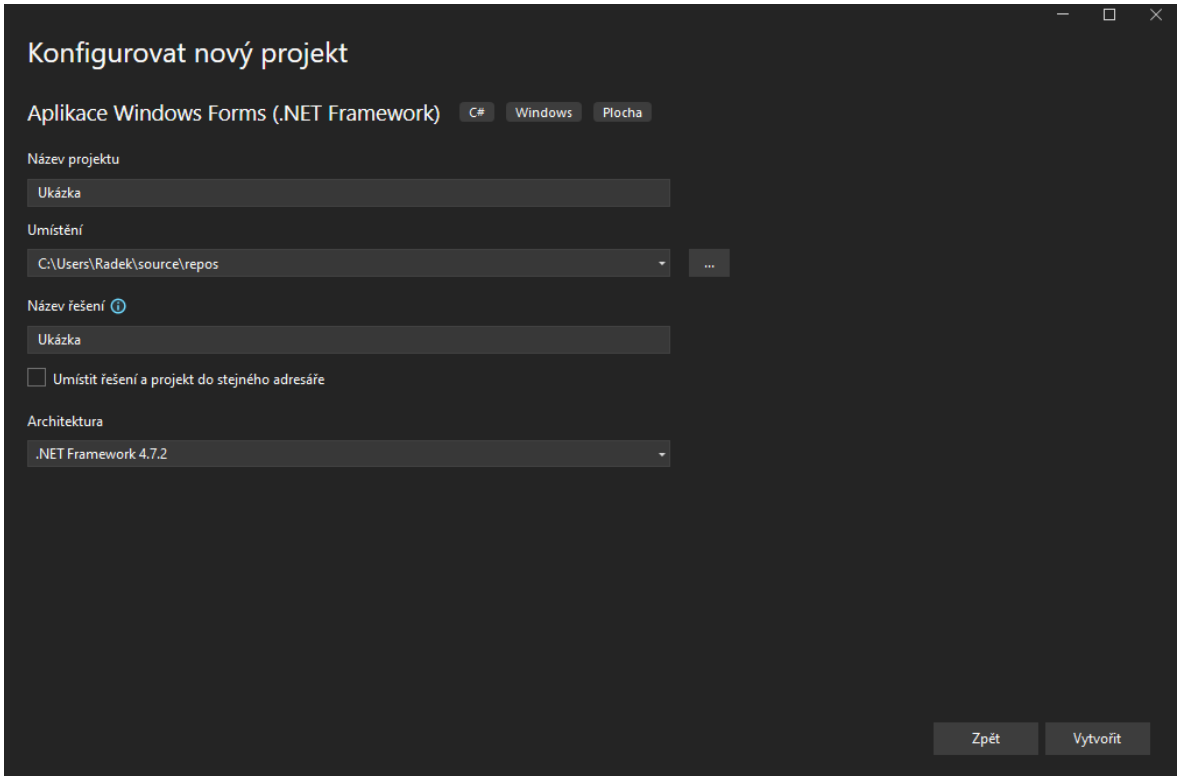


Obrázek 2 Vytváření nového projektu v MS Visual Studiu

Po vybrání šablony se zobrazí okno pro konfiguraci nového projektu. V tomto případě vybereme šablonu Aplikace Windows Forms (.NET Framework).

Nastavují se zde následující parametry:

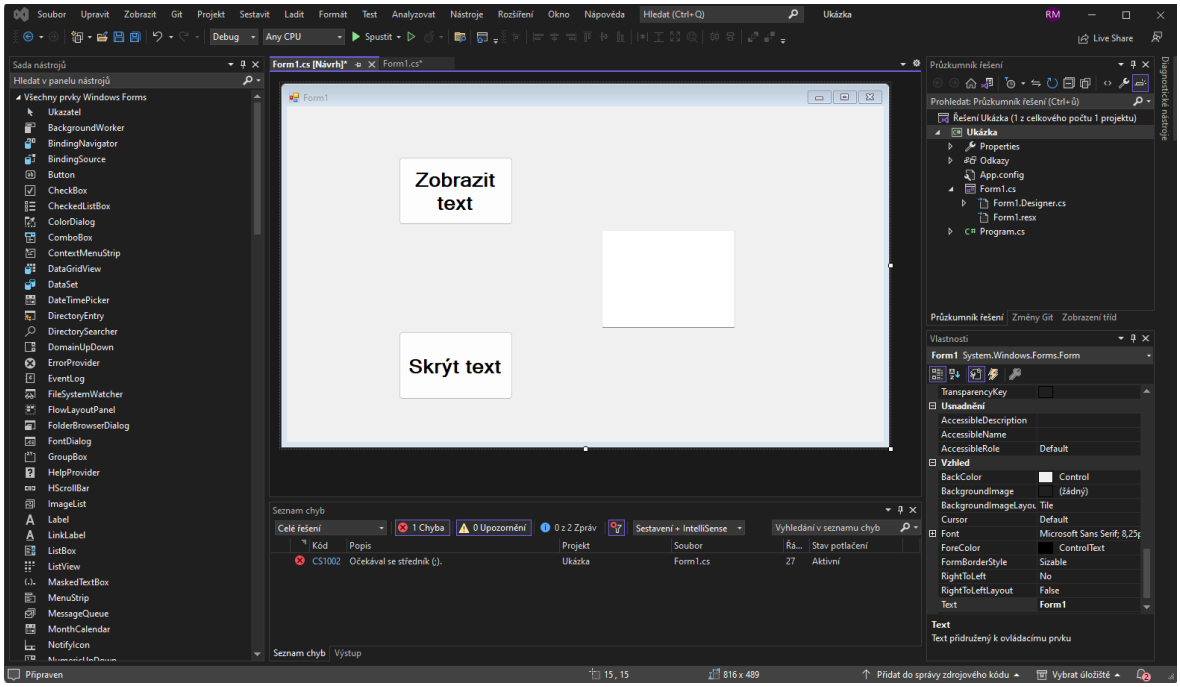
- Název Projektu
- Umístění
- Název řešení
    - Možnost umístit řešení a projekt do stejného adresáře nebo samostatně
- Architektura
    - .NET Framework 2.0
    - .NET Framework 3.0
    - .NET Framework 3.5
    - .NET Framework 4.7.2
    - .NET Framework 4.8



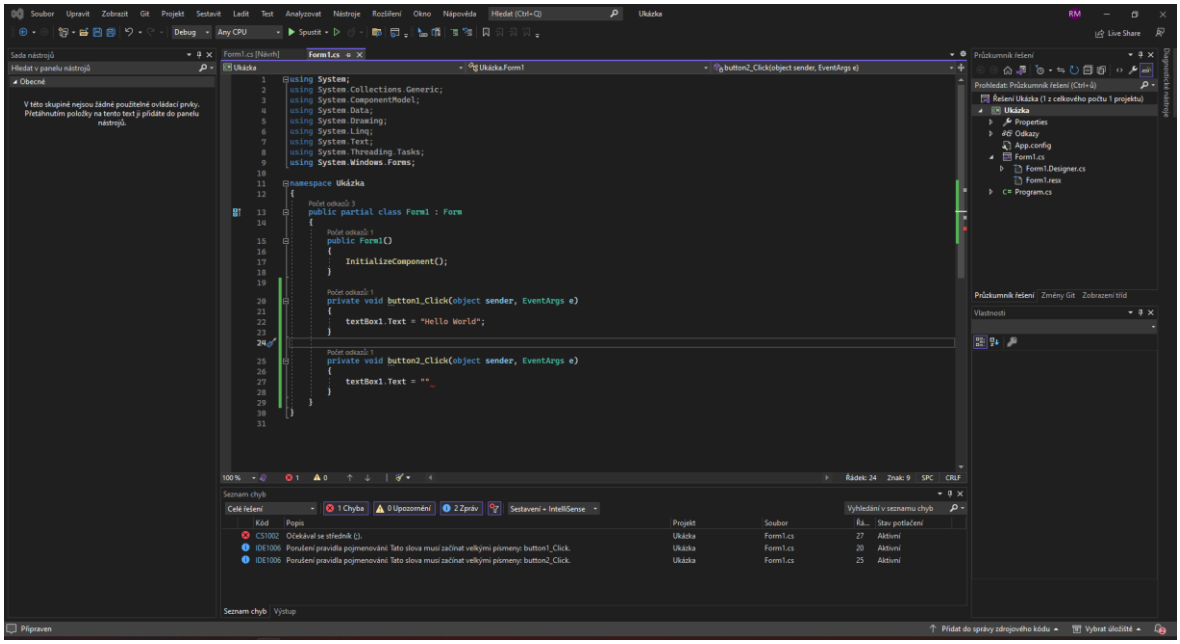
Obrázek 3 Okno konfigurace nového projektu

Po kliknutí na *Vytvořit* dojde k vytvoření nového projektu.

1.4 Okno aplikace



Obrázek 4 Okno aplikace MS Visual Studio 2022 – soubor otevřený v návrhovém režimu



Obrázek 5 Okno aplikace MS Visual Studio 2022 – soubor otevřený v textovém režimu

Okno aplikace je rozděleno na tyto části:

- **Menu** – v menu se nachází několik záložek, které jsou blíže popsány v kapitole 1.4.1
- **Nástrojová lišta** – nachází se zde několik funkcí pro rychlejší práci ve vývojovém prostředí. Můžeme odtud vytvořit nový projekt, otevřít soubor, spustit program, vrátit změny v kódu zpět nebo dopředu, …
- **Sada nástrojů** – nachází se zde prvky aplikace
- **Záložky** – zde jsou zobrazeny všechny otevřené soubory, díky tomu můžeme mezi nimi rychle přepínat.
- **Pracovní plocha** – zde je zobrazen aktuální otevřený soubor buď v režimu návrhu, jak můžeme vidět na obrázku 4 nebo v textovém režimu, který je zobrazen na obrázku 5.
- **Průzkumník řešení** – zde je zobrazena struktura projektu a najdeme zde všechny soubory, které jsou do něj zahrnuty.
- **Seznam chyb** – v této části se zobrazují chybové zprávy, které jsou rozděleny do třech kategorií podle důležitosti.
    - **Chyba** – má nejvyšší důležitost. Dokud se v kódu vyskytuje chyba, tak není možné projekt sestavit a spustit.

- **Upozornění** – střední důležitost. Projekt můžeme sestavit a spustit, ale může dojít k chybnému chování programu.
- **Zpráva** – nejnižší důležitost. Jedná se pouze o informační zprávy, které nemají vliv na chování programu.

- **Vlastnosti** – po kliknutí na prvek na pracovní ploše se zde zobrazí jeho vlastnosti, které můžeme upravovat.
- **Stavový řádek** – informační lišta

1.4.1 Menu

Struktura menu vývojového nástroje Microsoft Visual Studio:

- **Soubor** – tato záložka slouží pro práci se soubory a projekty, můžeme zde vytvářet nové projekty a soubory, otevírat již vytvořené projekty nebo projekt zavřít. Také se zde nachází možnost uložení projektu.
- **Upravit** – v této záložce najdeme funkce pro *přejít na* nebo *najít a nahradit*. Dále zde najdeme operace *vpřed* a *zpět*. Pro práci se schránkou se zde nachází funkce jako je vyjmutí, kopírování a vložení kódu nebo jeho odstranění.
- **Zobrazit** – tato záložka obsahuje přepínání mezi návrhářem a textovým editorem. Dále si zde můžeme zobrazit pomocná okna.
- **Git** – v této záložce se nachází funkce pro nastavení Gitu. Git slouží pro jednoduší správu verzí kódu. Můžeme si zde zobrazit změny kódu, provedené v průběhu času nebo obnovit určité verze kódu.
- **Projekt** – v této záložce se nachází funkce pro přidávání nových položek do projektu jako je například formulář, třídy, uživatelský ovládací prvek, atd. A dále si zde můžeme nastavit vlastnosti projektu.
- **Ladit** – V této záložce můžeme spustit ladění programu. Pro kontrolu stavu v daném bodě můžeme přidat do kódu zarážky.
- **Formát** – zde najdeme funkce pro nastavení formátovacích vlastností ovládacích prvků formuláře. Můžeme například jednotlivé prvky zarovnat, změnit velikost prvků, změnit vodorovné nebo svislé mezery mezi prvky a přenést prvky do popředí nebo do pozadí.

- **Test** – v této záložce se provádí testování výsledného programu
- **Analyzovat** – v této záložce můžeme spustit vyčištění kódu. Dojde například k odstranění nepoužívaných proměnných nebo odkazů.
- **Nástroje** – tato záložka obsahuje možnost přidání dalších nástrojů a funkcí do Visual Studia, připojení k databázi nebo k serveru a také zde můžeme měnit vlastnosti vývojového prostředí.
- **Rozšíření** – záložka pro správu rozšíření
- **Okno** – v záložce se nachází nastavení pro zobrazení okna a přepínání mezi otevřenými soubory.
- **Nápověda** – V této záložce si můžeme zobrazit nápovědu, tipy a triky, co je nového ve Visual Studiu, atd.

2 PROGRAMOVACÍ JAZYK C#

Programovací jazyk C# vyvinula firma Microsoft. Byl vytvořen s celým vývojovým prostředím .NET. C# vychází z programovacích jazyků C a C++ a podobá se programovacímu jazyku Java. [4]

Níže jsou uvedeny základní vlastnosti jazyka C#.

- Jedná se čistě o objektově orientovaný jazyk
- Podporuje atributové programování
- Využívá automatickou správu paměti (garbage collector – stará se o korektní uvolnění zdrojů aplikace)
- Podporuje zpracování chyb formou výjimek
- Obsahuje nativní podporu komponentového programování
- Podporuje typovou bezpečnost a podporuje řízení verzí
- Zabezpečuje zpětnou komptabilitu se stávajícím kódem na zdrojové i binární úrovni
- Jedná se o case sensitive programovací jazyk (rozlišuje velká a malá písmena) [4]

2.1 Struktura programu

Na níže uvedeném kódu je zobrazena obecná struktura kódu. Program je vytvořen v konzolové aplikaci.

```
 1. using System;
 2.
 3. namespace ConsoleApp1
 4. {
 5.     internal class Program
 6.     {
 7.         static void Main(string[] args)
 8.         {
 9.             Console.WriteLine("Hello World!");
10.         }
11.     }
12. }
```

Tento program má jedinou funkci, a to vypsání textu „Hello World!“ do konzole. Na prvním řádku kódu je napsán příkaz *using*, který importuje jmenný prostor *Systém*.

Na dalším řádku je příkaz *namespace*, který nám říká že kód je uložen ve jmenném prostoru ConsoleApp1.

Níže je definována třída *Program*. Pro definování třídy používáme klíčové slovo *class*.

Na dalším řádku je metoda *Main*, kterou musí mít každá aplikace v jazyce C#, protože metoda *main* je vstupním bodem spustitelného programu. V tomto případě funkce *main* obsahuje jen jeden příkaz a to *Console.WriteLine("Hello World!"*, který vypíše text „Hello World!“ do konzole.

2.2 Komentáře

Komentáře využívá programátor pro vepisování poznámek do kódu. Tyto poznámky nijak neovlivňují rychlost a funkčnost programu.

V jazyce C# máme dva základní typy komentářů. Prvním typem je jednořádkový komentář, který se uvozuje dvěma lomítky. Druhým typem je víceřádkový komentář. Víceřádkový komentář se označuje na začátku /* a pro jeho ukončení se na konec komentáře píše */.

Na příkladu níže jsou uvedeny oba způsoby psaní komentářů do kódu.

```
1. using System;
2.
3. // Aplikace pro vypsání text Hello World! do konzole
4. namespace ConsoleApp2
5. {
6.     internal class Program
7.     {
8.         static void Main(string[] args)
9.         {
10.            /* Příkaz pro vypsaní textu do okna konzole
11.            Do konzole se vypíše text Hello World!
12.            */
13.            Console.WriteLine("Hello World!");
14.        }
15.    }
16. }
```

2.3 Datové typy

V jazyce C# rozděluje datové typy na 2 druhy:

- Hodnotové
- Referenční

Hlavním rozdílem mezi hodnotovými a referenčními datovými typy je ten že hodnotový datový typ obsahuje přímo hodnotu a referenční datový typ obsahuje pouze referenci (odkaz) na místo v paměti, ve které je uložena hodnota proměnné. [5]

Hodnotové datové typy mají pevnou velikost. Proměnná většinou obsahuje jednu hodnotu, jako je například jedno číslo, jeden znak nebo logická pravda. Většinou se jich v programu vyskytuje velmi mnoho, zabírají málo místa a je potřeba s nimi pracovat co nejrychleji. Pro hodnotové datové typy se vyhrazuje dočasné uložiště. Využívá se zásobník nebo registry. [5]

Referenční datový typy jsou složitější, protože obsahují více hodnot a atributů. Zabírají více místa v paměti než hodnotové datové typy. Protože referenční datové typy nemají předem omezenou velikost, tak se vyhrazuje náhodně alokovaný blok z volné oblasti systémové paměti, kterým se říká hromady, nebo také haldy.[5]

Tabulka 1 Tabulka datových typů [6]

Datový typ	**.Net typ**	**Velikost/Přesnost**	**Rozsah**
sbyte	System.SByte	8 bitů	-128 až 127
byte	System.BByte	8 bitů	0 až 255
Short	System.Int16	16 bitů	-32 768 až 32 767
ushort	System.UInt16	16 bitů	0 až 65 535
Int	System.Int32	32 bitů	-2 147 483 648 až 2 147 483 647
uint	System.UInt32	32 bitů	0 až 4 294 967 295
long	System.Int64	64 bitů	-9 223 372 036 854 775 808 až 9 223 372 036 854 775 807
ulong	System.UInt64	64 bitů	0 až 18 446 744 073 709 551 615
float	System.Single	7 čísel	$+-1.5 * 10^{-45}$ až $+-3.4 * 10^{38}$

double	System.Double	16 čísel	+-5.0 * 10−324 až +-1.7 * 10308
char	System.Char	16 bitů	U+0000 až U+ffff
decimal	System.Decimal	29 čísel	+-1.0 * 10^{-28} až +-7.9 * 10^{28}
bool	System.Boolean	8 bitů	true/false
string	Systém.String	neomezeně	omezeno pouze velikostí dostupné paměti

2.4 Modifikátory

V jazyce C# jsou modifikátory umožňují upravit chování tříd, metod, polí, vlastností a dalších prvků jazyka. Pomocí modifikátorů můžeme například ovlivnit přístupnost a viditelnost těchto prvků. [7]

Tabulka 2 Modifikátory [7], [8], [9]

Modifikátor	**Popis**
Public	Členy jsou přístupné i vně příslušné třídy
Protected	Chráněné členy jsou přístupné pouze uvnitř své třídy nebo tříd, které jsou od dané třídy odvozeny
Private	Přístup je omezen pouze uvnitř třídy nebo struktury, ve které jsou soukromé členy deklarovány
Const	Používá se pro označení konstant (hodnoty nelze upravovat)
Readonly	Nastavení vlastnosti že hodnoty jsou pouze ke čtení
Sealed	Od takto označené třídy, nepůjde dále odvozovat další třídy

2.5 Pole

Pole je datová struktura, do které můžeme uložit více proměnných stejného typu. Proměnné, které jsou uložené v poli se nazývají prvky pole. Každý prvek v poli má svůj index, díky kterému můžeme jednotlivé prvky z pole získat. Prvky pole jsou indexovány od nuly a musí být všechny stejného typu. [10]

V jazyce C# můžeme vytvořit tyto typy polí.

- Jednorozměrná pole
- Vícerozměrná pole
- Vícenásobná pole

Jednorozměrné pole lze inicializovat a následné naplnit tímto způsobem.

```
1. //inicializace a naplnění pole i
2. int[] i = new int[] {1, 2 ,3};
```

Pole také můžeme první inicializovat a na dalším řádcích přidat hodnoty k jednotlivým indexům pole.

```
1. //inicializace pole y
2. int[] y = new int[3];
3.
4. //naplnění pole y
5. y[0] = 1;
6. y[1] = 2;
7. y[2] = 3;
```

Vícerozměrné pole se zapisuje podobně. Na příkladu níže je zobrazeno vytvoření 2D a 3D pole, vložení hodnot do pole a následné zobrazení hodnoty z pole do konsole.

```
1.  //inicializace dvojrozměrného pole
2.  int[,] pole_2D = new int[3, 3];
3.  //naplnění pole
4.  pole_2D[0, 0] = 5;
5.  //Výpis hodnoty z pole na indexu 0,0 do konzole
6.  Console.WriteLine(pole_2D[0, 0]);
7.
8.  //inicializace trojrozměrného pole
9.  int[,,] pole_3D = new int[3, 3, 3];
10. //naplnění pole
11. pole_3D[0, 0, 0] = 5;
12. //Výpis hodnoty z pole na indexu 0,0,0 do konzole
13. Console.WriteLine(pole_3D[0,0,0]);
```

U vícenásobných polí můžeme říct, že jsou to pole, ve kterých jsou vloženy další pole. Nejčastěji se používají při práci s 2D poli, kde každý řádek má jiný počet sloupců.

```
1.              //definování počtu řádků
2.              int[][] z = new int[5][];
3.              //definování počtu řádků v jednotlivých řádcích
4.              z[0] = new int[1];
5.              z[1] = new int[2];
6.              z[2] = new int[3];
7.              z[3] = new int[4];
8.              z[4] = new int[5];
```

2.6 Operátory

Operátory můžeme rozdělit podle jejich účelu do těchto skupin:

- Aritmetické operátory
- Logické operátory
- Operátor přiřazení
- Relační operátory [11]

2.6.1 Aritmetické operátory

Aritmetické operátory se využívají k základním matematickým operacím jako je sčítání, odčítání, násobení, dělení a modulo. Dále do této skupiny patří unární operátory, a to operátor přírůstku (inkrementace – zvýšení hodnoty o 1) a operátor dekrementu (snížení hodnoty o 1) [12]

V tabulce níže je uveden přehled matematických operátorů.

Tabulka 3 Aritmetické operátory

+	Součet
-	Rozdíl
*	Součin
/	Podíl
%	Modulo
++	Inkrementace
--	Dekrementace

V jazyce C# můžeme využívat složené operátory. Tyto operátory jsou složeny z operátoru přiřazení a daného aritmetického operátoru. Díky složeným operátorům můžeme provést vypočet a výsledek uložit do proměnné.

V tabulce níže je uveden seznam složených operátorů

Tabulka 4 Složené operátory

Operátor	Význam	Původní zápis	Nový zápis
+=	Součet a přiřazení	A = A + B	A += B
-=	Rozdíl a přiřazení	A = A - B	A -= B
*=	Součin a přiřazení	A = A * B	A *= B
/=	Podíl a přiřazení	A = A / B	A /= B
%/	Modlu a přiřazení	A = A % B	A %= B

2.6.2 Logické operátory

Logické operátory se využívají k provádění logických operací. Logické operátory používáme, když chceme spojit několik logických výrazů (například u cyklů nebo podmínkách)

Do této skupiny operátorů patří tyto dva operátory:

- Logická konjunkce – jejíž význam je „a zároveň“ a značí se znaky „&&“
- Logická disjunkce – její význam je „nebo“ a značí se znaky „||“

V následujících dvou tabulkách je uvedena forma zápisu těchto dvou operátorů a hodnoty výsledků.

Tabulka 5 Logická konjunkce

Logická konjunkce (&&)		
X	**Y**	**Z = X && Y**
True	True	True
True	False	False
False	True	False
False	False	False

Tabulka 6 Logická Disjunkce

Logická disjunkce (||)		
X	**Y**	**Z = X && Y**
True	True	True
True	False	True
False	True	True
False	False	False

S těmito logickými operátory se používá ještě operátor negace, který se značí znakem „!“. Funkce tohoto operátoru je vysvětlena v následující tabulce.

Negace (!)	
X	**Y = X!**
True	False
False	True

2.6.3 Operátor přiřazení

Operátor přiřazení patří k nejpoužívanějším operátorům ve všech programovacích jazycích, protože se využívá k přiřazení hodnoty do proměnné. V jazyce C# se pro tento účel používá znak „=“. [13]

2.6.4 Relační operátory

Relační operátory se používají k porovnání dvou hodnot. Výsledkem porovnání je buď logická hodnota *true* nebo *false*. Relační operátory se používají většinou u tvorby podmínek, které způsobují větvení programu.

V následující tabulce je vypsán seznam relačních operátorů a jejich význam.

Tabulka 7 Seznam relačních operátorů

Operátor	**Význam**
<	Menší než

>	Větší než
==	rovnost
!=	nerovnost
<=	Menší nebo rovno
>=	Větší nebo rovno

2.7 Podmínky

Během programování musíme reagovat na různé situace. Například pokud uživatel zadá hodnotu, podle které se dále rozhodne, co program vykoná. Tomu se říká že se program větví a k větvení programu používáme podmínky.

V jazyce C# používáme tyhle 3 podmínky:

- IF
- IF-ELSE-IF
- SWITCH

2.7.1 Podmínka IF

Podmínka IF se používá pro vyhodnocení jednoho logického výrazu. Obecná syntaxe podmínky IF je následující:

```
1. if (podmínka)
2. Příkazy
3. else
4. Příkazy
```

Jak podmínka IF funguje si vysvětlíme na následujícím programu, kde se vyhodnocuje, jestli uložená hodnota v proměnné *cislo* je větší, jak číslo 10.

```
 1.             int cislo;
 2.             cislo = 5;
 3.             if (cislo > 10)
 4.             {
 5.                 Console.WriteLine("cislo je větší jak 10");
 6.             }
 7.             else
 8.             {
 9.                 Console.WriteLine("cislo je menší jak 10");
10.             }
```

2.7.2 Podmínka IF-ELSE-IF

Pokud potřebujeme vyhodnotit více stavů určité proměnné, tak používáme podmínku IF-ELSE-IF. Jedná se o modifikaci podmínky IF, která nám umožní vícenásobné větvení. [14]

Obecná syntaxe podmínky IF-ELSE-IF:

```
1. if (podmínka 1)
2.      Příkazy
3. else if (podmínka 2)
4.      Příkazy
5. else if (podmínka N)
6.      Příkazy
7. else
8.      Příkazy
```

V následujícím příkazu je uveden příklad, ve kterém zjišťujeme, jestli je hodnota v proměnné *číslo* větší, menší nebo se rovná číslu 10.

```
 1.             int cislo;
 2.             cislo = 5;
 3.             if (cislo == 10)
 4.             {
 5.                 Console.WriteLine("cislo se rovná číslu 10");
 6.             }
 7.             else if (cislo > 10)
 8.             {
 9.                 Console.WriteLine("cislo je větší jak 10");
10.             }
11.             else
12.             {
13.                 Console.WriteLine("cislo je menší jak 10");
14.             }
```

2.7.3 Podmínka SWITCH

Poslední možností, jak můžeme program větvit je pomocí podmínky SWITCH. Pomocí podmínky SWITCH můžeme zapsat více podmínek pod sebou. Obecná syntaxe podmínky SWITCH je následující:

```
 1. switch (proměnná)
 2. {
 3. case hodnota1:
 4.     Příkazy
 5. break;
 6. case hodnota2:
 7.     Příkazy
 8. break;
 9. case hodnotaN:
10.     Příkazy
11. break;
```

```
12. default:
13.     Příkazy
14. break;
15. }
```

Z obecné syntaxe můžeme vidět, že aby podmínka SWITCH využívá pro svou funkci další pomocné příkazy.

- Case
- Break
- Default

Prvním z pomocných příkazů je příkaz *case*. Pokud se hodnota za *case* rovná hodnotě uložené v proměnné, tak se vyková program v daném bloku. Dalším příkazem je *break*. Tento příkaz musíme napsat na každý konec bloku *case*, abychom tím daný blok ukončili. Pokud by blok *case* nebyl ukončen příkazem *break*, tak by program prováděl příkazy v dalších blocích *case,* dokud by nenarazil na příkaz *break*. Posledním příkazem je příkaz *default*. Příkazy, které jsou uvedeny v tomto bloku se vykonají jen v případě, že daná hodnota proměnné se neshoduje s žádnou hodnotou uvedenou za příkazem *case*. [14]

V následujícím kódu je ukázáno, jak daná podmínka funguje na příkladu vypsání do konsole názvu měsíce podle čísla, které je uloženo v proměnné *cisloMesice:*

```
1. int cisloMesice = 6;
2.    string nazevMesice;
3.
4.    switch (cisloMesice)
5.    {
6.        case 1:
7.            nazevMesice = "Leden";
8.            break;
9.
10.       case 2:
11.           nazevMesice = "Únor";
12.           break;
13.       case 3:
14.           nazevMesice = "Březen";
15.           break;
16.       case 4:
17.           nazevMesice = "Duben";
18.           break;
19.       case 5:
20.           nazevMesice = "Květen";
21.           break;
22.       case 6:
23.           nazevMesice = "Červen";
24.           break;
25.           ...
26.       case 12:
```

```
27.                     nazevMesice = "Prosinec";
28.                     break;
29.                 default:
30.                     nazevMesice = "Číslo neodpovídá žádnému z měsíců";
31.                     break;
32.             }
33.             Console.WriteLine(nazevMesice);
```

2.8 Cykly

Pomocí cyklů můžeme provést opakování určité části programu na základě vyhodnocení logického výrazu. Mezi cykly, které využíváme v jazyce C# patří tyto čtyři:

- FOR
- WHILE
- DO… WHILE
- FOREACH

2.8.1 Cyklus FOR

Cyklus FOR nám umožňuje opakovat určitou část programu. Před provedením příkazů je vždy zkontrolována platnost podmínky. Cyklus FOR má stanovený pevný počet opakování [14]

Obecná syntaxe Cyklu FOR je následující:

```
1. for (proměnná; podmínka; příkaz)
2. {
3. Příkazy
4. }
```

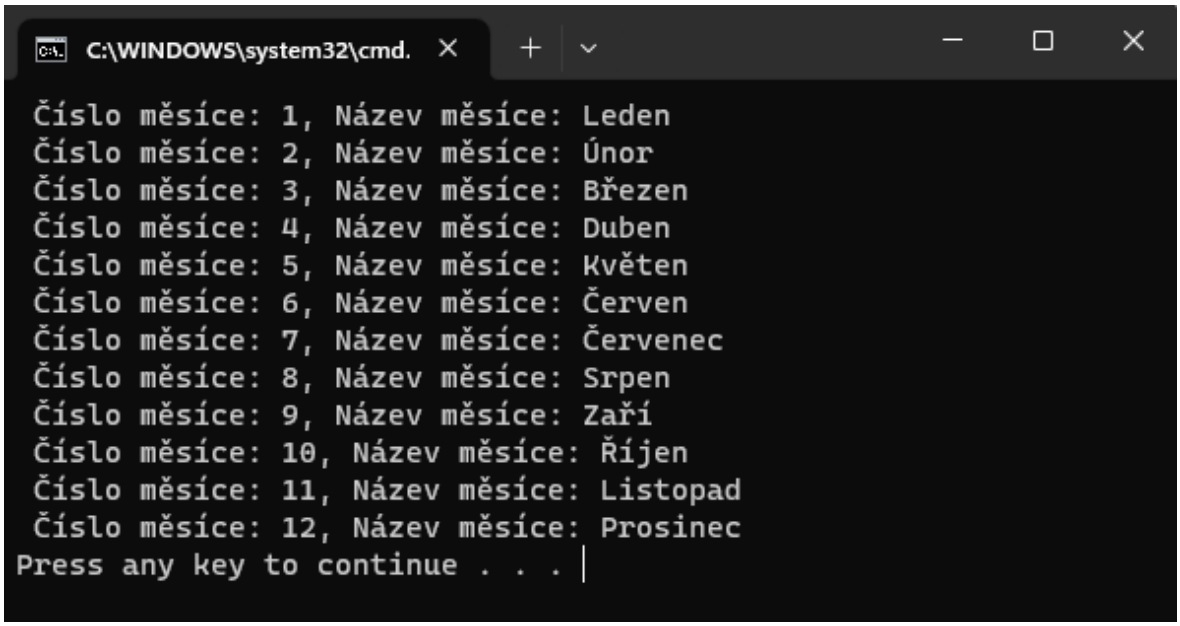
Cyklus FOR má tři parametry:

- **Proměnná** – zde se inicializuje lokální proměnná (nastavujeme zde počáteční hodnotu pro řídící proměnou)
- **Podmínka** – výraz, který se kontroluje před každým provedením příkazů. Jakmile podmínka nebude platit, tak se ukončí cyklus.
- **Příkaz** – nám určuje co se má stát v každém dalším kroku s řídící proměnnou (jestli se její hodnota má zvýšit nebo zmenšit) [14]

Níže je uvedena ukázka použití cyklu FOR. Program vypíše postupně do konsole čísla měsíců a jejich název.

```
            int cisloMesice = 1;
            string nazevMesice;
            for (int a = 0; a <= 11; a++)
            {

                switch (cisloMesice)
                {
                    case 1:
                        nazevMesice = "Leden";
                        break;
                    ...
                    case 12:
                        nazevMesice = "Prosinec";
                        break;
                    default:
                        nazevMesice = "Číslo neodpovídá žádnému z
měsíců";
                        break;
                }
               Console.WriteLine(" Číslo měsíce: {0}, Název měsíce: {1}
", cisloMesice, nazevMesice);
                cisloMesice++;
   }
```



Obrázek 6 Výpis do konzole ukázkového příkladu pro cyklus FOR

2.8.2 WHILE cyklus

Další možností, jak použít v programu cyklus je pomocí příkazu WHILE. Na rozdíl od cyklu FOR má cyklus WHILE jen parametr podmínky, která se kontroluje na začátku cyklu. Protože se podmínka kontroluje na začátku cyklu, tak může dojít k tomu, že příkazy uvnitř cyklu se neprovedou ani jednou. [14], [15]

Níže je uvedena obecná syntaxe cyklu WHILE:

```
1. while (podmínka)
2. {
3.      Příkazy
4. }
```

Cyklus WHILE se používá například v případech, kdy máme v podmínce metodu vracející hodnotu *true* nebo *false*.

Níže je uvedena ukázka použití cyklu WHILE. Program vypíše stejný seznam čísel a názvy měsíců do konzole jako to bylo u příkladu pro cyklus FOR.

```
            1. int cisloMesice = 1;
 2.            string nazevMesice;
 3.            while(cisloMesice <= 12)
 4.            {
 5.
 6.                switch (cisloMesice)
 7.                {
 8.                    case 1:
 9.                        nazevMesice = "Leden";
10.                       break;
11.                   ...
12.                   case 12:
13.                       nazevMesice = "Prosinec";
14.                       break;
15.                   default:
16.                       nazevMesice = "Číslo neodpovídá žádnému z
měsíců";
17.                       break;
18.               }
19.               Console.WriteLine(" Číslo měsíce: {0}, Název měsíce: {1}
", cisloMesice, nazevMesice);
20.               cisloMesice++;
21. }
```

2.8.3 Cyklus DO … WHILE

Cyklus DO … WHILE se podobá cyklu WHILE. Rozdíl mezi nimi je takový že, cyklus DO…WHILE má podmínku napsanou na konci cyklu. Díky tomu se nejdříve provedou příkazy uvnitř cyklu a až pak se kontroluje podmínka. Protože je podmínka napsána až na konci cyklu, tak se příkazy uvnitř cyklu provedou minimálně jednou, na rozdíl od cyklu WHILE. [14], [15]

Obecná syntaxe cyklu DO…WHILE je následující:

```
1. do
2. {
3.      Příkazy
4. }
```

```
5. while (podmínka);
```

Níže je uvedena ukázku programu pro výpis čísel a jim odpovídající názvy měsíců do konsole pomocí cyklu DO…WHILE.

```
            1. int cisloMesice = 1;
 2.             string nazevMesice;
 3.             do
 4.             {
 5.
 6.                 switch (cisloMesice)
 7.                 {
 8.                     case 1:
 9.                         nazevMesice = "Leden";
10.                         break;
11.                     ...
12.                     case 12:
13.                         nazevMesice = "Prosinec";
14.                         break;
15.                     default:
16.                         nazevMesice = "Číslo neodpovídá žádnému z
měsíců";
17.                         break;
18.                 }
19.                 Console.WriteLine(" Číslo měsíce: {0}, Název měsíce: {1}
", cisloMesice, nazevMesice);
20.                 cisloMesice++;
21.             }
22.             while (cisloMesice <= 12);
```

2.8.4 Cyklus FOREACH

Cyklus FOREACH jako jediný nevyhodnocuje podmínku, ale slouží k procházení polí.

Obecná syntaxe cyklu FOREACH:

```
1. foreach(datovýtyp proměnná in list)
2. {
3. Příkazy
4. }
```

Níže je uvedena ukázka cyklu foreach, díky kterému vypíšeme všechny prvky v poli do konsole. V tomto případě se jedná o názvy ročních období.

```
1.         string[] pole = new string[] { "jaro", "léto", "podzim", "zima"
};
2.
3.         foreach (string b in pole)
4.            {
5.               Console.WriteLine(b);
6.            }
```

2.9 Zpracování chyb a výjimky

Pokud dojde k výjimce, tak je to v programování výjimečná stav, který může nastat za běhu programu a může způsobit k havárii programu. Mezi časté výjimky patří například dělení nulou, program nenajde soubor, se kterým má pracovat, nedostatek paměti nebo jiných systémových zdrojů. [16]

Odhalení chyb v programu:

- Chyby odhalí kompilátor
- Chyby, které zjistíme při prvním spuštění programu
- Chyby, které způsobí havárii programu jen občas

Reakce programu, pokud dojde k výjimečné situaci:

- Program provede automatickou opravu a pracuje dál, aniž by uživatele vyrušil
- Program informuje uživatele a nabídne mu obnovení chodu
- Program informuje uživatele a sám se ukončí
- Program se zhroutí [16]

2.9.1 Ošetření výjimek

C# pracuje s výjimkami na způsobu hlídání bloku kódu. Operace, které se nemusí podařit a může u nich dojít k výjimce, uzavřeme do bloku, před která napíšeme slovo *try*. Za něj pak napíšeme *handler* nebo koncovku. Hlídaný blok musí obsahovat minimálně jeden *handler* nebo koncovku. *Handlerů* může být několik, ale koncovka může být jen jedna. [15]

Syntaxe kódu pro hledaný blok.

```
1. Try {příkazy} handler koncovka
```

Handler je část programu, díky níž můžeme ošetřit výjimku.

Syntaxe *handleru* je:

```
1. Catch {typ identifikátor} {tělo handleru}
```

Za klíčové slovo *Catch* píšeme ve složených závorkách typ výjimky, kterou může tento *handler* ošetřit. Identifikátor obsahuje odkaz na objekt, který má informace o výjimce. V těle *handleru* jsou příkazy, které danou výjimku vyřeší. [16]

V následující tabulce je uveden seznam několika předefinovaných výjimek z knihovny .NET framework

Tabulka 8 Tabulka výjimek z knihovny .NET Framework [4], [17]

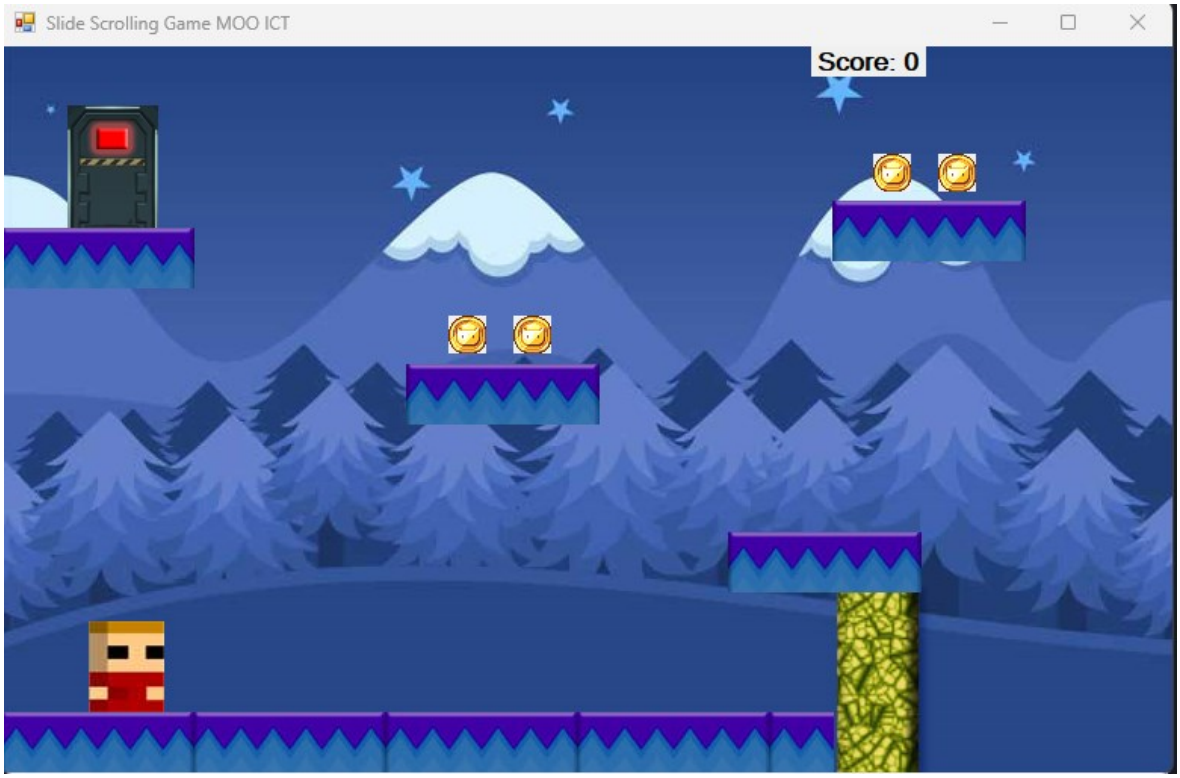
Výjimka	**Popis**
Exception	Základní třída pro všechny výjimky.
SystemException	Základní třída výjimek pro všechny druhy výjimek, které jsou generovány za běhu aplikace.
IndexOutOfRangeException	Přístup v prvku v poli mimo jeho rozsah.
NullReferenceException	Generována při pokusu pracovat s neinicializovanou funkcí.
ArgumentException	Základní třída pro všechny výjimky, které jsou způsobeny chybnými argumenty
ArgumentNullException	Je generována v případě, když neočekává parametr s null hodnotou.
InteropException	Výjimka, která je generována mimo CLR .NET Framework.
DivideByZeroException	Výjimka, která nevstane při pokusu dělení nulou
OutOfMemoryException	Vzniká, pokud není dostatek paměti
FileNotFoundException	Vzniká při pokusu otevřít neexistující soubor.

3 HRY VYTVOŘENÉ V JAZYCE C#

Tato kapitola se zabývá hrami vytvořenými v programovém prostředí Microsoft Visual Studio a naprogramovány v jazyce C#.

3.1 Side scrolling platform game

Side scrolling platform game je jednoduchá 2D hra ve které má hráč za úkol posbírat co největší množství peněz. Za sebrané peníze dostává hráč body, které se mu přičítají do skóre. Aby hráč ukončil hru, tak musí projít dveřmi. Dveře jsou ale zamčeny, proto musí hráč najít a sebrat klíč, pomocí kterého dveře odemkne. Pokud hráč projde dveřmi hra se ukončí a zobrazí se celkové skóre. Pokud hráč skočí mimo plošiny, na kterých může stát, tak spadne mimo mapu a umře. Jestli hráč umře, tak dojde k restartování hry a hráč přijde o všechno své skóre. Hra byla vytvořena v aplikaci Windows Forms (.NET Framework). [18]

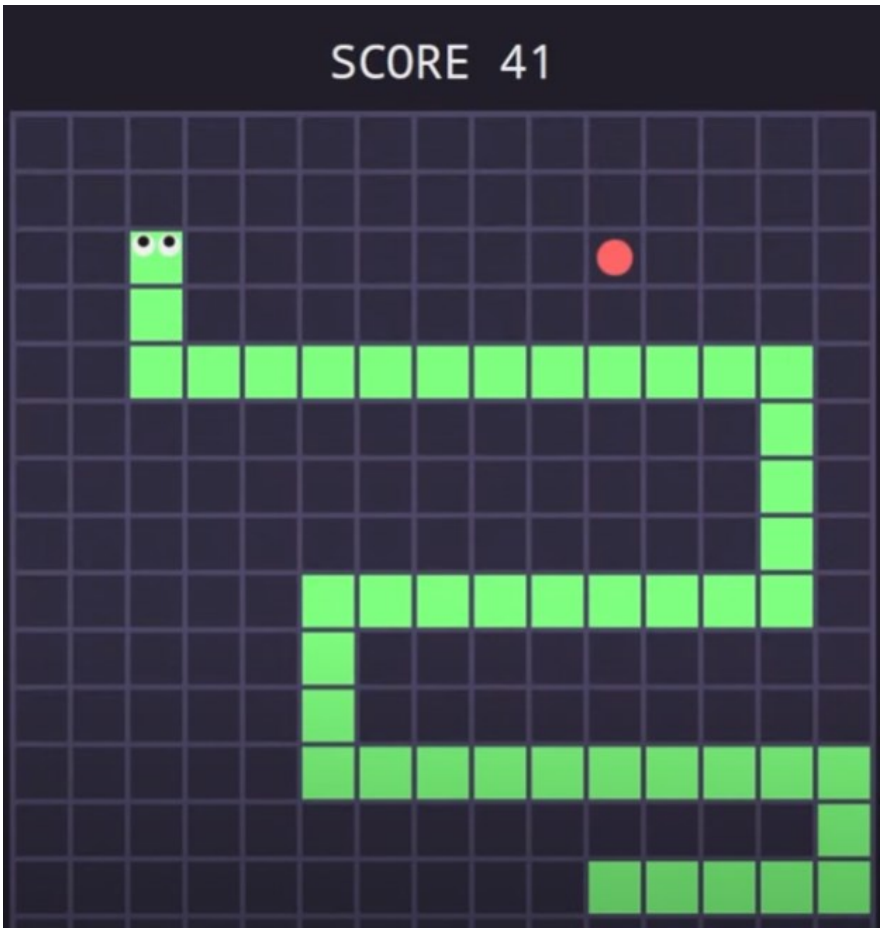


Obrázek 7 Side scrolling game [18]

3.2 Snake Game

Snake Game je jednoduchá 2D hra ve které hráč hraje za hada a jeho úkolem je sníst červenou tečku. Pokaždé když had sní červenou tečku, tak se jeho velikost zvětší o jeden

čtvereček. Hra se ukončí, jakmile had narazí do zdi nebo do vlastního těla. Hra je vytvořená v WPF Application. [19]



Obrázek 8 Snake Game [19]

3.3 Car racing game

Ve hře Car racing game hráč řídí bílé auto. Cílem hry je nasbírat co největší množství coinů, které se náhodně objevují na silnici. Aby hra byla obtížnější, tak se rychlost hráčova auta zvyšuje. Kromě coinů se na silnici náhodně objevují červené formule, kterým se hráč musí vyhýbat. Hra končí, jakmile hráč narazí do některé z červených formulí. Hra byla vytvořena ve Visual Studiu konkrétně v šabloně Windows Forms Application. [20]

Obrázek 9 Car racing game [20]

3.4 Helicopter Shooting Game

Jedná se o jednoduchou 2D počítačovou hru, ve které hráč pilotuje vrtulník. Ve hře se náhodně generují sloupy, kterým se hráč musí vyhýbat, pokud by do sloupu narazil, tak hra končí. Na mapě se také objevují nepřátelská vznášedla. Cílem hry je tato nepřátelská vznášedla sestřelit. Pokud hráč zasáhne nepřátelské vznášedlo, tak se mu přičte bod do skóre. V případě že nepřítel narazí do hráčova vrtulníku, tak hra končí. [21]

Obrázek 10 Helicopter Shooting Game [21]

II. PRAKTICKÁ ČÁST

4 SPECIFIKACE HRY

Tato kapitola je věnována popisu hry a jejím pravidlům

4.1 Popis hry

Thief Adventure je jednoduchá 2D vědomostní hra ve které hráč hraje za zloděje jehož cílem je získat dostatečný počet mincí v každé úrovni, aby mohl pokračovat do dalších úrovně. Hra obsahuje 5 úrovní. Více o hře je uvedeno v kapitole 4.2 Pravidla hry.

Hra obsahuje celkem 120 otázek. Hra obsahuje otázky z předmětů Krizové plánování a řízení, Systém bezpečnosti a veřejná správa a Bezpečnost informačních systémů. Otázky jsou z databáze vybírány náhodně, aby byla hra po každém spuštění jedinečná.

Aplikace obsahuje možnost si projít a odpovědět na jednotlivé otázky z databáze postupně.

Hru je možné spustit ve Visual Studiu nebo pomocí spustitelného souboru EXE.

4.2 Pravidla hry

Ve hře Thief adventure hraje hráč za postavu zloděje. V každé úrovni se nachází několik truhel a trezorů, které musí hráč otevřít. Za otevření truhel a trezorů dostává hráč mince. Cílem hry je otevřít co největší počet truhel a trezorů v jednotlivých úrovních pro získání mincí a dostat se do další úrovně. Pro postup do další úrovně je potřeba získat minimální počet mincí. S každou úrovní se potřebný počet mincí zvyšuje. Aby hráč otevřel truhlu nebo trezor musí je nejdřív odemknout, k tomu potřebuje paklíče. Pokud má hráč alespoň jeden paklíč může se pokusit o odemčení truhly nebo trezoru. Hráči se zobrazí otázka, na kterou musí odpovědět správně, aby truhlu nebo trezor odemkl, v opačném případě dojde ke zlomení paklíče a hráč o něj přijde. Pokud hráči nezbývají již žádné paklíče a nemá dostatek mincí pro postup do další úrovně, tak musí restartovat úroveň. V každé úrovni se nachází několik stráží, kteří hlídají truhly a trezory. Pokud stráže hráče chytnou, tak ho zatknou. V případě, kdy dojde k zatčení hráče, tak tím hráč přijde o všechny získané mince v dané úrovni a musí ji opakovat znovu od začátku.

Hra se ovládá pomocí myši a klávesnice. Šipky doleva a doprava, popřípadě klávesy A a D slouží pro pohyb hráče do stran. Šipkami nahoru a dolů nebo klávesami W a S může hráč šplhat po žebříku. Truhly a trezory se odemykají pomocí klávesy E nebo Enter.

5 REALIZACE HRY

Tato kapitola je věnována popisu hry z pohledu programátora.

5.1 Popis tvorby hry

Nejprve jsem se seznámil s prací ve Viusal Studiu. Následně jsem si našel hry, které byly vytvořeny ve Visual Studiu v jazyce C#. Hry jsou popsány v teoretické části.

Hra byla vyvíjena ve vývojovém prostředí Microsoft Visual Studio ve verzi 2022 v programovacím jazyce C#. Jako šablonu projektu jsem zvolil aplikaci Windows Forms (.NET Framework) a architekturu .NET Framework 4.8. Hra obsahuje celkem 8 formulářů Windows Forms a 2 excel soubory (databáze otázek a uložené hry)

5.2 Databáze otázek a odpovědí

Databázi otázek obsahuje otázky z předmětů Krizové plánování a řízení, Systém bezpečnosti a veřejná správa a Bezpečnost informačních systémů. Otázky jsem vytvořil podle mých znalostí, které jsem získal během studia daných předmětů a studijních materiálů. Otázky jsou uloženy v excel souboru Otazky.xlsx. Na každém řádku je vždy v první buňce zapsán název předmětu, ve druhé buňce je otázka ve třetí až šesté buňce jsou zapsány možné odpovědi na otázku a v sedmé buňce je zapsána správná odpověď.

Databázi otázek je možné jednoduše rozšířit dalšími otázkami zápisem na další řádky nebo nahrazením jiným excel souborem.

	A	B	C	D	E	F	G
31	Krizové plánování a řízení	Mezi vnitřní hrozby nepatří	Procesní hrozby	Sociální hrozby	Personální hrozby	Věcné hrozby	Sociální hrozby
32	Krizové plánování a řízení	Jaké má mít osobní vlastnosti auditor	objektivní, společenský, empati	objektivní, společenský, trpělivi	objektivní, nespolečenský, e	objektivní, společenský, em	objektivní, společenský, e
33	Krizové plánování a řízení	Vyberte co není výsledkem auditu:	Vyhovuje	Vyhovuje s podmínkou	Nevyhovuje	Všechny možnosti jsou správ	Všechny možnosti jsou sp
34	Krizové plánování a řízení	Jaké jsou základní funkce krizového mar	Prevence, Korekce, Protikrizová	Prevence, Taktika, Reakce, Obr	Prevence, Korekce, Protikrizo	Prevence, Připravenost, Rea	Prevence, Korekce, Protik
35	Krizové plánování a řízení	Vývojová stádia krize	Stadium symptomů, Akutní stadi	Stadium symptomů, Akutní sta	Stadium symptomů, Akutní s	Ani jedna z možností.	Stadium symptomů, Akutr
36	Krizové plánování a řízení	Vyberte jak jdou správně po sobě vývoj	1. Stadium symptomů 2. Akutní s	1. Stadium symptomů 2. Chron	1. Chronické stadium 2. Stadi	1. Akutní stadium 2. Stadium	1. Stadium symptomů 2. A
37	Krizové plánování a řízení	Specifika krizového managementu. Vyb	Nedostatek časum sil a prostřed	Krize vzniká obvykle nenadále	Základem je připravenost	Předvídatelnost vývoje krize	Předvídatelnost vývoje kri
38	Krizové plánování a řízení	Definice krizové komunikace	Výměna informací mezi odpově	Výměna informací mezi odpov	Výměna informací mezi odpo	Výměna informací mezi odp	Výměna informací mezi oc
39	Krizové plánování a řízení	Jaké jsou principy krizové komunikace?	Princip přímé odpovědnosti, Prin	Princip přímé odpovědnosti, Pr	Princip přímé odpovědnosti,	Princip nepřímé odpovědno	Princip přímé odpovědnos
40	Krizové plánování a řízení	Zásady úspěšné krizové komunikace. Vy	Nepovažujte „odkládání" za sprá	Držte se krizových plánů, nevy	Shodněte se s tiskem	Pravidelně aktualizujte krize	Držte se krizových plánů, r
41	Krizové plánování a řízení	Co nepatří mezi pravidla krizové komuni	Poskytovat omezené množství j	Využívat vizuální podpory.	Kontrolovat svou neverbální	Sdělení neopakovat.	Sdělení neopakovat.
42	Krizové plánování a řízení	Jaké jsou výhody Check listu? Vyberte c	Jednoduchost - lze snadno zkon	Účinnost - je velmi účinnou tec	Praktičnost - kontrolní seznar	Všechny možnosti jsou správ	Všechny možnosti jsou sp
43	Krizové plánování a řízení	Metoda What-if. Vyberte co není pravda	V praxi oblíbená, nenáročná na č	Složitější analytická technika p	Technika identifikace a analý	Je efektivní a účinná, pokud	Složitější analytická techni
44	Krizové plánování a řízení	Metoda FMEA. Vyberte co není pravda.	Využívá se v mnoha různých odv	Snižuje ztráty vyvolané nízkou	V případě složitějších systém	Zkracuje dobu řešení vývojo	V případě složitějších syst
45	Systém bezpečnosti a veře	Co je to bezpečnost?	Bezpečnost je stav, kdy jsou hro	Bezpečnost je stav bezpečnost	Bezpečnost je stav bezpečno	Ani jedna z možností.	Bezpečnost je stav bezpeč
46	Systém bezpečnosti a veře	Čemu ČR dává přednost u Bezpečnostní	Zastrašování zbrojením a demon	Aktivnímu přecházení ozbrojen	Aktivnímu předcházení ozbro	Ani jedna z možností.	Aktivnímu předcházení oz
47	Systém bezpečnosti a veře	Mezinárodní vztahy. Co je to unipolarita	Jeden stát významně převyšuje	Větší počet velmocí, jedna vyčr	Větší počet velmocí, jedna vy	Větší počet velmocí, jedna v	Jeden stát významně přev
48	Systém bezpečnosti a veře	Mezinárodní vztahy. Co je to Bipolarita?	Jeden stát významně převyšuje	Větší počet velmocí, jedna vyčr	Dvě ústřední supervelmoci rc	Dominující postavení tří a víc	Větší počet velmocí, jedna
49	Systém bezpečnosti a veře	Mezinárodní vztahy. Co je to Vyvážená r	Dvě ústřední supervelmoci rozd	Dominující postavení tří a více v	Jeden stát významně převyšu	Dvě ústřední supervelmoci r	Dominující postavení tří a v
50	Systém bezpečnosti a veře	Mezinárodní vztahy. Co je to Nevyvážen	Větší počet velmocí, jedna vyčr	Dvě ústřední supervelmoci roz	Jeden stát významně převyšu	Dvě ústřední supervelmoci r	Větší počet velmocí, jedna
51	Systém bezpečnosti a veře	Vyberte co nepatří mezi nástroje systém	Organizace kolektivní sebeobran	Smlouvy o odzbrojení.	Aktivní zahraniční politika.	Zbrojení proti nepříteli.	Zbrojení proti nepříteli.
52	Systém bezpečnosti a veře	Co je cílem organizace kolektivní bezpeč	Společný útok na nepřítele.	Společné zajištění kolektivní se	Společné zajištění kolektivní	Ani jedna z možností.	Společné zajištění kolektiv
53	Systém bezpečnosti a veře	Jak rozlišujeme nejzávažnější rizika bez	Vojenská rizika, politická rizika,	Dopravní rizika, ekonomická ri	Ovlivnitelná rizika, neovlivni	Vojenská rizika, nevojenská	Vojenská rizika, politická r
54	Systém bezpečnosti a veře	Kolik má Evropská unie členských států	EU má 27 členských států	EU má 30 členských států	EU má 24 členských států	EU má 18 členských států	EU má 27 členských států
55	Systém bezpečnosti a veře	Co je to proliferace zbraní ?	Opatření proti ničení zbraní.	Opatření proti dovozu a ničení	Opatření proti vývozu a rozšíi	Ani jedna z možností.	Opatření proti vývozu a ro
56	Systém bezpečnosti a veře	Kdy byla podepsána Versailleská mírová	28. dubna 1919	28. června 1919	28. června 1939	28. dubna 1939	28. června 1919
57	Systém bezpečnosti a veře	Ve kterém roce bylo zrušené Společens	v roce 1939	v roce 1846	v roce 1946	v roce 1945	v roce 1946
58	Systém bezpečnosti a veře	Datum vstupu ČR do EU	28. června 1939	1. ledna 1993	12. března 1999	1. května 2004	1. května 2004
59	Systém bezpečnosti a veře	Datum vstupu ČR do NATO	12. března 1999	28. června 1939	1. května 2004	1. ledna 1993	12. března 1999
60	Systém bezpečnosti a veře	Kdy vzniklo NATO?	28. června 1939	4. dubna 1949	12. března 1999	1. ledna 1993	4. dubna 1949
61	Systém bezpečnosti a veře	Kolik členů má parlament ČR	261 - 61 senátorů a 200 poslanců	271 - 71 senátorů a 200 poslanc	281 - 81 senátorů a 200 poslar	291 - 91 senátorů a 200 posla	281 - 81 senátorů a 200 po
62	Systém bezpečnosti a veře	Kdy bylo založeno OSN?	24. října 1945	4. dubna 1949	12. března 1945	28. června 1939	24. října 1945

Obrázek 11 Databáze otázek a odpovědí

5.3 Uložené hry

Informace o uložených hrách jsou uloženy v excel souboru SavedGame.xlsx. Na každém řádku v první sloupci je vždy uložena přezdívka hráče a v následujících sloupcích je vždy uložena informace o tom, jestli hráč dokončil danou úroveň nebo ne a získané mince v dané úrovni.

NickName	Level1	Coins	Level2	Coins	Level3	Coins	Level4	Coins	Level5	Coins
Hráč1	locked	0	locked	0	locked	0	locked	0	locked	0
Hráč2	unlocked	1200	locked	0	locked	0	locked	0	locked	0
Hráč3	unlocked	1200	unlocked	2200	locked	0	locked	0	locked	0
Hráč4	unlocked	1200	unlocked	2200	unlocked	3200	locked	0	locked	0
Hráč5	unlocked	1200	unlocked	2200	unlocked	3200	unlocked	4200	locked	0
Hráč6	unlocked	1000	locked	0	locked	0	locked	0	locked	0

Obrázek 12 Uložené hry

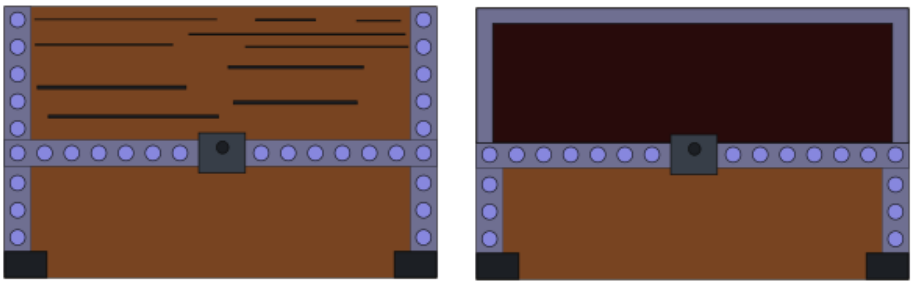
5.4 Grafický návrh hry

Pro vytvoření grafiky hry jsem použil program Inkscape. Inkscape je bezplatný a open source editor vektorové grafiky. [22]

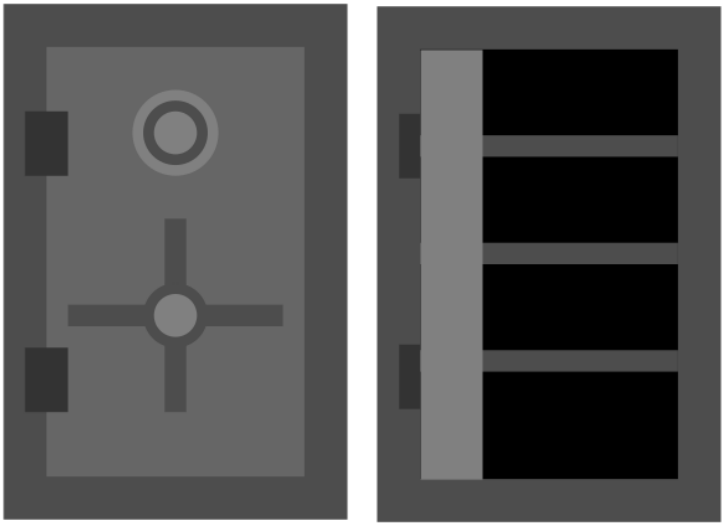
Nejdříve jsem vytvořil pozadí jednotlivých úrovní.

Obrázek 13 Pozadí úrovně 2

Následně jsem vytvořil obrázky pro truhly a trezory.

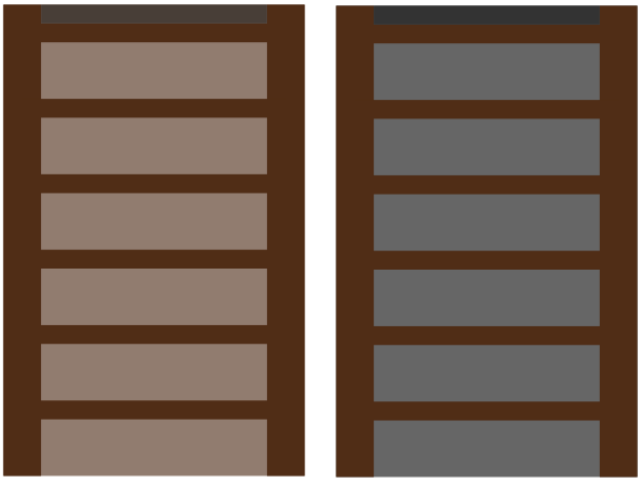


Obrázek 14 Grafický návrh zavřené a otevřené truhly



Obrázek 15 Grafický návrh zavřeného a otevřeného trezoru

Dále bylo potřeba vytvořit žebříky do úrovní.

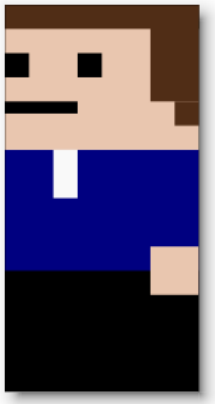


Obrázek 16 Grafický návrh žebříků

Následně jsem vytvořil grafický návrh postavy, kterou bude ovládat hráč.

Obrázek 17 Postava hráče

A nakonec jsem vytvořil grafický návrh stráží.



Obrázek 18 Postava stráže

5.5 Hlavní menu

Formulář MainMenu obsahuje tři tlačítka (spuštění hry, procvičení otázek, konec) a jedno textové pole pro zápis jména hráče.

Ukázka kódu:

```
1.          bool start = false;
2.          public static string nickname;
3.          string SaveName;
4.          string Level1 = "locked";
5.          string Level2 = "locked";
6.          string Level3 = "locked";
7.          string Level4 = "locked";
8.          string Level5 = "locked";
9.          int i = 1;
```

Po zapnutí hry se definují a deklarují proměnné, *nickname, Savename, Level1, Level2, Leve3, Level4, Level5 a i*;

- *start* – Tato proměnná slouží pro povolení nebo zakázání spuštění hry. Po spuštění hry se nastaví na hodnotu *false*, aby nebylo možné spustit hru bez zadání jména hráče.
- *nickname* – V této proměnné je uloženo jméno hráče, které hráč zadal do textového pole.
- *SaveName* – Uložená jména ze souboru SavedGame.xlsx.
- *Level1, Level2, Level3, Level4* a *Level5* – Do těchto proměnných se ukládá informace, jestli hráč dokončil danou úroveň. Po zapnutí hry se do těchto proměnných uloží slovo „locked“ (locked – úroveň nebyla dokončena, unlocked – úroveň byla dokončena)
- *i* – Pomocná proměnná, která obsahuje číslo řádku v excel souboru s uloženými jmény hráčů.

5.5.1 StartButton (Spustit hru)

Po kliknuté na tlačítko *spustit hru* se nejdřív vyhodnotí podmínka, jestli je textové pole prázdné. Pokud je prázdné (hráč nenapsal své jméno do textového pole), tak se objeví okno se zprávou, aby hráč zadal své jméno. A proměnná *start* se nastaví na hodnotu *false,* aby nebylo možné spustit hru.

Ukázka kódu:

```
1.              if (textBoxNickName.Text == "")
2.              {
3.                  MessageBox.Show("Zadejte jméno hráče do textového pole v
levém horním rohu.");
4.                  start = false;
5.              }
```

V případě, kdy textové pole není prázdné, tak se do proměnné *nickname* vloží řetězec z textového pole. Následně se otevře excel soubor SavedGame.xlsx (souboru jsou uloženy jména hráčů a počet mincí, které získali v jednotlivých úrovních) a do proměnné *SaveName* se uloží první jméno z tabulky.

Následně probíhá cyklus *while* do té doby, než se najde shoda mezi zadaným jménem do textového pole a jménem hráče uloženého v databázi hráčů nebo dokud cyklus nenarazí na prázdné políčko v databázi.

V případě, kdy se zadané jméno shoduje s některým jménem z databáze, tak se excel soubor zavře, proměnná *start* se nastaví na hodnotu *true* a opakování cyklu se zastaví pomocí příkazu *break*.

V případě, kdy cyklus dojde až k prázdné buňce v databázi hráčů, tak se do databáze na tento řádek, kde se nachází prázdná buňka vytvoří záznam o novém hráči (Do souboru SavedGame.xlsx se uloží na nový řádek jméno hráče, k jednotlivým úrovním se napíše slovo „locked“ a k získaným mincím v jednotlivých úrovních se napíše hodnota 0).

Ukázka kódu:

```
            else
            {
                nickname = textBoxNickName.Text;

                Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\SavedGame.xlsx", 1);
                SaveName = excel.ReadCell(1, 0);

                while (true)
                {
                    if (SaveName == nickname)
                    {
                        MessageBox.Show("hráč nalezen" +
Environment.NewLine + "Pokracujte kliknutim na OK");
                        excel.Close();
                        start = true;
                        break;
                    }
                    else if (SaveName == "")
                    {
                        excel.WriteToCell(i, 0, nickname);
                        excel.WriteToCell(i, 1, "locked");
                        excel.WriteToCell(i, 2, "0");
                        excel.WriteToCell(i, 3, "locked");
                        excel.WriteToCell(i, 4, "0");
                        excel.WriteToCell(i, 5, "locked");
                        excel.WriteToCell(i, 6, "0");
                        excel.WriteToCell(i, 7, "locked");
                        excel.WriteToCell(i, 8, "0");
                        excel.WriteToCell(i, 9, "locked");
                        excel.WriteToCell(i, 10, "0");
                        excel.Save();
                        excel.Close();
                        start = true;
                        break;
                    }
                    else
                    {
                        i++;
                        SaveName = excel.ReadCell(i, 0);
                    }
```

```
40.                 }
41.             }
```

Následně se vyhodnotí podmínka, jestli se proměnná *start* rovná hodnotě *true.* Pokud ano, tak se otevře excel soubor SavedGame.xlsx a do proměnných *Level1* až *Level5* se uloží informace, jestli hráč danou úroveň již dokončil.

Následně se zjišťuje, která z proměnných *Level5* až *Level1* se rovná hodnotě „unlocked“. Podle toho, ve které úrovni hráč skončil se následně zobrazí message box s dotazem, jestli chce hráč pokračovat v dané úrovni nebo ne. Pokud hráč zvolí „Ano“, tak se spustí daná úroveň. Když hráč zvolí „Ne“, tak se zobrazí message box s dotazem, jestli chce hráč spustit úroveň 1. Po kliknutí na „Ano“ se zavolá funkce *RestartGame* (dojde k vymazání herního postupu) a spustí se první úroveň.

Pokud hráč nedokončil žádnou s úrovní, tak se automaticky zavolá funkce *Level1Start* (spustí se první úroveň).

Ukázka kódu:

```
 1.          if (start == true)
 2.             {
 3.                 Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\SavedGame.xlsx", 1);
 4.                 Level1 = excel.ReadCell(i, 1);
 5.                 Level2 = excel.ReadCell(i, 3);
 6.                 Level3 = excel.ReadCell(i, 5);
 7.                 Level4 = excel.ReadCell(i, 7);
 8.                 Level5 = excel.ReadCell(i, 9);
 9.                 excel.Close();
10.
11.                 if (Level5 == "unlocked")
12.                 {
13.                     if (MessageBox.Show("Hra byla dokončena. Přejete si
začít znovu ?", "Začít znovu ?", MessageBoxButtons.YesNo) ==
DialogResult.Yes)
14.                     {
15.                         if (MessageBox.Show("Určitě chcete pokračovat ?
Dosažený herní postup a skóre bude smazáno.", "Začít znovu ?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
16.                         {
17.                             RestartGame();
18.                             Level1Start();
19.                         }else
20.                         {
21.                             start = false;
22.                             i = 1;
23.                         }
24.                     }
25.                     else
26.                     {
27.                         start = false;
```

```
28.                         i = 1;
29.                     }
30.                 }
31.
32.                 else if (Level4 == "unlocked")
33.                 {
34.                     if (MessageBox.Show("Přejete si načíst úroveň 5?",
"Načíst úroveň 5?", MessageBoxButtons.YesNo) == DialogResult.Yes)
35.                     {
36.                         Level5 frmLevel5 = new Level5();
37.                         frmLevel5.ShowDialog();
38.                         this.Hide();
39.                     }
40.                     else if (MessageBox.Show("Přejete si načíst úroveň
1? Dosažený herní postup a skóre bude smazáno. ", "Začít znovu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
41.                     {
42.                         RestartGame();
43.                         Level1Start();
44.                     }
45.                     else
46.                     {
47.                         start = false;
48.                         i = 1;
49.                     }
50.                 }
51.
52.                 else if (Level3 == "unlocked")
53.                 {
54.                     if (MessageBox.Show("Přejete si načíst úroveň 4?",
"Načíst úroveň 4?", MessageBoxButtons.YesNo) == DialogResult.Yes)
55.                     {
56.                         Level4 frmLevel4 = new Level4();
57.                         frmLevel4.ShowDialog();
58.                         this.Hide();
59.                     }
60.                     else if (MessageBox.Show("Přejete si načíst úroveň
1? Dosažený herní postup a skóre bude smazáno. ", "Začít znovu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
61.                     {
62.                         RestartGame();
63.                         Level1Start();
64.                     }
65.                     else
66.                     {
67.                         start = false;
68.                         i = 1;
69.                     }
70.                 }
71.
72.                 else if (Level2 == "unlocked")
73.                 {
74.                     if (MessageBox.Show("Přejete si načíst úroveň 3?",
"Načíst úroveň 3?", MessageBoxButtons.YesNo) == DialogResult.Yes)
75.                     {
76.                         Level3 frmLevel3 = new Level3();
```

```
 77.                         frmLevel3.ShowDialog();
 78.                         this.Hide();
 79.                     }
 80.                     else if (MessageBox.Show("Přejete si načíst úroveň
1? Dosažený herní postup a skóre bude smazáno. ", "Začít znovu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
 81.                     {
 82.                         RestartGame();
 83.                         Level1Start();
 84.                     }
 85.                     else
 86.                     {
 87.                         start = false;
 88.                         i = 1;
 89.                     }
 90.                 }
 91.
 92.                 else if (Level1 == "unlocked")
 93.                 {
 94.                     if (MessageBox.Show("Přejete si načíst úroveň 2?",
"Načíst úroveň 2?", MessageBoxButtons.YesNo) == DialogResult.Yes)
 95.                     {
 96.                         Level2 frmLevel2 = new Level2();
 97.                         frmLevel2.ShowDialog();
 98.                         this.Hide();
 99.                     }
100.                     else if (MessageBox.Show("Přejete si načíst úroveň
1? Dosažený herní postup a skóre bude smazáno. ", "Začít znovu?",
MessageBoxButtons.YesNo) == DialogResult.Yes)
101.                     {
102.                         RestartGame();
103.                         Level1Start();
104.                     }
105.                     else
106.                     {
107.                         start = false;
108.                         i = 1;
109.                     }
110.                 }
111.                 else
112.                 {
113.                     Level1Start();
114.                 }
115.             }
116.         }
```

5.5.2 Ostatní funkce a tlačítka

Funkce *Level1Start* slouží pro otevření formuláře pro první úroveň.

Ukázka kódu:

```
1.         private void Level1Start()
2.         {
3.             Level1 frmLevel1 = new Level1();
4.             frmLevel1.ShowDialog();
```

```
5.              this.Hide();
6.          }
```

Po zavolání funkce *RestartGame* dojde k vymazání herního postupu.

```
 1.          private void RestartGame()
 2.          {
 3.              Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\SavedGame.xlsx", 1);
 4.              excel.WriteToCell(i, 1, "locked");
 5.              excel.WriteToCell(i, 2, "0");
 6.              excel.WriteToCell(i, 3, "locked");
 7.              excel.WriteToCell(i, 4, "0");
 8.              excel.WriteToCell(i, 5, "locked");
 9.              excel.WriteToCell(i, 6, "0");
10.              excel.WriteToCell(i, 7, "locked");
11.              excel.WriteToCell(i, 8, "0");
12.              excel.WriteToCell(i, 9, "locked");
13.              excel.WriteToCell(i, 10, "0");
14.              excel.Save();
15.              excel.Close();
16.          }
```

Po kliknutí na tlačítko Procvičení otázek se otevře formulář *OtazkyCviceni.*

Ukázka kódu:

```
1.          private void cviceni_Click(object sender, EventArgs e)
2.          {
3.              OtazkyCviceni otazkyCV = new OtazkyCviceni();
4.              otazkyCV.ShowDialog();
5.              this.Hide();
6.          }
```

Po kliknutí na tlačítko *Konec* nebo na červený křížek v pravém horním rohu dojde k ukončení aplikace.

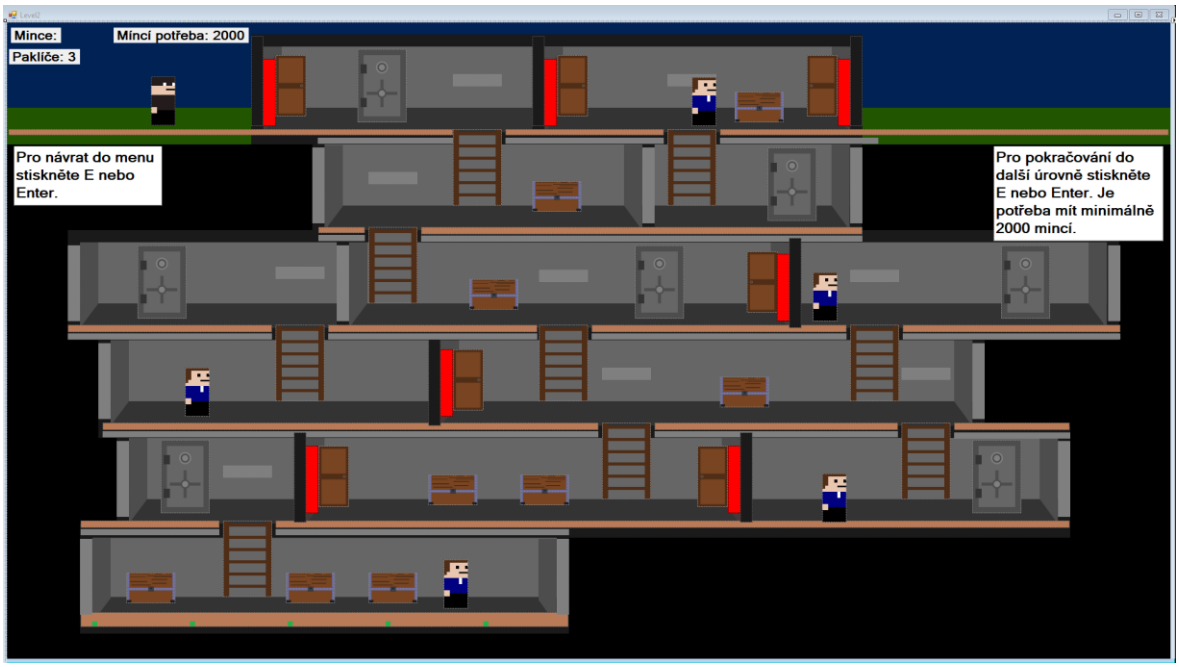
Ukázka kódu:

```
 1.          private void CloseGame(object sender, FormClosedEventArgs e)
 2.          {
 3.              Application.Exit();
 4.          }
 5.
 6.          private void EndButton_Click(object sender, EventArgs e)
 7.          {
 8.              Application.Exit();
 9.          }
```

5.6 Úrovně

Formuláře jednotlivých úrovní jsou tvořeny několika picture boxy (pozadí, postava hráče, postavy nepřátel, žebříky, truhly, trezory, dveře…), text boxy (informační zprávy pro hráče), labely (mince získané za danou úroveň, počet paklíčů). Pomocí timer (časovač) se nastavuje rychlost hry. Kód pro jednotlivé úrovně je podobný.

Všechny následující ukázky kódu jsou vzíty z úrovně 2.



Obrázek 19 Ukázka návrhu formuláře druhé úrovně

Po otevření formuláře jednotlivých úrovní se deklarují a definují proměnné *goLeft, goRight, goUp, goDown, climbing, use, answer, coin, lockpick, enemySpeed, playerSpeed, SetPlayerSpeed, playerSpeedUpDown, SetPlayerSpeedUpDown, gravity, SetGravity SaveName a nickname a hitWall.*

Ukázka kódu:

```
1.          bool goLeft, goRight, use, goUp, goDown, climbing, answer;
2.
3.          int coin = 0;
4.          int lockpick = 3;
5.
6.          int SetPlayerSpeed = 5;
7.          int SetPlayerSpeedUpDown = 5;
8.          int SetGravity = 10;
9.
10.         int enemySpeed1 = 5;
11.         int enemySpeed2 = 5;
12.         int enemySpeed3 = 5;
```

```
13.         int enemySpeed4 = 3;
14.         int enemySpeed5 = 3;
15.
16.         string SaveName;
17.         string nickname = MainMenu.nickname;
18.         int playerSpeed;
19.         int playerSpeedUpDown;
20.         int hitWall = 0;
21.         int gravity;
```

- *goLeft, goRight, goUp, goDown* – tyto proměnné slouží pro ovládání postavy hráče (pohyb postavy doleva, doprava, nahoru a dolů)
- *climbing* – pomocná proměnná, která informuje že se postava hráče nachází na žebříku
- *use* – proměnná pro otevírání truhel a trezorů
- *answer* – pomocná proměnná, která informuje o tom, jestli hráč správně odpověděl na otázku
- *SetPlayerSpeedUpDown* – nastavení rychlost pohybu hráče po žebříku
- *playerSpeedUpDown* – rychlost hráče na žebříku
- *SetPlayerSpeed* – nastavení rychlosti pohybu hráče
- *playerSpeed* – rychlost pohybu hráče
- *Coin* – počet nasbíraných mincí
- Lockpick – počet paklíčů
- *enemySpeed* = nastavení rychlosti pohybu stráží
- *SaveName* – do této proměnné se ukládají jména hráčů z excel souboru SavedGame.xlxs
- *Nickname* – v této proměnné je uloženo jméno hráče, z textového pole v hlavním menu
- *hitWal*l – pomocná proměnná, která informuje o tom jestli hráč narazil do zdi (0 – hráč nenarazil do zdi, 1- hráč narazil do zdi vpravo, 2 – hráč narazil do zdi vlevo)
- gravity – rychlost jakou bude hráč padat, pokud skočí z žebříku

5.6.1 Level2_Load

Po spuštění úrovně se nastaví rychlost pohybu hráče po zemi a žebříku.

Ukázka kódu:

```
1.          private void Level2_Load(object sender, EventArgs e)
2.          {
3.              playerSpeed = SetPlayerSpeed;
4.              playerSpeedUpDown = SetPlayerSpeedUpDown;
5.          }
```

5.6.2 MainTimerEvent

V této funkci se nastavují všechny události ve hře.

5.6.2.1 Mince a paklíče

Do textového pole se *txtScore* se zapíše slovo „Mince“ a hodnota uložená v proměnné *coin*. A do textového pole *txtLockpick* se zapíše slovo „Paklíče“ a hodnota uložená v proměnné *lockpick.*

Ukázka kódu:

```
1.              txtScore.Text = "Mince: " + coin;
2.              txtLockpick.Text = "Paklíče: " + lockpick;
```

5.6.2.2 Pohyb postavy hráče

Nejdříve se vyhodnotí podmínka, jestli se proměnná *climbing* rovná hodnotě *false* (to znamená že hráčova postava nešplhá po žebříku). Pokud se, proměnná *climbing* rovná *false* tak se k ypsilonové souřadnici hráče přičte hodnota uložená v proměnné *gravity* to znamená, že dojde k posunutí postavy hráče směrem dolů.

Další podmínka vyhodnocuje, jestli se proměnná *goLeft* rovná hodnotě *true* (hráč zmáčkl šipku doleva na klávesnici) a zároveň jestli je hráčova postava od levého kraje vzdálena víc jak 40 pixelů. Pokud ano, tak se od x-ové souřadnice postavy hráče odečte hodnota uložená v proměnné *playerSpeed*. To znamená že dojde k posunutí postavy hráče směrem doleva. A změní se obrázek postavy hráče pro pohyb doleva.

Následující podmínka je podobná té předchozí, ale tentokrát se jedná o pohyb postavy směrem doprava.

Ukázka kódu:

```
1.              if (climbing == false)
```

```
 2.              {
 3.                  player.Top += gravity;
 4.              }
 5.
 6.              if (goLeft == true && player.Left > 40)
 7.              {
 8.                  player.Left -= playerSpeed;
 9.                  player.Image = Properties.Resources.player_L;
10.              }
11.
12.              if (goRight == true && player.Left + (player.Width + 40) <
this.ClientSize.Width)
13.              {
14.                  player.Left += playerSpeed;
15.                  player.Image = Properties.Resources.player_R;
16.              }
17.
18.             gravity = SetGravity;
```

5.6.2.3 Gravitace

Následující podmínky vyhodnocují, jestli se hráčova postava stojí na nějaké platformě (hráčův picture box se dotýká nějakého picture boxu, který je označen tagem „platform“). Pokud ano tak se do *player.Top* (ypsilonová souřadnice hráče) uloží hodnota která se rovná ypsilonové souřadnici platformy – výšku hráče + 5 (hodnota +5 se přičítá z důvodu, aby se hráčova postava stále dotýkala platformy). Pokud hráčova postava stojí na platformě a zároveň hráč zmáčkne šipku dolů (*goDown* = *true*), tak se proměnná *playerSpeedUpDown* nastaví na hodnotu 0, aby nedošlo k zaseknutí postavy hráče.

Ukázka kódu:

```
 1.              foreach (Control x in this.Controls)
 2.              {
 3.                  if (x is PictureBox && (string)x.Tag == "platform")
 4.                  {
 5.                      if (player.Bounds.IntersectsWith(x.Bounds) && goDown
== true)
 6.                      {
 7.                          playerSpeedUpDown = 0;
 8.                          player.Top = x.Top - player.Height + 5;
 9.                          gravity = 0;
10.                          goDown = false;
11.                      }
12.                      else if  (player.Bounds.IntersectsWith(x.Bounds))
13.                      {
14.                          player.Top = x.Top - player.Height + 5;
15.                          gravity = 0;
16.                          goDown = false;
17.                          playerSpeedUpDown = SetPlayerSpeedUpDown;
18.                      }
19.                  }
```

5.6.2.4 Šplhání po žebříku

Další podmínka vyhodnocuje, jestli se hráčova postava dotýká žebříku. Pokud ano tak se vyhodnocují další podmínky.

Aby hráčova postava mohla šplhat po žebříku směrem nahoru, tak se musí dotýkat žebříku a zároveň se proměnná *goUp* musí rovnat hodnotě *true* (hráč drží šipku nahoru na klávesnici). Pokud se *goUp* rovná hodnotě *true* tak se proměnná *climbing* nastaví na hodnotu *true*, do proměnné gravity se uloží hodnota 0 a od ypsilonové souřadnice hráče se bude odečítat hodnota uložená v proměnné *playerSpeedUpDown* (hráč se bude pohybovat směrem nahoru)

Následující podmínka je podobná té předchozí s rozdílem že proměnná *goDown* se musí rovnat hodnotě *true* (hráč drží šipku dolů na klávesnici). Proměnné *climbing* a gravity se nastaví na stejné hodnoty jako u předchozí podmínky a k ypsilonové souřadnici postavy hráče se bude přičítat hodnota uložená v proměnné *playerSpeedUpDown* (postava hráče se bude pohybovat směrem dolů).

Další podmínka vyhodnocuje případ, kdy se hráčova postava pouze dotýká nebo stojí na žebříku (hráč nedrží šipku nahoru ani dolů). V tomto případě se proměnné *climbing* a *gravity* nastaví na stejné hodnoty jako to bylo u předchozích podmínek.

Pokud neplatí ani jedna z podmínek tak se proměnná *climbing* nastaví na hodnotu *false* (hráč nešplhá po žebříku)

Jakmile se hráčova postava dotkne nějakého picture boxu označeného tagem „MaxHeight“, tak se proměnná *goUp* nastaví na hodnotu *false*, aby nedošlo k zaseknutí postavy hráče, když vylezl po žebříku do maximální výšky.

Ukázka kódu:

```
 1.                 if (x is PictureBox && (string)x.Tag == "ladder")
 2.                 {
 3.
 4.                     if (player.Bounds.IntersectsWith(x.Bounds) && goUp
== true)
 5.                     {
 6.                         climbing = true;
 7.                         player.Top -= playerSpeedUpDown;
 8.                         gravity = 0;
 9.                     }
10.                     else if (player.Bounds.IntersectsWith(x.Bounds) &&
goDown == true)
11.                     {
12.
```

```
13.                         climbing = true;
14.                         player.Top += playerSpeedUpDown;
15.                         gravity = 0;
16.                     }
17.                     else if (player.Bounds.IntersectsWith(x.Bounds))
18.                     {
19.                         climbing = true;
20.                         gravity = 0;
21.                     }
22.                     else
23.                     {
24.                         climbing = false;
25.                     }
26.                 }
27.
28.                 if (x is PictureBox && (string)x.Tag == "MaxHeight")
29.                 {
30.                     if (player.Bounds.IntersectsWith(x.Bounds))
31.                     {
32.                         if (goUp == true)
33.                         {
34.                             goUp = false;
35.                         }
36.                     }
37.                 }
```

5.6.2.5 Narazení do zdi

V případě, kdy se hráčova postava narazí do stěny, tak se vyhodnocují podmínky, jestli šel hráč doprava nebo doleva.

V případě, kdy šel hráč doprava (proměnná *goRight* se rovná hodnotě *true*) a narazí do zdi, tak se hráčova postava posune o pět pixelů směrem doleva, aby nedošlo k zaseknutí postavy hráče do zdi. Následně se hráčova rychlost postavy nastaví na nulu a pomocná proměnná *hitWall* na 1 (0 – hráč se nedotýká zdi, 1 – hráč narazil do zdi na pravé straně, 2 - hráč narazil do zdi na levé straně, 3 – hráč narazil do stropu)

Pokud hráč půjde doleva (*goLeft* se rovná *true*) a narazí do zdi tak se hráčova postava posune o 5 pixelů směrem doprava. Opět se rychlost hráče nastaví na nulu a proměnná *hitWall* se nastaví na 2.

Pokud hráč šplhá po žebříku a narazí do stropu, tak se proměnná playerSpeedUpDown nastaví na hodnotu 0 a hráčova postava se posune o 5 pixelů směrem dolů. Proměnná *hitWall* se nastaví na hodnotu 3.

Dále se vyhodnocují podmínky, jestli jde hráč směrem od stěny, do které narazil. V případě že ano, tak se proměnná *hitWall* nastaví na hodnotu 0 a hráčova rychlost pohybu po zemi a žebříku se nastaví na původní hodnoty.

Ukázka kódu:

```
 1.                 if (x is PictureBox && (string)x.Tag == "wall")
 2.                 {
 3.
 4.                     if (player.Bounds.IntersectsWith(x.Bounds))
 5.                     {
 6.                         if (goRight == true)
 7.                         {
 8.                             player.Left -= 5;
 9.                             playerSpeed = 0;
10.                             goRight = false;
11.                             hitWall = 1;
12.                         }
13.                         else if (goLeft == true)
14.                         {
15.                             player.Left += 5;
16.                             playerSpeed = 0;
17.                             goLeft = false;
18.                             hitWall = 2;
19.                         }
20.                         else if (goUp == true)
21.                         {
22.                             player.Top += 5;
23.                             playerSpeedUpDown = 0;
24.                             goUp = false;
25.                             hitWall = 3;
26.                         }
27.                     }
28.                 }
29.
30.                 if (hitWall == 1 && (goLeft == true || goUp ==true ||
goDown == true))
31.                 {
32.                     playerSpeed = SetPlayerSpeed;
33.                     hitWall = 0;
34.                 }
35.                 else if (hitWall == 2 && (goRight == true || goUp ==
true || goDown == true))
36.                 {
37.                     playerSpeed = SetPlayerSpeed;
38.                     hitWall = 0;
39.                 }
40.                 else if (hitWall == 3 && (goRight == true || goLeft ==
true ||goDown == true))
41.                 {
42.                     playerSpeedUpDown = SetPlayerSpeedUpDown;
43.                     hitWall = 0;
44.                 }
```

5.6.2.6 Opuštění hry

Pokud se hráč nachází od levého okraje méně jak 60 pixelů, tak se zobrazí text box s dotazem, jestli se chce hráč vrátit do hlavního menu. Pokud hráč zmáčkne E nebo Enter na klávesnici, tak se zavolá funkce *RestartGame.* Která je více popsána v kapitole 5.6.5.

Ukázka kódu

```
            if (player.Left < 60)

            {
                TextExitMenu.Visible = true;

                if (use == true)
                {
                    use = false;
                    goRight = false;
                    goLeft = false;
                    RestartGame();
                }
            }
            else
            {
                TextExitMenu.Visible = false;
            }
```

5.6.2.7 Postup do další úrovně

První se vyhodnocuje podmínka, jestli má hráčova postava x-ové souřadnice větší jak 1800. Pokud má tak se zobrazí text box s informací, kolik potřebuje hráč minimálně mincí, aby mohl postoupit do další úrovně. Pokud ne, tak se textový box skryje.

V případě, kdy je zobrazen textový box, tak se vyhodnocuje podmínka, jestli se proměnná *use* rovná *true* (Hráč zmáčkl písmeno E nebo Enter na klávesnici). Dále se vyhodnocuje podmínka, jestli hráč má požadovaný počet mincí, pro postup do další úrovně. Pokud ano, tak se následně otevře excel soubor se jmény hráčů a bude se postupně procházet databáze s uloženými jmény, dokud se nebude rovna proměnná *SaveName* (jména uložená v excel souboru SavedGame.xlxs) s proměnou *nickname* (jméno hráče). Jakmile se tyto dvě proměnné budou rovnat, tak se do excelu na řádek, na kterém se nachází jméno hráče zapíše do třetí buňky slovo *unlock*ed (hráč dokončil druhou úroveň) a do čtvrté buňky počet mincí, které hráč získal v dané úrovni. Následně se otevře formulář s další úrovní.

V případě že hráč nebude mít dostatečný počet mincí pro postup do další úrovně, tak se zobrazí message box, který informuje hráče, že nemá dostatečný počet mincí.

Ukázka kódu:

```
            if (player.Left > 1800)

            {
                TextNextLevel.Visible = true;

                if (use == true)
                {
                    use = false;
                    goRight = false;
                    goLeft = false;
                    if (coin >= 2000)
                    {

                        Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\SavedGame.xlsx", 1);
                        int i = 1;
                        SaveName = excel.ReadCell(1, 0);

                        while (true)
                        {
                            if (SaveName == nickname)
                            {
                                excel.WriteToCell(i, 3, "unlocked");
                                excel.WriteToCellNumber(i, 4, coin);
                                excel.Save();
                                excel.Close();
                                break;
                            }
                            else
                            {
                                i++;
                                SaveName = excel.ReadCell(i, 0);
                            }
                        }
                        Level3 nextlevel = new Level3();
                        nextlevel.Show();
                        this.Hide();
                    }
                    else
                    {
                        MessageBox.Show("Nemáte dostatek mincí pro
postup do další úrovně.");
                    }
                }
            }
            else
            {
                TextNextLevel.Visible = false;
            }

```

5.6.2.8 Pokud hráče chytne nepřítel

Pro zjištění, jestli byl hráč zatknut stráží se nejprve vyhodnocuje podmínka, jestli se hráčův picture box dotknul nějakého picture boxu označeného tagem „enemy“. Jakmile se nějaká

postava stráží dotkne postavy hráče, tak se herní čas zastaví a hráči se pomocí message boxu zobrazí zpráva s informací, že byl zatčen, a nakonec se zavolá funkce pro restartování hry, která je více popsaná v kapitole 5.6.5.

Ukázka kódu:

```
1.                 if ((string)x.Tag == "enemy")
2.                 {
3.                     if (player.Bounds.IntersectsWith(x.Bounds))
4.                     {
5.                         GameTimer.Stop();
6.                         MessageBox.Show("Byl jsi zatcen!" +
Environment.NewLine + "Klikni Ok pro restart hry");
7.                         RestartGame();
8.                     }
9.                 }
```

5.6.2.9 Pohyb nepřátel

Kód pro všechny nepřátele je podobný, vždy záleží na dané úrovni a patře v jakém se nepřítel nachází.

Jako první se nastaví rychlost nepřátel a směr jakým se budou pohybovat po zapnutí dané úrovně (*enemySpeed* – kladné číslo směr doprava, záporné číslo směr doleva) Dále se vyhodnocuje podmínka, ve které je uvedeno, v jakém případě se má postava nepřítele otočit. Je to například pokud nepřítel narazí do zdi, do zavřených dveří nebo jen do místa kde se dveře nachází.

V případě, kdy je aspoň jedna z těchto podmínek pravda, tak se rychlost nepřítele nastaví na opačnou hodnotu a nepřítel se bude pohybovat opačným směrem.

V další podmínce se vyhodnocuje, jestli je proměnná *enemySpeed* větší nebo menší než nula. V případě, kdy je větší, než nula tak se ze složky s obrázky nahraje na picture box daného nepřítele obrázek pojmenovaný „nepritel_P“ (nepřítel jde směrem doprava) v opačném případě se nahraje obrázek pojmenovaný „nepritel_L“ (nepřítel jde směrem doleva)

Ukázka kódu:

```
1. enemy2.Left += enemySpeed2;
2.
3.             if (enemy2.Bounds.IntersectsWith(wall5.Bounds) ||
enemy2.Bounds.IntersectsWith(wall6.Bounds) ||
enemy2.Bounds.IntersectsWith(door4.Bounds) && door4.Enabled == true)
4.             {
5.                 enemySpeed2 = -enemySpeed2;
```

```
 6.
 7.                 if (enemySpeed2 > 0)
 8.                     enemy2.Image = Properties.Resources.nepritel_p;
 9.                 else
10.                     enemy2.Image = Properties.Resources.nepritel_L;
11.             }
```

5.6.2.10 Odemčení dveří

Aby došlo k odemčení dveří, tak se první zjišťuje jestli proměnná *door.Enabled* se rovná hodnotě *true* (*true* – dveře jsou zamčeny). Následně se vyhodnocuje, jestli se postava hráče dotkla zamčených dveří. Pokud ano tak se zavolá funkce *UnlockDoor*, pomocí které zjistíme, jestli hráč odpověděl správně na otázku. Pokud hráč odpoví správně, tak se proměnná *answer* nastaví na hodnotu *true*. Funkce *UnlockDoor* je více popsaná v kapitole 5.6.3

V další podmínce se vyhodnocuje, jestli se proměnná *answer* rovná hodnotě *true*. Pokud je to pravda, tak se nastaví proměnná *door.Enabled* na hodnotu *false* (false – dveře jsou odemčeny) a picture box pro otevřené dveře se nastaví na viditelný.

Ukázka kódu:

```
 1.             if (door1.Enabled == true)
 2.             {
 3.                 if (player.Bounds.IntersectsWith(door1.Bounds))
 4.                 {
 5.                     UnlockDoor();
 6.
 7.                     if (answer == true)
 8.                     {
 9.                         door1.Enabled = false;
10.                         openDoor1.Visible = true;
11.                     }
12.                 }
13.             }
```

5.6.2.11 Odemčení truhel a trezorů

Odemykání truhel a trezorů funguje na stejném principu. První se vyhodnocuje podmínka, jestli hráč stojí u truhly nebo trezoru, dále se musí proměnná *use* rovnat hodnotě *true* (hráč zmáčkl písmeno E na klávesnici) a proměnná *Enabled* truhly nebo trezoru musí být nastavena na hodnotě *true* (*true* – truhla je zamčena).

Pokud je vše splněno, tak se zavolá funkce *UnlockSafe* ve které se vyhodnotí, jestli hráč odpověděl správně na otázku. Pokud odpoví správně tak se proměnná *answer* nastaví na hodnotu *true*. Funkce *UnlockSafe* je více popsána v kapitole 5.6.4.

V další podmínce se vyhodnocuje, jestli se proměnná *answer* rovná hodnotě *true*. Pokud ano tak se proměnná *Enabled* dané truhly nebo trezoru nastaví na hodnotu *false*. K proměnné *coin* se přičtou mince, které byly v dané truhle nebo trezoru. A obrázek picture boxu dané truhly nebo trezoru se nastaví na otevřený.

Ukázka kódu pro odemykání truhel:

```
 1.             if (player.Bounds.IntersectsWith(chest1.Bounds) && use ==
true && chest1.Enabled == true)
 2.             {
 3.                 UnlockSafe();
 4.
 5.                 if (answer == true)
 6.                 {
 7.                     chest1.Enabled = false;
 8.                     chest1.Image = Properties.Resources.chest_o;
 9.                     coin += 100;
10.                 }
11.             }
```

Ukázka kódu pro odemykání trezorů:

```
 1.             if (player.Bounds.IntersectsWith(safe1.Bounds) && use ==
true && safe1.Enabled == true)
 2.             {
 3.                 UnlockSafe();
 4.
 5.                 if (answer == true)
 6.                 {
 7.                     safe1.Enabled = false;
 8.                     coin += 300;
 9.                     safe1.Image = Properties.Resources.safe_o;
10.                 }
11.             }
```

5.6.3 Funkce UnlockDoor

Tato funkce slouží pro odemykání dveří.

Po zavolaní této funkce se nejdříve zastaví časovač hry. Následně se vyhodnocuje podmínka, jestli alespoň jedna proměnná *goRight* nebo *goLeft* se rovná hodnotě *true*. Pokud je podmínka splněna, tak se otevře nový formulář, který slouží pro zobrazení otázky a čtyř možných odpovědí na otázku. Deklaruje se nová proměnná *formOdpoved* do které se uloží hodnota uložená v proměnné *Otazky.answer* z formuláře otázky. Kód formuláře Otazky.cs je podrobněji popsán v kapitole 5.7.

Dále se vyhodnocuje podmínka, jestli se proměnná *formOdpoved* rovná hodnotě 1. Pokud se bude rovnat hodnotě 1, tak to znamená že hráč odpověděl správně na otázku. V případě, kdy je podmínka splněna, tak se proměnná *answer* nastaví na hodnotu *true*, v opačném

případě se nastaví na hodnotu *false* a vyhodnocuje se podmínka, jestli se proměnná *formOdpoved* nerovná číslu 2. Pokud by se rovnala číslu 2 znamenalo by to, že hráč nevybral ani jednu z možných odpovědí na otázku a klikl na tlačítko *zpět*. V případě, kdy je podmínka splněna, tak se zobrazí message box s informací, že hráč odpověděl špatně na otázku a od proměnné *lockpick* se odečte jednička (hráč přijde o jeden paklíč).

Dále aby nedošlo k zaseknutí hráče do dveří tak se vyhodnocuje podmínka, jestli šel hráč směrem doprava nebo doleva. V případě, kdy šel doprava (*goRight* se rovná *true*), tak se hráčova postava posune o 5 pixelů směrem doleva a proměnná *goRight* se nastaví na hodnotu *false* a spustí se časovač hry. V opačném případě, kdy hráč šel směrem doleva (*goLeft* se rovná hodnotě *true*), tak se posune hráčova postava směrem doprava o 5 pixelů, proměnná *goLeft* se nastaví na hodnotu *false* a opět se spustí časovač hry.

Ukázka kódu:

```
1.          private void UnlockDoor()
2.          {
3.              GameTimer.Stop();
4.
5.              if (goRight == true || goLeft == true)
6.              {
7.                  Otazky frmOtazky = new Otazky();
8.                  frmOtazky.ShowDialog();
9.                  int formOdpoved = Otazky.answer;
10.
11.                 if (formOdpoved == 1)
12.                 {
13.                     answer = true;
14.                 }
15.                 else
16.                 {
17.                     answer = false;
18.                     if (formOdpoved != 2)
19.                     {
20.                         MessageBox.Show("Špatná odpověď, zkuste to
znovu.");
21.                         lockpick -= 1;
22.                     }
23.                 }
24.
25.                 if (goRight == true)
26.                 {
27.                     player.Left -= 5;
28.                     goRight = false;
29.                     GameTimer.Start();
30.                 }
31.                 else if (goLeft == true)
32.                 {
33.                     player.Left += 5;
34.                     goLeft = false;
```

```
35.                     GameTimer.Start();
36.                 }
37.             }
38.         }
```

5.6.4 Funkce UnlockSafe

Funkce *UnclockSafe* slouží pro odemykání truhel a trezorů.

Po zavolání této funkce se opět první zastaví časovač hry, jako tomu bylo u funkce *UnlockDoor*. Následně se vyhodnocuje podmínka, jestli se proměnná *use* rovná hodnotě *true* (hráč zmáčkl písmeno E nebo Enter na klávesnici). V další podmínce se vyhodnocuje, jestli má hráč dostatek paklíčů. Pokud se bude proměnná *lockpick* rovnat hodnotě nula, tak to znamená že hráč již nemá žádné paklíče. V tom případě se zobrazí message box s touto informací.

Pokud má hráč dostatek paklíčů, tak se otevře nový formulář s otázkou, na kterou musí hráč odpovědět správně. A deklaruje se nová proměnná *formOdpoved2* do které se uloží hodnota z proměnné *Otazky.answer* z formuláře Otazky.cs. Kód formuláře Otazky.cs je více popsán v kapitole 5.6.

Dále se zjišťuje, jestli hodnota v proměnné *formOdpoved2* se rovná číslu jedna. Pokud ano, tak hráč odpověděl správně na otázku, a proto se proměnná *answer* nastaví na hodnotu *true*. V případě, kdy hráč odpoví špatně na otázku, tak se proměnná *answer* nastaví na hodnotu *false* a vyhodnocuje se další podmínka. V této podmínce se zjišťuje, jestli se proměnná *formOdpoved2* nerovná číslu dva. Pokud se nerovná tak hráč odpověděl špatně na otázku. Hráči se zobrazí message box s informací, že vybral špatnou odpověď a hodnota v proměnné *lockpick* se zmenší o 1. (Hráč přišel o jeden paklíč).

Dále se nastaví hodnota v proměnné *use* na *false,* aby nedošlo k opakovanému otvírání formuláře s otázkami a hodnoty v proměnných *goLeft* a *goRight* se taky nastaví na *false*, aby se hráčova postava zůstala stát na místě v případech, kdy hráč držel šipku doprava nebo doleva a zároveň se pokusil o odemčení truhly nebo trezoru. A nakonec se spustí časovač hry.

Ukázka kódu:

```
1.         private void UnlockSafe()
2.         {
3.             GameTimer.Stop();
4.
5.             if (use == true)
6.             {
```

```
 7.                 if (lockpick == 0)
 8.                 {
 9.                     use = false;
10.                     answer = false;
11.                     MessageBox.Show("Nemáte dostatek paklíčů k odemčení
zámku.");
12.                 }
13.                 else
14.                 {
15.                     Otazky frmOtazky = new Otazky();
16.                     frmOtazky.ShowDialog();
17.                     int formOdpoved2 = Otazky.answer;
18.
19.                     if (formOdpoved2 == 1)
20.                     {
21.                         answer = true;
22.                     }
23.                     else
24.                     {
25.                         answer = false;
26.                         if (formOdpoved2 != 2)
27.                         {
28.                             MessageBox.Show("Špatná odpověď, zlomil jste
paklíč.");
29.                             lockpick -= 1;
30.                         }
31.                     }
32.                 }
33.
34.                 use = false;
35.                 goLeft = false;
36.                 goRight = false;
37.
38.                 GameTimer.Start();
39.             }
40.         }
```

5.6.5 Funkce RestartGame

Tato funkce slouží pro restartování dané úrovně nebo pro vrácení do hlavního menu.

Po zavolání této funkce se zobrazí message box s informací, jestli chce hráč zkusit danou úroveň znovu nebo se chce vrátit do menu. V message boxu jsou umístěny tlačítka *Retry* a *Cancel.* Tlačítko *Retry* slouží pro restartování úrovně a tlačítko *Cancel* slouží pro návrat do hlavního menu. Pokud hráč klikne na tlačítko *Retry*, tak se zavře okno s aktuálním formulářem a otevře se nové okno s danou úrovní. Po stisknutí tlačítka *Cancel* dojde k zavření aktuálního formuláře a otevře se formulář hlavního menu.

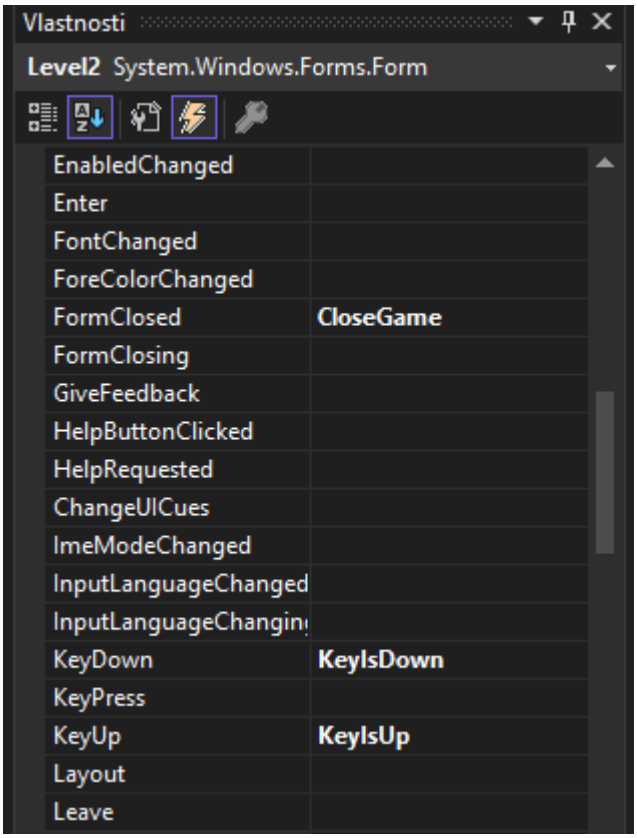
Ukázka kódu:

```
 1.         private void RestartGame()
 2.         {
```

```
 3.             if (MessageBox.Show("Stiskněte OPAKOVAT pro restart úrovně.
Po Stisknutí ZRUŠIT se vrátíte do menu.", "Načíst poslední level",
MessageBoxButtons.RetryCancel) == DialogResult.Retry)
 4.             {
 5.                 Level2 frmLevel2 = new Level2();
 6.                 frmLevel2.Show();
 7.                 this.Hide();
 8.             }
 9.             else
10.             {
11.                 MainMenu newWindow = new MainMenu();
12.                 newWindow.Show();
13.                 this.Hide();
14.             }
15.         }
```

5.6.6 Funkce pro nastavení tlačítek ve hře

Pro zjištění, jestli je dané tlačítko na klávesnici zmáčknuto nebo ne se využívají funkce *KeyIsDown* a *KeyIsUp*. Tyto funkce je nutné nastavit v okně daného formuláře. Po kliknutí na daný formulář, vybereme ve vlastnostech formuláře záložku události a v ní najdeme položky *KeyDown* a *KeyUp*. A k nim následně napíšeme názvy funkcí, jak můžeme vidět na obrázku 20.



Obrázek 20 Vlastnosti – Level2

5.6.6.1 KeyIsDown

V této funkci se nastavují jednotlivé proměnné na hodnotu *true* pokud hráč zmáčkne danou klávesu na klávesnici.

Šipka doleva nebo A – *goLeft* (Pohyb hráče směrem doleva)

Šipka doprava nebo D – *goRight* (Pohyb hráče směrem doprava)

E nebo Enter – use (Tlačítko pro otevření truhel nebo trezorů)

Šipka nahoru nebo W – *goUp* (Pohyb hráče směrem nahoru)

Šipka dolů nebo S – *goDown* (Pohyb hráče směrem dolů)

Ukázka kódu:

```
1.          private void KeyIsDown(object sender, KeyEventArgs e)
2.          {
3.              if (e.KeyCode == Keys.Left || e.KeyCode == Keys.A)
4.              {
5.                  goLeft = true;
6.              }
7.
8.              if (e.KeyCode == Keys.Right || e.KeyCode == Keys.D)
9.              {
10.                 goRight = true;
11.             }
12.
13.             if (e.KeyCode == Keys.E || e.KeyCode == Keys.Enter)
14.             {
15.                 use = true;
16.             }
17.
18.             if (e.KeyCode == Keys.Up || e.KeyCode == Keys.W)
19.             {
20.
21.                 goUp = true;
22.             }
23.
24.             if (e.KeyCode == Keys.Down || e.KeyCode == Keys.S)
25.             {
26.                 goDown = true;
27.             }
28.         }
```

5.6.6.2 KeyIsUP

V této funkci se nastavují jednotlivé proměnné na hodnotu *false* pokud hráč pustí danou klávesu na klávesnici.

Šipka doleva nebo A – *goLeft* (Pohyb hráče směrem doleva)

Šipka doprava nebo D – *goRight* (Pohyb hráče směrem doprava)

E nebo Enter – use (Tlačítko pro otevření truhel nebo trezorů)

Šipka nahoru nebo W – *goUp* (Pohyb hráče směrem nahoru)

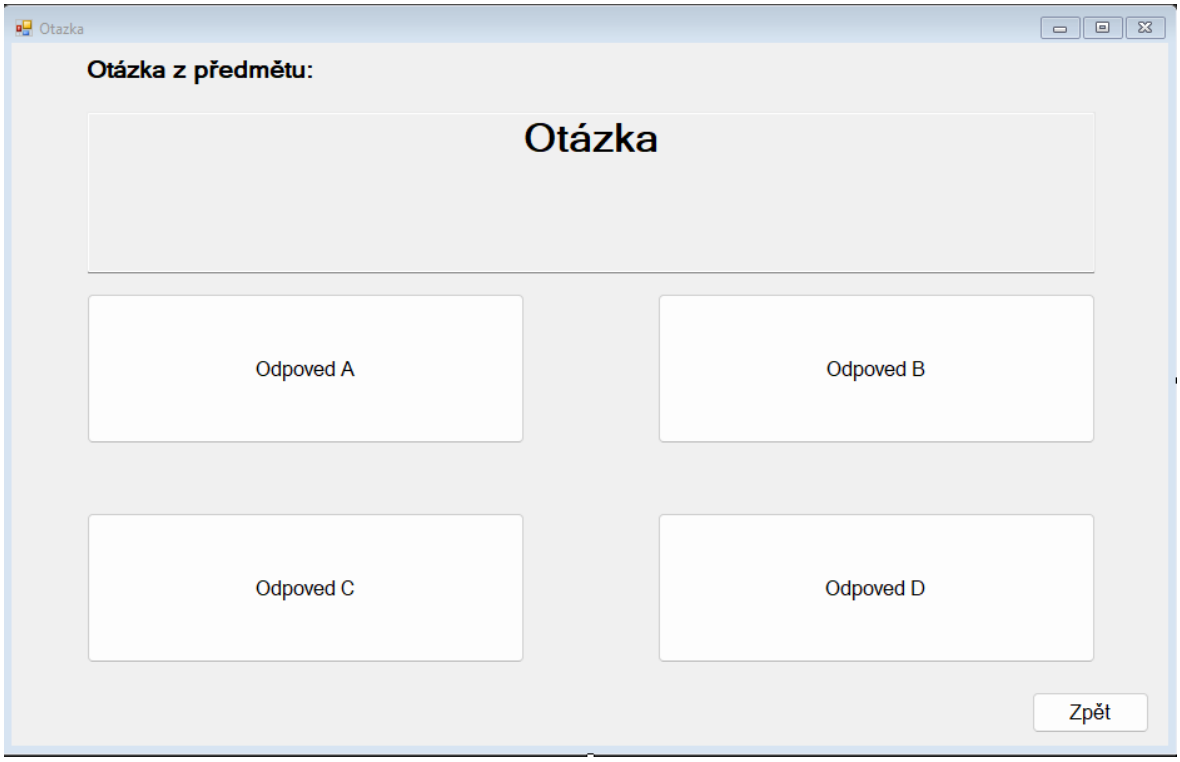
Šipka dolů nebo S – *goDown* (Pohyb hráče směrem dolů)

Ukázka kódu:

```
1.          private void KeyIsUp(object sender, KeyEventArgs e)
2.          {
3.              if (e.KeyCode == Keys.Left || e.KeyCode == Keys.A)
4.              {
5.                  goLeft = false;
6.              }
7.
8.              if (e.KeyCode == Keys.Right || e.KeyCode == Keys.D)
9.              {
10.                 goRight = false;
11.             }
12.
13.             if (e.KeyCode == Keys.E || e.KeyCode == Keys.Enter)
14.             {
15.                 use = false;
16.             }
17.
18.             if (e.KeyCode == Keys.Up || e.KeyCode == Keys.W)
19.             {
20.                 goUp = false;
21.             }
22.
23.             if (e.KeyCode == Keys.Down || e.KeyCode == Keys.S)
24.             {
25.                 goDown = false;
26.             }
27.         }
```

5.7 Otazky.cs

Tento formulář slouží pro zobrazení otázek a možných odpovědí. Je zde umístěn jeden textový box, který zobrazuje otázku a pět tlačítek. Čtyři tlačítka jsou pro odpovědi a jedno tlačítko pro navrácení do hry.



Obrázek 21 Formulář Otazky

Po spuštění formuláře se definují a deklarují proměnné *RightAnswer* a *answer*. Dále se po načtení zavolá funkce *OpenFile*.

Ukázka kódu:

```
1.          string RightAnswer = "";
2.          public static int answer = 2;
3.
4.          private void Otazky_Load(object sender, EventArgs e)
5.          {
6.              OpenFile();
7.          }
```

RightAnswer – řetězec do kterého se uloží správná odpověď na otázku

Answer – do této proměnné se ukládá číslo, podle toho, jak hráč odpověděl na otázku (0 – špatná odpověď, 1 – správná odpověď, 2 – hráč neodpověděl)

5.7.1 OpenFile

Funkce *OpenFile* nejdříve otevře excel s databází otázek. Následně se do proměnné *rInt* uloží náhodné číslo z rozsahu 0 až 119 pro náhodný výběr řádku z databáze otázek. Na každém řádku se nachází jedna otázka s možnými odpověďmi a správnou odpovědí. Následně se z excel souboru uloží do labelu *Subject* název předmětu, do textového boxu *Question* otázka a do tlačítek *AnswerA*, *AnswerB, AnswerC a AnswerD* možné odpovědi na

otázku. Do proměnné *RightAnswer* se uloží správná odpověď na otázku. A dojde k zavření excel souboru.

Ukázka kódu:

```
 1.         public void OpenFile()
 2.         {
 3.             Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\Otazky.xlsx", 1);
 4.             Random r = new Random();
 5.             int rInt = r.Next(0, 119);
 6.
 7.             Subject.Text = "Otázka z předmětu:" + excel.ReadCell(rInt,
0);
 8.             Question.Text = excel.ReadCell(rInt, 1);
 9.             AnswerA.Text = excel.ReadCell(rInt, 2);
10.             AnswerB.Text = excel.ReadCell(rInt, 3);
11.             AnswerC.Text = excel.ReadCell(rInt, 4);
12.             AnswerD.Text = excel.ReadCell(rInt, 5);
13.             RightAnswer = excel.ReadCell(rInt, 6);
14.             excel.Close();
15.         }
```

5.7.2 Tlačítka

Funkce tlačítek *AnswerA, AnswerB, AnswerC, AnswerD* je totožná. Nejdříve se vyhodnocuje podmínka, jestli se text tlačítka rovná proměnné *RightAnswer* ve které je uložená správná odpověď na otázku.

Pokud bude podmínka platit, tak se do proměnné *answer* uloží hodnota 1 (to znamená že hráč odpověděl správně na otázku) a formulář s otázkou se zavře. V opačném případě se do proměnné *answer* uloží hodnota 0 (hráč odpověděl špatně na otázku) a opět se formulář s otázkou zavře.

Ukázka kódu pro tlačítko *AnswerA*:

```
 1.         private void AnswerA_Click(object sender, EventArgs e)
 2.         {
 3.             if (AnswerA.Text == RightAnswer)
 4.             {
 5.                 answer = 1;
 6.                 this.Hide();
 7.             }
 8.             else
 9.             {
10.                 answer = 0;
11.                 this.Hide();
12.             }
13.         }
```

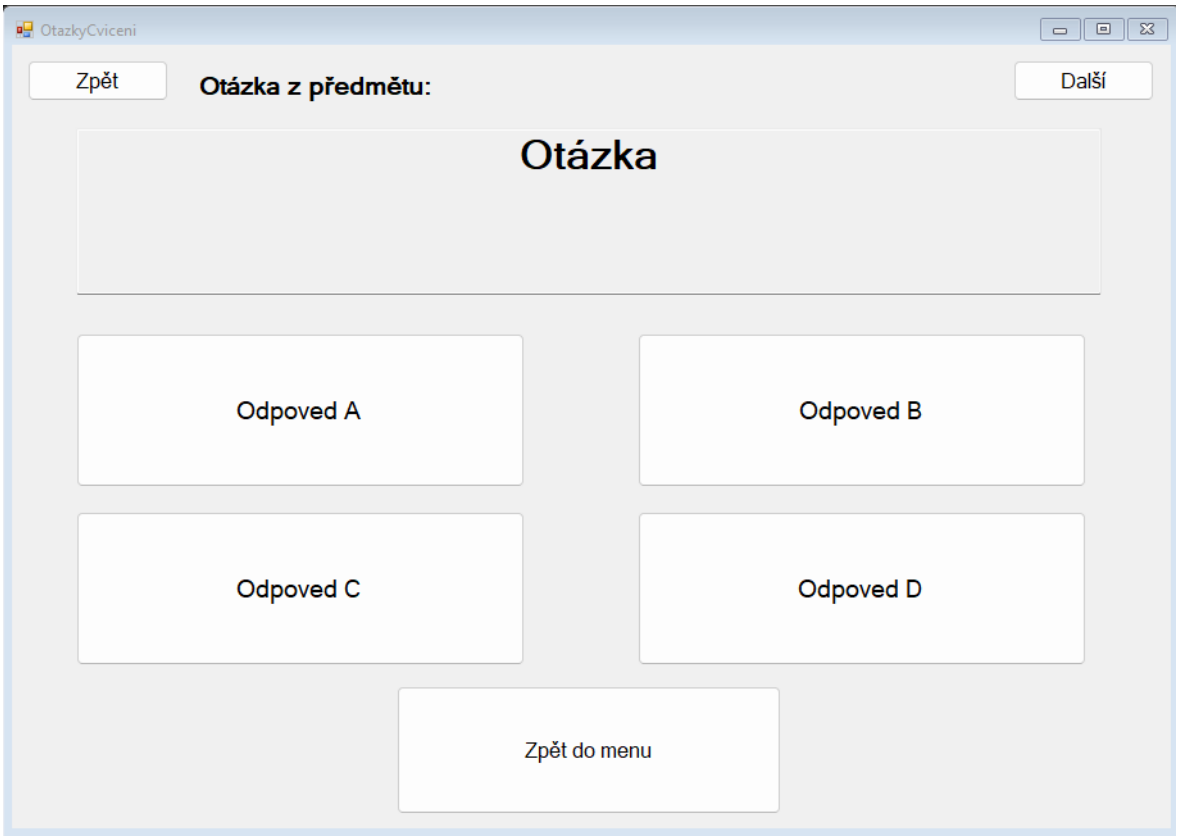
Tlačítko *zpět* slouží pro zavření formuláře a navrácení do hry. Po stisknutí tlačítka *BackButton* se formulář s otázkou zavře.

Ukázka kódu:

```
1.          private void BackButton_Click(object sender, EventArgs e)
2.          {
3.              this.Hide();
4.          }
```

5.8 OtazkyCviceni

Tento formulář slouží pro procházení všech otázek uložených v excel souboru Otazky.xlsx.



Obrázek 22 Formulář OtazkyCviceni

Nejdříve se deklarují proměnné *RightAnswer* a *i*. Následně se otevře excel soubor, ve kterém jsou uloženy otázky. Po načtení formuláře se zavolá funkce *OpenFile*.

Ukázka kódu:

```
1.          string RightAnswer = "";
2.          int i = 0;
3.          Excel excel = new
Excel(System.Windows.Forms.Application.StartupPath + "\\Otazky.xlsx", 1);
4.
5.          private void OtazkyCviceni_Load(object sender, EventArgs e)
6.          {
7.              OpenFile();
8.          }
```

RightAnswer – řetězec do kterého se uloží správná odpověď na otázku

i – pomocná proměnná do které se ukládá číslo otázky, která bude zobrazena.

5.8.1 OpenFile

Po zavolání této funkce se do labelu *Question* uloží otázka z excelu, následně se do tlačítek *AnswerA, AnswerB, AnswerC a AnswerD* se možné odpovědi, a nakonec se do proměnné *RightAnswer* uloží správná odpověď na otázku.

Ukázka kódu:

```
1.          public void OpenFile()
2.          {
3.              Question.Text = excel.ReadCell(i, 1);
4.              AnswerA.Text = excel.ReadCell(i, 2);
5.              AnswerB.Text = excel.ReadCell(i, 3);
6.              AnswerC.Text = excel.ReadCell(i, 4);
7.              AnswerD.Text = excel.ReadCell(i, 5);
8.              RightAnswer = excel.ReadCell(i, 6);
9.          }
```

5.8.2 Tlačítka

V této části jsou popsány funkce jednotlivých tlačítek

5.8.2.1 Tlačítka s odpověďmi

Funkce tlačítek *AnswerA, AnswerB, AnswerC a AnswerD* je totožná. Nejdříve se vyhodnocuje, jestli se text uložený v daném tlačítku rovná textu, který je uložený v proměnné *RightQuestion*. Pokud ano, tak se vyhodnocuje další podmínka, jestli se jedná o poslední otázku z databáze otázek nebo ne. Pokud ano, tak se zobrazí message box s informací, že to byla poslední otázka z databáze. Pokud ne, tak se proměnná i zvětší o jedna a znovu se zavolá funkce *OpenFile*.

Pokud hráč odpoví špatně na otázku, tak se zobrazí message box s informací, že odpověděl špatně.

Ukázka kódu pro tlačítko *AnswerA*:

```
1.              if (AnswerA.Text == RightAnswer)
2.              {
3.                  if (i == 119)
4.                  {
5.                      MessageBox.Show("Je zobrazena poslední otázka z
databáze." + Environment.NewLine + "Můžete se vrátit k předchozím otázkám
nebo se vrátit do menu.");
6.                  }
7.                  else
8.                  {
9.                      i++;
```

```
10.                     OpenFile();
11.                 }
12.             }
13.             else
14.             {
15.                 MessageBox.Show("Špatná odpověď." + Environment.NewLine
+ "Zkuste to znovu.");
16.             }
17.         }
```

5.8.2.2 Tlačítka další a zpět

Tlačítka *další* a *zpět* slouží pro zobrazení další otázky nebo zobrazení předchozí otázky.

Po kliknutí na tlačítko *Next* se vyhodnocuje podmínka, jestli se jedná o poslední otázku z databáze. Pokud ano, tak se zobrazí message box s informací, že je zobrazena poslední otázka. Pokud ne, tak se proměnná i zvětší o jedna a zavolá se funkce *OpenFile*.

Ukázka kódu:

```
 1.         private void Next_Click(object sender, EventArgs e)
 2.         {
 3.             if (i == 119)
 4.             {
 5.                 MessageBox.Show("Je zobrazena poslední otázka z
databáze." + Environment.NewLine + "Můžete se vrátit k předchozím otázkám
nebo se vrátit do menu.");
 6.             }
 7.             else
 8.             {
 9.                 i++;
10.                 OpenFile();
11.             }
12.         }
```

Po kliknutí na tlačítko *Back* se vyhodnocuje podmínka, jestli se proměnná i nerovná nule. Pokud by se proměnná i rovnala nule znamenalo by to, že je zobrazena první otázka z databáze.

Pokud se proměnná *i* nerovná nule tak se její hodnota zmenší o jedna a zavolá se funkce *OpenFile*. Pokud se bude rovnat nule, tak se zobrazí message box s informací, že je zobrazena první otázka.

Ukázka kódu:

```
 1.         private void Back_Click(object sender, EventArgs e)
 2.         {
 3.             if (i != 0)
 4.             {
 5.                 i--;
 6.                 OpenFile();
```

```
 7.             }
 8.             else
 9.             {
10.                 MessageBox.Show("Nelze se vrátit zpět." +
Environment.NewLine + "Je zobrazena první otázka");
11.             }
12.         }
```

5.8.2.3 Tlačítko Zpět do menu

Tlačítko *BackToMenu* slouží pro navrácení do hlavního menu. Po kliknutí na toto tlačítko se otevře nové okno s formulářem hlavního menu a současné okno se zavře.

Ukázka kódu:

```
1.         private void BackToMenu_Click(object sender, EventArgs e)
2.         {
3.
4.             MainMenu newWindow = new MainMenu();
5.             newWindow.Show();
6.             this.Hide();
7.         }
```

5.9 Třída excel

Třída excel slouží pro práci s excel soubory. Nejdříve se deklarují a definují proměnné pro práci s Excel soubory. V proměnné *path* je uložená cesta k excel souboru. V proměnné *Sheet* je uložena hodnota vyjadřující stranu listu v excel souboru.

Ukázka kódu:

```
 1.         string path = "";
 2.         _Application excel = new _Excel.Application();
 3.         Workbook wb;
 4.         Worksheet ws;
 5.
 6.         public Excel(string path, int Sheet)
 7.         {
 8.             this.path = path;
 9.             wb = excel.Workbooks.Open(path);
10.             ws = wb.Worksheets[Sheet];
11.         }
```

5.9.1 Čtení z excel souboru

Funkce *ReadCell* slouží pro čtení hodnot v jednotlivých buňkách v excel souboru. V proměnné *i* je uloženo číslo řádku a v proměnné *j* číslo sloupce.

Ukázka kódu:

```
1.          public string ReadCell(int i, int j)
2.          {
3.              i++;
4.              j++;
5.              if (ws.Cells[i, j].Value2 != null)
6.                  return ws.Cells[i, j].Value2;
7.              else
8.                  return "";
9.          }
```

5.9.2 Zápis do excelu

Funkce *WriteToCell* slouží pro zápis textu do jednotlivých buněk v excel souboru. V proměnné *i* je uloženo číslo řádku a v proměnné *j* číslo sloupce buňky v excel souboru. V proměnné *s* je uložený text, který se má zapsat do buňky.

```
1.          public void WriteToCell(int i, int j, string s)
2.          {
3.              i++;
4.              j++;
5.              ws.Cells[i, j].Value2 = s;
6.          }
```

Funkce *WriteToCellNumber* slouží pro zápis čísel do jednotlivých buněk v excel souboru. Opět je v proměnné *i* uloženo číslo řádku a v proměnné *j* číslo sloupce buňky v excel souboru. V proměnné *c* je uloženo číslo, které se má zapsat do buňky.

Ukázka kódu:

```
1.          public void WriteToCellNumber(int i, int j, int c)
2.          {
3.              i++;
4.              j++;
5.              ws.Cells[i, j].Value2 = c;
6.          }
```

5.9.3 Ukládání a zavření excel souboru

Funkce *Save* slouží pro uložení excel souboru.

Ukázka kódu:

```
1.          public void Save()
2.          {
3.              wb.Save();
4.          }
```

Funkce *Close* slouží pro zavření excel souboru.

Ukázka kódu:

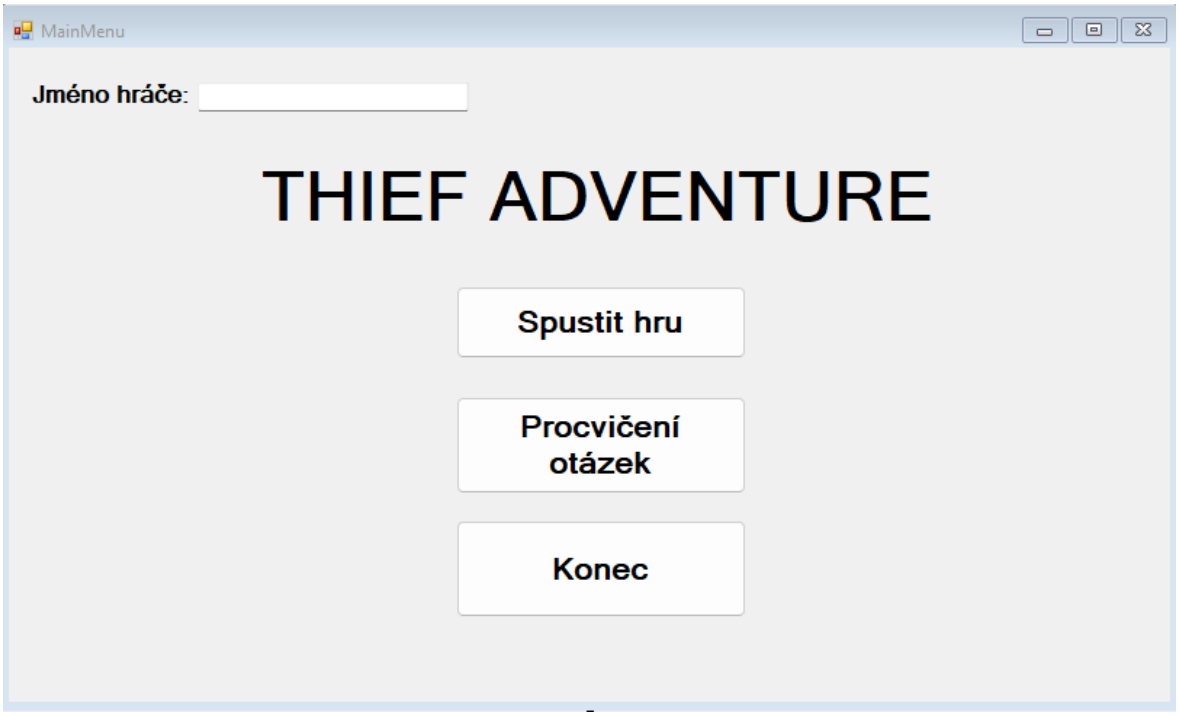
```
1.          public void Close()
2.          {
3.              wb.Close();
4.          }
```

POPIS HRY Z POHLEDU HRÁČE

V této kapitole je hra popsána z pohledu hráče.

5.10 Hlavní menu

Po spuštění hry se zobrazí hlavní menu. V hlavním menu se nachází tři možnosti. Hráč si může vybrat, jestli chce spustit hru, procvičit si otázky nebo ukončit hru. V levém horním rohu se nachází pole, do kterého hráč musí napsat své jméno, aby mohl spustit hru.



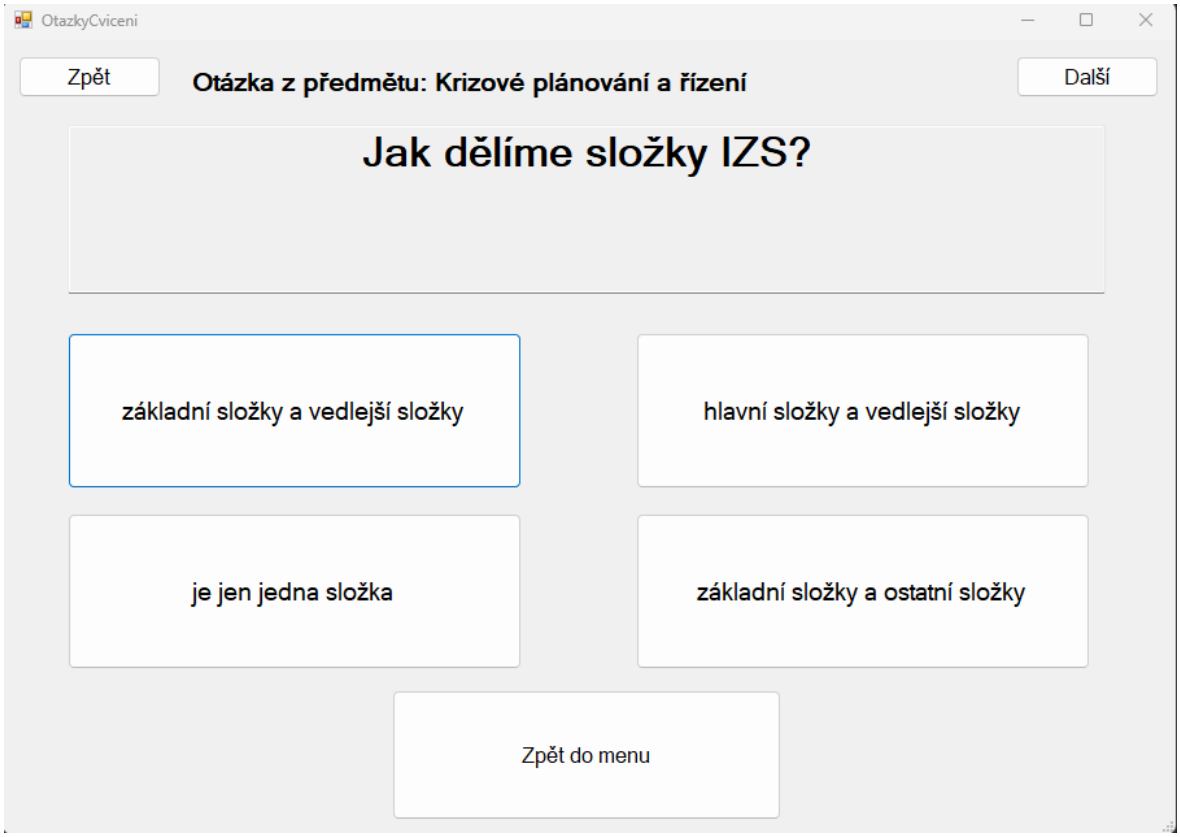
Obrázek 23 Hlavní menu

5.11 Konec

Po kliknutí na tlačítko Konec se hra vypne. Hru je možné také kdykoliv vypnout kliknutím na červený křížek v pravém horním rohu.

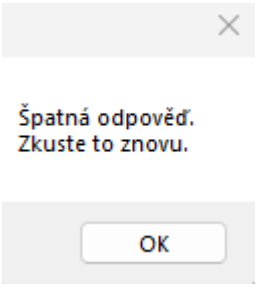
5.12 Procvičení otázek

Po kliknutí na tlačítko procvičení otázek se zobrazí nový formulář, ve kterém je zobrazena otázka se čtyřmi možnými odpověďmi.



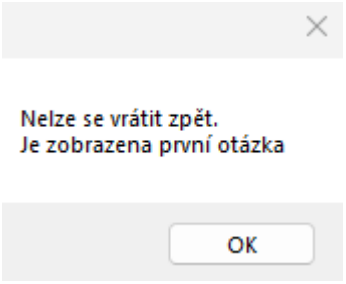
Obrázek 24 Procvičení otázek

Pokud hráč odpoví správně na otázku tak se zobrazí automaticky další otázka. Pokud odpoví špatně tak se zobrazí Dialogové okno s informací, že hráč odpověděl špatně.



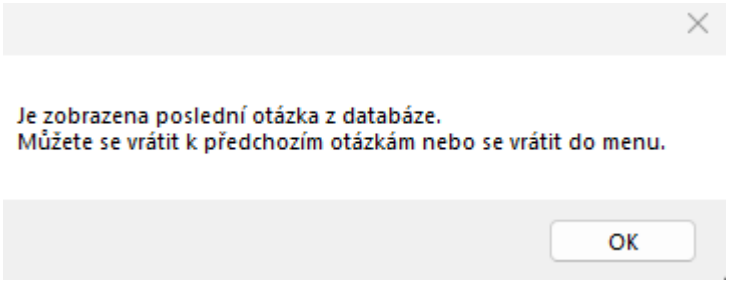
Obrázek 25 Špatná odpověď

Otázky je možné přeskočit pomocí tlačítka *Další* anebo se vrátit k přechozí otázce pomocí tlačítka *Zpět*. Pokud je zobrazena první otázka a hráč klikne na tlačítko *zpět*, tak se zobrazí dialogové okno, které hráče informuje že je zobrazena první otázka.



Obrázek 26 První otázka z databáze

Pokud hráč odpoví na poslední otázku z databáze nebo klikne na tlačítko *Další*, když je zobrazena poslední otázka, tak se zobrazí dialogové okno s informací, že je zobrazena poslední otázka.

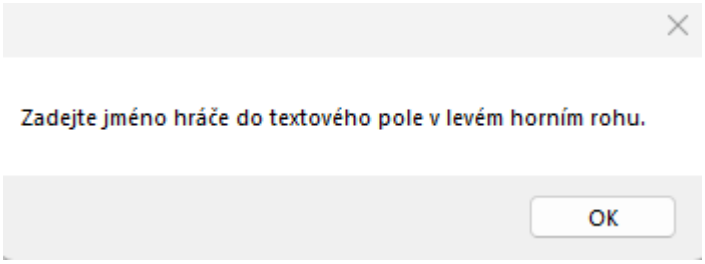


Obrázek 27 Poslední otázka z databáze

Po kliknutí na tlačítko *Zpět do menu* se okno s otázkami zavře a zobrazí se hlavní menu.

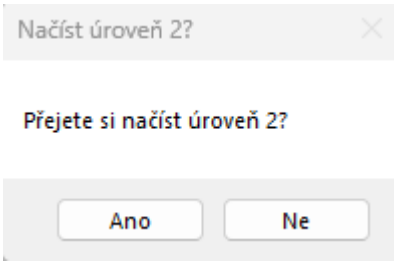
5.13 Spustit hru

Pokud hráč klikne na tlačítko *Spustit hru* a nenapsal své jméno do textového pole v levém horním rohu, tak se zobrazí okno s informací, aby hráč zadal své jméno do textového pole.



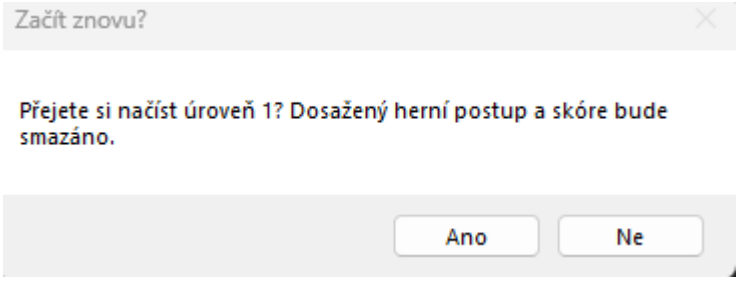
Obrázek 28 Jméno hráče

Pokud se jedná o nového hráče, tak se automaticky spustí první úroveň hry. Pokud se jedná o hráče, který již dokončil nějakou úroveň, tak se zobrazí dialogové okno s možností načíst úroveň, kterou zatím hráč nedokončil. Pokud hráč zvolí *Ano*, tak se spustí úroveň hry, kterou hráč ještě nedokončil.



Obrázek 29 Další úroveň

Pokud hráč zvolí tlačítko *Ne*, tak se zobrazí dialogové okno s informací, jestli chce hráč načíst první úroveň.

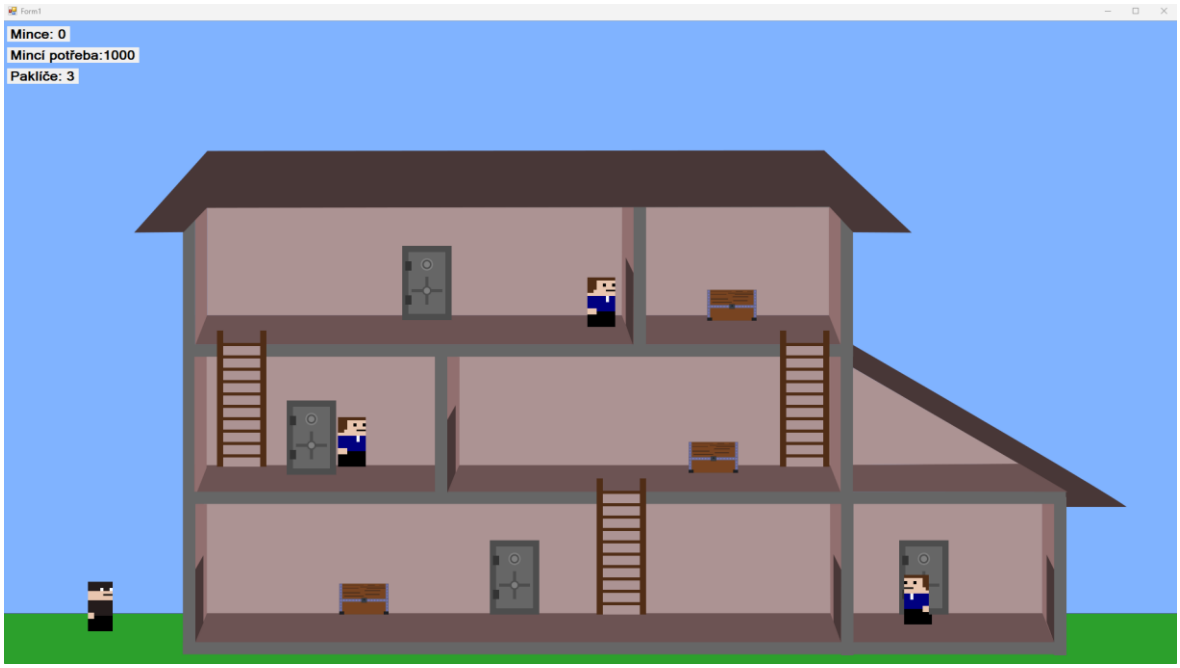


Obrázek 30 Začít znovu

Pokud hráč klikne na *Ano* tak dojde k vymazání herního postupu hráče a budou smazány informace o získaných mincích v jednotlivých úrovních. A následně se spustí první úroveň hry.

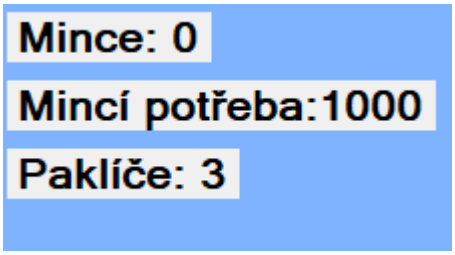
5.14 První úroveň

Na obrázku 31 je zobrazena první úroveň hry. V první úrovni musí hráč získat 1000 mincí pro postup do další úrovně. V domě se nachází 3 truhly a 4 trezory, které hlídají tři stráže. První úroveň hry slouží k vysvětlení ovládání hry. Proto tato úroveň jako jediná obsahuje možnost zobrazení tutoriálu.



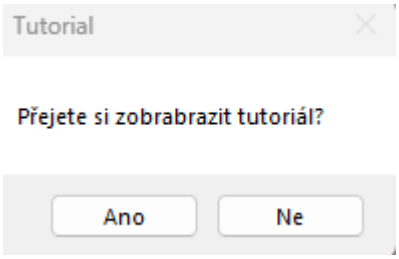
Obrázek 31 První úroveň

V levém horním rohu jsou zobrazeny informace o počtu mincí, které hráč získal, počet celkových mincí, které musí získat pro postup do další úrovně a počet paklíčů.



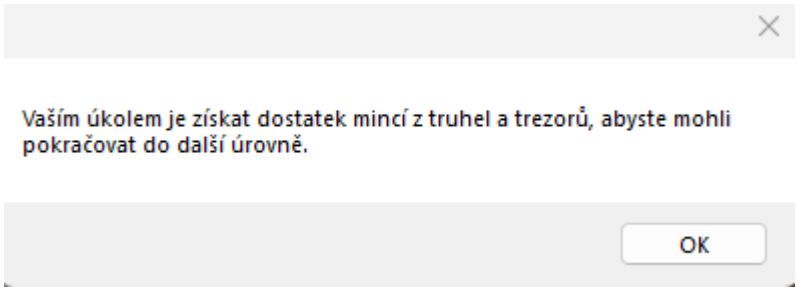
Obrázek 32 Informace

Po spuštění první úrovně se zobrazí dialogové okno s otázkou, jestli chce hráč zobrazit tutoriál.



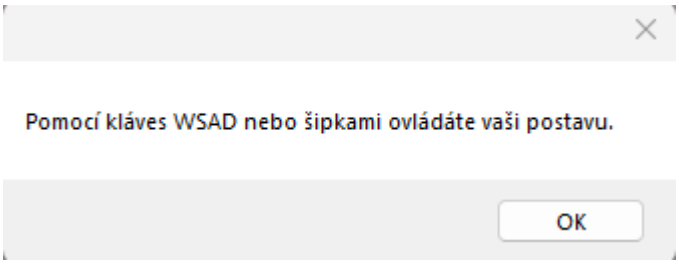
Obrázek 33 Tutoriál

Po kliknutí na *Ano* se zobrazí další dialogové okno s informací, jaký je cíl hry,



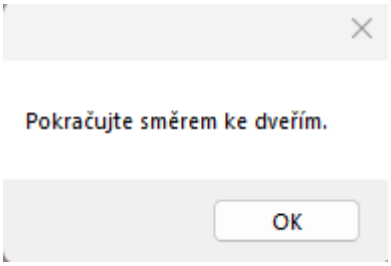
Obrázek 34 Cíl hry

V následujícím dialogovém okně je vysvětleno, jak hráč ovládá svoji postavu.



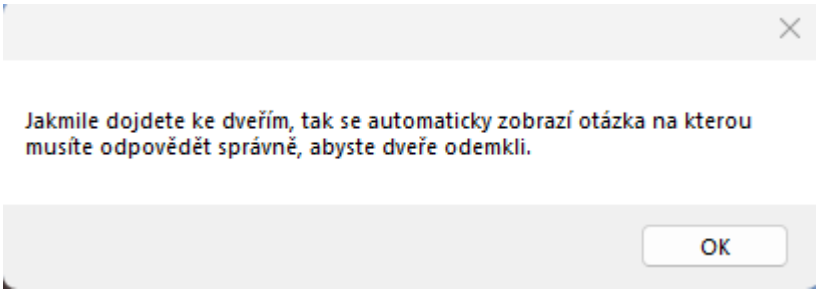
Obrázek 35 Ovládaní

Následně se zobrazí další dialogové okno, které hráče informuje kam má s postavou jít.



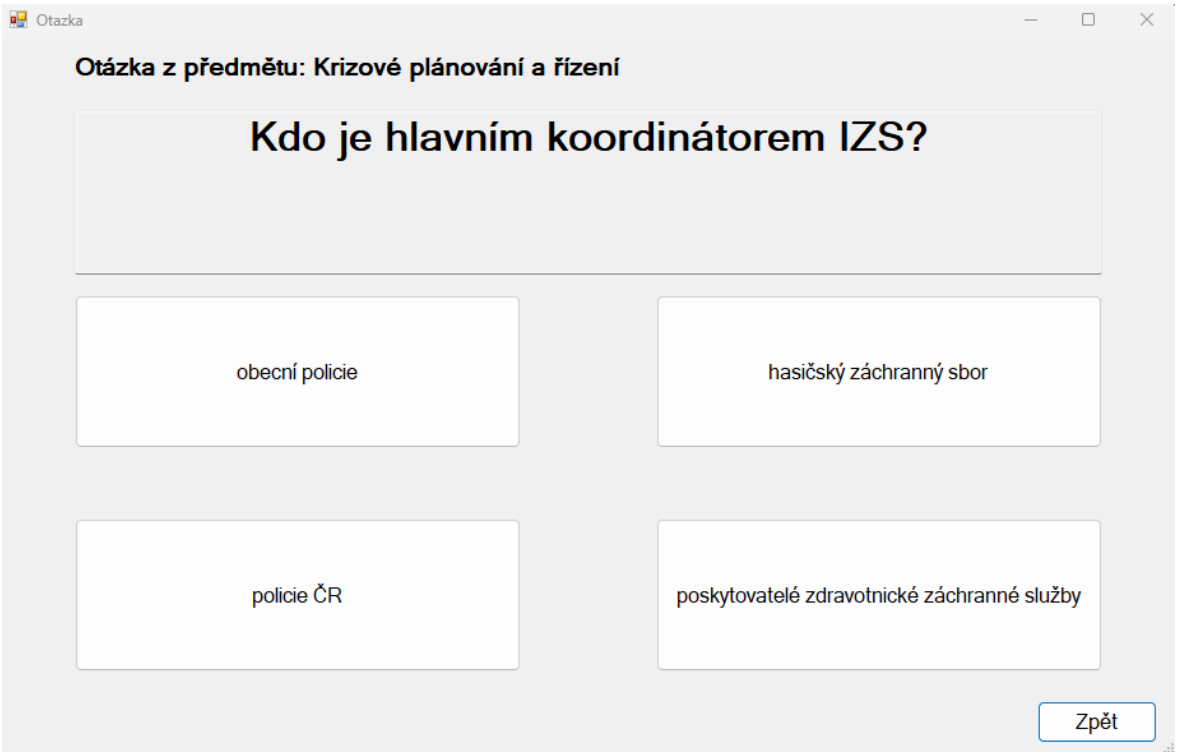
Obrázek 36 První úkol

Jakmile hráč dojde až ke dveřím, tak se zobrazí další informační okno.



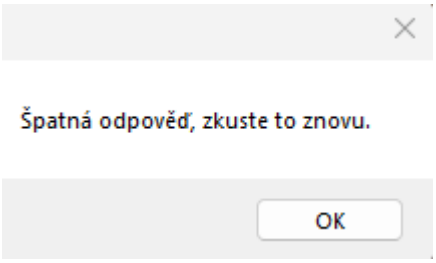
Obrázek 37 Informace, jak otevřít dveře

Po kliknutí na *OK* se zobrazí formulář s otázkou, na kterou musí hráč odpovědět správně.



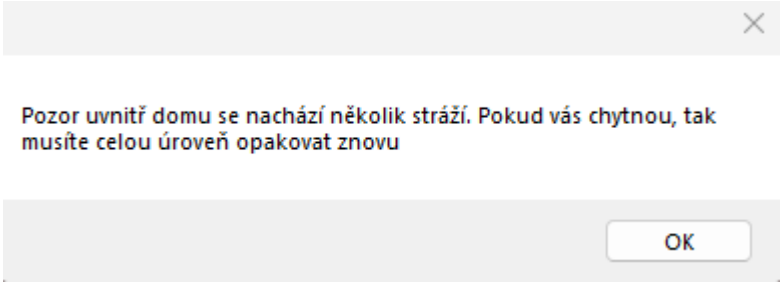
Obrázek 38 Otázka a možné odpovědi

V případě, kdy hráč odpoví špatně na otázku zobrazí se okno s informací, že hráč zvolil špatnou odpověď.



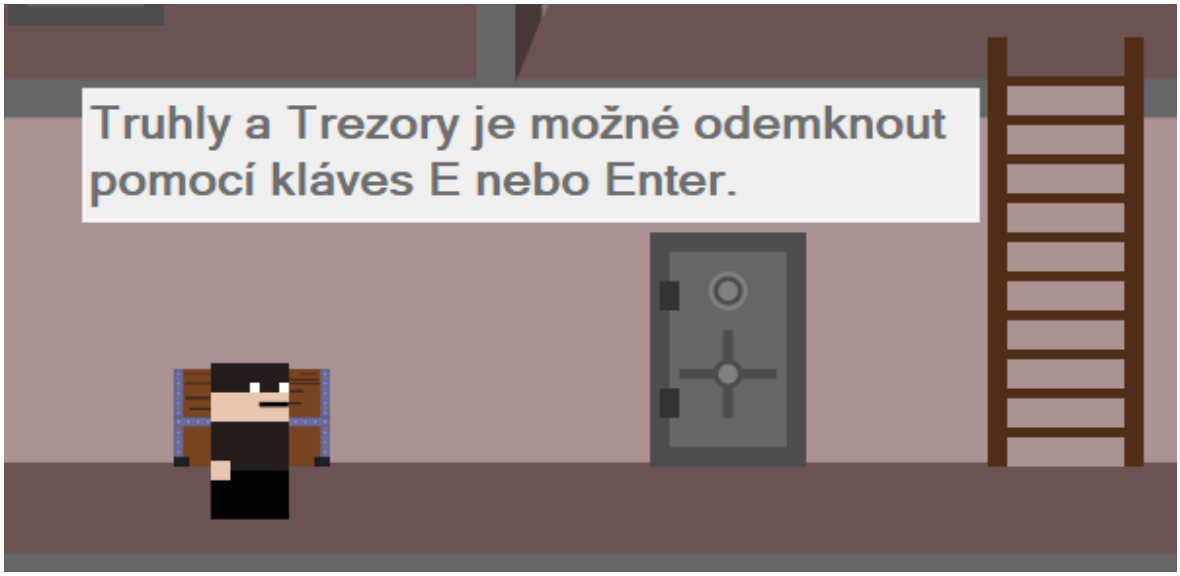
Obrázek 39 Špatná odpověď

Pokud hráč odpoví správně na otázku, tak se zobrazí další dialogové okno informujícího hráče, aby si dával pozor na stráže uvnitř domu.



Obrázek 40 Varování před strážnými

Jakmile hráč stojí u truhly nebo trezoru může se pokusit o jejich odemčení. Pomocí kláves *E* nebo *Enter* aktivuje odemykání truhel. Po zmáčknutí jedné z kláves se zobrazí opět otázka, jako tomu bylo u odemykání dveří.

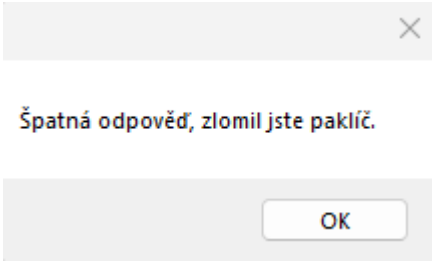


Obrázek 41 Otevírání truhel a trezorů

Jakmile hráč odpoví správně na otázku, tak získá mince uvnitř truhly nebo trezoru a změní se obrázek daného předmětu na otevřený stav.

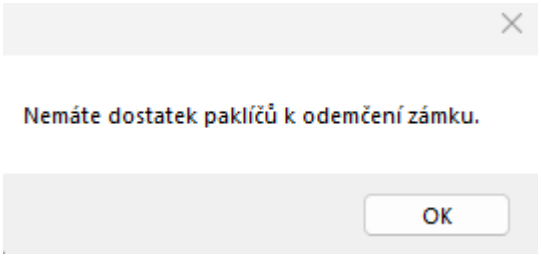
Obrázek 42 Otevřená truhla a trezor

Pokud hráč odpoví špatně na otázku, tak se zobrazí okno s informací, že zlomil paklíč.



Obrázek 43 Zlomený paklíč

Pokud hráč zlomí všechny paklíče, nebude už moct otevřít další truhly ani trezory. Při pokusu o otevření truhly nebo trezoru se zobrazí okno s informací, že hráč již nemá paklíče.

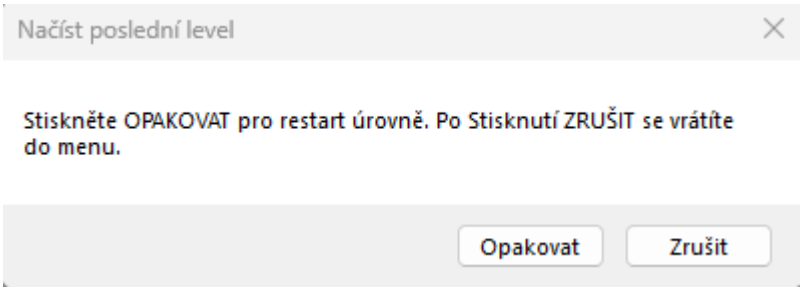


Obrázek 44 Hráč nemá paklíče

Pokud hráč dojde k levé straně úrovně, zobrazí se text s informací, co má hráč udělat, jestli se chce vrátit do menu.

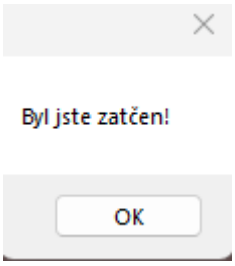
Obrázek 45 Zpět do menu

Pokud hráč zmáčkne E nebo Enter u okraje mapy, když je zobrazen text na obrázku 45, tak se zobrazí okno s informací, jestli chce hráč opakovat úroveň nebo jestli se chce vrátit do menu.



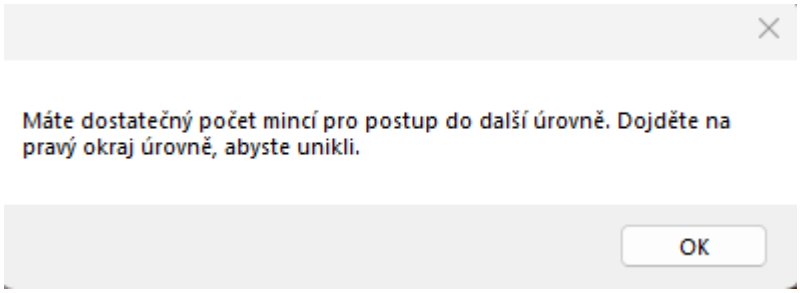
Obrázek 46 Opakování úrovně nebo návrat do menu

Jakmile hráče chytne některý ze strážců, tak se zobrazí okno s informací, že byl hráč zatčen a následně se zobrazí dialogové okno pro opakování úrovně.



Obrázek 47 Zatčení hráče

Jakmile hráč nasbírá dostatek mincí pro postup do další úrovně, tak se zobrazí okno s informací, kam má hráč dojít, aby pokračoval do další úrovně.

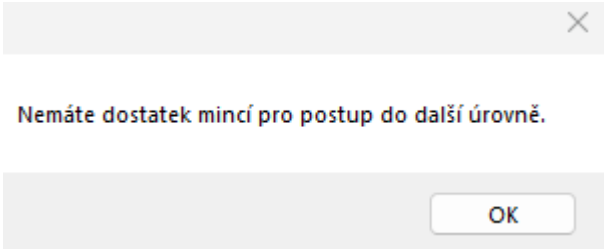


Obrázek 48 Informace pro pokračování do další úrovně

Jakmile hráč dojde k pravé části okna dané úrovně, tak se zobrazí text s informací, co má zmáčknout, aby pokračoval do další úrovně. Pokud má hráč dostatečný počet mincí a zmáčkne klávesu *E* nebo *Enter*, když se jeho postava nachází na správném místě, tak se okno se současnou úrovní zavře a spustí se další úroveň.

Obrázek 49 Pokračování do další úrovně

Pokud hráč nemá dostatek mincí pro postup do další úrovně, tak se zobrazí okno s informací, kolik mincí potřebuje nasbírat v dané úrovni, aby mohl pokračovat do další úrovně.



Obrázek 50 Nedostatek mincí pro postup do další úrovně

5.15 Druhá úroveň

Ve druhé úrovni musí hráč nasbírat 2000 mincí, aby mohl postoupit do další úrovně. Hráče zde může odemknout 9 truhel a 7 trezorů. V úrovni se nachází 5 stráží. Hráč má na začátku k dispozici tři paklíče.

Obrázek 51 Úroveň 2

5.16 Třetí úroveň

Ve třetí úrovni se nachází 13 trezorů a 12 truhel. Hráč musí v této úrovni nabírat celkem 3000 mincí, aby mohl pokračovat do další úrovně. V této úrovni se nachází 5 stráží a hráč má k dispozici 3 paklíče.

Obrázek 52 Úroveň 3

5.17 Čtvrtá úroveň

Ve čtvrté úrovni musí hráč získat 3500 mincí, aby mohl pokračovat do další úrovně. V této úrovni se nachází 13 truhel a 9 trezorů a chrání je 5 stráží. Ve čtvrté úrovni má hráč k dispozici 5 paklíčů.

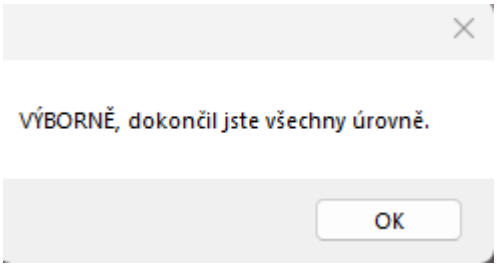
Obrázek 53 Úroveň 4

5.18 Pátá úroveň

Pátá úroveň je poslední úrovní ve hře. Hráč zde musí nasbírat 4000 mincí, aby dokončil hru. V této úrovni se nachází 11 trezorů a 16 truhel, které jsou chráněny 8 strážemi. V páté úrovni má hráč k dispozici 5 paklíčů.

Obrázek 54 Úroveň 5

Jakmile hráč nasbírá dostatečný počet mincí v poslední úrovni a dojde k okraji úrovně, aby ji opustil. Tak se zobrazí okno s informací, že dokončil všechny úrovně. Po kliknutí na *OK,* se zobrazí menu hry.



Obrázek 55 Dokončeny všechny úrovně

ZÁVĚR

Cílem diplomové práce bylo vytvořit 2D počítačovou hru v programu Visual Studio pomocí programovacího jazyka C#. Dále má hra sloužit k procvičení vědomostí z oboru BTSM.

V teoretické části práce je popsáno Visual Studio, jsou zde uvedeny užitečné funkce editoru kódu, tvorba nového projektu a popis vývojového prostředí. Ve druhé části je popsán programovací jazyk C#. Je zde uveden stručný popis jazyka C# a jeho funkcí, včetně ukázek kódu. V poslední části jsou popsány již vytvořené 2D hry, které byly vytvořeny pomocí programovacího jazyka C# ve vývojovém prostředí Visual Studio.

V praktické části je popsán vývoj 2D hry. Hra je nejdříve popsána z pohledu programátora, je zde uveden popis tvorby grafiky hry, použité funkce a ukázky kódu. Nakonec je hra popsána z pohledu hráče, spolu s popisem jsou uvedeny obrázky všech oken, která se můžou zobrazit hráči při hraní.

Hra Thief Adventure obsahuje celkem 5 úrovní. V každé úrovni přibývá počet mincí, které hráč musí získat z truhel a trezorů, aby se dostal do další úrovně. K otevření truhel a trezorů jsou potřeba paklíče. Aby hráč odemknul truhlu nebo trezor, tak musí odpovědět správně na otázku. Pokud hráč odpoví správně, tak získá mince z dané truhly nebo trezoru, v opačném případě dojde ke zlomení paklíče. Po dokončení jednotlivých úrovní se uloží do excel souboru informace, že byla dána úroveň dokončena a počet získaných mincí. Díky tomu nemusí hráč začínat při dalším spuštění hry od první úrovně, ale může pokračovat v úrovni kde skončil. Hru je možné spustit pomocí Visual Studia nebo spustitelného exe souboru.

Hra je určena pro hráče, kteří mají znalosti z předmětů tykajících se oboru BTSM. Jelikož se otázky nachází v samotném excel souboru, tak je možné databázi jednoduše rozšířit dalšími otázkami nebo je nahradit otázkami z jiných oborů. Hra obsahuje celkem 120 otázek. V databázi otázek jsou otázky z předmětů Systém bezpečnosti a veřejná správa, Krizové plánování a řízení a Bezpečnost informačních systémů. Otázky se ve hře zobrazují v náhodném pořadí. Hra také obsahuje možnost si jednotlivé otázky projít postupně.

Vytvořená 2D hra může sloužit k procvičení otázek z oboru BTSM zábavnou formou.

SEZNAM POUŽITÉ LITERATURY

[1] Visual Studio 2022. *Visual* [online]. [cit. 2023-02-18]. Dostupné z: https://visualstudio.microsoft.com/cs/

[2] Funkce editoru kódu. *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/visualstudio/ide/writing-code-in-the-code-and-text-editor?view=vs-2022

[3] První seznámení s ladicím programem sady Visual Studio. *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/visualstudio/debugger/debugger-feature-tour?view=vs-2022

[4] Programovací jazyk C# Marek Běhálek. *DOCPLAYER* [online]. [cit. 2023-02-18]. Dostupné z: https://docplayer.cz/298028-Programovaci-jazyk-c-marek-behalek.html

[5] Datové typy hodnotové a referenční. *Uzlabina2* [online]. [cit. 2023-02-18]. Dostupné z: http://uzlabina2.aspone.cz/dathr.aspx

[6] Lekce 4 - Typový systém podruhé: Datové typy v C# .NET *Itnetwork* [online]. [cit. 2023-02-18]. Dostupné z: https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-typovy-system-podruhe-datove-typy-string

[7] Modifikátory přístupu (Referenční dokumentace jazyka C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/keywords/access-modifiers

[8] Readonly – modifikátor (Referenční dokumentace jazyka C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/keywords/readonly

[9] Sealed (Referenční dokumentace jazyka C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/keywords/sealed

[10] Pole (Průvodce programováním v C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/programming-guide/arrays/

[11] Operátory a výrazy jazyka C# (referenční dokumentace jazyka C#). Microsoft [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/operators/

[12] Aritmetické operátory (Referenční dokumentace jazyka C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/operators/arithmetic-operators

[13] Operátory přiřazení (referenční dokumentace jazyka C#). *Microsoft* [online]. [cit. 2023-02-18]. Dostupné z: https://learn.microsoft.com/cs-cz/dotnet/csharp/language-reference/operators/assignment-operator

[14] Beginning C# 7 Programming with Visual Studio 2017, Benjamin Perkins, Jacob Vibe Hammer, Jon D. Reid, Vydavatel: John Wiley & Sons, 2018, ISBN: 1119458722, 9781119458722, počet stran 912.

[15] Cykly. *Itnetwork* [online]. [cit. 2023-02-18]. Dostupné z: https://www.itnetwork.cz/csharp/zaklady/c-sharp-tutorial-cykly-for-while

[16] Výjimky, generické třídy a metody. *Moodle.sspbrno* [online]. [cit. 2023-02-18]. Dostupné z: https://moodle.sspbrno.cz/pluginfile.php/5463/mod_resource/content/0/CSHARP/Vyjimky_genericke_tridy_a_metody.pdf

[17] Výjimky a zpracování chyb v.net/c# Zpracování chyb Výjimky Logování Code Contracts. *DOCPLAYER* [online]. [cit. 2023-02-18]. Dostupné z: https://docplayer.cz/70538186-Vyjimky-a-zpracovani-chyb-v-net-c-zpracovani-chyb-vyjimky-logovani-code-contracts.html

[18] C# Tutorial Create a side scrolling platform game in visual studio. *Moo ICT – Project Based Tutorials* [online]. [cit. 2023-02-25]. Dostupné z: https://www.mooict.com/c-tutorial-create-a-side-scrolling-platform-game-in-visual-studio/

[19] Snake project. *Ottobotcode* [online]. [cit. 2023-02-25]. Dostupné z: https://ottobotcode.com/snakeAssets/

[20] Car Racing Complete Game in c#. *YouTube* [online]. [cit. 2023-02-25]. Dostupné z: https://www.youtube.com/watch?v=xyggRDkoOwU&ab_channel=miniprogrammers

[21] C# Tutorial – Create a helicopter flying and shooting game in visual studio. *Moo ICT – Project Based Tutorials* [online]. [cit. 2023-02-25]. Dostupné z: https://www.mooict.com/c-tutorial-create-a-helicopter-flying-and-shooting-game-in-visual-studio/

[22] What is Inkscape ? . *Inkscape* [online]. [cit. 2023-03-23]. Dostupné z: https://inkscape.org/about/

[23] Bezpečnost informačních systémů. *Digitální knihovna UTB* [online]. [cit. 2023-04-28]. Dostupné z: https://digilib.k.utb.cz/handle/10563/52082

[24] Krizové plánování a řízení. *Moodle UTB* [online]. [cit. 2023-04-28]. Dostupné z: https://moodle.utb.cz/mod/folder/view.php?id=497877

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

BTSM Bezpečnostní technologie, systémy a management

IDE integrated development environment

MC Microsoft

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: OBSAH CD

Obsah přiloženého CD:

- DP v elektronické podobě
- Hra Thief Adventure