Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Počítačová 2D hra typu RPG – Začarovaný les

Tomáš Zlatohlávek

Bakalářská práce
2024

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Tomáš Zlatohlávek**
Osobní číslo:	**I19161**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Počítačová 2D hra typu RPG – Začarovaný les**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

V teoretické části bude představen framework Monogame, který byl použit při tvorbě této hry. Rozebrány budou některé klíčové části kódu, použité datové struktury, systém ukládání a načítání hry apod.
Implementační část obsahuje samotný kód hry v jazyce C#, který bude zveřejněný na stránce Github. Bude vytvořen herní manuál pro rychlé pochopení principu hry, schopností jednotlivých postav, představení významu atributů a jejich roli ve hře.

Rozsah pracovní zprávy: **min. 30 stran**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam doporučené literatury:

THE MONOGAME TEAM. MonoGame documentation [online]. 2012 [cit. 2023-10-19]. Dostupné z: https://docs.monogame.net/
SALIN, Louis a Rami MORRAR. Game Development with MonoGame [online]. Apress Berkeley, CA, 2021 [cit. 2023-10-19]. ISBN 978-1-4842-7771-3. Dostupné z: https://doi.org/10.1007/978-1-4842-7771-3
PAVLEAS, Jebediah, Jack KENG-WEI CHANG, Kelvin SUNG a Robert ZHU. Learn 2D Game Development with C# [online]. Apress Berkeley, CA, 2014 [cit. 2023-10-19]. ISBN 978-1-4302-6605-1. Dostupné z: https://doi.org/10.1007/978-1-4842-7771-3

Vedoucí bakalářské práce: **Ing. Jan Panuš, Ph.D.**
Katedra informačních technologií

Datum zadání bakalářské práce: **15. prosince 2023**
Termín odevzdání bakalářské práce: **10. května 2024**

L.S.

Ing. Zdeněk Němec, Ph.D. v.r.
děkan

Ing. Jan Panuš, Ph.D. v.r.
vedoucí katedry

V Pardubicích dne 28. února 2024

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 30. 4. 2024

Tomáš Zlatohlávek

PODĚKOVÁNÍ

Rád bych zde poděkoval vedoucímu bakalářské práce panu Ing. Janu Panušovi za odborné vedení a jeho pomoc při realizaci této bakalářské práce. Dále bych rád poděkoval všem učitelům a pracovníkům Univerzity Pardubice, především z fakulty Elektrotechniky a Informatiky, kteří mi byli nápomocni při mém vzdělávání na této škole. V neposlední řadě také mojí rodině, spolužákům, přítelkyni a přátelům za pomoc a oporu během mých studií, pro která jsem se v pozdějším věku rozhodl.

ANOTACE

Bakalářská práce se věnuje tvorbě počítačové 2D RPG hry s názvem Začarovaný les. Bude v ní nejprve představen C# framework MonoGame, který tato hra využívá a rozebrány jeho hlavní části, které jsou při vývoji této hry využity. Rovněž budou rozebrány klíčové části kódu samotné hry. Zdrojový kód lze nalézt na uložišti Github, kde je kompletně dostupný, stejně tak i příručka a dokumentace hry. Tato práce tedy může sloužit jako berlička, pro vytvoření podobného stylu hry v tomto frameworku.

KLÍČOVÁ SLOVA

počítačová hra, framework MonoGame, RPG, programovací jazyk C#, vývoj počítačových her, programování her, uložiště Github

TITLE

Computer 2D RPG game – Enchanted forest

ANNOTATION

The bachelor's thesis is devoted to the creation of a computer 2D RPG game called Enchanted Forest. It will first introduce the C# framework MonoGame, which this game uses, and analyze its main parts, that are used in the development of this game. Key parts of the game code itself will also be broken down. The source code can be found on the Github repository, where it is fully accessible, as well as the game's manual and documentation. This work can serve as a crutch to create a similar style of pc game in this framework.

KEYWORDS

computer game, framework MonoGame, RPG, C# programming language, pc game development, game programing, Github repository

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM ZDROJOVÝCH KÓDŮ

SEZNAM ZKRATEK

RPG	Role Playing Game
2D	Two-dimensional
3D	Three-dimensional
MGCB	MonoGame Content Builder
UI	User Interface
AI	Artificial Intelligence
C#	C-Sharp
OS	Operating System
iOS	iPhone Operating System
iPadOS	iPad Operating System
PC	Personal Computer

ÚVOD

V teoretické části bakalářské práce je nejprve představen framework MonoGame a jeho instalace na operačním systému Windows. Dále také vytvoření projektu ve Visual Studiu 2022 a rozebrány jednotlivé části třídy Game.cs, která je součástí šablony frameworku MonoGame. Vysvětlen a názorně předveden je i nástroj MonoGame Content Builder a pro ukázku představeny známé hry vytvořené ve frameworku MonoGame.

V praktické části se bakalářská práce věnuje tvorbě počítačové 2D RPG hry s názvem Začarovaný les pomocí již zmíněného frameworku MonoGame ve vývojovém prostředí Visual Studio 2022. Obsahuje prvotní návrh v programovém prostředí Raptor, motivaci, analýzu a na ukázkách kódu jsou rozebrány hlavní části hry. Především se jedná o zdrojové kódy postavy, schopností, systému boje, logiky nepřátel, ukládání a načítání hry, mapového systému a jeho struktury. Zdrojový kód je kompletně dostupný na uložišti GitHub, stejně tak i příručka a dokumentace hry.

Tato práce má za cíl posloužit jako berlička při tvorbě podobného typu hry ve frameworku MonoGame a představit některé jeho možnosti teoreticky a prakticky.

1 FRAMEWORK MONOGAME

1.1 Představení

MonoGame je jednoduchý a účinný .NET framework pro vytváření her na stolní počítače, herní konzole a mobilní zařízení za použití programovacího jazyka C#.

Obrázek 1 – Logo frameworku MonoGame Zdroj: [1]

Jedná se o náhradu již nepodporovaného frameworku XNA od společnosti Microsoft. Mezi přední vlastnosti a možnosti tohoto frameworku patří například 2D a 3D renderování, přehrávání zvuků a hudby, ovládání pomocí vstupů z klávesnice, myši, dotykových a herních zařízení jako je ovladač, nebo joystick. Obsahuje také různé optimalizace, nebo knihovnu matematických operací optimalizovanou právě pro hry. [1; 7, s. 4]

Tento framework nelze označit přímo za engine, jelikož zde není žádný editor scén (jako například u Unity, nebo Unreal Enginu). Obsahuje však veškeré nástroje, které lze využít při tvorbě vlastního enginu. Je tedy i vhodný pro lidi, které zajímá jak vytvořit hru bez předem vyšlápnutých kolejí již existujícím enginem a nakouknout tak pod pokličku herního vývojářství. Jedná se zároveň o open-source projekt, který je spravovaný komunitou. Je tedy prostor pro pomoc s tvořením kvalitní dokumentace, tutoriálů, video návodů apod.

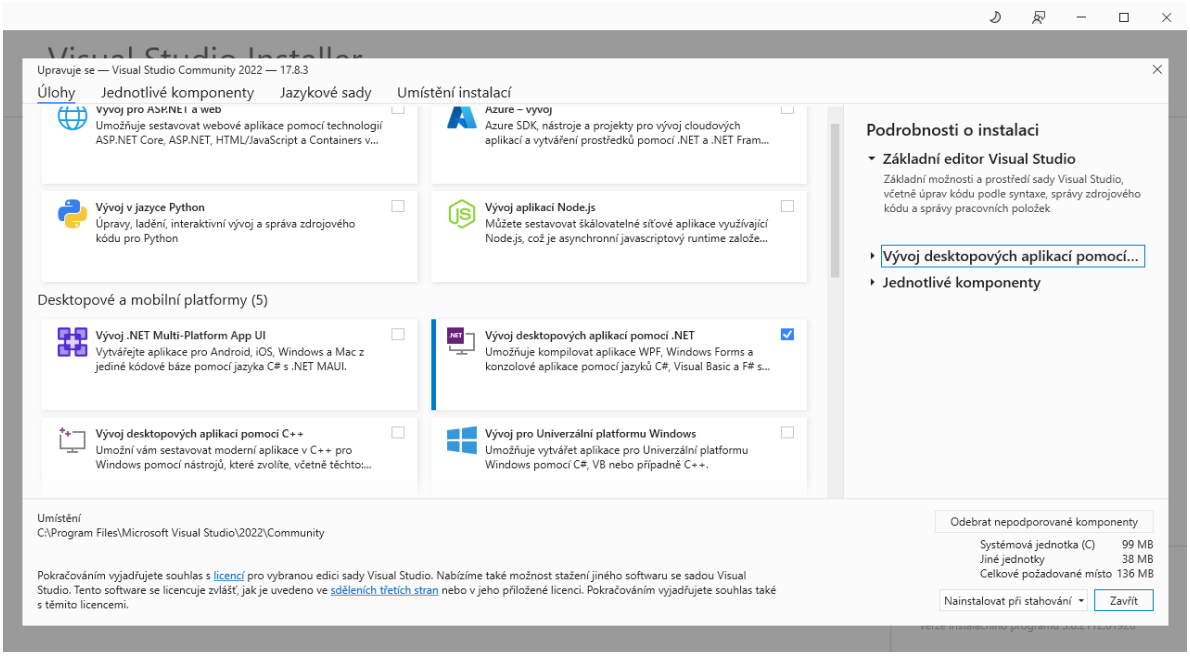
1.2 Instalace

MonoGame framework je primárně určen pro operační systémy Windows, macOS a Linux a dokáže pracovat s většinou .NET kompatibilních nástrojů, ale tvůrci doporučují pracovat s vývojovým prostředím Visual Studio 2022. Alternativou k těmto nástrojům může být JetBrains Rider, nebo Visual Studio Code. V této práci bude uvedeno pouze nastavení a instalace pro vývojové prostředí Visual Studio 2022 v operačním systému Windows. Další možnosti instalace a nastavení jsou k nalezení na internetových stránkách frameworku MonoGame pod odkazem na dokumentaci. [2; 7, s. 18]

1.2.1 Nastavení vývojového prostředí Visual Studio 2022 v systému Windows

Pokud vývojové prostřední Visual Studio 2022 není nainstalované, tak je k nalezení na internetových stránkách společnosti Microsoft. Komunitní edice je zcela zdarma pro individuální vývojáře, akademické využití a open source projekty. Edice Professional je určená pro individuální využití a edice Enterprise pro organizace. Komunitní edice je pro potřeby MonoGame dostačující a byla využita při tvorbě hry Začarovaný les. [3]

Při instalaci, nebo dodatečně pomocí softwaru Visual Studio Installer je nutné, aby byla vybrána možnost „Vývoj desktopových aplikací pomocí .NET“ (viz Obrázek 2). Tato možnost



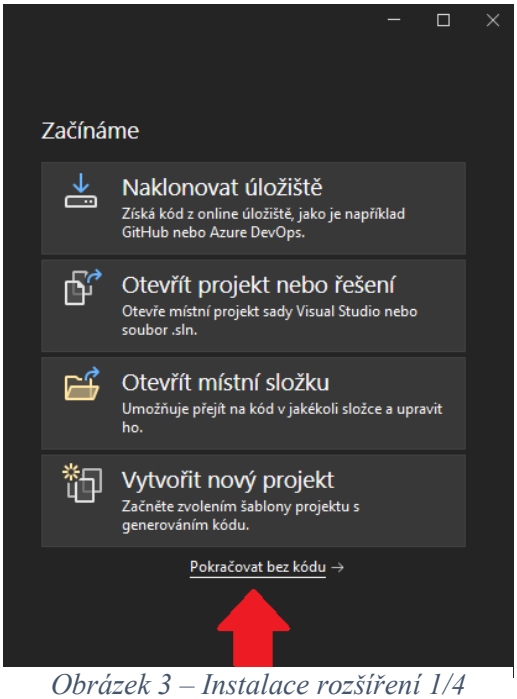
Obrázek 2 – Instalace komponent Zdroj: Visual Studio 2022

je naprosto nezbytná pro všechny cílové platformy. Volitelné možnosti jsou dále „Vývoj .NET Multi-Platform App UI“, pro vývoj na zařízení s OS Andoid, iOS, nebo iPadOS a „Vývoj pro Univerzální platformu Windows“ pro vývoj do služby Windows store, nebo na herní konzole Xbox.

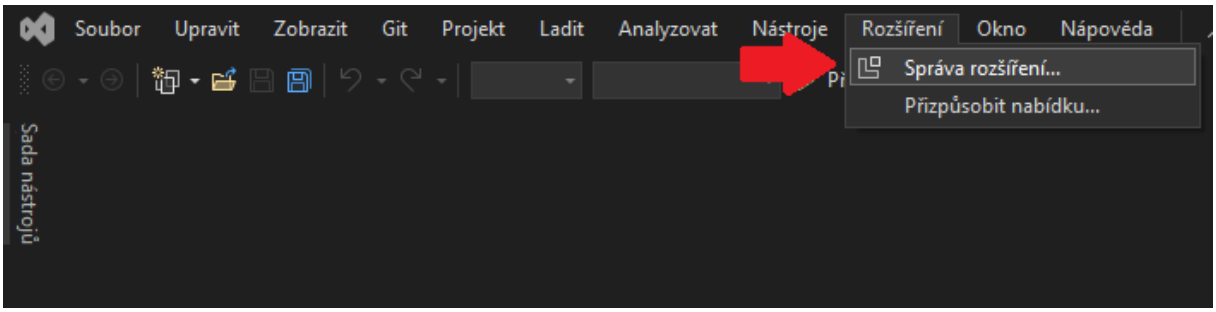
Instalaci samotného rozšíření, díky kterému je možné vytvořit MonoGame projekty z předem vytvořených šablon je potřebné provést nejprve spuštěním Visual Studia 2022.

Při spuštění Visual Studia bude nabídnuto několik možností. Vybráním možnosti „Pokračovat bez kódu“ se vývojové prostředí spustí, aniž by byl otevřený nějaký projekt (viz Obrázek 3).

Následně je nutné v horním panelu zvolit možnost „Rozšíření“ a po rozevření kontextové nabídky pokračovat kliknutím na tlačítko „Správa rozšíření…“ (viz Obrázek 4).

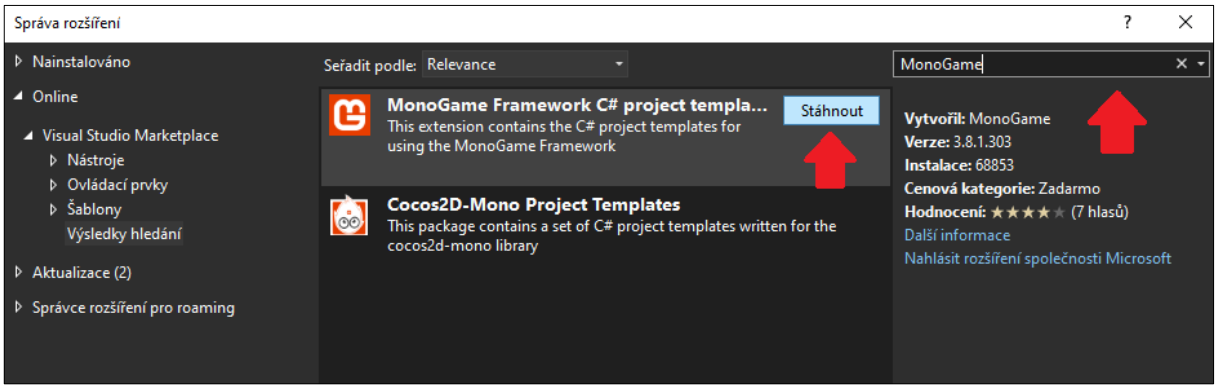


Obrázek 3 – Instalace rozšíření 1/4 Zdroj: Visual Studio 2022



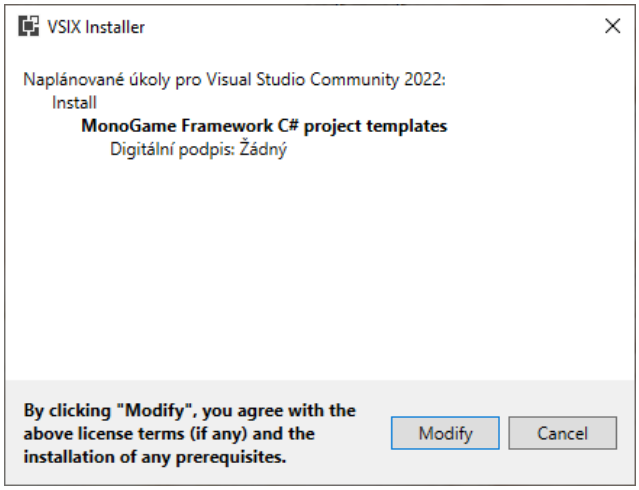
Obrázek 4 – Instalace rozšíření 2/4 Zdroj: Visual Studio 2022

Ve správci rozšíření lze nyní v pravé horní části vyhledat výraz „MonoGame“ a u výsledku „MonoGame Framework C# project templates“ kliknout na tlačítko „Stáhnout“ (viz Obrázek 5). Následně je pro zahájení instalace potřebné ukončit Visual Studio 2022 a zavřít veškerá okna spojená s touto aplikací.



Obrázek 5 – Instalace rozšíření 3/4 Zdroj: Visual Studio 2022

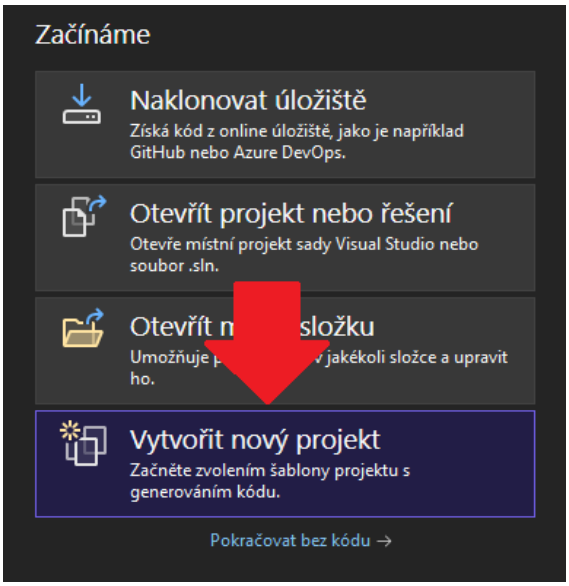
Po uzavření Visual Studia 2022 se otevře okno „VSIX Installer“ (viz Obrázek 6), kde bude uživatel požádán o potvrzení změn týkajících se instalace rozšíření projektových šablon pro MonoGame Framework. Klepnutím na tlačítko „Modify“ vyjádří uživatel svůj souhlas, toto rozšíření bude nainstalováno a při vytváření projektu si nyní bude moci vybrat z předem připravených šablon frameworku MonoGame.



Obrázek 6 – Instalace rozšíření 4/4 Zdroj: Visual Studio 2022

1.2.2 Vytvoření MonoGame projektu ve Visual Studiu 2022

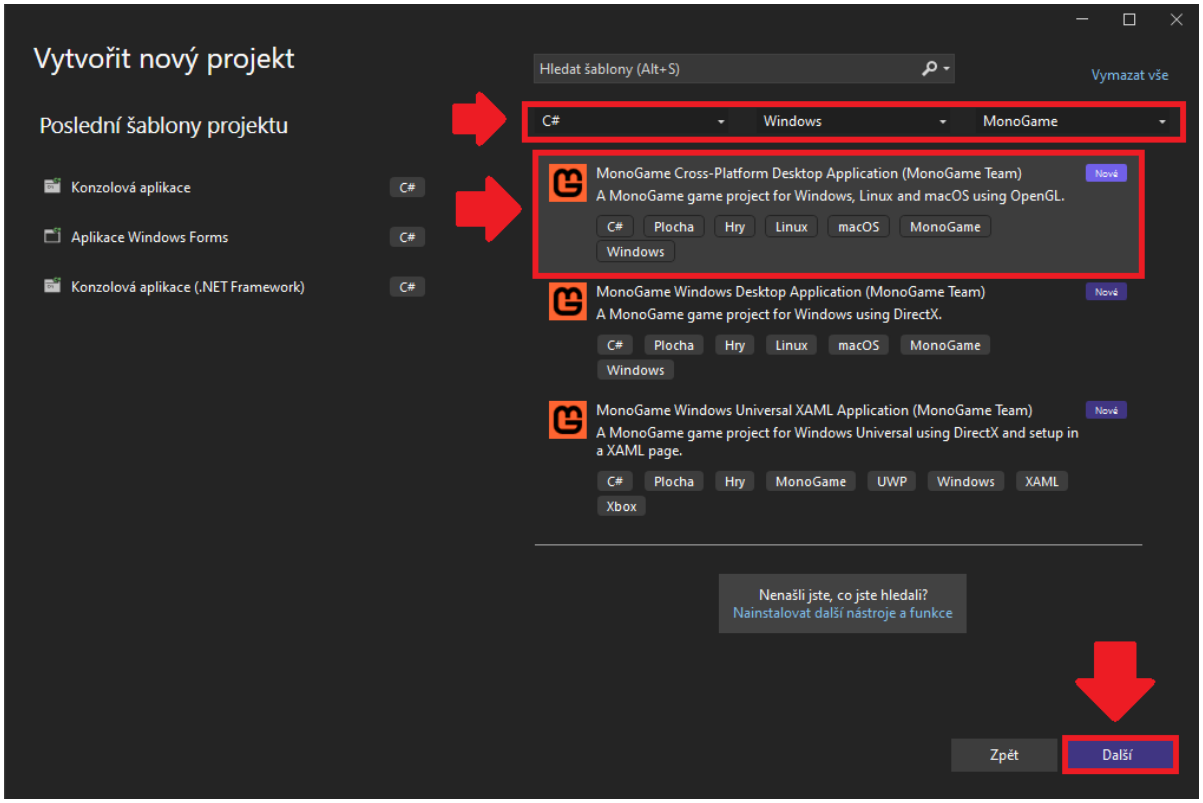
Pokud je Visual Studio 2022 správně nainstalováno společně s rozšířením pro MonoGame Framework, tak je možné vytvořit projekt s pomocí předem připravených šablon při spuštění programu kliknutím na možnost „Vytvořit nový projekt“ (viz Obrázek 7). [4; 7, s. 40]



Obrázek 7 – Vytvoření MonoGame projektu 1/2 Zdroj: Visual Studio 2022

V okně pro vytvoření nového projektu (viz Obrázek 8) je doporučeno pro snazší orientaci správně nastavit filtry zobrazené v horní části obrázku. Pro testovací účely lze vybrat šablonu „MonoGame Cross-Platform Desktop Application (MonoGame Team)“ a kliknout na tlačítko „Další“.

Na dalším okně bude vyplněn název projektu (název projektu nesmí obsahovat mezery), případně lze přizpůsobit umístění projektu a následně stačí kliknout na tlačítko „Vytvořit“.



Obrázek 8 – Vytvoření MonoGame projektu 2/2 Zdroj: Visual Studio 2022

Pro otestování funkčnosti lze spustit projekt stisknutím klávesy F5, nebo kliknutím na zelenou ikonu trojúhelníku. Po spuštění by se mělo objevit herní okno s modrou plochou.

1.3 Hlavní části frameworku

V této části budou rozebrány jednotlivé části generovaného kódu a další nástroje přidružené frameworku MonoGame.

1.3.1 Herní třída

Uvnitř souboru třídy Game.cs, který je jádrem MonoGame projektu, lze nalézt několik kritických sekcí, potřebných ke spuštění a běhu hry. Jsou jimi: [6; 9, s. 22]

- Direktivy using, které poskytují jednoduchý přístup k různým komponentům frameworku MonoGame (viz Zdrojový kód 1, řádky 1-3). Tyto direktivy mají prefix „Microsoft.Xna.Framework", protože MonoGame je open-source re-implementací frameworku XNA od Microsoftu a z důvodu kompatibility používá stejné jmenné prostory.
- Definici herní třídy, která je srdcem MonoGame projektu (viz Zdrojový kód 1, řádek 7). Třída Game1 dědí z třídy Game, která poskytuje všechny hlavní metody. Název této třídy není tak důležitý, protože běžně je součástí projektu pouze jedna.
- Instanční proměnné tříd GraphicsDeviceManager a SpriteBach (viz Zdrojový kód 1, řádky 9 a 10) jsou dvěma výchozími proměnnými šablony frameworku MonoGame. Obě z nich jsou využívány pro vykreslování na obrazovku.
- Herní konstruktor s inicializací proměnných (viz Zdrojový kód 1, řádky 12-17). V tomto případě je vytvořena instance třídy GraphicsDeviceManager a je nastaven kořenový adresář pro soubory herního obsahu.
- Inicializační metoda pro inicializování hry při startu (viz Zdrojový kód 1, řádky 19-23). Tato metoda je volána po konstruktoru, ale před hlavní herní smyčkou (metody Update a Draw).
- Metody pro načtení a odebrání obsahu (viz Zdrojový kód 1, řádky 25-29). V šabloně lze nalézt pouze metodu pro načtení obsahu, tato metoda je volána skrze inicializační metodu.

- Aktualizační metoda Update pro úpravu herní logiky v pravidelných intervalech (viz Zdrojový kód 1, řádky 31-37). Šablona také obsahuje kód pro vypnutí hry.
- Vykreslovací metoda Draw pro vykreslení grafických prvků v pravidelných intervalech na obrazovku. (viz Zdrojový kód 1, řádky 39-44).

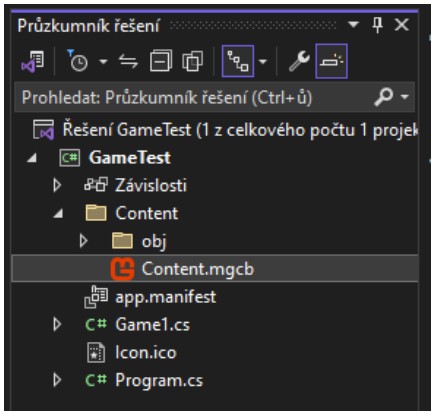
```
1. using Microsoft.Xna.Framework;
2. using Microsoft.Xna.Framework.Graphics;
3. using Microsoft.Xna.Framework.Input;
4.
5. namespace GameTest
6. {
7.     public class Game1 : Game
8.     {
9.         private GraphicsDeviceManager _graphics;
10.        private SpriteBatch _spriteBatch;
11.
12.        public Game1()
13.        {
14.            _graphics = new GraphicsDeviceManager(this);
15.            Content.RootDirectory = "Content";
16.            IsMouseVisible = true;
17.        }
18.
19.        protected override void Initialize()
20.        {
21.            // TODO: Add your initialization logic here
22.            base.Initialize();
23.        }
24.
25.        protected override void LoadContent()
26.        {
27.            _spriteBatch = new SpriteBatch(GraphicsDevice);
28.            // TODO: use this.Content to load your game content here
29.        }
30.
31.        protected override void Update(GameTime gameTime)
32.        {
33.            if (GamePad.GetState(PlayerIndex.One).Buttons.Back ==
   ButtonState.Pressed || Keyboard.GetState().IsKeyDown(Keys.Escape))
34.                Exit();
35.            // TODO: Add your update logic here
36.            base.Update(gameTime);
37.        }
38.
39.        protected override void Draw(GameTime gameTime)
40.        {
41.            GraphicsDevice.Clear(Color.CornflowerBlue);
42.            // TODO: Add your drawing code here
43.            base.Draw(gameTime);
44.        }
45.    }
46.}
```

Zdrojový kód 1 – Třída Game1.cs a její sekce

1.3.2 Manažer obsahu – MonoGame Content Builder Tool (MGCB Editor)

Společně s instalací rozšíření pro MonoGame byl nainstalován i MGCB Editor, který slouží pro správu přidaného obsahu do hry. Zde bude ukázáno jak přidat soubor skrz tento editor do projektu frameworku MonoGame. [5; 7, s. 75]

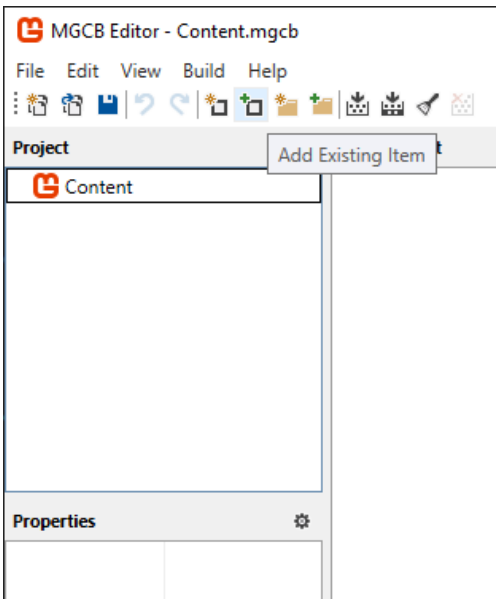
Nejprve je nutné otevřít projekt, který využívá MonoGame šablonu ve Visual Studiu 2022. V průzkumníku řešení je pak nutné otevřít soubor Content.mgcb pod složkou Content (viz Obrázek 9).



Obrázek 9 – MonoGame Content Builder Tool Zdroj: Visual Studio 2022

Po otevření by mělo být vidět otevřené okno MGCB Editoru. Existuje vícero možností přidání obsahu, je možné v editoru některé položky přímo vytvořit (například fonty). Další možností je přidat existující obsah a to buď jednotlivě, nebo celou složku. [8, s. 10]

V této ukázce bude přidán jeden obrázek pomocí položky nabídky nástrojů s názvem „Add Existing Item“ (viz Obrázek 10). Nyní stačí pouze vybrat požadovaný soubor a po upozornění, že se soubor nachází mimo náš projekt, vybrat možnost „Copy the file to the directory“, kliknout na tlačítko „Add“ a uložit změny v MGCB editoru pomocí známé ikony diskety, nebo klávesovou zkratkou „CTRL+S“.



Obrázek 10 – Přidání obrázku v MGCB Editoru Zdroj: Visual Studio 2022

Ve třídě Game1.cs je nyní nutné deklarovat proměnnou typu Texture2D pro přidaný obrázek a v metodě LoadContent() přidat načtení obrázku, kde bude jako parametr název souboru obrázku bez přípony typu souboru (viz Zdrojový kód 2). Stojí za povšimnutí, že třída, která se stará o obrázky (textury) se nazývá Texture2D a je uvedena ve špičatých závorkách jako datový typ v kolekci načítaného obsahu. Pokud bychom chtěli přidat jiný typ souboru, například zvuk, je nutné změnit jak datový typ deklarace, tak i datový typ ve špičatých závorkách v metodě LoadContent().

```
34.protected override void LoadContent()
35.{
36.    _spriteBatch = new SpriteBatch(GraphicsDevice);
37.    _obrazek = Content.Load<Texture2D>("obrazek");
38.}
```

Zdrojový kód 2 – Načtení obrázku ve třídě Game1.cs

Nyní načtený obrázek zbývá pouze vykreslit v metodě Draw. V ukázce je uvedeno pouze statické vykreslení na pozici (viz Zdrojový kód 3) a skládá se nejprve z otevření dávky obrázků

```
48.protected override void Draw(GameTime gameTime)
49.{
50.    GraphicsDevice.Clear(Color.CornflowerBlue);
51.    _spriteBatch.Begin();
52.    _spriteBatch.Draw(_obrazek, new Vector2(0, 0), Color.White);
53.    _spriteBatch.End();
54.    base.Draw(gameTime);
55.}
```

Zdrojový kód 3 – Vykreslení obrázku ve třídě Game1.cs

(řádek 51). Následně z přidání obrázku, který je třeba vykreslit společně s potřebnými parametry, do dávky (řádek 52). A uzavření dávky obrázků (viz řádek 53), čímž budou obrázky připraveny k vykreslení na obrazovku. Při zobrazování více obrázků musí být umístěny mezi metody „Begin“ a „End“ přesně v pořadí v jakém mají být vykresleny.

1.4 Příklady známých her

Framework MonoGame byl využit pro celou řadu známých her jako je například Stardew Valley, Streets of Rage 4, Carrion, Celeste a mnoho dalších.

1.4.1 Stardew Valley

Jedná se o RPG z prostředí venkova s otevřeným koncem. Věnujete se farmaření a skrz něj se snažíte vytvořit prosperující domov. Hra obsahuje i multiplayer a je určena pro PC, Mac, Linux, Xbox, PS4 a Switch. Existují i porty na iOS a Android. [10]

Obrázek 11 – Ukázka ze hry Stardew Valley – Zdroj: [10]

Hru vytvořil herní designer Eric Barone, který pod aliasem ConcernedApe hru vyvíjel zcela sám po celé 4 roky. Naučil se vše potřebné pro vývoj takové hry, jako je vytváření herní hudby, grafiky, programování a herní design.

Hra vyšla na PC 26. února 2016 a během 2 měsíců se prodal 1 milion kopií této hry. Za pár měsíců se hra rozšířila na Mac i Linux a vyšla pro herní konzole Xbox One a Playstation 4 v prosinci 2016, na Nintendo Switch později v roce 2017 a na iOS v říjnu 2018. Dále přibyla i rozšířená jazyková podpora a v roce 2018 vyšel multiplayer.

Síťový kód tehdy vytvořil programátor Tom Coxon. Hra se stále vyvíjí a přibývá stále další a další obsah. K březnu 2022 bylo prodáno přes 20 milionů kopií této hry napříč všemi platformami a 13 milionů z toho bylo pro osobní počítače.

1.4.2 **Celeste**

Hra Celeste se žánrově řadí mezi plošinovkové adventury. Jedná se o příběh mladé dívky jménem Madeline, která se snaží zdolat horu Celeste. Tato hora je velmi vysoká, nebezpečná, a aby toho nebylo málo, tak zde straší. Tato slečna má totiž pocit, že nic jiného ve svém životě nedokáže a tak se vydá na tuto strastiplnou cestu plnou nástrah a různých monster. [11]

Obrázek 12 – Ukázka ze hry Celeste – Zdroj: [11]

Mezi podporované platformy se řadí PC a Mac, Nintendo Switch, Playstation 4 a Xbox One. Hra byla vydána a publikována celkem v 11 světových jazycích vývojářským studiem Matt Makes Games Inc. dne 25. ledna 2018. Mezi další známou hru od tohoto vývojářského studia patří hra TowerFall. Ovládání hry je velmi rychlé a přívětivé a určitě potěší nejednoho fanouška plošinových her. [12]

2 POČÍTAČOVÁ 2D HRA TYPU RPG – ZAČAROVANÝ LES

2.1 Představení

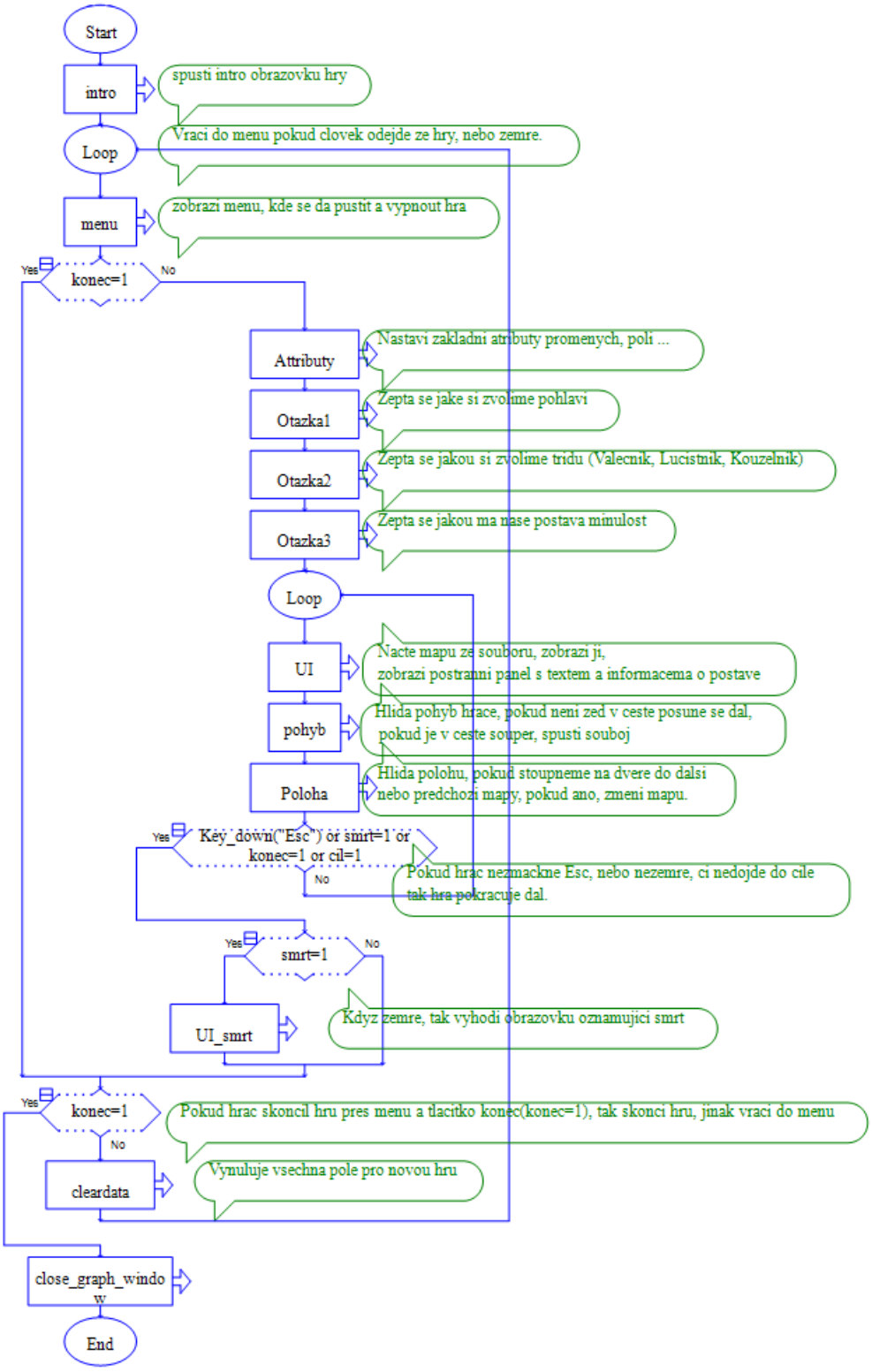
Počítačová hra Začarovaný les je 2D RPG tahová hra, kde si vyberete jednu ze tří tříd (Válečník, Lučištník a Kouzelník), postupně porážíte své nepřátele promyšlenými tahy, procházíte úrovněmi, až se dostanete na konec Začarovaného lesa. Cílem hry je tedy vyčistit les od banditů, kteří se v něm usadili a obsadit jejich tábor. Hra obsahuje ale také režim duelu, kde si uživatel může zvolit, že budou proti sobě hrát dvě postavy a každá z postav může být ovládána buď hráčem, nebo umělou inteligencí. Lze si tedy zahrát proti jinému člověku, nebo například i sledovat jak proti sobě hrají dvě počítačem řízené postavy.

Hry takové typu bývají většího rozsahu a zaberou spoustu práce při svém vývoji. Často vyžadují celý tým programátorů, designerů, zvukařů, animátorů, grafiků, scénáristů apod. Při vývoji této hry tedy bylo přihlédnuto ke skutečnosti, že hru dělá pouze jednotlivec a tak bylo rozhodnuto, že se hra bude soustředit na přesun po mapě a souboje a určité vybalancování mezi jednotlivými třídami.

2.1.1 Prvotní návrh v programovacím prostředí Raptor

Prvotní návrh této hry vznikl v programovacím prostředí Raptor, který je založený na vývojových diagramech (viz Obrázek 13). Syntaxe je v tomto prostředí omezena na minimum a program je sestavován vizuálně pomocí vývojových diagramů. Na obrázku lze vidět pouze kostru celého programu. Jednotlivé bloky vývojových diagramů obsahují samostatné úseky grafů, zajišťující popisovanou činnost v jednotlivých komentářích vyznačených zelenou barvou. Když se hra v softwaru Raptor pustí, tak lze sledovat grafické okno, které je vytvořeno a současně i krokový běh ve vývojových diagramech. Toto krokování připomíná režimy ladění jiných programovacích nástrojů, kde lze krokově sledovat posloupnost zdrojového kódu. Hru lze samozřejmě i zkompilovat a pustit bez tohoto režimu, který průběh hry zpomaluje. Projekt vytvořený v softwaru Raptor je možné stáhnout z GitHubu a odkaz na něj lze nalézt v seznamu použité literatury. [14; 15]

Programovací prostředí Raptor sice nabízí možnost konverze vývojových diagramů do jiných programovacích jazyků, ale výsledek není vždy uspokojující. Proto se autor této práce rozhodl pro manuální předělání a vylepšení svého nápadu v objektově orientovaném programovacím jazyku. Tento návrh byl tedy postupně přepsán do jazyka C# ve frameworku MonoGame.



Obrázek 13 – Prvotní návrh v programovacím prostředí Raptor Zdroj: Vlastní

2.1.2 **Motivace**

Hlavním důvodem pro vytvoření této práce byl autorům zájem o podobný žánr počítačových her a touha vytvořit něco vlastního v podobném stylu. Rovněž i možnost sdílet nadšení pro tvorbu počítačových her s případnými čtenáři této práce a probudit v nich zápal pro vlastní tvorbu ve frameworku MonoGame.

Počítačové hry hrají čím dál větší roli v oblasti zábavy a těší se stále vzrůstající oblíbenosti. Cílem této práce je přiblížit framework MonoGame a nabídnout alternativu pro tvorbu her k oblíbeným herním enginům jako jsou Unity, Unreal Engine, Godot, Cocos2d nebo GameMaker.

2.1.3 **Analýza**

Během analýzy prvotního návrhu se autor rozhodl, že pro realizaci svého projektu využije framework MonoGame, který je vhodný pro 2D hry a vyniká svou jednoduchostí. Tento framework je schopný splnit funkční požadavky stanovené autorem práce. Mezi ně patří vytvoření tahové RPG hry, kde se postava s určitými atributy a ovládána hráčem postupně zlepšuje porážením nepřátel. Hráč je schopný si hru uložit, může vytvářet vlastní mapy, obsahuje různé úrovně obtížnosti nepřátel, pohyb po mapě, inventář atd. Požadavkem také bylo rozšíření hry o jednoduchou animaci při pohybu po mapě, rozšíření počtu schopností oproti původnímu návrhu a přidání režimu duel pro rychlou hru dvou hráčů proti sobě.

I po výkonnostní stránce je framework MonoGame dostačující a vzhledem k tomu, že se jedná pouze o tahové RPG, tak ani požadavky na výkonnost nejsou nijak vysoké. Množství uživatelů nemá na výkon vliv, neboť hra neobsahuje žádné síťové prvky.

Z analýzy také vyplynulo, že pro data nebude potřebné vytvoření databáze a veškerá data, jako jsou obrázky, zvuk a hudba, budou součástí projektu a výsledně zkompilovaného softwaru. Hra neobsahuje žádná uživatelská data a neshromažďuje žádné informace o uživateli, není tedy nutnost provádět nějaká bezpečnostní opatření pro ochranu dat uživatele.

2.1.4 **Zdrojový kód**

Jak již bylo řečeno, tato hra je psána v jazyku C# a celý zdrojový kód je k zobrazení na uložišti GitHub. [13]

2.2 Postava

Každá postava má své jméno, pohlaví, minulost, úroveň (level), počet získaných zkušeností, počet zkušeností potřebných na další úroveň, zdraví, manu, sílu, obratnost, inteligenci, brnění, schopnosti a svůj inventář.

2.2.1 Vlastnosti

Výchozí hodnoty pro některé vlastnosti jsou představeny na následující ukázce kódu.

```
215. Zkusenosti = 0;
216. ZkusenostiNext = 25 + Level * 25;
217. ZivotyMax = 70 + Level * 10;
218. ManaMax = 30 + Level;
219. Brneni = 1;
220. Sila = 5;
221. Obratnost = 5;
222. Inteligence = 5;
```

Zdrojový kód 4 – Výchozí hodnoty vlastností postav ze třídy Postava.cs

Z kódu je patrné, že s každou úrovní roste postavě počet životů a mírně i mana, která je využita při používání silných magických schopností. Postavy válečníka a lučištníka mají pouze jednu takovou schopnost, oproti tomu kouzelník má takových schopností pět. Zisk zkušeností je vysvětlen v další části.

U každé třídy se bonus získané síly, obratnosti a inteligence mění, typicky se dá říci, že válečník má nejvyšší bonus k síle a zároveň má i nejvyšší hodnotu brnění, lučištník nejvyšší bonus k obratnosti a menší bonus k brnění a kouzelník nejvyšší bonus k inteligenci a je mu přidáno 60 many. Každá třída má samozřejmě i své vlastní schopnosti, které k ní tematicky sedí. Konkrétní úpravy vlastností postavy válečníka, lučištníka a kouzelníka můžete vidět na ukázce kódu.

```
226.  case Trida.Valecnik:
227.      Sila += 3 + (Level - 1) * 2;
228.      Obratnost += -1 + (Level - 1);
229.      Inteligence += -2 + (Level - 1);
230.      Brneni += 2;

239.  case Trida.Lucistnik:
240.      Sila += -1 + (Level - 1) ;
241.      Obratnost += 3 + (Level - 1) * 2;
242.      Inteligence += -1 + (Level - 1);
243.      Brneni += 1;

252.  case Trida.Kouzelnik:
253.      Sila += -2 + (Level - 1);
254.      Obratnost += -1 + (Level - 1) ;
255.      Inteligence += 3 + (Level - 1) * 2;
256.      ManaMax += 60;
```

Zdrojový kód 5 – Úpravy vlastností pro třídy postav ze třídy Postava.cs

Dalším faktorem, který ovlivňuje vlastnosti postav je minulost postavy. Lze si ji představit jako setkání s určitou osobou, která ovlivnila charakter postavy a něco jí naučila. Hráč má na výběr ze třech možností a ty jsou buď setkání s rytířem, lovcem anebo s mágem. Rytíř poskytne postavě +1 k síle a +2 k brnění. Lovec poskytne postavě +1 k obratnosti a +20 životů. Mág poskytne postavě +1 k inteligenci a +20 many.

Nejmenší vliv na vlastnosti postavy má pohlaví. To jednoduše přidá +1 k síle pokud je postava muž anebo +1 k obratnosti pokud je postava žena.

2.2.2 **Význam vlastností**

K čemu ale vlastně vlastnosti jako síla, obratnost, inteligence a brnění slouží? Síla, obratnost a inteligence určují, jak silné budou schopnosti. Válečníkovy schopnosti jsou často založené na síle, u lučištníka zase na obratnosti a kouzelník nejlépe zužitkuje inteligenci. Každá

třída má však určitý prospěch i z ostatních vlastností. Konkrétní využití vlastností postavy pro sílu schopností lze nalézt v podkapitole efekty schopností.

Brnění má pro každou postavu obranný charakter a udává, o kolik bodů bude poškození z nepřátelské schopnosti, nebo efektu menší. Rozdílný efekt má brnění na magické a nemagické útoky. V případě nemagických (fyzických) útoků, je poškození zmenšeno přesně o udanou hodnotu brnění. Pokud se však jedná o magický útok, tak je efektivita brnění poloviční. Pokud je hodnota brnění postavy liché číslo a zaútočil by na ni někdo magickým útokem, tak se dělená hodnota brnění zaokrouhlí směrem k sudému číslu. Lze si tedy představit, že kdyby měla postava hodnotu brnění 3 a někdo by na ni zaútočil magickým poškozením za 10 bodů, tak její hodnota brnění proti magické schopnosti bude mít následně efektivní hodnotu 2 a magická schopnost jí tak ubere 8 bodů poškození.

2.2.3 **Získávání zkušeností a typy soupeřů**

Hráčova postava začíná s úrovní (levelem) jedna. Na další úroveň potřebuje získat 50 zkušenostních bodů, pokud se jí to povede, množství potřebných zkušeností na přechod na vyšší úroveň vzroste o 25 bodů. Na úrovni 3 je tedy potřebné získat 100 zkušenostních bodů, aby hráč povýšil na úroveň 4.

Nejprve bude hráč porážet nepřátele, kteří jsou ještě zelenáči, proto jsou vyznačeni zelenou barvou. Úroveň zelenáčů, může být buď jedna, nebo dvě a za jejich poražení je hráč odměněn 25 zkušenostními body. Středně těžcí soupeři mají úroveň 3, nebo 4 a jsou vyznačeni žlutou barvou a za jejich porážku postava obdrží 50 zkušenostních bodů. Těžcí soupeři mají barvu červenou a jejich úroveň je vždy alespoň 5, nebo pokud má hráč vyšší úroveň, tak vždy odráží úroveň hráče. Porazit červeného soupeře na začátku hry je tak s největší pravděpodobností nemožné, ale když se to hráči povede, získá 100 zkušenostních bodů.

2.2.4 **Inventář**

Každá postava disponuje svým inventářem. Ve hře se vyskytují pouze dvě položky inventáře a těmi jsou lahvičky zdraví a many. Tyto lahvičky lze sebrat na mapě a slouží jako záchrana při boji proti obzvláště silným soupeřům. Hráč by se měl typicky snažit porazit zelené a žluté nepřátele bez použití lahviček, může se totiž stát, že je bude potřebovat v boji proti červeným nepřátelům. Červení nepřátelé totiž můžou mít některou (je možné, že i obě) z lahviček také.

2.3 Vlastnosti Schopností

Schopnosti jsou důležitou součástí každé postavy. Představují způsob, jakým bude hráč při soubojích schopen porazit svého soupeře a jakým se i jeho soupeř bude bránit a útočit. Každá schopnost má určité vlastnosti, které rozhodují o povaze dané schopnosti (viz Zdrojové kódy 6, 7 a 8).

Mezi tyto vlastnosti patří druh schopnosti, který je reprezentován výčtovým typem a pomáhá s identifikací dané schopnosti. Dále také prodleva (cooldown, nebo také CD),

```
244. Schopnosti.Add(new Schopnost(Druh.Utok_Mecem, 0, 0, 0, false));
245. Schopnosti.Add(new Schopnost(Druh.Obrana_Stitem, 3, 0, 0, false));
246. Schopnosti.Add(new Schopnost(Druh.Bojovy_Pokrik, 2, 0, 0, false));
247. Schopnosti.Add(new Schopnost(Druh.Regenerace, 2, 0, 20, false));
248. Schopnosti.Add(new Schopnost(Druh.Uder_stitem, 5, 0, 0, false));
249. Schopnosti.Add(new Schopnost(Druh.Vrh_sekerou, 3, 0, 0, false));
250. Schopnosti.Add(new Schopnost(Druh.Berserk, 2, 0, 0, false));
```

Zdrojový kód 6 – Přidání schopností válečníka z třídy Postava.cs

což je doba, kterou musí postava počkat, než bude schopna použít schopnost znovu. Do této doby se počítá i aktuální odehrané kolo. Když je tedy použita schopnost, která má prodlevu 3 kola, tak následující dvě kola tuto schopnost nelze použít. Fáze schopnosti je další faktor, který určuje, jak dlouho bude trvat, než se schopnost použije. Většina schopností má fázi nastavenou na 0, což znamená, že je schopnost použita okamžitě. V této verzi hry mají pouze

```
244. Schopnosti.Add(new Schopnost(Druh.Bodnuti_Dykou, 0, 0, 0, false));
245. Schopnosti.Add(new Schopnost(Druh.Uskok, 3, 0, 0, false));
246. Schopnosti.Add(new Schopnost(Druh.Strelba_Lukem, 0, 1, 0, false));
247. Schopnosti.Add(new Schopnost(Druh.Magicky_sip, 0, 1, 20, true));
248. Schopnosti.Add(new Schopnost(Druh.Rychlost, 6, 0, 0, false));
249. Schopnosti.Add(new Schopnost(Druh.Lesni_bobule, 4, 0, 0, false));
250. Schopnosti.Add(new Schopnost(Druh.Jedova_sipka, 3, 0, 0, false));
```

Zdrojový kód 7 – Přidání schopností lučištníka z třídy Postava.cs

dvě schopnosti svoji výchozí fázi nastavenou na 1, což znamená, že na provedení akce schopnosti, musí postava jedno kolo počkat. Některé schopnosti lze také sesílat, pouze pokud má postava dostatečnou zásobu many, proto každá schopnost obsahuje údaj o ceně many. A dalším velmi důležitým údajem je skutečnost, zda je útočná schopnost magického typu, což je vyjádřeno buď pravdou, nebo nepravdou. Na přiložených zdrojových kódech jsou jednotlivé parametry konstruktoru třídy Schopnost zadány v pořadí, v jakém jsou zde vysvětleny, tedy od druhu schopnosti, až po skutečnost jestli je schopnost magického útočného typu (způsobuje magické poškození), či nikoliv.

Mezi schopnosti patří i používání lahviček zdraví, nebo many, které jsou součástí inventáře. Kromě přidruženého druhu schopnosti a faktu, že vypití lektvaru je nemagická schopnost, jsou na všech pozicích nuly. Lahvičky tedy nemají žádnou prodlevu, vypít je lze okamžitě a jejich vypití nestojí žádnou manu.

```
257. Schopnosti.Add(new Schopnost(Druh.Uder_Holi, 0, 0, 0, false));
258. Schopnosti.Add(new Schopnost(Druh.Magicky_Stit, 3, 0, 10, false));
259. Schopnosti.Add(new Schopnost(Druh.Ohniva_Koule, 0, 0, 20, true));
260. Schopnosti.Add(new Schopnost(Druh.Ledove_Kopi, 0, 0, 20, true));
261. Schopnosti.Add(new Schopnost(Druh.Vysati_zivota, 3, 0, 10, true));
262. Schopnosti.Add(new Schopnost(Druh.Vysati_many, 4, 0, 10, true));
263. Schopnosti.Add(new Schopnost(Druh.Magicke_soustredeni, 5, 0, 0, false));
```

Zdrojový kód 8 – Přidání schopností kouzelníka z třídy Postava.cs

2.4 Efekty Schopností

V předchozí části bylo rozebráno, kterými schopnostmi každá třída oplývá a jaké vlastnosti jednotlivé schopnosti můžou mít. V této části budou rozebrány účinky jednotlivých schopností, ať už na postavu hráče, nebo soupeře. Při výpočtu hodnot poškození, léčení, nebo přidání many jednotlivých schopností je využit rozptyl o velikosti úrovně postavy. Pokud je například úroveň postavy jedna, tak výsledný rozptyl bude buď -0.5, 0 anebo 0.5. Jedná se o dodatečný modifikátor, který přidává s vyšší úrovní postavy větší náhodnost. Výpočty hodnot pro jednotlivé schopnosti budou uvedeny na následujících ukázkách zdrojových kódu (viz Zdrojový kód 9, 10 a 11).

2.4.1 Válečník

Nejprve budou rozebrány schopnosti válečníka. Z vložených ukázek zdrojového kódu (viz Zdrojový kód 6 a 9) je patrné, že mezi schopnosti válečníka patří útok mečem, obrana štítem, bojový pokřik, regenerace, úder štítem, vrh sekerou a berserk.

Útok mečem je základní válečníkův útok, lze ho použít vždy, okamžitě a nestojí žádnou manu. Pokud je schopnost použita s bojovým pokřikem, je způsobeno dvojnásobné poškození.

Obrana štítem je obranná schopnost a nehledě na to zda nepřítel útočí první, nebo druhý, vždy se schopnost aktivuje, nezáleží tedy na pořadí schopností. Obranu štítem lze znovu použít až za 3 kola (včetně aktuálního). Tedy pokud je použita schopnost v prvním kole tak v kole dvě a tři ještě obranu štítem použít nelze. Teprve až v kole 4 se dá obrana štítem znovu použít. Pokud se touto schopností válečník brání, tak veškeré nemagické útoky nemají šanci obránci ubrat životy. Pokud je však proti této schopnosti použit magický útok, tak schopnost zasáhne

svůj cíl s polovičním poškozením. Dá se tedy říct, že magická poškození částečně poškodí svůj cíl, i když se dotyčná osoba brání.

Bojový pokřik je posilující schopnost, při jejím použití bude následující útok, nebo schopnost dvakrát silnější. Tato schopnost vytváří i pozitivní efekt u hráčovy postavy, kde lze vidět, jak dlouho bude pokřik aktivní. Schopnost bojový pokřik nelze použít znovu v příštím kole, nelze tedy několik kol za sebou zařvat, aby efekt pokřiku vydržel déle. Kromě dvojnásobného poškození, nebo léčení ze schopnosti regenerace, vylepšuje pokřik i další schopnosti, což bude vysvětleno u příslušných schopností.

Regenerace je jediná válečníkova schopnost jak se léčit, jedná se o schopnost, která stojí 20 bodů many. Nelze ji tedy použít, pokud postava nemá dostatek many. Pokud je tato schopnost použita s bojovým pokřikem, tak je efektivita léčení dvojnásobná. Při použití schopnosti se musí na její opětovné seslání jedno kolo čekat. U všech schopností, které doplňují

```
25. // Valecnik
26. case Druh.Utok_Mecem:
27.     return (int)Math.Round(postava.Sila + postava.Obratnost / 2.0 + rozptyl);
28. case Druh.Regenerace:
29.     return (int)Math.Round(postava.Inteligence + postava.Sila + rozptyl);
30. case Druh.Bojovy_Pokrik:
31.     return 0;
32. case Druh.Obrana_Stitem:
33.     return 0;
34. case Druh.Uder_stitem:
35.     return (int)Math.Round(postava.Sila / 2.0 + rozptyl);
36. case Druh.Vrh_sekerou:
37.     return (int)Math.Round(postava.Sila / 2.0 + postava.Obratnost / 2.0 + rozptyl);
38. case Druh.Berserk:
39.     double procenta = Math.Max(1 - postava.Zivoty / (double)postava.ZivotyMax, 0.1);
40.     return (int)Math.Round(procenta * postava.Sila + postava.Sila / 2.0 +
        postava.Obratnost / 2.0 + rozptyl);
```

Zdrojový kód 9 – Výpočet hodnot pro schopnosti válečníka ze třídy Schopnost.cs

životy, nebo manu (včetně lahviček), je třeba dávat pozor, aby hráčova postava nekrvácela vinou válečníkovy sekery, nebyla pod účinky jedu lučištníka, nebo nehořela účinkem ohnivé koule kouzelníka. Veškeré tyto účinky snižují účinnost léčení a zisku many o jednu třetinu.

Úder štítem je schopnost, která působí pouze malé poškození, ale omráčí svého soupeře v aktuálním kole. Je třeba tedy tuto schopnost použít, pokud je postava hráče na tahu jako první, jinak tato schopnost nemá efekt (vyjma přerušení vícefázových útoků jako je střelba lukem a magický šíp). Pokud je tato schopnost použita s bojovým pokřikem, tak kromě zvýšení

poškození se zvýší i doba omráčení o jedno kolo. Lze tak tedy na 2 kola omráčit svého nepřítele, což válečníkovi poskytne znatelnou výhodu, která je vykoupená faktem, že tuto schopnost nemůže příští 4 kola použít znovu. Úder štítem vytváří efekt omráčení na svém protihráči, kde lze vidět, jak dlouho bude efekt ještě trvat.

Vrh sekerou je schopnost, která soupeři neubere tolik poškození okamžitě, ale vytvoří na svém soupeři efekt krvácení, který trvá 2 kola. Efekt krvácení ubere na začátku každého kola svému soupeři drobnou část životů. Samotné krvácení je považováno za magickou schopnost brnění tedy proti krvácení nemá takový efekt. Krvácení se dá prodloužit o jedno kolo bojovým pokřikem, který zvýší i prvotní zásah sekerou. Bojový pokřik však na samotné poškození od krvácení nemá efekt. Stejně jako efekt omráčení, je i efekt krvácení viditelný u soupeřovi postavy.

Berserk je schopnost válečníka, která je specifická tím, že její poškození se odvíjí od zdraví válečníka. Čím více je válečník zraněný, tím větším útokem zaútočí. Při padesáti procentním zdraví má schopnost Berserk srovnatelné poškození s útokem mečem, nevyplatí se jí tedy používat, pokud má postava životů více. Největší zranění tato schopnost způsobí na sklonku života s jedním bodem zdraví. V kombinaci s bojovým pokřikem se tak stává doslova smrtící schopností. Při nízkém zdraví však hrozí, že soupeř porazí hráčovu postavu.

2.4.2 **Lučištník**

Nyní budou rozebrány schopnosti lučištníka. Na ukázkách zdrojových kódů (viz Zdrojový kód 7 a 10), lze vidět, že schopnosti lučištníka jsou bodnutí dýkou, úskok, střelba lukem, magický šíp, rychlost, lesní bobule a jedová šipka.

Bodnutí dýkou je základní schopnost lučištníka. Má obdobné vlastnosti jako útok mečem, ale místo síly je hlavní atributem pro výpočet poškození obratnost.

Úskok je obdoba válečníkovy obrany štítem, jediný rozdíl je v názvu schopnosti

Střelba lukem je dvoufázová schopnost, v první fázi útoku hráč pouze natáhne tětivu svého luku. Výstřel je pak proveden v následujícím kole. Důležité je dávat si pozor, aby útok postavy nebyl přerušen omráčením pomocí schopnosti úder štítem. Jedná se o nejsilnější nemagickou schopnost ve hře. Velmi lehko se dá vykrýt, pokud však je použita dvakrát za sebou, tak druhý útok bývá smrtící. Nestojí žádnou manu a ani nemá prodlevu.

Magický šíp je také dvoufázová schopnost a funguje obdobně jako střelba lukem, s tím rozdílem, že se jedná o nejsilnější magickou schopnost ve hře a stojí 20 bodů many. I když se tedy bude soupeř bránit, stále mu ubere polovinu svého poškození.

```
43. // Lucistnik
44. case Druh.Bodnuti_Dykou:
45.     return (int)Math.Round(postava.Sila / 2.0 + postava.Obratnost + rozptyl);
46. case Druh.Strelba_Lukem:
47.     return (int)Math.Round(2 * (postava.Sila + postava.Obratnost + rozptyl));
48. case Druh.Uskok:
49.     return 0;
50. case Druh.Magicky_sip:
51.     return (int)Math.Round(2 * (postava.Sila + postava.Obratnost + postava.Inteligence /
        2.0 + rozptyl));
52. case Druh.Rychlost:
53.     return 0;
54. case Druh.Lesni_bobule:
55.     return (int)Math.Round(postava.Obratnost + postava.Inteligence / 2.0 + postava.Sila /
        2.0 + rozptyl);
56. case Druh.Jedova_sipka:
57.     return (int)Math.Round(postava.Obratnost / 2.0 + postava.Inteligence / 2.0 + rozptyl);
```

Zdrojový kód 10 – Výpočet hodnot pro schopnosti lučistníka ze třídy Schopnost.cs

Rychlost je schopnost, která je silně vázána na element štěstí. Vytváří efekt, který podle délky svého trvání udává šanci na vyhnutí se útoku. Pokud rychlost bude trvat ještě 4 kola, tak existuje 60% šance na uhnutí. Při délce trvání 3, 2 a 1 je 40%, 30% a 20% šance na uhnutí. Ideální je tedy rychlost použít, když je hráč na tahu jako první, jedině tehdy uplatní 60% šanci na uhnutí. Je dobré tuto schopnost kombinovat s dvoufázovými útoky jako je střelba z luku, nebo magický šíp. Zvyšuje se tak šance, že se postavě hráče během této doby nic nestane a útoky proběhnou, jak mají. Rychlost nelze příštích 5 kol od použití použít znovu.

Lesní bobule jsou výborným zdrojem vitamínů a poskytují tak lučištníkovi schopnost vyléčit se. Léčení je o něco více silné, než základní útok a 3 kola po použití této schopnosti ji nelze použít znovu. Obdobně jako u válečníkovy regenerace je potřebné dávat pozor na efekty redukující léčení.

Jedová šipka vytváří léčení redukující efekt, který účinkuje na 3 kola a podobně jako vrh sekerou ubírá každé kolo drobnou část poškození svému cíli.

2.4.3 **Kouzelník**

Jako poslední budou rozebrány schopnosti kouzelníka. Z ukázek zdrojového kódu (Zdrojový kód 8 a 11) je patrné, že kouzelník oplývá schopnostmi úder holí, magický štít, ohnivá koule, ledové kopí, vysátí života, vysátí many a magické soustředění.

Úder holí je základní útok kouzelníka, je podobný útoku mečem, nebo bodnutí dýkou, ale jeho poškození je menší. Důvodem je, že kouzelník se soustředí především na souboj kouzel a také fakt, že ani jedno pohlaví nepřidává bonus k inteligenci, která je právě pro boj s magickou holí důležitá. Útok ale podobně jako všechny ostatní nestojí žádnou manu.

Magický štít je téměř stejný jako válečníkova obrana štítem, nebo lučištníkův úhyb. Rozdíl je zde pouze v tom, že stojí manu a to konkrétně 10 bodů many. Pokud tedy kouzelník přijde o všechnu manu, nemůže se bránit.

Ohnivá koule je důležitou schopností každého kouzelníka, dává vysoké poškození a na dvě kola vytvoří efekt hoření, který snižuje účinnost léčení a také každé kolo způsobí drobné poškození. Další výhodou je, že lze efekt hoření nasčítat a tím ho prodloužit jelikož schopnost nemá žádnou prodlevu. Tato schopnost stojí v základu 20 bodů many.

```
59. // Kouzelnik
60. case Druh.Uder_Holi:
61.     return (int)Math.Round(postava.Sila / 4.0 + postava.Obratnost / 4.0 +
        postava.Inteligence + rozptyl);
62. case Druh.Ohniva_Koule:
63.     return (int)Math.Round(postava.Inteligence + postava.Sila + rozptyl);
64. case Druh.Ledove_Kopi:
65.     return (int)Math.Round(postava.Inteligence + postava.Obratnost + rozptyl);
66. case Druh.Magicky_Stit:
67.     return 0;
68. case Druh.Vysati_zivota:
69.     return (int)Math.Round(postava.Inteligence + postava.Sila / 3.0 + rozptyl);
70. case Druh.Vysati_many:
71.     return (int)Math.Round((5.0 + postava.Inteligence + postava.Obratnost / 3.0) / 3.0 +
        rozptyl);
72. case Druh.Magicke_soustredeni:
73.     return 0;
```

Zdrojový kód 11 – Výpočet hodnot pro schopnosti kouzelníka ze třídy Schopnost.cs

Ledové kopí je další důležitou schopností, způsobí také vysoké poškození a na dvě kola vytvoří efekt mráz, který snižuje soupeřovu sílu, obratnost a inteligenci o 20%. Tato schopnost by se tedy dala považovat za kombinaci útočné a obranné schopnosti, případný útok mrznoucího soupeře pak způsobí menší poškození. Stejně jako efekt hoření, tak i efekt mráz

lze nasčítat a vytvořit tak dlouhotrvající ochranu proti poškození. V běžném světě by se efekt hoření a efekt mráz vyrušili, ale zde na soupeři zůstanou, dají se tedy tyto dva efekty kombinovat. Tato schopnost stojí také 20 bodů many.

Vysátí života je schopnost kouzelníka, která mu umožňuje přenést část životů soupeře na sebe a vyléčit se tak. Stejně jako u ostatních schopností, které léčí i zde je třeba dát si pozor na efekty, které snižují účinnost léčení. I když ukradená hodnota zdraví bude stále stejná, vstřebat se jí kouzelníkovy povede jen ze dvou třetin. Tato schopnost stojí 10 bodů many. Příští 2 kola nelze tuto schopnost znovu použít.

Vysátí many je jak už název napovídá schopností, která pomůže přenést protivníkovu manu na postavu hráče, pokud tedy ještě nějakou manu má. Tato schopnost způsobí ale i drobné poškození svému soupeři, i když soupeř již žádnou manu nemá. Pokud je kouzelník pod vlivem léčení redukujících efektů, tak i vstřebání many bude menší. Zde je ale na rozdíl od vysátí životů zredukováno i poškození many soupeře. Tuto schopnost nelze použít znovu po další 3 kola.

Magické soustředění je důležitou schopností, pokud chce kouzelník svou manu šetřit. Není totiž nic horšího, než kouzelník bez many. Když je schopnost použita, tak vytvoří na postavě efekt soustředění, který značí, že všechny magické schopnosti budou stát polovinu many. Efekt je aktivní ještě příští 3 kola a schopnost nelze znovu použít další 4 kola.

2.5 Herní systémy

V této části bude rozebrána logika herního kódu a jednotlivé části jejich systémů. Pro úplný herní kód a případnou kompilaci kódu lze navštívit stránku na GitHubu. [13]

2.5.1 Systém boje

Souboj probíhá mezi dvěma postavami, které se ve svých tazích střídají a vybírají schopnosti s různými efekty. Postavy může ovládat hráč, nebo jednoduchá umělá inteligence (AI). Postava, které klesne jako první zdraví na nulu, nebo níže prohrává a druhá postava vítězí. V závislosti na módu hry, to může znamenat pokračování ve hře, nebo konec hry (souboje).

Souboj se skládá z několika fází. Mezi hlavní z nich patří fáze, kdy hráč/počítač vybírá schopnost, používá schopnost, a kdy jsou zhodnoceny a provedeny efekty nad jednotlivými postavami (hoření, jed, krvácení apod.). Mezi další pak patří i fáze, které určují konec souboje a kdo vyhrál, začátek souboje a fáze kdy je zahájeno nové kolo.

Při výběru schopnosti (viz Zdrojový kód 12) je nejdříve ověřeno, jestli postava není omráčena. Pokud je postava omráčena, tak je celý proces výběru schopnosti přeskočen, tato skutečnost je oznámena a postava v tomto tahu žádnou schopnost nevybere a ani nepoužije.

```
298. case Faze.VyberPrvni:
299.     if (prvni.Efekty.Omraceni == 0)
300.     {
301.         messagePrvni = prvni.Name + " vybírá schopnost";
302.         if (!ZacarovanyLes.delayed)
303.         {
304.             string message = souboj.VyberPrvniSchopnosti(ref
    vybranaDruhy, ref vybranaPrvni);
305.             if (message != "")
306.             {
307.                 messagePrvni = message;
308.                 ZacarovanyLes.NastavitZpozdeniHry(3);
309.                 faze = Faze.VyberDruhy;
310.             }
311.         }
312.     }
313.     else
314.     {
315.         messagePrvni = prvni.Name + " je omráčený";
316.         ZacarovanyLes.NastavitZpozdeniHry(3);
317.         faze = Faze.VyberDruhy;
318.     }
319.     break;
```

Zdrojový kód 12 – Výběr schopnosti prvního hráče ve třídě GameState.cs

Pokud postava není omráčena, tedy efekt omráčení je roven nule, tak je oznámeno, že postava vybírá schopnost. Schopnost vybírá hráč kliknutím na tlačítko jedné ze schopností během souboje (viz Obrázek 14), pokud je na řadě počítač tak hra projde pár podmínek dle obtížnosti soupeře a vyhodnotí, jakou schopnost by mohla použít dle předem daných priorit v závislosti na situaci.

Dokud hráč nebo počítač nevyberou jednu ze schopností, tak zpráva (message) zůstává po volání metody VyberPrvniSchopnosti prázdná a nemůže tedy hra pokročit do další fáze výběru, která probíhá obdobně i u druhého hráče. Mezi fázemi je nastavené i drobné zpoždění,

Obrázek 14 – Výběr schopnosti hráčem Zdroj: Vlastní

aby uživatel stíhal sledovat dění na obrazovce.

Další fází po výběru schopností je samotné používání těchto schopností. Metoda UtokSchopnosti ve třídě Souboj.cs obsahuje necelých 300 řádků kódu. Ukázka zdrojového kódu by proto byla příliš dlouhá a tak zde bude vypsána pouze část, ve které se objevuje nejvíce schopností (viz Zdrojový kód 13). Zde uvedené schopnosti mají poměrně jednoduchou logiku. Způsobují primárně pouze poškození anebo vytvářejí na postavě soupeře různé efekty schopností. Ostatní schopnosti než zde uvedené také kromě odlišné logiky vypisují i jiné zprávy (message).

```
198. case Druh.Utok_Mecem:
199. case Druh.Berserk:
200. case Druh.Bodnuti_Dykou:
201. case Druh.Uder_Holi:
202. case Druh.Uder_stitem:
203. case Druh.Vrh_sekerou:
204. case Druh.Jedova_sipka:
205.     if (BraniSe(vybranaDruhy, rychlost))
206.     {
207.         spravceMedii.BowMiss.Play();
208.         message = naTahu.Name + " se schopností " +
   PomocneMetody.SchopnostToString(vybranaNaTahu.Druh) + " netrefil";
209.     }
210.     else
211.     {
212.         spravceMedii.Hit.Play();
213.         skPosk = ZautocNaPostavu(ref cil, efektyNaTahu.Pokrik > 0 ?
   poskozeni * 2 : poskozeni, vybranaNaTahu.Magicka);
214.         message = naTahu.Name + " útočí schopností " +
   PomocneMetody.SchopnostToString(vybranaNaTahu.Druh) + " za " + skPosk +
   " poškození";
215.         if (vybranaNaTahu.Druh == Druh.Berserk)
216.             spravceMedii.Battlecry.Play();
217.         if (vybranaNaTahu.Druh == Druh.Uder_stitem)
218.             efektyDruhy.Omraceni += efektyNaTahu.Pokrik > 0 ? 2 : 1;
219.         if (vybranaNaTahu.Druh == Druh.Vrh_sekerou)
220.             efektyDruhy.Krvaceni += efektyNaTahu.Pokrik > 0 ? 3 : 2;
221.         if (vybranaNaTahu.Druh == Druh.Jedova_sipka)
222.             efektyDruhy.Jed += 3;
223.     }
224.     break;
```

Zdrojový kód 13 – Část metody UtokSchopnosti z třídy Souboj.cs

O aplikování poškozujících efektů se stará metoda AplikujEfekty ve třídě Souboj.cs (viz Zdrojový kód 14). Cílová postava zde je terčem poškození, parametr efekt předá v místě volání metody informaci, že některý efekt byl použit a je potřeba chvíli počkat při výpisu. Naopak pokud žádný efekt použit nebyl, tak se tím předejde zbytečnému čekání. Správce médií je zde uveden jako parametr, aby bylo možné přehrát zvuk. Lze také pozorovat, že jednotlivé

efekty mají třetinový účinek hlavního atributu každé třídy. A nakonec je předána i zpráva pro výpis bojové události.

```
37.public string AplikujEfekty(Postava cil, ref bool efekt, SpravceMedii
   spravceMedii)
38.{
39.    string message = "";
40.    Postava utocnik = cil == Prvni ? Druhy : Prvni;
41.    Efekty efektyCil = cil == Obrance ? EfektyObrance : EfektyUtocnika;
42.    if (efektyCil.Horeni > 0)
43.    {
44.        int poskozeni = ZautocNaPostavu(ref cil,
   (int)Math.Round(utocnik.Inteligence / 3.0), true);
45.        message = cil.Name + " hoří za " + poskozeni + " poškození";
46.        spravceMedii.Fireball.Play();
47.        efekt = true;
48.    }
49.    if (efektyCil.Krvaceni > 0)
50.    {
51.        int poskozeni = ZautocNaPostavu(ref cil,
   (int)Math.Round(utocnik.Sila / 3.0), false);
52.        message = cil.Name + " krvácí za " + poskozeni + " poškození";
53.        efekt = true;
54.    }
55.    if (efektyCil.Jed > 0)
56.    {
57.        int poskozeni = ZautocNaPostavu(ref cil,
   (int)Math.Round(utocnik.Obratnost / 3.0), false);
58.        message = cil.Name + " je otrávený za " + poskozeni + " poškození";
59.        efekt = true;
60.    }
61.    return message;
62.}
```

Zdrojový kód 14 – Metoda AplikujEfekty z třídy Souboj.cs

2.5.2 Logika nepřátel

Logika nepřátel svou složitostí odpovídá úrovni nepřítele. Jak již bylo zmíněno dříve, tak snadní soupeři jsou označeni zelenou barvou, středně těžcí soupeři barvou žlutou a nejtěžší soupeři barvou červenou. V případě hry 1v1 je vybrán vždy těžký (červený) soupeř a představuje tak zajímavou výzvu pro napínavé souboje.

Snadný soupeř je typický svou náhodností a může použít schopnosti, které v dané chvíli nedávají vůbec smysl. Jeho úroveň (level) je vždy jedna, nebo dva. Na ukázce kódu (viz Zdrojový kód 15) je patrné, že je schopnost vybírána tak dlouho, dokud se nenajde první schopnost, kterou by mohl použít i hráč dle nastavených pravidel. Musí tedy postava počítače

```
13.case Majitel.Pocitac_Lehky:
14.start00:
15.     sch = pocitac.Schopnosti[souboj.Kostka.Next(0,
   pocitac.Schopnosti.Count)];
16.     if (sch.Cd > 0 || (efektyPocitac.Soustredeni == 0 && sch.CenaMany >
   pocitac.Mana) || (efektyPocitac.Soustredeni > 0 && (sch.CenaMany / 2) >
   pocitac.Mana))
17.     {
18.         goto start00;
19.     }
20.     if ((sch.Druh == Druh.Lahvicka_Many && inventarPocitac.LahvickyMany
   == 0) || (sch.Druh == Druh.Lahvicka_Zdravi &&
   inventarPocitac.LahvickyZdravi == 0))
21.     {
22.         goto start00;
23.     }
24.     return sch;
```

Zdrojový kód 15 – Část Metody VyberSchopnostAI z třídy AI.cs

splňovat podmínku úrovně many, schopnost nesmí být v prodlevě, a pokud se jedná o použití lahviček zdraví a many tak musí mít alespoň jednu v inventáři.

Středně těžký soupeř stále zachovává prvek náhodnosti, ale nikdy se nebude bránit zbytečně. Obranu si tedy schovává na silnější schopnosti, které by ho mohli ohrozit. Úroveň tohoto soupeře je vždy buď tři anebo čtyři.

Těžký soupeř má minimálně úroveň 5 (mimo typ hry duel), nebo stejnou jako hráč a neexistuje u něj prvek náhodnosti. Vždy vybírá pomocí předem nastavených podmínek vhodnou schopnost. Tento soupeř je ale pořád jen tak dobrý, jako ten kdo ho vytvořil, není tedy neomylný a není zdaleka tak propracovaný jako dobrý lidský soupeř.

2.5.3 Systém ukládání a načítání

Hra obsahuje systém ukládání a načítání hry, který může pomoci při nutnosti ukončení hry před jejím dohráním. Hra se jednoduše pomocí klávesy F5 uloží během rozehrané hry do binárního souboru s příponou „.save“ a načte pomocí klávesy F9 v podstatě v jakékoliv části hry. Soubor uložení obsahuje důležité informace o postavě hráče, včetně informací o všech mapách a pozice hráče v nich.

```
61.public static void Uloz()
62.{
63.    try
64.    {
65.        string adresar = Directory.GetCurrentDirectory() + "\\saves";
66.        Directory.CreateDirectory(adresar);
67.        System.Diagnostics.Debug.WriteLine(adresar);
68.        using SaveFileDialog saveFileDialog1 = new SaveFileDialog
69.        {
70.            InitialDirectory = adresar,
71.            Filter = "save files (*.save)|*.save|All files (*.*)|*.*",
72.            FilterIndex = 1,
73.            RestoreDirectory = true
74.        };
75.
76.        if (saveFileDialog1.ShowDialog() == DialogResult.OK)
77.        {
78.            Serializuj(Path.GetFullPath(saveFileDialog1.FileName));
79.        }
80.    }
81.    catch (Exception ex)
82.    {
83.        System.Diagnostics.Debug.WriteLine("Uloz:" + ex.Message);
84.    }
85.}
```

Zdrojový kód 16 – Metoda Uloz z třídy UkladaniNacitani.cs

Uložení do souboru (viz Zdrojový kód 16) obstarává metoda Uloz ve třídě UkladaniNacitani.cs. Nejprve vytvoří složku „saves“ v adresáři se hrou a v ní uživatel může uložit svými slovy pojmenovaný soubor dle funkčnosti a rozhraní třídy SaveFileDialog. Po potvrzení uložení se serializují potřebná data (viz Zdrojový kód 17) a soubor se uloží.

```
10.private static void Serializuj(string file)
11.{
12.    using BinaryWriter binWriter = new BinaryWriter(File.Open(file,
   FileMode.Create));
13.    ZacarovanyLes.utocnik.Write(binWriter);
14.    ZacarovanyLes.maps.Write(binWriter);
15.    binWriter.Close();
16.}
```

Zdrojový kód 17 – Metoda Serializuj z třídy UkladaniNacitani.cs

Načítání hry ze souboru obstarává metoda Nacti ve třídě UkladaniNacitani.cs (viz Zdrojový kód 18). Hra opět případně vytvoří složku „saves“ pokud neexistuje. Pomocí

třídy OpenFileDialog hra zobrazí uživatelské rozhraní s výchozím umístěním ve složce „saves“ pro načtení binárních souborů uložených her s příponou „.save“. Při potvrzení vybraného souboru nyní projde hra procesem deserializace (viz Zdrojový kód 19) a načte postavu společně s informacemi o mapách, pro které jsou současně i aktualizovány textury částí mapy.

```
34.public static void Nacti()
35.{
36.     try
37.     {
38.         string adresar = Directory.GetCurrentDirectory() + "\\saves";
39.         Directory.CreateDirectory(adresar);
40.         System.Diagnostics.Debug.WriteLine(adresar);
41.         using OpenFileDialog openFileDialog = new OpenFileDialog()
42.         {
43.             InitialDirectory = adresar,
44.             Filter = "save files (*.save)|*.save|All files (*.*)|*.*",
45.             FilterIndex = 1,
46.             RestoreDirectory = true
47.         };
48.
49.         if (openFileDialog.ShowDialog() == DialogResult.OK)
50.         {
51.             Deserializuj(Path.GetFullPath(openFileDialog.FileName));
52.         }
53.     }
54.     catch (Exception ex)
55.     {
56.         System.Diagnostics.Debug.WriteLine("Nacti:" + ex.Message);
57.     }
58.}
```

Zdrojový kód 18 – Metoda Nacti z třídy UkladaniNacitani.cs

Hra se následně přepne do pohledu pohybu po mapě (ZacarovanyLes.mapstate) a pokračuje tam, kde uživatel svou hru uložil.

```
17.private static void Deserializuj(string file)
18.{
19.     using BinaryReader binReader = new BinaryReader(File.Open(file,
   FileMode.Open));
20.     ZacarovanyLes.game.ChangeCurrentState(ZacarovanyLes.menuState);
21.     ZacarovanyLes.gameState = null;
22.     ZacarovanyLes.utocnik = Postava.Read(binReader);
23.     ZacarovanyLes.maps = MapManager.Read(binReader);
24.     ZacarovanyLes.mapState = new MapState(ZacarovanyLes.game,
   ZacarovanyLes.content);
25.     ZacarovanyLes.mapState.LoadContent();
26.     ((MapState)ZacarovanyLes.mapState).UpdateTextures();
27.     ZacarovanyLes.game.ChangeCurrentState(ZacarovanyLes.mapState);
28.     binReader.Close();
29.}
```

Zdrojový kód 19 – Metoda Deserializuj z třídy UkladaniNacitani.cs

2.5.4 Mapový systém a tvorba vlastních map

```
6. public class Map
7. {
8.     public Objekt[,] Objekty { get; set; }
9.     public Vector2 PoziceHrace { get; set; }
10.
11.    public Map(Objekt[,] objekty, Vector2 poziceHrace)
12.    {
13.        Objekty = objekty;
14.        PoziceHrace = poziceHrace;
15.    }
16.}
```

Zdrojový kód 20 – Třída Map.cs

Každá načtená mapa ve hře (viz Zdrojový kód 21) obsahuje informace o objektech uložené v dvojrozměrném poli, které se na mapě vyskytují. Může se jednat o trávu, kámen, strom, hráče, dveře do další a předchozí mapy, soupeře tří obtížností, lahvičku zdraví i many a dveře ukončující hru. Dále také poslední pozici hráče na mapě, tato informace je užitečná zejména při přechodu na jinou mapu a následném vrácení zpět.

```
10.public List<Map> Maps { get; set; }
11.public Map Aktualni { get; set; }
12.
13.public MapManager(List<Map> maps, Map aktualni)
14.{
15.    Maps = maps;
16.    Aktualni = aktualni;
17.}
18.
19.public MapManager(List<Map> maps)
20.{
21.    Maps = maps;
22.    if (maps.Count > 0)
23.        Aktualni = maps[0];
24.}
25.
26.public void DalsiMapa()
27.{
28.    int index = Maps.IndexOf(Aktualni) + 1;
29.    if (index < Maps.Count && index >= 0)
30.    {
31.        Aktualni = Maps[index];
32.    }
33.}
34.
35.public void PredchoziMapa()
36.{
37.    int index = Maps.IndexOf(Aktualni) - 1;
38.    if (index < Maps.Count && index >= 0)
39.    {
40.        Aktualni = Maps[index];
41.    }
42.}
```

Zdrojový kód 21 – Část třídy MapManager.cs

Třída MapManager.cs má tyto mapy uložené v dynamické datové struktuře List a současně i mapu na které se aktuálně hráč nachází. Pro přechod do další a předchozí mapy jsou využity metody DalsiMapa a PredchoziMapa. Při zjištění indexu aktuální mapy v Listu, lze pak zjistit, zda nějaká předchozí anebo další mapa existuje a podle toho se buď přesunout a nastavit aktuální mapu na odpovídající mapu z Listu, nebo se nepřesunout. Pořadí vložených map tedy určuje i pořadí jak jdou mapy postupně za sebou. Tento systém je velmi jednoduchý, a pokud by bylo žádoucí větvení cest, musel by se upravit.

Mapy jsou načítány z textových dokumentů v adresáři hry. Lze je tedy upravit a vytvořit tak své vlastní dobrodružství. Více informací o tom jak mapy správně upravit lze najít v dokumentaci na GitHubu. [13]

ZÁVĚR

Framework MonoGame je dobrým nástrojem, který má určitě své místo v herním vývoji. Nicméně není zdaleka tak využívaným. Pořád je ale možné najít kvalitní dokumentaci, která zájemcům o tento framework může dobře posloužit. Náročnost tohoto frameworku spočívá především v ovládnutí jazyka C# a samotný framework je velmi jednoduchý na používání. Již vytvořené enginy práci při vývoji počítačových her usnadňují, je však nutné naučit se samotný engine, který se od ostatních může velmi lišit.

Pokud by se jednalo o hru pro komerční účely tak by vývoj hry Začarovaný les rozhodně nebyl u konce. Pořád je možné na projektu udělat spoustu práce, i co se týče optimalizace a zpřehlednění kódu, nebo samotných funkcí hry. V této fázi vývoje by hra určitě nemohla konkurovat čím dál více rozšířenějšímu trhu s počítačovými hrami. Postrádá líbivou grafiku, animace soubojů, další položky inventáře, NPC, dialogy s NPC atd. Proto jsou počítačové hry většinou vytvářeny celým týmem programátorů, grafiků, animátorů, scénáristů a pro jednotlivce je tvorba počítačové hry často výzvou.

Cíle stanovené autorem práce se podařilo splnit, ať už jde o teoretickou průpravu a představení frameworku MonoGame, nebo praktickou část ve které byla představena počítačová hra Začarovaný les. Schopnosti postav byli rozšířeny o 3 další k původnímu počtu 4 schopností, celkem tedy má nyní každá třída k dispozici 7 unikátních schopností. Plynulá animace pohybu po mapě byla rovněž přidána společně se zpřehledněním systému boje.

POUŽITÁ LITERATURA

[1] MONOGAME FOUNDATION, INC. *MonoGame documentation* [online]. 2024 [cit. 2024-04-24]. Dostupné z: https://docs.monogame.net/articles/index.html

[2] MONOGAME FOUNDATION, INC. *Setting up your development environment for Windows.* MonoGame Documentation [online]. 2024 [cit. 2024-04-22]. Dostupné z: https://docs.monogame.net/articles/getting_started/1_setting_up_your_development_environment_windows.html

[3] MICROSOFT. *Visual Studio 2022 IDE* [online]. 2024 [cit. 2024-05-02]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

[4] MONOGAME FOUNDATION, INC. *Creating a new project.* MonoGame Documentation [online]. 2024 [cit. 2024-04-25]. Dostupné z: https://docs.monogame.net/articles/getting_started/2_creating_a_new_project_vs.html

[5] MONOGAME FOUNDATION, INC. *Adding Content.* MonoGame Documentation [online]. 2024 [cit. 2024-04-26]. Dostupné z: https://docs.monogame.net/articles/getting_started/4_adding_content.html

[6] MONOGAME FOUNDATION, INC. *Understanding the code.* MonoGame Documentation [online]. 2024 [cit. 2024-05-05]. Dostupné z: https://docs.monogame.net/articles/getting_started/3_understanding_the_code.html

[7] CAPELLMAN, Jarred a Louis SALIN. *MonoGame Mastery: Build a Multi-Platform 2D Game and Reusable Game Engine* [online]. Apress Berkeley, CA, 2020, 323 s. [cit. 2023-11-04]. ISBN 978-1-4842-6308-2. Dostupné z: https://doi.org/10.1007/978-1-4842-6309-9

[8] SALIN, Louis a Rami MORRAR. *Game Development with MonoGame: Build a 2D Game Using Your Own Reusable and Performant Game Engine* [online]. Apress Berkeley, CA, 2021 [cit. 2023-10-19]. ISBN 978-1-4842-7771-3. Dostupné z: https://doi.org/10.1007/978-1-4842-7771-3

[9] PAVLEAS, Jebediah, Jack KENG-WEI CHANG, Kelvin SUNG a Robert ZHU. *Learn 2D Game Development with C#* [online]. Apress Berkeley, CA, 2014 [cit. 2023-10-19]. ISBN 978-1-4302-6605-1. Dostupné z: https://doi.org/10.1007/978-1-4842-7771-3

[10] Stardew Valley – Press. BARONE, Eric. *Stardew Valley* [online]. 2023 [cit. 2023-10-19]. Dostupné z: https://www.stardewvalley.net/press/

[11] PRESCOTT, Shaun. *Celeste review*. PC Gamer [online]. 2018, 1–5 [cit. 2023-11-03]. Dostupné z: https://www.pcgamer.com/celeste-review/

[12] THORSON, Madeline Stephanie. MATT MAKES GAMES INC. *Celeste press kit* [online]. 2018 [cit. 2023-11-03]. Dostupné z: https://www.dropbox.com/sh/87lycjzncb0o9ns/AACcA3s5Uyzkeicn3-26qLBWa?dl=0&preview=Celeste+Press+Kit.pdf

[13] ZLATOHLÁVEK, Tomáš. *Zdrojový kód hry Začarovaný les. GitHub* [online]. 2024 [cit. 2024-05-02]. Dostupné z: https://github.com/tomaszlat/ZacarovanyLes

[14] ZLATOHLÁVEK, Tomáš. *Původní návrh v programovacím prostředí Raptor. GitHub* [online]. 2024 [cit. 2024-05-03]. Dostupné z: https://github.com/tomaszlat/ZacarovanyLesRaptor

[15] MAHAJAN, AKASH. *Raptor homepage* [online]. 2023 [cit. 2024-05-03]. Dostupné z: https://raptor.martincarlisle.com/