UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2017 Martin Fryml

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Herní launcher a hra pro platformu Windows

Martin Fryml

Bakalářská práce

2017

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2016/2017**

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Martin Fryml**
Osobní číslo:	**I14090**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Herní launcher a hra pro platformu Windows**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování:

Cílem práce bude popsat vývoj launcheru jako softwarového prostředku pro verzování aplikace a vyvinout a popsat vývoj 2D hry Bloody Rain pro platformu Windows.

V teoretické části bude představen programovací jazyk Ruby s knihovnou RGSS, techniky navázání spojení s web serverem a metody stažení aktualizace aplikace pomocí launcheru.

Praktickým výstupem práce bude dvoudimenzionální hra v kresleném fantasy stylu. Žánr hry bude Role playing game. Soubojový systém je založený na "turn-based strategy", kde se celý souboj odehrává po kolech. Pořadí útoků nepřátel a hráče je algoritmicky definován dle zohledňovaných charakteristik.
Součástí řešení bude i softwarových launcher, který naváže spojení s web serverem, ověří verzi hry a v případě dostupnosti nové verze nabídne aktualizaci hry. Launcher detekuje změněné soubory a stáhne je. Hráč nemusí stahovat znovu celou hru při každé aktualizaci. Launcher poskytne základní nastavení hry ještě před spuštěním.

V textu práce budou popsány klíčové třídy, metody a postup vývoje samotné hry.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování bakalářské práce: **tištěná**

Seznam odborné literatury:

***DILLE, Flint. The ultimate guide to video game writing and design. New York: Watson-Guptill Publications, 2007. ISBN 158065066X.**
***PECINOVSKÝ, Rudolf. OOP: Naučte se myslet a programovat objektově. Brno: Computer Press, a.s., 2010. ISBN 978-80-251-2126-9.**
***KNUTH, D. E.: Umění programování - Základní algoritmy, Brno, Computer Press 2008, ISBN: 978-80-251-2025-5.**
***WRÓBLEWSKI, Piotr. Algoritmy: datové struktury a programovací techniky Vyd. 1. Překlad Marek Michalek, Bogdan Kiszka. Brno: Computer Press, 2004 351 s. ISBN 80-251-0343-9.**
***KEOGH, Jim; DAVIDSON, Ken. Datové struktury bez předchozích znalostí : průvodce pro samouky. Vyd 1. Brno : Computer Press, 2006. 223 s. ISBN 80-251-0689-6.**

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2016**
Termín odevzdání bakalářské práce: **12. května 2017**

Univerzita Pardubice Česká republika

L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 31. března 2017

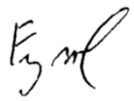
Prohlášení autora

Prohlašuji, že jsem tuto práci 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.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

V Pardubicích dne 8. 5. 2017



Martin Fryml

PODĚKOVÁNÍ

Chtěl bych poděkovat celé své rodině za neustálou podporu během studia a tvorby bakalářské práce. Dále svým přátelům, kteří při mě drželi, velké díky patří jmenovitě Ondřeji Koskovi. V neposlední řadě bych chtěl poděkovat i svému vedoucímu Josefu Brožkovi za konzultace a vedení práce.

ANOTACE

Cílem práce bylo navrhnout a realizovat vývoj launcheru jako softwarového prostředku pro verzování aplikace a popsat vývoj 2D hry Bloody Rain pro platformu Windows.

V teoretické části je představen programovací jazyk Ruby s knihovnou RGSS, techniky navázání a spojení s web serverem a metody stažení aktualizace aplikace pomocí launcheru.

Praktickým výstupem práce je dvoudimenzionální hra v kresleném fantasy stylu. Společně s ní byl vytvořen i softwarový launcher, který ověří verzi hry a v případně dostupnosti aktualizace nabídne její stažení.

KLÍČOVÁ SLOVA

Launcher, hra, Windows, vývoj, Ruby, C#, XML, RPG Maker

TITLE

Game launcher and Windows game development.

ANNOTATION

This thesis covers the development of a game launcher as a software tool for application versioning. It also follows the development of 2D game Bloody Rain for Windows platform.

The theoretical part concentrates on programming language Ruby with scripting library RGSS, establishing connections with a web server and methods of downloading updates through the launcher.

The practical part deals with a 2D game in old-school fantasy style. The game launcher is included altogether with the game. This software launcher is able to check and offer an update for the game.

KEYWORDS

Launcher, game, Windows, development, Ruby, C#, XML, RPG Maker

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

API	Application Programming Interface
FTP	File Transfer Protocol
MIT	Massachusetts Institute of Technology
RGSS	Ruby Game Scripting System
RPG	Role-playing game
RTP	Run Time Package
SDK	Software development kit
UML	Unified Modeling Language
WPF	Windows Presentation Foundation

TYPOGRAFICKÉ KONVENCE

V rámci této práce jsou pro zvýšení srozumitelnosti dodržovány zde uvedené konvence. Jedná se převážně o použití:

Kurzívou se vyznačují názvy konkrétních metod, názvy vlastních tříd, názvy souborů a cesty k umístění souboru.

`Bezpatkové písmo Verdana` o velikosti 11 se světle šedým stínováním je použito pro zobrazení zdrojových kódů.

ÚVOD

Cílem práce je vytvořit fantasy 2D hru v kresleném old-school stylu. Hra by měla obsahovat tahové souboje, úkoly a velkou variaci map. Pro hru bude také vytvořen herní launcher.

Od herního launcheru se očekává, že dokáže navázat spojení se severem, zkontrolovat verzi hry, která se nachází na serveru, porovnat ji s verzí lokální hry a nabídnout stažení aktualizace. Aktualizace lokální hry by měla probíhat po jednotlivých souborech, ne jako celek. Aby v případě poškození souboru nebo dostupné aktualizace hráč stahoval pouze dané soubory a ne celou hru. Očekává se přívětivé uživatelské rozhraní a kontrola lokálních souborů.

Hra by měla být plně kompatibilní s operačním systémem Windows a měla by mít ošetřené veškeré logické, designové i programovací chyby. Vzhledem k omezenému času se očekává minimálně půl hodiny hratelného času, 5 úkolů, několik map, plná funkčnost soubojového systému a úvodních map.

Bakalářská práce bude rozdělena do dvou části, teoretické (kapitoly 1 a 2) a praktické (kapitoly 3 a 4). V teoretické části budou popsány použité technologie a programovací jazyky. Praktická část bude obsahovat implementaci důležitých částí herního launcheru i 2D hry, včetně částí zdrojových kódu.

Velkou motivací pro mě byl předmět Vývoj počítačových her, který mě inspiroval k vytvoření vlastní hry. K vývoji herního launcheru mě vede zájem o lepší poznání práce s vlákny, jazyk C# a aktualizace souborů ve stylu herní platformy Steam. Realizaci samotné bakalářské práce předcházelo studium přenosu souborů po síti, jazyka C# a funkčnosti vláken v operačních systémech.

Osobním cílem je plné dokončení herního launcheru a následné pokračování vývoje 2D hry. Hra by měla být v plně hratelném stavu, aby se dala prezentovat veřejnosti (například ve službách Greenlight či itch.io).

1 PROGRAMOVACÍ JAZYKY

Programovací jazyk je nástroj pro zápis instrukcí, které má zařízení (nejčastěji počítač) provést. Pro člověka by bylo velmi obtížné psát přímo strojový kód a instrukce pro počítač. Proto byly vyvinuty programovací jazyky, které interpretují řešení mezi programátorem a technickými prostředky. Vyšší programovací jazyky jako například C nebo Java se podobají anglickému jazyku více než první generace programovacích jazyků. K převodu na strojový kód se využívá tzv. kompilátorů.

V dnešním světě elektroniky a techniky máme k dispozici desítky až stovky typů programovacích jazyků. Je jen na programátorovi jaký programovací jazyk si ke své práci vybere. Ne každý jazyk se hodí na všechno. Jsou jazyky určené pro vývoj webových aplikací, pro práci s hardwarem na úrovni jádra operačního systému nebo desktopové aplikace s příjemným uživatelským rozhraním.

Následující podkapitoly obsahují výčet všech použitých programovacích jazyků, včetně jejich popisu a důvodu použití. Mimo to je uveden i jeden jazyk značkovací.

Kapitola vychází ze zdrojů [1 – 11].

1.1 C#

C# (vyslovováno anglicky jako /si: ša:p/) je vysokoúrovňový objektově orientovaný programovací jazyk vyvinutý firmou Microsoft v roce 2000. Jako základ pro nový jazyk C# si vybrali jazyky C++ a Java. Tento jazyk se převážně používá pro tvorbu formulářových aplikací ve Windows, databázových programů a webových aplikací. Úmyslem C# bylo vyvážit sílu jazyka C++ a tu spojit se snadným použitím a možností rychlého programování, které poskytovali například jazyky Visual Basic a Delphi.

1.1.1 Cíle jazyka

C# byl vytvořený tak, aby byl jednoduchý, moderní, mnohoúčelový a objektově orientovaný jazyk. Jazyk a jeho implementace poskytuje podporu hlídání hranic polí, automatický garbage collector, silnou typovou kontrolu a detekci neinicializovaných proměnných.

C# má také automatickou správu paměti, kterou spravuje garbage collector. Uživatel se tedy nemusí zabývat destruktory z C++ nebo s manuálním uvolňováním paměti. Ačkoliv aplikace psané v C# mají být ekonomické a nemají plýtvat přidělený procesorový čas, nemohou se výkonově měřit aplikacím psanými v jazyce C nebo Assembly.

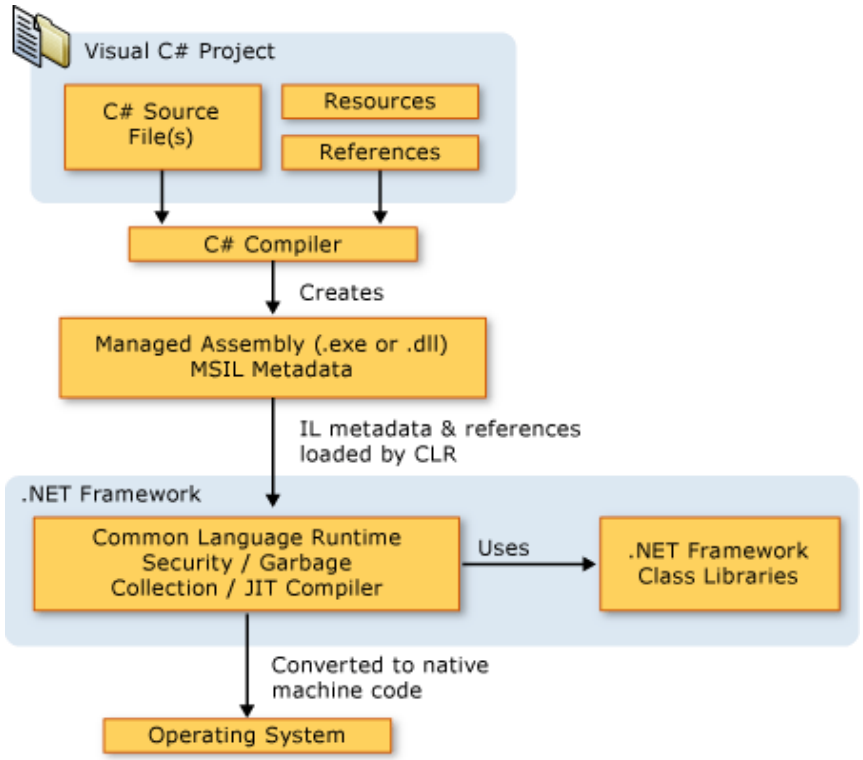
1.1.2 Vlastnosti jazyka

Ukazatele v C# neexistují a nebezpečné operace jako např. přímé manipulace s pamětí, nejsou dovoleny. Kompilátor zabrání jejich provedení a díky tomu je typově bezpečnější. Nelze provést implicitní konverze z celočíselných typů na boolean.

Na rozdíl od VB .NET je C# case sensitive, což znamená, že rozlišuje velká a malá písmena. Proměnná „value“ není to samé jako „Value“. Také není nutné deklarovat metody dopředu před jejich použití, jako tomu bylo například u C++. Pořadí deklarace a použití metod je irelevantní.

Od Javy byla přebrána dědičnost a statické metody. C# stejně jako Java nepodporuje vícenásobnou dědičnost a neexistují v něm globální proměnné. Globální proměnné nahrazují statické metody.

Mimo mnoha dalších výhod poskytuje C# oddělení kódu pomocí jmenných prostorů. Tento koncept je založený ja balíčkovacím systému Javy a funguje téměř totožně pouze s jinou syntaxí. Na obr. 1 jsou ilustrovány vztahy mezi soubory, kompilátorem a převedení zdrojového kódu do strojového jazyka.



Obrázek 1 – Ilustrace vztahů od zdrojového kódu až po vykonání programu[1]

[1] Zdroj: [2]

1.1.3 WinForms

Windows Forms, zkráceně WinForms, je knihovna tříd grafického rozhraní. Je původní část .NET Frameworku. Poskytuje platformu pro tvorbu desktopových aplikací. Jako výchozí nastavení vzhledu oken se používá aktuální vzhled operačního systému. Jak název napovídá, WinForms jsou dostupné pouze na operačních systémech Windows. Celá knihovna je postavená nad Win32 API.

Rozhraní WinForms je možné rozšířit pomocí uživatelských balíčků a frameworků. Mezi nejpopulárnější rozšíření patří Pdfium.Net SDK, které umožňuje čtení, úpravu a extrakci textů ze souborů formátu PDF, balíček obsahující komponentu pro zobrazení křivek a informací týkající se audio souborů a grafický framework MetroFramework (Modern UI), upravující základní vzhled Windows Forms komponentů.

Nejjednodušší způsob pro získání a nainstalování balíčků pro WinForms je použití balíčkovacího manažera NuGet od společnosti Microsoft.

1.1.4 Vlákno a multithreading

Vlákno (anglicky thread) je „odlehčený proces“, pomocí něhož se snižuje režie operačního systému při změně kontextu (content switch). Vlákno je vykonávací část procesu. Každý běžící proces tedy musí mít alespoň jedno vlákno. Vlákna mezi sebou sdílí paměťový prostor a další datové struktury.

Každé vlákno musí mít vlastní:

- Stav, ve kterém se vlákno nachází.
- Stack, ve kterém ukládá parametry funkcí, návratovou adresu a lokální proměnné.
- Registry, kam se ukládají mezivýsledky vlákna.

Je možné spustit více vláken současně. Taková vlákna mohou běžet pseudoparalelně na jednojádrových procesorech a paralelně (současně) na vícejádrových procesorech. Tento běh více vláken se nazývá multithreading.

Typickým příkladem v C# je nevhodně navržená Windows Form aplikace, která běží v jednom vlákně. Pokud se začne vykonávat časově náročnější metoda uvnitř aplikace, GUI není schopno reagovat na vstupy uživatele, protože je blokováno prací uvnitř metody. Výsledkem je okno aplikace, které uživatelsky nereaguje. Uživatel v době vykonávání metody není schopen akci přerušit nebo jakkoliv manipulovat s oknem.

Programování pomocí vláken s sebou přináší kromě výhod také problémy. K některým proměnným mohou mít potřebu přistupovat dvě vlákna současně. Pokud by tento stav nebyl ošetřen, může první vlákno změnit data uvnitř proměnné, se kterou právě pracuje. Čtení druhého vlákna může skončit chybou, protože část původních dat je již přepsána jinými. Touto problematikou se zabývají témata řízení přístupu do kritické sekce.

Multithreading je ve správných rukách velmi mocný nástroj a pro některé funkce aplikace je naprosto nezbytný.

1.2 Ruby

Ruby je interpretovaný, objektově orientovaný, skriptovací programovací jazyk. Obsahuje jednoduchou syntaxi, která se snadno učí a zároveň je dostatečný výkonný, že dokáže konkurovat jazykům, jako jsou Perl nebo Python. Vytvořil ho jediný autor, Jukihiro Macumoto.

1.2.1 Cíle a vlastnosti jazyka

Cílem tvůrce Matsumota bylo vytvořit programovací jazyk, který spojí funkce z jeho oblíbených programovacích jazyků (Perl, Smalltalk, Eiffel, Ada a Lisp) do jednoho nového jazyka. Chtěl, aby Ruby byl přirozený, ne jednoduchý.

Pokusil se tak vytvořit ideální syntaxi, která bude mocnější než Perl a zároveň více objektově orientovaná než Python. V Ruby je tedy všechno objekt. Každému bitu informace a kódu se dá přiřadit instanční proměnné a metody.

Ruby je vnímán jako flexibilní jazyk, neboť umožňuje uživatelům volně měnit veškeré jeho části. Podstatné části mohou být odstraněny nebo redefinovány dle potřeb uživatele.

Stejně jako většina ostatních jazyků, i Ruby podporuje pouze jednu dědičnost. Ruby obsahuje koncept modulů, které reprezentují kolekce metod a proměnných. Třídy tak můžou získat všechny metody obsažené v modulu a není zapotřebí dědičnost.

Velkou výhodu tvoří zabudovaný garbage collector, který stejně jako v C# a Javě odstraňuje veškeré objekty, na které již neexistují žádné reference. Není třeba se starat o uvolňování paměti.

Ruby lze zdarma využívat, kopírovat, modifikovat a distribuovat. Navíc je vysoce přenosný. Může být spuštěn na jakékoliv platformě. Od Linuxu, přes mnoho typů Unixu až po Windows a Mac OS X.

1.2.2 Skript

Skript je program či kus kódu zapsaný ve skriptovacím jazyce. Jedná se o souvislou sérii příkazů vykonávající určitý úkol. Obvykle bývá uložen jako soubor.

Jeho největší výhodou je, že se nemusí kompilovat (resp. o kompilaci se stará interpreter). Uživatel nemusí mít nainstalovaný kompilátor a při jakékoliv změně kódu jej nemusí znovu kompilovat. Proto se skripty nejčastěji používají v dynamických stránkách (v podobě PHP nebo JavaScriptu) a v hrách. Jsou snadné na údržbu i správu kódu.

Nevýhodou je nižší rychlost. Překlad instrukcí nebude nikdy tak rychlý jako ze zkompilovaného programu. Navíc uživatel musí mít nainstalovaný překladač (interpreter), který bude schopný skript přeložit a spustit. Překladač musí být spuštěný po celou dobu běhu programu, což znamená i větší požadavky na paměť.

1.3 XML

XML je zkratkou pro rozšiřitelný značkovací jazyk Extensible Markup Language, který byl vyvinut konsorciem W3C. XML slouží u popisu, přenosu a uložení dat. V pravém slova smyslu nejde o programovací jazyk. Sám o sobě nedokáže vůbec nic.

Formát tohoto jazyka je velmi podobný HTML. Oba jazyky obsahují značkovací symboly, které popisují svůj obsah. Data v XML jsou známá jako samopopisná či samovysvětlující, což znamená, že struktura dat je uložena společně s daty. Když dorazí data na místo určení, není třeba předem vytvořit strukturu pro ukládání dat. Je již dynamicky uložena v XML a stačí ji jen načíst a přečíst její data.

Základní stavěcí blok XML dokumentu je element definovaný značkou. Každý element musí začínat a končit značkou. XML neobsahuje žádné předdefinované značky, je třeba si vlastní značky vytvořit a definovat. Každá značka musí mít tzv. pár a nesmí se křížit. Elementy lze vnořit do dalších elementů a tvořit tak hierarchické struktury. Základním a jediným povinným elementem je kořen (anglicky root).

Kromě elementů obsahuje XML ještě atributy. Atribut popisuje charakteristiku elementu a je umístěn v počátečním elementu. Element může obsahovat libovolné množství atributů oddělené mezerami. Atribut se zapisuje za element rovnítkem a jeho data jsou ve formě stringu (použitím uvozovek). Atributy musí mít unikátní názvy. Nelze mít v jednom elementu dvakrát atribut ID. Tyto hodnoty by se přepsaly a došlo by tak k chybě.

Příklad obsahu XML souboru

```
<?xml version="1.0" encoding="UTF-8"?>
<Koren>
	<Element>
		<PodElement>Data</PodElement>
		<PodElement atribut="Data atributu">Data elementu</PodElement>
	</Element>
	<ElementBezDat atribut="Další data atributu"/>
</Koren>
```

Síla XML spočívá v jeho jednoduchosti a přenositelnosti. Může obsahovat velké množství informací a poskytnout smysluplnou, čitelnou a jednoduchou strukturu a organizaci dat. Data jsou pro člověka snadno čitelná a lze je snadno manuálně upravovat. XML je multiplatformní a multitechnologické.

2 POUŽITÉ TECHNOLOGIE

Tato kapitola popisuje všechny technologie, které byly použity pro tvorbu herního launcheru a 2D hry. Každá použitá technologie je v kapitole popsána a odůvodněna, z jakého důvodu byla použita. Mezi technologiemi nejsem znovu uvedeny programovací jazyky, které byly představeny v předchozí kapitole.

Kapitola vychází ze zdrojů [12 – 14].

2.1 FTP

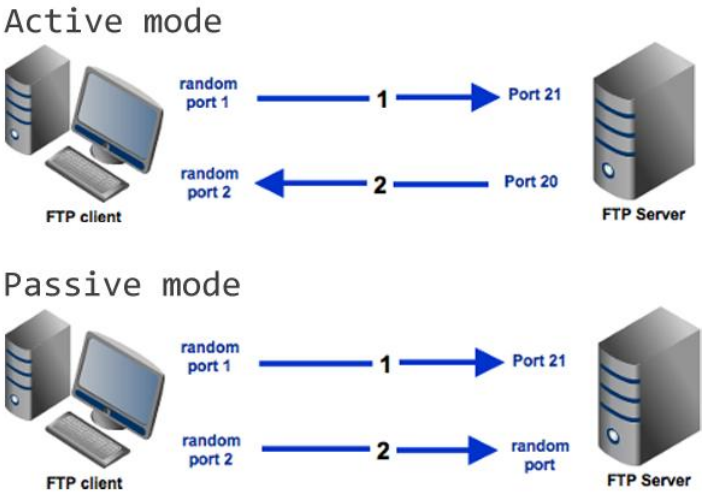
V jazyku Internetu je FTP zkratkou pro File Tranfer Protocol. Jedná se o síťový protokol odpovědný za přenos souborů z jednoho počítače na druhý pomocí protokolu TCP z rodiny TCP/IP.

Nejčastěji se FTP vyskytuje ve formě serveru a klienta, kdy server je například web server a klient je uživatel, který chce nahrát na web nová data. Uživatel se k FTP serveru přihlašuje pomocí unikátního přihlašovacího jména a hesla. Tento uživatelský účet musí být předem na serveru vytvořen.

FTP je platformě nezávislý, což znamená, že lze použít na jakémkoliv operačním systému. Využívá dvou TCP portů, kterými jsou porty 20 a 21. FTP server naslouchá na portu 21 a zjišťuje, zda se nechce nějaký FTP klient připojit. Tento port přenáší příkazy klienta. Samotná data se přenáší na portu 20.

Velkou nevýhodou FTP jsou bezpečností rizika a problémy. Veškerá přenášená data, ale i přihlašovací jméno a heslo je zasíláno jako běžný text (plain text). Takto nechráněné a nešifrované informace mohou být zachyceny nebo upraveny. Pokud je to možné, je vždy lepší použít jedno z rozšíření pro FTP. Taková rozšíření mají označení FTPS (případně FTP/SSL) a zajišťují bezpečný přenos dat a informací pomocí protokolu FTP.

FTPS se často zaměňuje se SFTP nebo Secure FTP. SFTP je označení pro SSH file transfer protocol, což je protokol a program pro zabezpečený a šifrovaný přenos souborů. Secure FTP je zabezpečené spojení FTP pomocí SSH tunelu.



Obrázek 2 – Tvorba kanálů v aktivním a pasivním režimu[2]

Připojení se realizuje v jednom ze dvou různých režimů. Prvním režimem je aktivní režim. V tomto režimu naváže klient komunikační kanál a server vytvoří datový kanál. V druhém (pasivním) režimu navazuje oba kanály klient. Vytvoření kanálů v obou režimech je ilustrováno na obr. 2. Pasivní režim se často používá v situacích, kdy FTP server nedokáže vytvořit datový kanál. Jedním z hlavních důvodů proč by FTP server nedokázal vytvořit datový kanál je síťový firewall, který zablokuje daný port.

Technologie FTP byla použita z důvodu zjišťování dat na serveru a získávání informací o těchto souborech. Nikoliv kvůli samotnému stažení souborů.

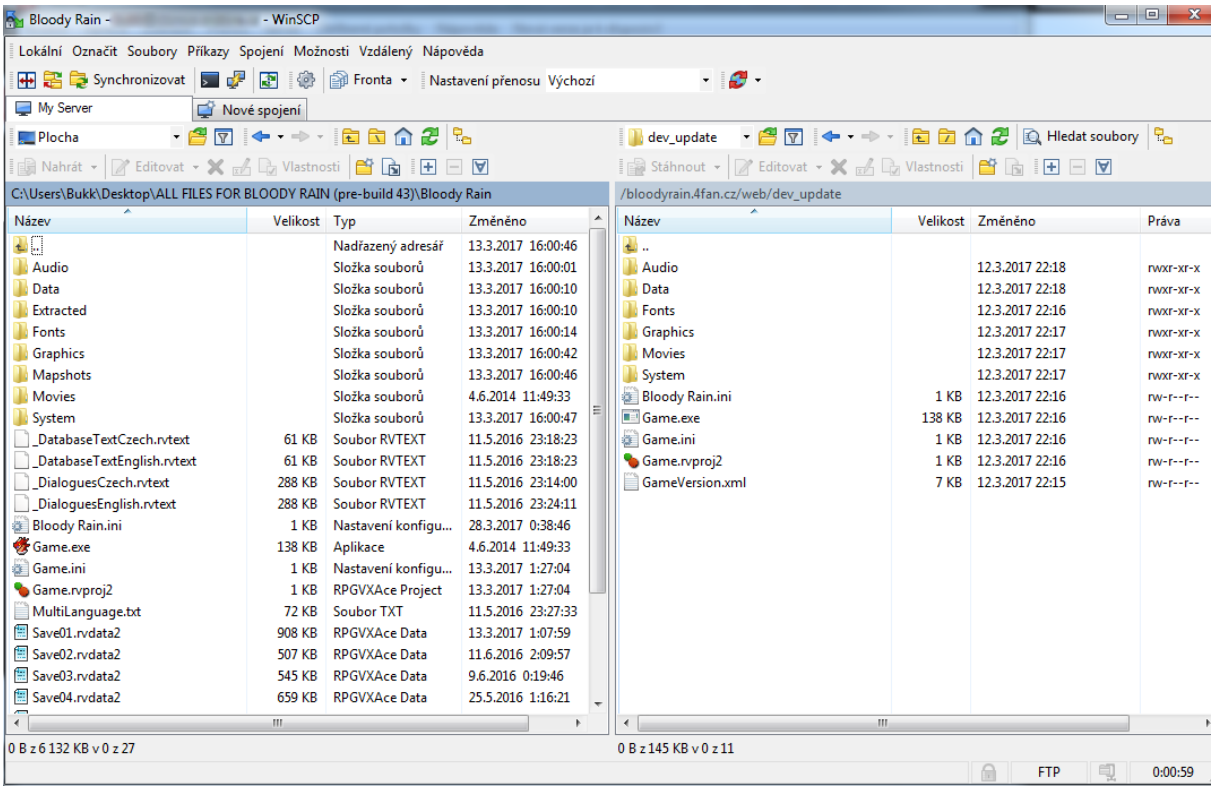
2.1.1 WinSCP

WinSCP je open source SFTP a FTP klient pro platformu Windows. Hlavní důvod jeho použití je bezpečný přenos souborů mezi klientem a serverem. WinSCP podporuje všechny základní operace, včetně stahování, nahrávání, přejmenovávání souboru a adresářů.

Uživatel má na výběr ze dvou uživatelských rozhraní s možností konfigurace. Uživatelské rozhraní lze zvolit během instalace nebo později v nastavení. Příjemné uživatelské prostředí umožňuje uživateli snadno pracovat s lokálními a serverovými soubory (viz obr. 3).

[2] Zdroj: [13]

WinSCP byl v herním launcheru použit pro konzolové použití a jako prostředník pro získávání dat ze serveru. V launcheru se tedy spouští na pozadí, aby navázal spojení se serverem a získal informace o souborech na serveru. Získává pouze informace o souborech, jako jsou například velikost a umístění. Není použit pro samotné stažení souborů ze serveru. Jeho grafický potenciál je v launcheru nevyužit a uživatel ani neví, že tato aplikace byla rozbalena do dočasného adresáře *%temp%* a běží na pozadí v průběhu aktivního spojení. Grafické uživatelské rozhraní je použito pouze pro nahrání nových souborů na server.



Obrázek 3 – Uživatelské rozhraní WinSCP

2.2 Frameworky

Framework je softwarová struktura, která může být například ve formě knihoven. Jeho cílem je ulehčit programátorovi práci a poskytnout mu podporu během programování. Nejčastěji se jedná o ucelený soubor tematicky zaměřených knihoven.

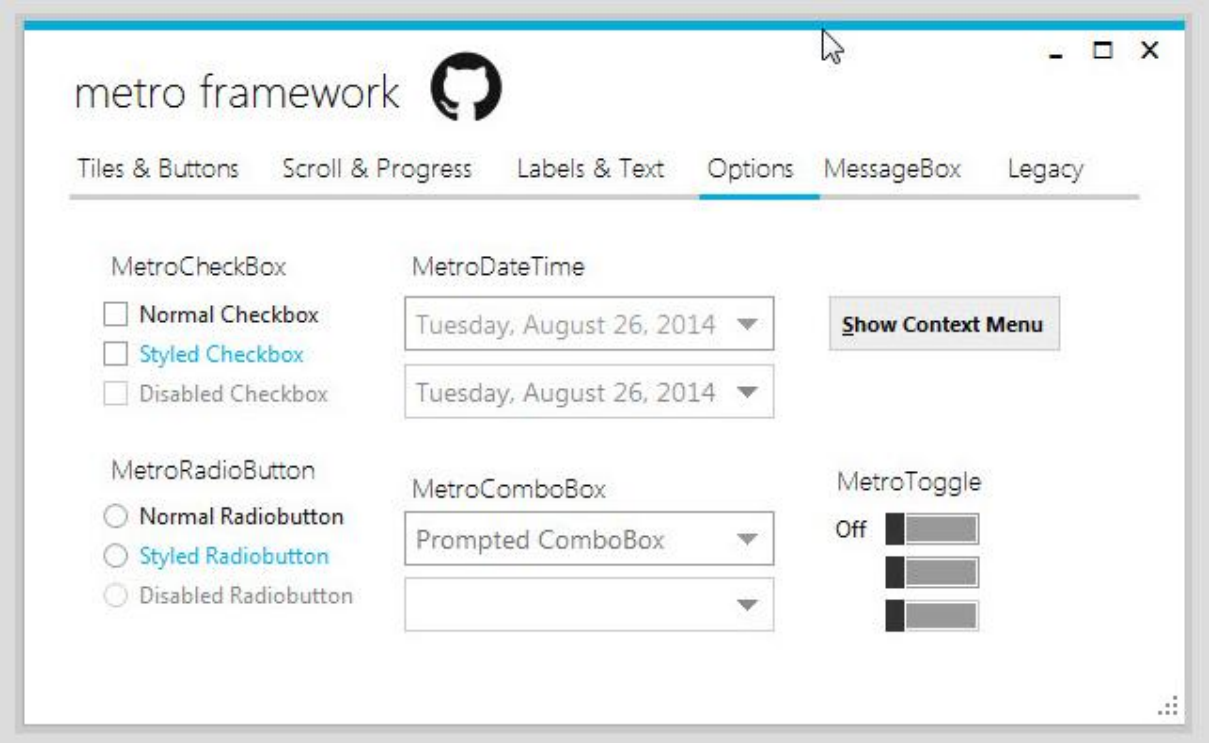
Jeden framework v sobě může obsahovat více knihoven, podpůrné programy nebo i celá vývojová prostředí. Často obsahuje i předdefinované třídy a funkce, které zpracovávají vstupy nebo komunikují se systémovým softwarem.

Kapitola vychází ze zdrojů [15 – 19].

2.2.1 MetroFramework / Modern UI

MetroFramework, také známý pod jménem Modern UI či Metro UI, je grafický framework pro .NET Windows Form aplikace. Lze jej použít pro aplikace psané v C# i VB.NET.

Ačkoliv se jedná o framework určený pro vylepšení komponentů WinForm aplikací, neobsahuje však veškeré komponenty, které poskytuje Windows Form. Mezi podporované komponenty patří často používaná tlačítka, rozbalovací seznamy, přepínače, dialogové okno a další.



Obrázek 4 – Jeden ze dvou vzhledů MetroFrameworku[3]

V základu jsou k dispozici dva barevné motivy. Světlý „Light“ (viz obr. 4) a tmavý „Dark“. Oba motivy si lze libovolným způsobem přizpůsobit dle potřeby, případně vytvořit zcela nový. Tvorba nového motivu se realizuje pomocí XML souboru s názvem *themes.xml*. Příklad části nastavení vlastního motivu znázorňuje obr. 5.

[3] Zdroj: [18]

```
<property name='Button.FontSize.*' value='11' type='System.Single' />
<property name='Button.MetroFontWeight.*' value='Bold' type='MetroFramework.Drawing.MetroFontWeight' />

<property name='Button.BackColor.Normal' value='#EEEEEE' type='System.Drawing.Color' />
<property name='Button.BackColor.Disabled' value='#CCCCCC' type='System.Drawing.Color' />
<property name='Button.BackColor.Hover' value='#666666' type='System.Drawing.Color' />
<property name='Button.BackColor.Press' value='#333333' type='System.Drawing.Color' />

<property name='Button.ForeColor.Normal' value='#000000' type='System.Drawing.Color' />
<property name='Button.ForeColor.Disabled' value='#888888' type='System.Drawing.Color' />
<property name='Button.ForeColor.Hover' value='#FFFFFF' type='System.Drawing.Color' />
<property name='Button.ForeColor.Press' value='#FFFFFF' type='System.Drawing.Color' />
```

Obrázek 5 – Ukázka nastavení vlastního motivu

MetroFramework je sdílený formou svobodné MIT licence, která umožňuje volné šíření, modifikaci a jakékoliv použití. Lze jej získat pomocí balíčkovacího manažera NuGet zadáním příkazu:

```
Install-Package MetroModernUI
```

2.2.2 RGSS3

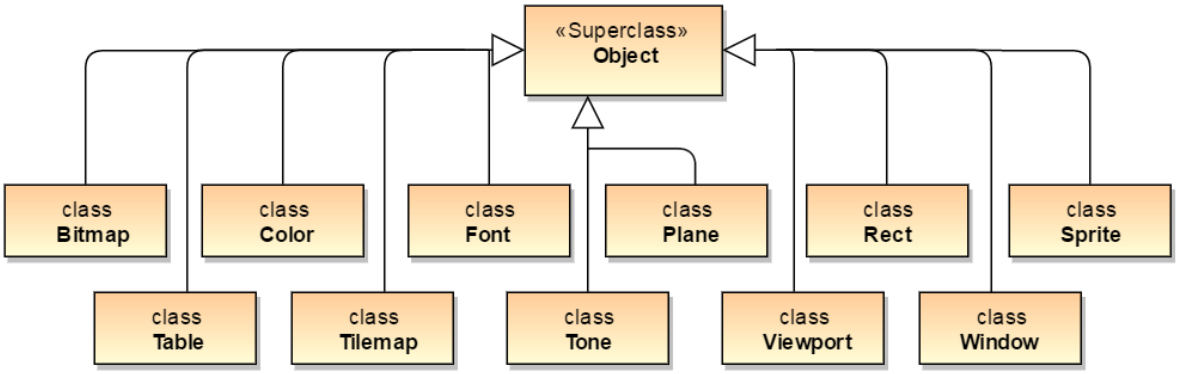
RGSS je zkratkou pro Ruby Game Scripting System. RGSS3 je již třetí verzí od jeho vydání. Využívá objektově orientovaný, skriptovací jazyk Ruby pro vývoj dvou dimenzionálních her na platformu Windows. Obsahuje třídy pro práci s bitmapami, okny, nastavením barev a fontů a moduly pro zpracování audia, grafiky a vstupních dat. Největší část RGSS3 tvoří datové struktury.

Kapitola vychází ze zdrojů [20 a 21].

Třídy zabudované v RGSS3

RGSS3 obsahuje několik zabudovaných tříd, které se starají o jádro hry. Jedná se o třídy Bitmap, Color, Font, Plane, Rect, Sprite, Table, Tilemap, Tone, Viewport a Window. Každá z těchto tříd dědí od supertřídy Object. Jako příklad budou v této podkapitole uvedeny pouze dvě třídy.

UML diagram základních vztahů mezi třídami je zobrazen na obr. 6.



Obrázek 6 – Třídy zabudované v RGSS3

Třída Window

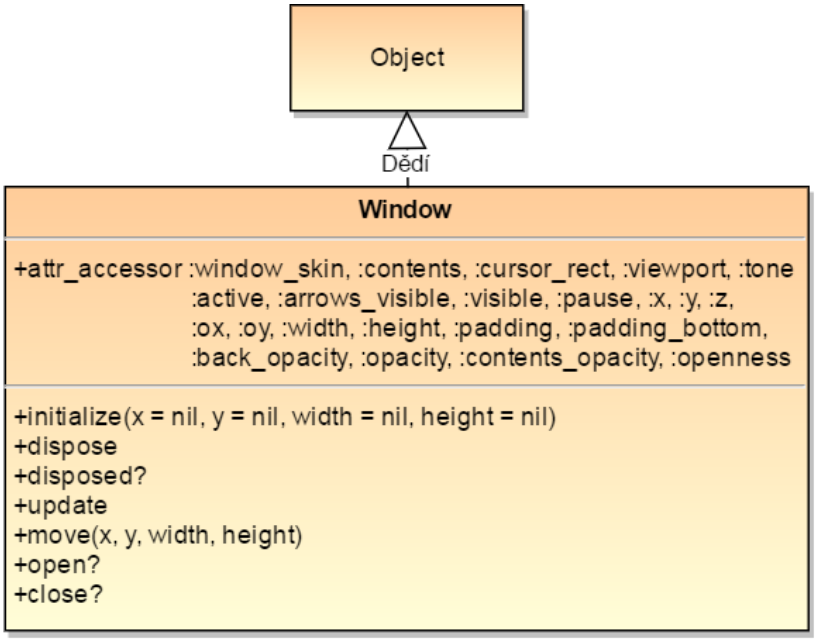
Pomocí třídy Window se vytváří objekty oken, které mohou obsahovat aktivní grafické prvky. Okna se pomocí metody *update* každým framem aktualizují (ve výchozím nastavení obsahuje 1 vteřina 60 framů). Lze do nich umístit animace, případně zachytávat události způsobené hráčem. Příkladem takového okna je okno inventáře, nápověda nebo dialogové okno. UML diagram třídy Window zobrazuje obr. 7.

Použití třídy Window

```
Window.new([x, y, width, height])
```

Vytvoří objekt okna. Poloha a velikost může být manuálně zadána. Pokud se zavolá metoda bez jakýchkoliv parametrů, určí se poloha a velikost dle potřeby.

UML diagram třídy Window



Obrázek 7 – UML diagram třídy Window

Třída Bitmap

Třída Bitmap zajišťuje vykreslení grafiky, její nastavení a řádné zrušení. Kromě obrázků a grafických objektů dokáže vykreslit i text a aplikovat rozmazání obrazu. Třída obsahuje jednu vlastnost Font, která se používá pro vykreslení textu v metodě *draw_text* a určuje font vykreslovaného textu. Některé metody jsou navíc přetížené a požadují jiné vstupní parametry. Na obr. 8 je zobrazen UML diagram této třídy.

Použití třídy Bitmap

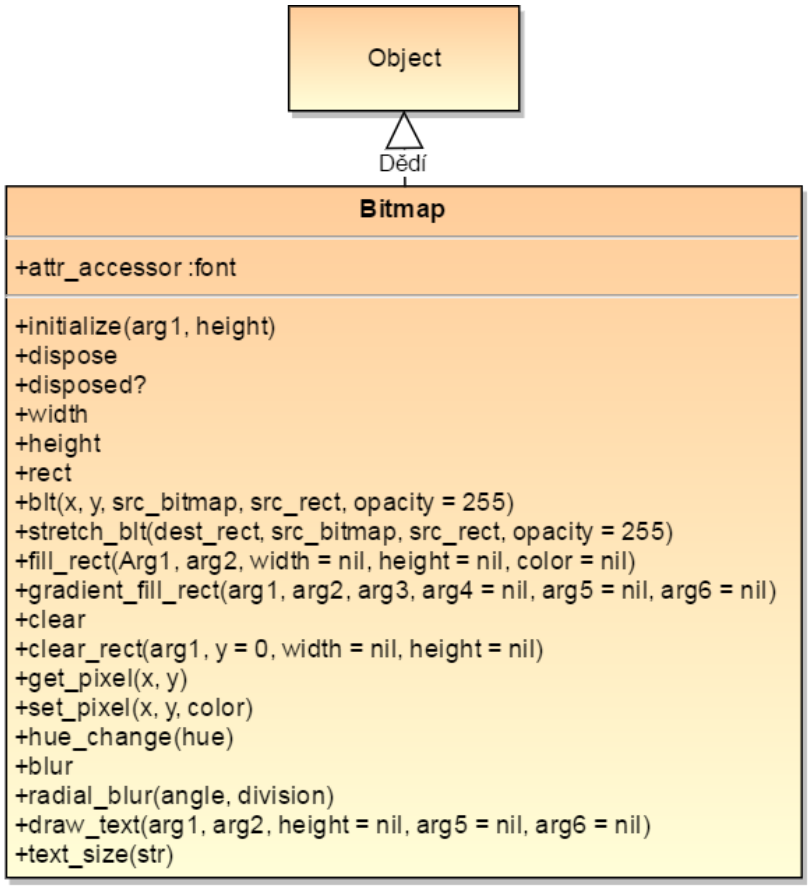
```
Bitmap.new(filename)
```

Načte grafický soubor a vytvoří bitmapový objekt. Automaticky prohledává soubory v RTP a v šifrovaných archivech. Příponu souboru lze vynechat.

```
Bitmap.new(width, height)
```

Vytvoří bitmapový objekt zadané velikosti.

UML diagram třídy Bitmap



Obrázek 8 – UML diagram třídy Bitmap

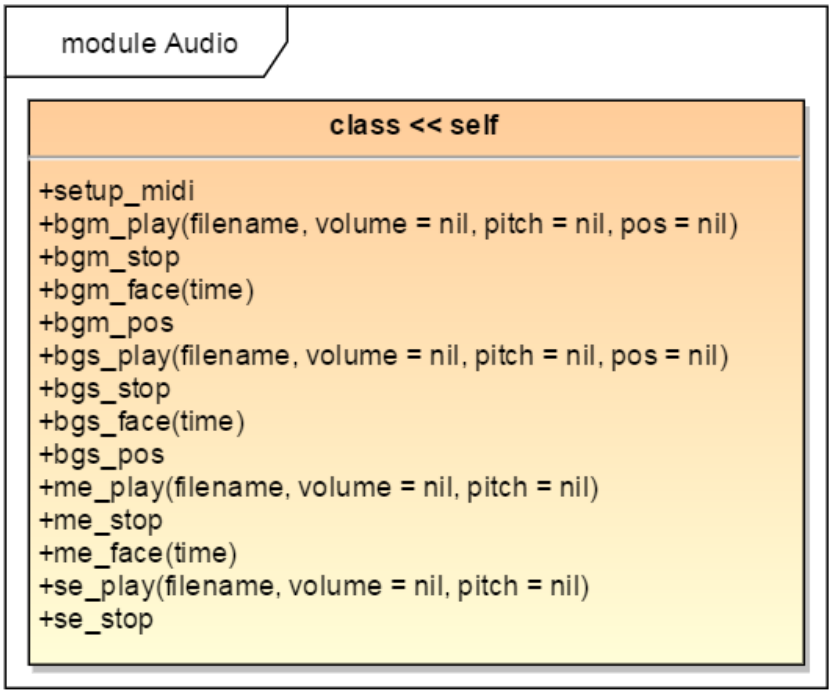
Moduly v RGSS3

RGSS3 obsahuje čtyři moduly. Tyto moduly se starají o zpracování audia (Audio module), grafiky (Graphics module) a uživatelských vstupů (Input module). Poslední modul s názvem RPG Module obsahuje datové struktury s RPG prvky.

Modul Audio

Modul Audio zpracovává hudbu a zvuky. Jako standardní audio formát využívá ztrátový OGG Vorbis. Výhodou tohoto formátu je jeho malá velikost a licence. Je zdarma i pro komerční využití. Nevýhodou je ztrátový formát, který je u her ve stylu retra zanedbatelný.

Modul umožňuje základní operace, jako jsou přehrání, pozastavení nebo postupné zeslabování zvuku. Obsah modulu Audio zobrazuje obr. 9.

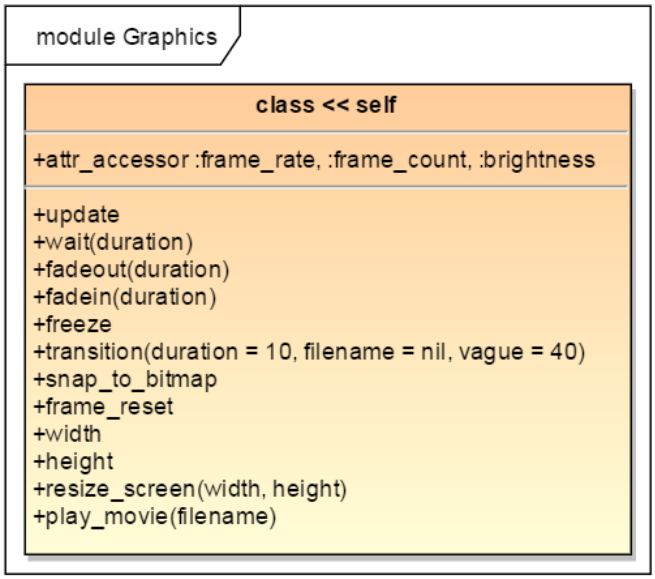


Obrázek 9 – UML diagram modulu Audio

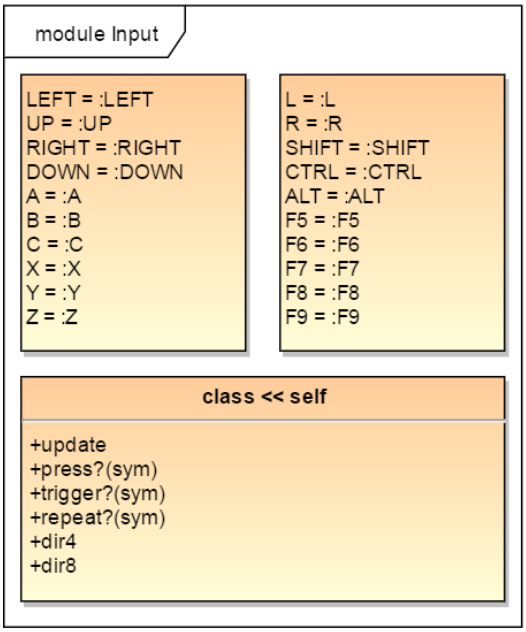
Modul Graphics

Grafický modul obsahuje zpracování grafiky jako takové. Podobně jako třída Window i tento modul obsahuje metodu *update*, která je volaná po jednom framu.

Tato metoda nám poskytuje aktualizaci grafiky na obrazovce a posun času. Dále můžeme provádění aktualizací pozastavit, nastavit plynulé přechody mezi grafickými prvky nebo aktivně čekat. Lze i ručně nastavit jas a kolikrát za vteřinu se obraz obnoví (frame rate). Není to však doporučeno. Obsah modulu Graphics zobrazuje obr. 10.



Obrázek 10 – UML diagram modulu Graphics



Obrázek 11 – UML diagram modulu Input

Modul Input

Posledním modulem je Input. Zpracovává vstupy a události z ovladače nebo klávesnice. Obsahuje metody, které detekují, zda bylo dané tlačítko stlačeno, drženo nebo opakovaně stlačeno. Modul je primárně zaměřen na snímání aktivity herního ovladače (gamepadu) s podporou klávesnice. Obsah modulu Input zobrazuje obr. 11.

2.3 Herní engine

Herní engine je druh softwarového frameworku nebo vývojářského kitu (SDK), který obsahuje zdrojové kódy a nástroje potřebné k vytvoření videohry. Jednoduchou hru lze vytvořit i bez použití herního enginu, ale bude to velmi pracné, neoptimalizované a pomalé. Herní enginy poskytují mocné nástroje, které nejen vývoj hry urychlí a lépe ji optimalizují, ale také není k vývoji zapotřebí tolik pokročilého programování. To umožní vývojáři se více soustředit na gameplay a mechaniky hry. Odpadá tak čas strávený nad řešení komplexních matematických rovnic a výpočtů (např. různé renderování grafiky nebo pohyb).

Mezi populární herní enginy patří CryEngine, Unity 3D, Frostbite nebo Unreal Engine. Každý z těchto enginů obsahuje grafický renderer, který vykresluje grafiku audio subsystém, který se stará o audio, resource manager, který umožňuje správu zdrojů a skripty. Konkrétní komponenty se můžou lišit. Umělá inteligence (AI) bývá většinou vyvíjena mimo hlavní engine, aby byla více sofistikovaná pro danou platformu.

Kapitola vychází ze zdrojů [22 a 23].

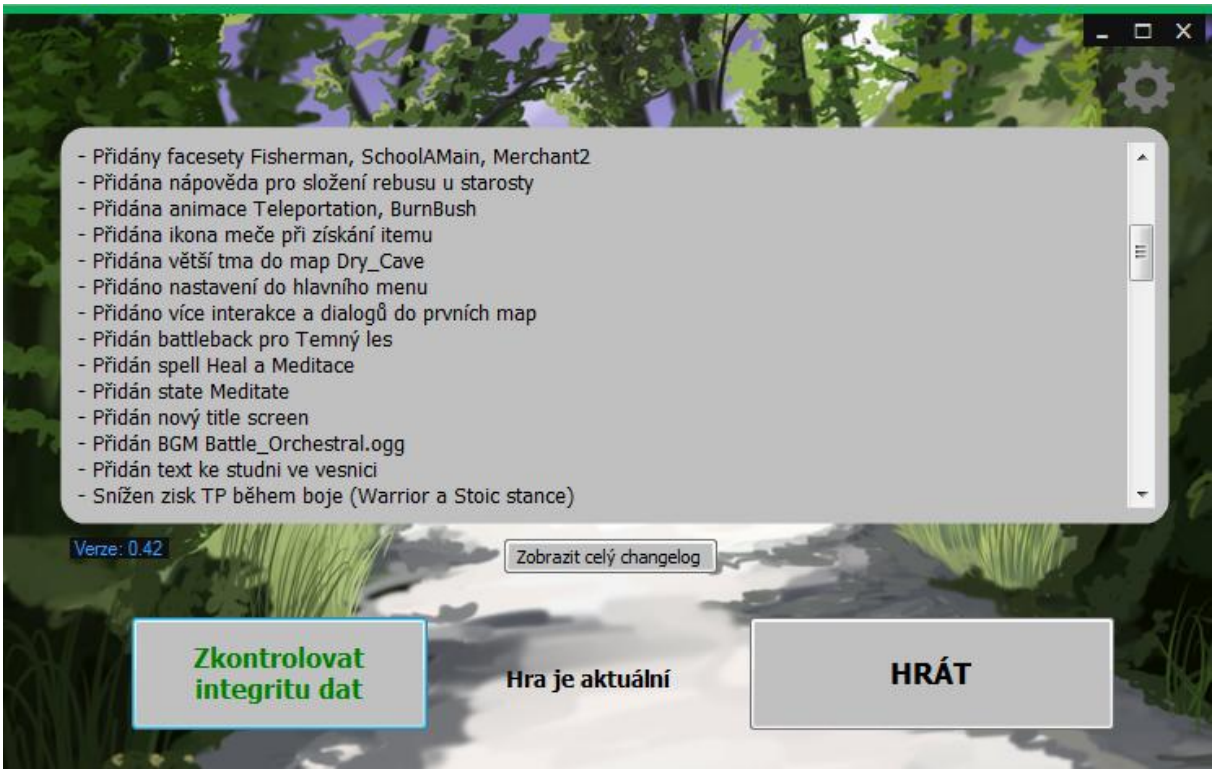
2.3.1 RPG Maker

RPG Maker je název celé série herních enginů, které se od roku 1992 zaměřují na vývoj RPG her. Pro vývoj hry byl zvolen RPG Maker VX Ace (obr. 12), jeden z nejnovějších a nejlepších enginů z celé série. RPG Maker VX Ace má plnou podporu stínů, databáze charakterů, předmětů, dovedností, přehrávání ogg formátů a editor map. Stejně jako předcházející dvě verze, i tato používá programovací jazyk Ruby a framework RGSS verze 3. Hry vytvořené v RPG Maker VX Ace lze exportovat do spustitelného souboru pro platformu Windows. Tato verze byla zvolena pro její programovací jazyk Ruby a velkou komunitu lidí. Nejnovější verze MV používá místo Ruby programovací jazyk JavaScript a má jiné rozlišení textur.

Obrázek 12 – Vývojové prostředí RPG Maker VX Ace

3 HERNÍ LAUNCHER

Herní launcher (dále pouze launcher) je softwarový prostředník mezi hráčem a samotnou hrou. Launcher poskytuje hráči možnost nastavit hru ještě před jejím spuštěním, získat nejnovější aktualizace a spustit samotnou hru. V nastavení launcheru má hráč možnost nastavit velikost okna, celoobrazovkový režim nebo také zakázat kontrolu aktualizací. Obr. 13 znázorňuje vzhled launcheru po získání a porovnání verzí.

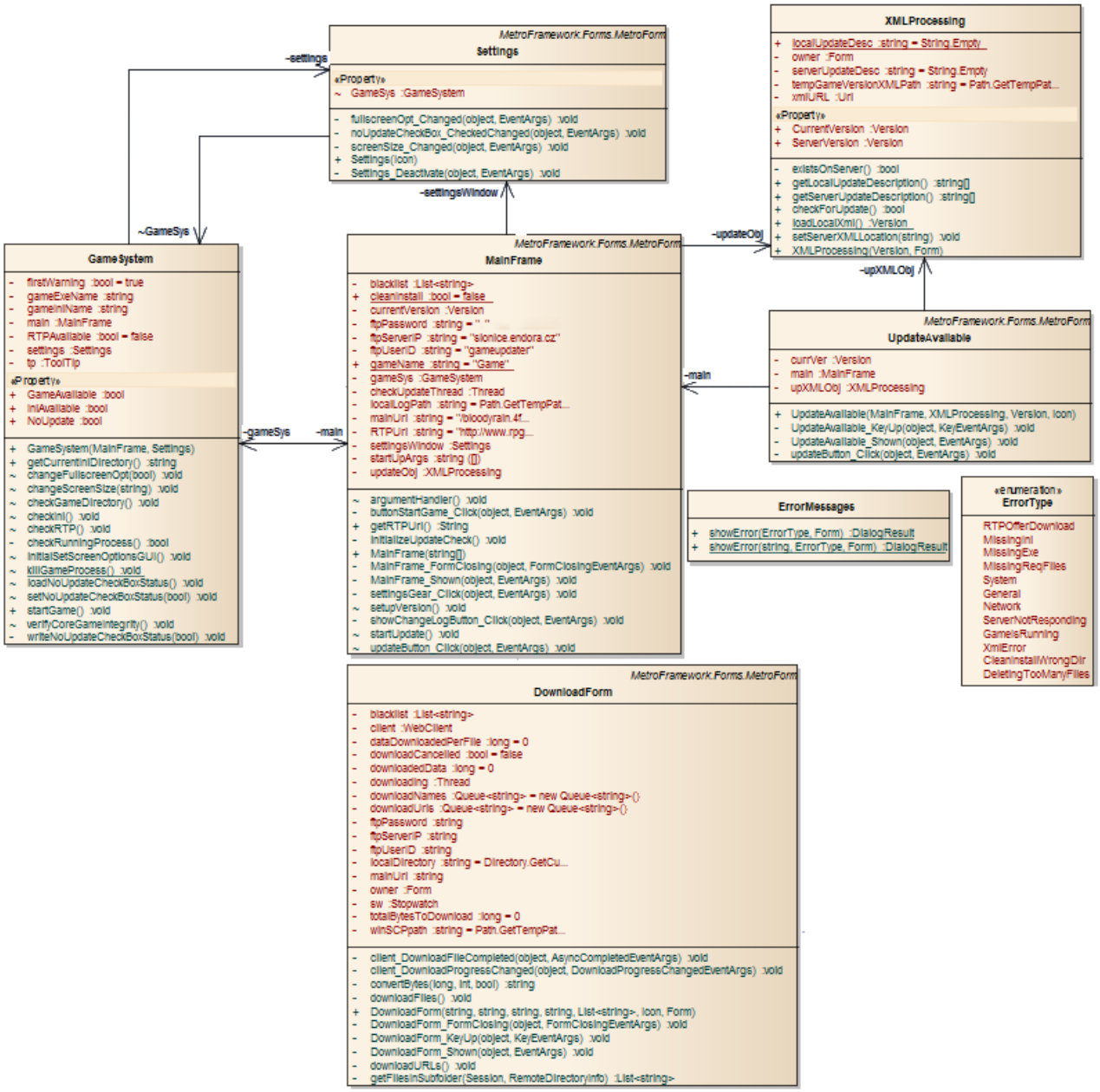


Obrázek 13 – Herní launcher

Launcher po spuštění naváže spojení se serverem a porovná serverovou verzi s lokální verzí hry. V případě dostupné aktualizace nabídne hráči možnost automatické aktualizace.

Pro případ poškození některých souborů hry, obsahuje launcher i systém kontroly integrity dat. Tento systém zkontroluje veškerá herní data, porovná je s nejnovějšími daty na serveru a vyhodnotí, zda je třeba soubory znovu stáhnout. Díky tomuto systému hráč stáhne jen velmi malé množství nových dat a nemusí kvůli jednomu poškozenému souboru stahovat znovu celou hru.

Pro realizaci herního launcheru byl použit programovací jazyk C# a grafický framework MetroFramework. UML diagram launcheru zobrazuje obr. 14.



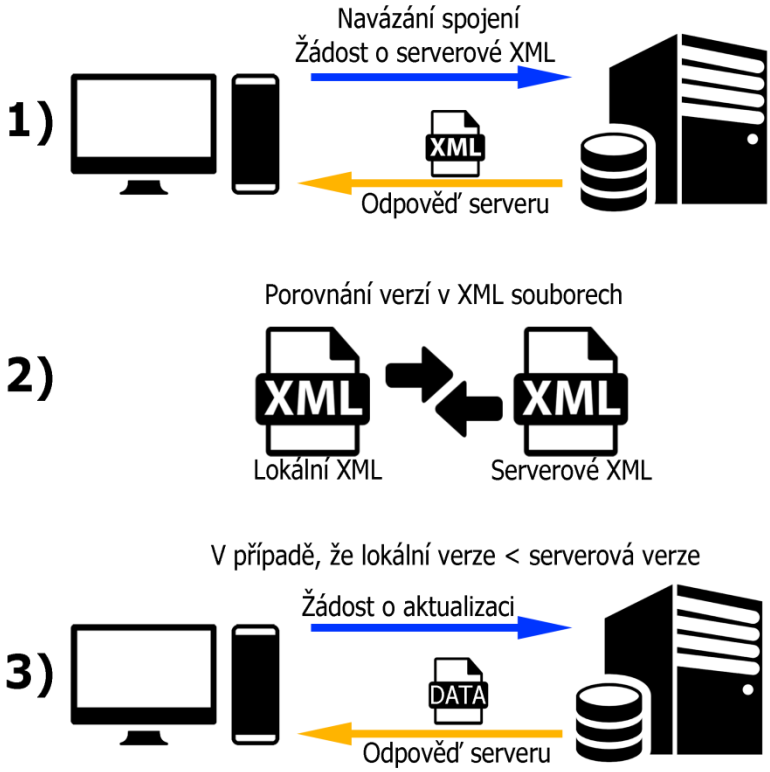
Obrázek 14 – UML diagram herního launcheru

Launcher se skládá ze čtyř dialogových oken (*Mainframe*, *DownloadForm*, *Settings* a *UpdateAvailable*) a tří pomocných tříd (*ErrorMessages*, *GameSystem* a *XMLProcessing*). Mezi největší a nejkomplexnější třídy patří *DownloadForm*, která zajišťuje veškerý přenos a kontrolu integrity dat a *Mainframe*, která poskytujte funkce pro celé hlavní okno.

Třídy *GameSystem* a *Settings* provádí kontrolu existence konfiguračního a spouštěcího souboru hry a dostupnost RTP. Zároveň umožňují zobrazit okno nastavení ještě před spuštěním hry. Nastavení z launcheru je uloženo do konfiguračního souboru hry.

XMLProcessing zpracovává XML soubory a vrací hodnoty elementů (např. číslo verze).

3.1 Model fungování



Obrázek 15 – Model fungování launcheru

Po spuštění se launcher pokusí automaticky navázat spojení se serverem. V případě úspěšného spojení zažádá o serverový soubor *GameVersion.xml*, který se stáhne do dočasného adresáře *%temp%*. Následně se načtou informace jak ze staženého a lokálního souboru. Tyto XML soubory obsahují element s verzí hry. Obě verze se vzájemně porovnají.

V případě, že serverová verze je vyšší než lokální, znamená to, že server obsahuje novější verzi a je dostupná aktualizace. Tato skutečnost je hráči oznámena formou dialogového okna, které obsahuje obě verze a výčet všech změn co byly provedeny v nové verzi. Celý tento průběh znázorňuje obr. 15.

Pokud hráč souhlasí s aktualizací hry, launcher opět naváže spojení se serverem, zkontroluje veškeré lokální soubory a porovná je s novými soubory na serveru. Ze serveru se stáhnou pouze soubory, které se liší velikostí. To má za následek, že hráč nemusí stahovat vždy celou hru, ale pouze aktualizované soubory (například nové grafické prvky, mapy nebo skripty). Po dokončení stahování se nahradí lokální XML soubor serverovým a dočasný soubor v *%temp%* adresáři se odstraní. Hráč může stažení aktualizace odmítnout a pokračovat v hraní staré verze.

3.2 Soubor GameVersion.xml

Soubor *GameVersion.xml* je datový soubor uchovávající informace o verzi hry. Je napsán pomocí značkovacího jazyka XML. Obsahuje datum vydání, verzi hry a popis všech změn. Ke kódování byl použit formát UTF-8.

3.2.1 Struktura souboru GameVersion.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<gameVersion>
	<update>
		<releaseDate>20.03.2017</releaseDate>
		<version>0.41</version>
		<description>Patch notes</description>
	</update>
</gameVersion>
```

Kořenovým elementem je *gameVersion*, který obsahuje podelement *update*. Ten obsahuje veškeré informace týkající se aktuální verze hry. Tento podelement byl vytvořen za účelem oddělit možné další informace, které by soubor XML mohl obsahovat, a netýkali by se přímo verzovacího systému.

Veškerá data o aktualizaci jsou místo atributů řešena pomocí XML elementů. Odřádkované elementy jsou více přehledné pro případnou manuální editaci. Na obr. 16 je ukázka alternativního řešení pomocí atributů.

```
<?xml version="1.0" encoding="UTF-8"?>
<gameVersion>
    <update releaseDate="20.11.2016" version="0.41" description="Patch Notes"/>
</gameVersion>
```

Obrázek 16 – Alternativní řešení pomocí atributů

Element releaseDate

Element *releaseDate* slouží pouze jako informativní část, která zobrazuje den vydání dané aktualizace. Datum není nikde v programu dále zpracováváno ani zobrazeno.

Element version

Element *version* obsahuje verzi hry. Jeho obsah se zpracovává do datového formátu Version, který se následně porovnává a určuje, zda je dostupná aktualizace. Základní formát verze je 0.1, kde 0 značí hlavní číslo verze a 1 její subverze. Tento formát lze rozšířit o třetí číslo (0.1.2), které značí drobnou aktualizaci. Jiný formát ani žádné nenumerické znaky nejsou podporovány. Oba formáty jsou vypsány v GUI pod seznamem změn.

Element description

Posledním elementem je *description*, který obsahuje veškeré poznámky o změnách v dané verzi. V tomto elementu se většinou vyskytují řádově desítky řádků poznámek. Všechny poznámky jsou zpracovány do ListBoxu, aby si je hráč mohl pohodlně přečíst. Poznámky respektují konce řádků a zalamování textu. Zároveň poskytují podporu pro diakritiku. Lze do něj umístit téměř cokoliv. Jedinou podmínku je zachování validity XML souboru a dodržení kódování UTF-8.

3.2.2 Parsování XML souboru

Čtení a následné parsování XML souboru lze řešit pomocí několika různých tříd. Mezi standardní „čtečky“ XML souborů patří XmlReader a XmlTextReader. Pro tvorbu celých instancí XML souborů lze použít třídy XmlDocument nebo XDocument.

Pro launcher byla vybrána třída XmlDocument, která se pokusí načíst celý soubor *GameVersion.xml* ze současného adresáře, kde se launcher nachází. Po načtení se vybere hlavní uzel *update*, který obsahuje požadované informace.

Celý proces načítání souboru je obalen ochranným blokem try-catch pro případ selhání. Může se stát, že soubor v adresáři neexistuje, XML soubor není validní nebo obsahuje neplatné znaky mimo kódování UTF-8. Jednotlivé chybové stavy jsou oznámeny vhodnou chybovou hláškou.

```
XmlDocument doc = new XmlDocument();
doc.Load(Directory.GetCurrentDirectory() + "\\GameVersion.xml");
XmlNode node = doc.DocumentElement.SelectSingleNode("update");
Version currentVersion = Version.Parse(node["version"].InnerText);
string localDescription = node["description"].InnerText;
```

3.3 Chybové hlášky

Launcher během svého běhu může narazit na různé chybové stavy a tuto skutečnost musí oznámit uživateli. Jedná se převážně o ztrátu spojení se serverem, nečekané přerušení stahování, chybějící soubory nebo špatné umístění. Pro řešení tohoto problému byla implementována třída *ErrorMessages* obsahující pouze chybové hlášky a varování. Tato třída zajišťuje zobrazení okna s příslušnou chybou a její možné řešení.

Třída také obsahuje výčtový typ *ErrorType*, obsahující výčet nejčastějších chyb (vizte tabulka 1). Následuje výčet dostupných typů chyb.

```
public enum ErrorType {
        General, MissingIni, MissingExe, MissingReqFiles, XmlError,
        RTPOfferDownload, Network, ServerNotResponding, System,
        GameIsRunning, CleanInstallWrongDir, DeletingTooManyFiles  };
```

Tabulka 1 – Tabulka chybových hlášek

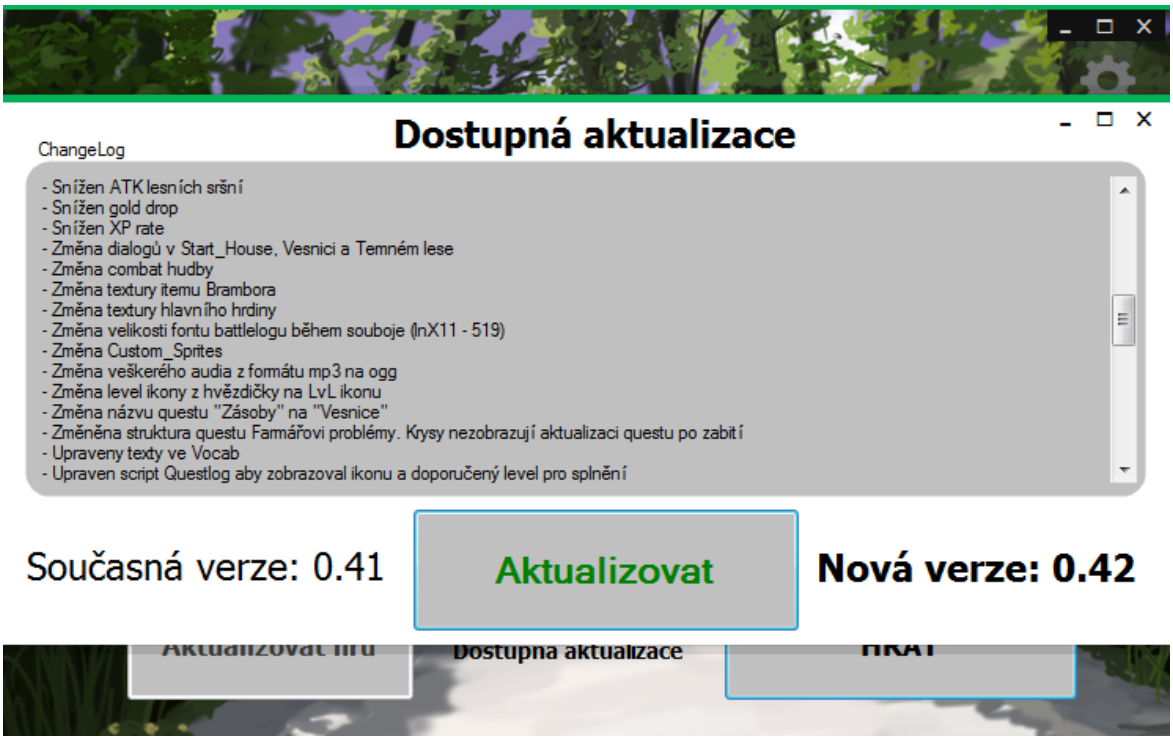
Název chyby	**Vysvětlení**
General	Obecná nespecifikovaná chyba.
MissingIni, MissingExe, MissingReqFiles	Chybí inicializační soubor, spustitelný soubor nebo soubor nutný ke spuštění.
XmlError	Chyba při čtení/prací s XML souborem.
RTPOfferDownload	RTP není k dispozici. Nabídne RTP ke stažení.
Network	Síťová chyba, přerušení spojení.
ServerNotResponding	Server neodpovídá.
System	Systémová a neočekávané pády.
GameIsRunning	Druhá instance hry již běží. Launcher nabídne její ukončení.
CleanInstallWrongDir	Spouštěcí argument *-cleanInstall* nenalezl žádné herní soubory.
DeletingTooManyFiles	Spouštěcí argument *-cleanInstall* detekoval neočekávaně velké množství souborů pro odstranění.

Chybová hláška se vyvolá zavoláním statické metody *showError*, která jako vstupní parametr požaduje pouze typ chyby. O vše ostatní se postará switch, který na základě chyby vyhodnotí vážnost chyby, nastaví vzhled chybového hlášení (tlačítka, text a zvukový signál) a nakonec dané chybové okno zobrazí. Následuje část metody *showError*.

```
public static DialogResult showError(string msg = "", ErrorType errType,
Form owner = null)
{
        // Výchozí hodnoty
        string errorMessage = msg;
        string title = "Nastala chyba";
        MessageBoxIcon msgIcon = MessageBoxIcon.Warning;
        MessageBoxButtons buttons = MessageBoxButtons.OK;

        /* Switch určující druh chyby */
        switch (errType) { … }

        return MetroMessageBox.Show(owner, errorMessage, title, buttons,
                                            msgIcon);
}
```

3.4 Stažení aktualizace

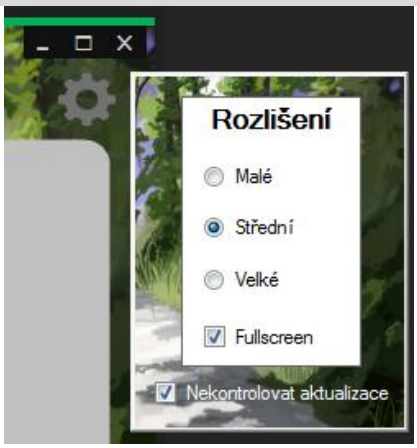
V případě, že je na serveru dostupná aktualizace, zobrazí se dialogové okno obsahující číslo staré i nové verze a výčet všech změn, které byly provedeny v nové verzi hry (viz obr. 17). Hráč se může rozhodnout zda hru aktualizuje nebo si ponechá starou verzi.



Obrázek 17 – Oznámení dostupné aktualizace

Aktualizace jsou kontrolovány při každém spuštění launcheru. Tuto automatickou kontrolu aktualizací lze vypnout v nastavení launcheru. Aby bylo možné kontroly aktualizací vypnout, je nutné mít stáhnutý konfigurační soubor *Game.ini*, do kterého se zapíše hodnota *NoUpdate*. Vypnuté aktualizace jsou zobrazeny na obr. 18.

```
[Launcher]
NoUpdate=1
```

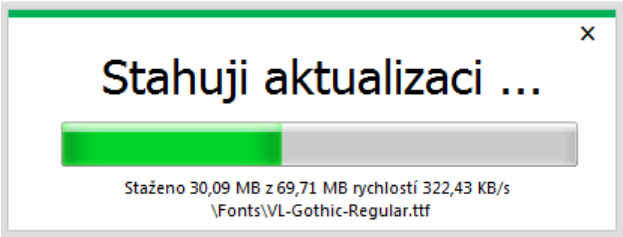


Obrázek 18 – Nastavení launcheru a hry

Celé stahování běží v druhém vlákně, takže okna jsou plně uživatelsky responsivní. Uživatel může kdykoliv stahování přerušit a aktuální průběh lze zobrazit v okně. Kdyby stahování neběželo v druhém vlákně, okno stahování by bylo pro uživatele neovladatelné, nemohl by stahování přerušit a uživatel by si mohl myslet, že program neodpovídá.

```
Thread downloading = new Thread(delegate(){ downloadURLs(); });
downloading.Start();
```

Po zahájení aktualizace se zobrazí okno s průběhem (viz obr. 19), kde hráč může sledovat procentuální stav, aktuálně stahovaný soubor a rychlost stahování. *GameVersion.xml* se stahuje až jako poslední. Kdyby došlo k chybě tak launcher opět nabídne aktualizaci, stáhne poškozené soubory a soubory, které hře ještě chybí ke spuštění.



Obrázek 19 – Stažení aktualizace

Po spuštění vlákna *downloading* se začne vykonávat metoda *downloadURLs*. Ta nejdříve naváže FTP spojení se serverem pomocí WinSCP klienta. Pokud je navázání spojení úspěšné, launcher shromáždí veškeré soubory na serveru, které jsou potřeba pro aktualizaci. Pro každý soubor na serveru se vytvoří URL odkaz a lokální umístění. Následně je zkontrolováno, zda místní soubor již neexistuje. Kontrola probíhá formou porovnání velikostí serverového a místního souboru. Po dokončení metody máme k dispozici dvě fronty, obsahující URL adresy ke stažení a lokální umístění.

Pokud vše proběhlo bez chyby, spustí se samotné stahování souborů metodou *downloadFiles*. Tato metoda je spuštěna rekurzivně. To znamená, že po stažení prvního souboru je opět spuštěna, aby stáhla další soubor ve frontě. Ke stažení souborů se nevyužívá FTP spojení, ale zabudované třídy WebClient. Tato třída poskytuje metody ke stažení a odesílání dat ze zadané adresy.

Zpracování adres a zahájení stahování v metodě *downloadFiles* je zobrazeno v následujícím zdrojovém kódu.

```
if (downloadUrls.Any()) { // Pokud jsou ve frontě nějaké adresy, pokračuj
        client = new WebClient();
        client.DownloadProgressChanged+=client_DownloadProgressChanged;
        client.DownloadFileCompleted += client_DownloadFileCompleted;
        var url = downloadUrls.Dequeue();
        var filename = downloadNames.Dequeue();
        client.DownloadFileAsync(new Uri(url), filename);

        // Zobrazení aktuálního souboru v GUI
        this.Invoke((MethodInvoker) delegate {
                currentFileLabel.Text =
                        filename.Replace(Directory.GetCurrentDirectory(), "");  });
} else { /* Stahování dokončeno */ }
```

Na každého nového web klienta jsou navázány události *DownloadProgressChanged* a *DownloadFileCompleted*.

Událost *DownloadProgressChanged* zajišťuje aktualizaci ukazatele průběhu. Počítá počet stažených dat a spolu s rychlostí stahování je vykresluje do GUI. Data jsou počítána v bytech, ale pro lepší přehlednost jsou převedena do vyšších jednotek (KB, MB).

Událost *DownloadFileCompleted* je aktivována po stažení aktuálního souboru. Tato událost zavolá zpětně metodu *downloadFiles*, aby se mohl stáhnout další soubor ve frontě.

3.5 Validace souborů

Validace souborů probíhá jako součást metody stažení aktualizace. Ověřuje integritu dat a zároveň kontroluje existující soubory, aby hráč nemusel stahovat vždy celou hru, ale pouze potřebné, nové nebo poškozené soubory. Kontrola probíhá formou porovnání velikostí v bytech. Každá změna souboru ovlivní velikost souboru a ta je detekována. Nejdříve launcher zjistí, zda lokální soubor existuje. Pokud existuje, porovná se jeho velikost s velikostí serverového souboru. V případě, že se velikosti shodují, soubor je ze seznamu stahovaných souborů vyřazen, protože se jedná o aktuální a správně fungující soubor. Kontrola je v kódu realizována následujícím způsobem.

```
foreach (string fileURLlocation in serverFiles) {
        /*  Příprava souborů ke stažení */
        string fileLocalLocation = Directory.GetCurrentDirectory();
        RemoteFileInfo rfInfo = session.GetFileInfo(fileURLlocation);
        long fileSizeDownload = rfInfo.Length;
        if (File.Exists(fileLocalLocation)) {
                long localFileSize = new FileInfo(location).Length;
                if (fileSizeDownload == localFileSize) continue;
        }
        /*  Samotné stažení souborů  */
}
```

4 2D HRA BLOODY RAIN

Tato kapitola popisuje důležité části 2D hry s názvem Bloody Rain. Jedná se o hru žánru RPG, která byla vyvinuta pomocí herního enginu RPG Maker VX Ace, frameworku RGSS3 a programovacího jazyka Ruby. Hra je určena pouze pro operační systémy Windows.

Bloody Rain obsahuje velké množství řídících prvků a skriptů, které jsou propojeny s frameworkem RGSS3. Z tohoto důvodu jsou popsány pouze některé klíčové funkce a zbytek kapitoly je pojat se značnou mírou abstrakce.

Obr. 20 zobrazuje hráče v herním světě.

Obrázek 20 – Hráč v herním světě

4.1 Technické detaily

Hra Bloody Rain je stylizovaná do „old-school“ stylu. Tento styl je znám především pro své grafické zpracování. Veškerá grafika má malé rozlišení a pixelový vzhled. Z tohoto důvodu je herní rozlišení pouze 544x416 pixelů. Je možné zvětšit rozlišení hry do tří různých rozlišení, ale textury budou stále stejné. Hra neobsahuje textury s vysokým rozlišením, pouze zvětší původní textury.

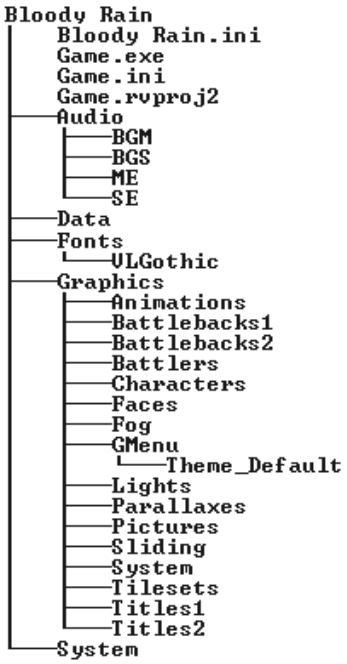
Hru lze také spustit v celoobrazovkovém módu. Textury budou opět zvětšeny z jejich původní velikosti a hráčům se širokoúhlými obrazovkami budou okraje vyplněny černými pruhy.

Díky malému rozlišení a pixelových textur je hra relativně nenáročná na systém. Požadavky na systém jsou uvedeny v tabulce 2.

Tabulka 2 – Požadavky na systém

	Minimální	Doporučené
Operační systém	Microsoft Windows XP / Vista / 7	Microsoft Windows 7 / 8 / 8.1 / 10
CPU	1.0 GHz Intel Pentium nebo lepší	2.0 GHz Quad Core nebo lepší
RAM	256 MB nebo více	512 MB nebo více
HDD	400 MB	400 MB

Strukturu hry zobrazuje obr. 21. Hlavní adresář obsahuje projekt, společně s inicializačním a spouštěcím souborem. Podadresáře jsou rozděleny na audio, data, fonty, grafiku a systém.



Obrázek 21 – Adresářová struktura Bloody Rain

4.2 Konfigurační soubor

Konfigurační soubor *Game.ini* obsahuje uživatelsky modifikovatelné nastavení hry. Kromě nastavení jazyka, stínů a grafiky, obsahuje také umístění skriptů a RGSS frameworku. Bez tohoto konfiguračního souboru nelze hru spustit. Konfigurační soubor je rozdělen do pěti kategorií:

- Game,
- Language,
- Graphics Settings,
- Fullscreen,
- Launcher.

Každá konfigurační část je načítána individuálně příslušným skriptem. I když je třeba konfigurační soubor otevřít několikrát a v celé hře se vyskytuje více načítacích metod, tento styl načítání zvyšuje přehlednost jednotlivých skriptů.

Poslední konfigurační část je *Launcher*. Ta byla použita pro uložení nastavení launcheru a ke spuštění samotné hry není potřeba.

Konfigurační část Game

Nejdůležitější konfigurační částí je část *Game*, která je umístěná na začátku konfiguračního souboru. Bez této části není možné hru spustit. Obsahuje umístění knihovny frameworku RGSS3, umístění všech skriptů, verzi RTP, název hry a její popis.

Tuto část načítá jádro herního enginu a při exportu hry je načítací skript uložen do spustitelného souboru hry. Žádnou ze značek v části *Game* by neměl hráč měnit ani s ní nijak manipulovat. Jakákoliv nezkušená manipulace může vést k pádům hry. Pro spuštění hry je však nezbytné, aby tato část byla veřejně umístěna v konfiguračním souboru.

Značky *CreationDate* a *CloudDate* slouží pouze pro synchronizaci s cloudovou službou Steam. Ve finálním produktu je možné tyto dvě značky odstranit.

```
[Game]
RTP=RPGVXAce
Library=System\RGSS301.dll
Scripts=Data\Scripts.rvdata2
Title=Bloody Rain
Description=Dobrodruzne, temne RPG
CreationDate=1401875373
CloudDate=1427059559
```

Konfigurační část Language

Konfigurační část *Language* obsahuje pouze jednu stejnojmennou značku. Tato značka určuje jazyk hry, ve kterém se hra spustí. V nastavení hry lze jazyk změnit. Pokud tato značka v konfiguračním souboru chybí, použije se výchozí jazyk, kterým je čeština.

```
[Language]
Language=Czech
```

V současné chvíli je k dispozici pouze česká lokalizace. Anglická lokalizace obsahuje základ a přípravy pro překlad. Není však dokončena.

Nastavení jazyka se z konfiguračního souboru načítá do instanční proměnné *language*, jak je zobrazeno v následujícím zdrojovém kódu.

```
GAME_INI_ENTRY = /^Language=(.+)$/
def load_language
	saved_lang = nil
	open("Game.ini") { |f|
	    f.each_line { |line|
	        GAME_INI_ENTRY.match(line) { |m|
	        saved_lang = m[1].to_sym
	        break
	     }
	  }
	}
	@language = saved_lang if saved_lang
end
```

Konfigurační část Graphics Settings

Konfigurační část *Graphics Settings* se vztahuje k nastavení grafiky, stínů, světla a mlhy. Nastavení pochází z grafického frameworku Khas Graphics Library, který je ve hře použit. Stejně jako ostatní části, i tuto část lze nastavit přímo v nastavení hry.

Značka *light_size* je jedinou značkou kde se udává číselná hodnota velikosti světla. Zbylé značky mají pouze stavy ON a OFF.

```
[Graphics Settings]
[static_shadows ON]
[dynamic_shadows ON]
[soft_shadows ON]
[light_size 100]
[light_opacity ON]
[fog ON]
```

Hodnoty z konfiguračního souboru se načtou do instančního pole *settings*. Z pole *settings* se poté získají jednotlivé hodnoty, které jsou použity v dalších metodách frameworku Khas Graphics Library. Načítání do pole probíhá následujícím způsobem.

```
def load
	return unless File.file?("Game.ini")
	File.readlines("Game.ini").each do |line|
		next unless line.start_with? '[Graphics Settings]'
		@settings[line.khas_command.to_sym] = (line.khas_value.is_int? ?
			line.khas_value.to_i : line.khas_value)
			if @settings.include?(line.khas_command.to_sym)
	end
end
```

Konfigurační část Fullscreen

Poslední konfigurační částí je *Fullscreen*. V této části se konfiguruje fullscreen mód a velikost herního okna. Okno hry lze změnit z původní, malé velikosti (0) na středně velkou (1) až velkou velikost (2). Velikost okna ve fullscreen a windowed módu určují značky *FullscreenRatio* a *WindowedRatio*. Jakákoliv jiná hodnota než 0, 1 a 2 není přípustná.

Značka *Fullscreen* je číselnou variantou boolean, nabývající hodnot 0 a 1. Určuje zda má být hra spuštěna ve fullscreenu. Toto nastavení lze změnit v nastavení launcheru nebo klávesovými zkratkami ve hře.

```
[Fullscreen++]
Fullscreen=0
FullscreenRatio=0
WindowedRatio=2
```

Nastavení velikosti okna se načítá do instančních proměnných *fullscreen, fullscreen_ratio* a *windowed_ratio*, vizte zdrojový kód.

```
def load
	return unless File.file?("Game.ini")
	File.readlines("Game.ini").each do |line|
		next unless line.start_with? '[Fullscreen++]'
		@fullscreen = line.partition('=').last.to_i if line.include?
							"Fullscreen="
		@fullscreen_ratio = line.partition('=').last.to_i if line.include?
							"Fullscreen_ratio="
		@windowed_ratio = line.partition('=').last.to_i if line.include?
							"Windowed_ratio="
	break if line.include? "WindowedRatio"
	end
end
```

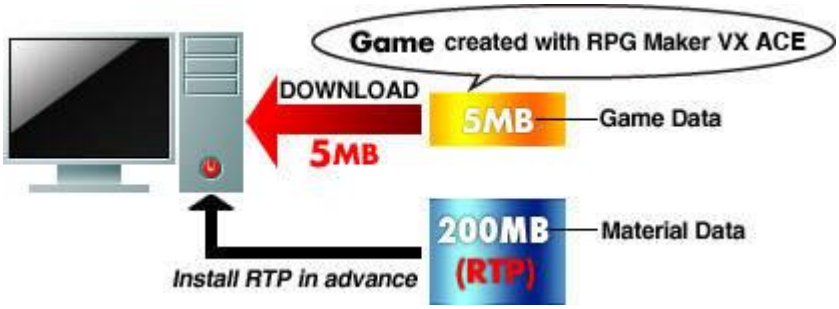
4.3 RTP

Runtime Package (RTP) je kolekce materiálů RPG Makeru. Jedná se o systém, který výrazně snižuje celkovou velikost hry. Obsahuje grafiku, hudbu (.ogg) a knihovny, které jsou potřeba při tvorbě hry v RPG Maker VX Ace. Bez některých souborů, které obsahuje RTP nelze hru vytvořenou v RPG Makeru spustit.

Kapitola vychází ze zdroje [24].

Stahování s nainstalovaným RTP

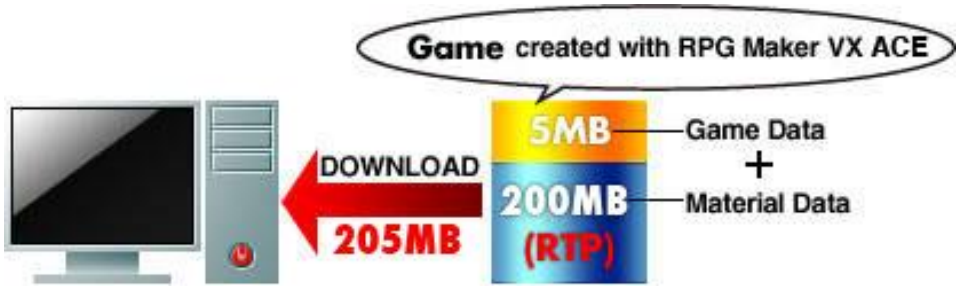
Pokud je RTP předem nainstalován v systému, většina materiálů potřebná ke spuštění hry je již připravena k použití. Hráč poté stáhne jen minimální množství dat, které RTP neobsahuje a může hrát. Na obr. 22 je zobrazen příklad, kde hráč má předem nainstalovaný RTP a stahuje pouze herní data o velikosti 5 MB.



Obrázek 22 – Stahování s předem nainstalovaným RTP[4]

Stahování bez nainstalovaného RTP

Pokud si hráč nenainstaluje RTP předem, musí stáhnout herní data a také všechny materiály potřebné ke spuštění. To má za následek mnohem větší velikost souborů, které musí hráč stáhnout. Obr. 23 zobrazuje příklad, kdy hráč nemá nainstalovaný RTP a musí stáhnout herní data i materiály. Celková velikost ke stažení se značně zvýšila z původních 5 MB na 205 MB.



Obrázek 23 – Stahování bez nainstalovaného RTP[5]

[4] Zdroj: [24]
[5] Zdroj: [24]

4.4 Hlavní menu

Hlavní menu je nejdůležitější část hry. Je to úvodní obrazovka, kterou hráč uvidí po zapnutí hry (viz obr. 24). V hlavním menu může hráč začít novou hru, načíst uloženou pozici, nastavit hru nebo ze hry odejít.

Obrázek 24 – Hlavní menu Bloody Rain

K vytvoření interaktivních položek (Nová hra, Pokračovat, Nastavení a Ukončit) byly použity metody a třídy z RGSS3 frameworku. Jedná se o třídy *Scene_Base* a *Window_Command*.

Třída *Window_TitleCommand* vytváří list interaktivních položek, které RGSS3 framework vykreslí do hlavního menu.

```
class Window_TitleCommand < Window_Command
        attr_reader  :list

        def add_command(name, symbol, enabled = true)
           @list.push({:name=>name, :symbol=>symbol, :enabled=>enabled})
        end

        def make_command_list
           add_command("Nová hra", :new_game)
           add_command("Pokračovat", :continue, continue_enabled)
           add_command("Nastavení",  :options)
           add_command("Konec", :shutdown)
        end
end
```

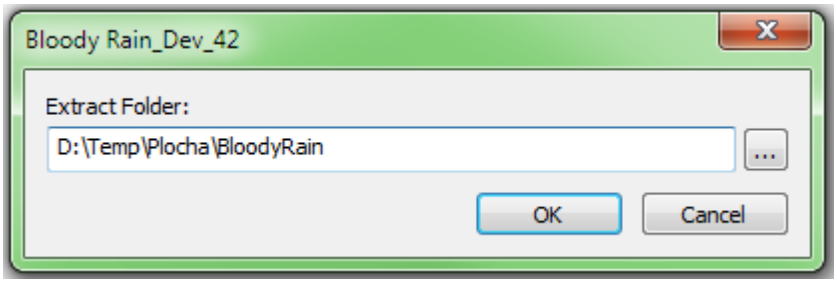
Aby vytvořené interaktivní texty v listu mohly fungovat, je na ně nastaven handler. Každý handler nastaví příslušnému textu (například *:new_game*) metodu, která se vykoná po interakci. Pro položku listu *:new_game* se vykoná metoda *command_new_game*.

K volání jednotlivých scén se používá SceneManager, který spravuje veškeré scény ve hře. Následuje zdrojový kód třídy *Scene_Title*.

```
class Scene_Title < Scene_Base
	def create_command_window
	   @command_window = Window_TitleCommand.new
	   @command_window.set_handler(:new_game,
	                               method(:command_new_game))
	   @command_window.set_handler(:continue,
	                               method(:command_continue))
	   @command_window.set_handler(:options,
	                               method(:command_options))
	   @command_window.set_handler(:shutdown,
	                               method(:command_shutdown))
	end

	def command_new_game
	   DataManager.setup_new_game
	   close_command_window
	   fadeout_all
	   $game_map.autoplay
	   SceneManager.goto(Scene_Map)
	end

	def command_continue
	   close_command_window
	   SceneManager.call(Scene_Load)
	end

	def command_options
	   SceneManager.call(Scene_System)
	end

	def command_shutdown
	   close_command_window
	   fadeout_all
	   SceneManager.exit
	end
end
```

4.5 Instalace

Instalace hry i launcheru je velmi jednoduchá. Na rozdíl od ostatních komplexních her, Bloody Rain stačí pouze extrahovat z původního archivu. Hra je dostupná pomocí samorozbalovacího .exe souboru (vizte obr. 25), kde hráč pouze určí umístění hry nebo extrahováním archivu (.rar).



Obrázek 25 – Samorozbalovací archiv Bloody Rain

Díky launcheru může hráč přeskočit manuální instalaci. Uživatel pouze spustí launcher a ten se o stažení a extrakci veškerých souborů postará. Launcher není třeba jakýmkoliv způsobem instalovat. Veškeré soubory potřebné pro běh aplikace jsou obsaženy uvnitř launcheru a ten si je rozbalí dle potřeby, vizte zdrojový kód.

```
if (!File.Exists(Directory.GetCurrentDirectory() + "\\MetroFramework.dll"))
                File.WriteAllBytes(Directory.GetCurrentDirectory() +
                "\\MetroFramework.dll", Properties.Resources.MetroFramework);

if (!File.Exists(Directory.GetCurrentDirectory() + "\\RoundedCorners.dll"))
                File.WriteAllBytes(Directory.GetCurrentDirectory() +
                "\\RoundedCorners.dll", Properties.Resources.RoundedCorners);

if(!File.Exists(winSCPpath))
            File.WriteAllBytes(winSCPpath, Properties.Resources.WinSCP);
```

4.6 Ovládání hry

Základní ovládání hry se může zdá pro současnou moderní dobu netradiční a zastaralé. K pohybu se místo klasických kláves WSAD používají šipky. Stejně, jako tomu bylo u starších her z devadesátých let.

RPG Maker původně poskytoval ovládání pro konzolové ovladače a jeho novější verze přináší i částečný port pro klávesnici. Tento port umožňuje ovládání hry klávesnicí a poskytuje podporu pro několik kláves. Počet použitelných kláves na klávesnici je roven počtu kláves na standardním konzolovém ovladači. Proto klávesové zkratky některých funkcí hry jsou „zvláštně“ umístěny (například mapa na klávese W místo M). Mapa kláves je uvedena v tabulce 3.

Existuje řešení pro plnou podporu klávesnice, kde je třeba ručně rekonfigurovat jednotlivé klávesy a částečně tak modifikovat modul Input. Toto řešení je však komplexní a pro účel této hry naprosto zbytečné. Podpora malého množství kláves je v tomto případě dostačující.

Tabulka 3 – Tabulka ovládání

Akce	**Klávesa**
Pohyb doprava	→
Pohyb doleva	←
Pohyb nahoru	↑
Pohyb dolů	↓
Sprint	SHIFT
Interakce	Mezerník / Enter
Zrušení, Menu	ESC / X
Přeskočení dialogu	S
Questlog	Q
Mapa	W
Lucerna	E
Fullscreen	F5
Změna rozlišení	F6

ZÁVĚR

Výsledkem práce je dokončený herní launcheru a demo verze 2D hry pojmenované jako Bloody Rain. Při tvorbě bakalářské práce jsem si rozšířil znalosti zejména v oblasti herního vývoje. Dále jsem získal znalosti o programovacích jazycích C# a Ruby, práci s vlákny, jak navázat spojení FTP pomocí externích aplikací v C# a o stahování souborů po síti.

Herní launcher implementuje grafický framework MetroFramework a SFTP/FTP klienta WinSCP, který umožňuje připojení pomocí FTP na server. Díky grafickému frameworku vypadá herní launcher elegantně i přesto, že využívá starší WinForm design (na místo WPF).

Jednou z klíčových výzev byl způsob stažení souborů ze serveru. Možností bylo mnoho, ale většina z nich nevyhovovala zadaným požadavkům. Soubory šlo zabalit do archivu (rar nebo zip), ten ze serveru stáhnout a na straně klienta jej pouze rozbalit. Tento způsob však vyžadoval vždy stažení celé hry a ne jednotlivých částí. Ani stažení pomocí FTP spojení nevyhovovalo. WinSCP neposkytuje události, které by sdělovali počet aktuálně přenesených dat. Vzhledem k tomuto chybějícímu údaji nešel zobrazit aktuální průběh stahování. Východiskem byla .NET třída WebClient, která umožňuje sledování přenesených dat a asynchronní stahování souborů. Třída WebClient byla použita, protože poskytuje všechny potřebné události a informace o stahování.

Pro další potenciální vývoj by bylo možné lépe zabezpečit spojení mezi klientem a serverem. Přenos dat pomocí třídy WebClient není zabezpečený. Jedním z možných vylepšení je použití šifrovaného spojení.

Launcher je určen pro snadnou a bezstarostnou aktualizaci a stažení hry Bloody Rain. Ta byla vyvinuta pomocí herního enginu RPG Maker VX Ace. Tento herní engine má jistá omezení a vývoj v programovacím jazyce Ruby nebyl snadný. I přesto je to jeden z nejlepších herních enginů pro vývoj právě 2D RPG her.

Práci jsem splnil v celém rozsahu jak po stránce praktické, tak i po stránce teoretické. Do budoucna budu dále vyvíjet příběh a prostředí Bloody Rain.

Mám velké ambice a plány pro další vývoj hry. V současné chvíli obsahuje Bloody Rain okolo tří hodin hratelného herního času, 8 rozsáhlých map, 17 menších map (například obchody a budovy) a 23 úkolů, které hráč může při hraní splnit. Hráč během své cesty navštíví malé vesnice, lesní krajiny, pobřeží i jedno velké město.

POUŽITÁ LITERATURA

[1] Standard ECMA-334 *C# Language Specification* [online]. 4th Edition. ECMA International, 2006, 553 s. [cit. 2017-04-18]. Dostupné z: http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf

[2] SHARP, John. *Microsoft Visual C# 2010: krok za krokem*. Brno: Computer Press, 2010, 696 s. Krok za krokem (Computer Press). ISBN 978-80-251-3147-3.

[3] Introduction to the C# Language and the .NET Framework. *Microsoft Developer Network* [online]. July 20, 2015 [cit. 2017-04-18]. Dostupné z: https://msdn.microsoft.com/en-us/library/z1zx9t92.aspx

[4] GANESH, Arun. C# and its Features. *C# Corner* [online]. June 9, 2001 [cit. 2017-04-18]. Dostupné z: http://www.c-sharpcorner.com/article/C-Sharp-and-its-features/

[5] Windows Forms. *Microsoft Developer Network* [online]. ©2017 Microsoft [cit. 2017-04-18]. Dostupné z: https://msdn.microsoft.com/en-us/library/dd30h2yb%28v=vs.110%29.aspx

[6] C# - Multithreading. *Tutorialspoint* [online]. ©2017 Tutorials Point [cit. 2017-04-18]. Dostupné z: https://www.tutorialspoint.com/csharp/csharp_multithreading.htm

[7] About Ruby. *Ruby Programming Language* [online]. [b.r.] [cit. 2017-04-18]. Dostupné z: https://www.ruby-lang.org/en/about/

[8] FULTON, Hal a Jiří KOUTNÝ. *Ruby – kompendium znalostí pro začátečníky i profesionály*. Brno: Zoner press, 2009, 768 s. Encyklopedie Zoner Press. ISBN 978-80-7413-018-2.

[9] FLANAGAN, David a Yukihiro MATSUMOTO. *The Ruby Programming Language*. Sebastopol, CA: O'Reilly Media, 2008, 448 s. ISBN 978-0596516178.

[10] What is script? *Computer Hope* [online]. April 26, 2017 [cit. 2017-04-18]. Dostupné z: http://www.computerhope.com/jargon/s/script.htm

[11] ROUSE, Margaret. What is XML (Extensible Markup Language)? *Search Microservices* [online]. December, 2014 [cit. 2017-04-18]. Dostupné z: http://searchmicroservices.techtarget.com/definition/XML-Extensible-Markup-Language

[12] What is FTP and how can you upload files using it? *NTC Hosting* [online]. ©2002-2017 {ntc}hosting.com [cit. 2017-04-18]. Dostupné z: https://www.ntchosting.com/encyclopedia/ftp/file-transfer-protocol/

[13] YUANTAO. What is the difference between active and passive FTP? *Stack Overflow* [online]. December 10, 2013, 21:44 SEČ [cit. 2017-04-18]. Dostupné z: http://stackoverflow.com/questions/1699145/what-is-the-difference-between-active-and-passive-ftp

[14] PŘIKRYL, Martin. WinSCP documentation. *WinSCP* [online]. August 4, 2014 [cit. 2017-04-18]. Dostupné z: https://winscp.net/eng/docs/start

[15] CHRISTENSSON, P. Framework Definition. *Tech Terms* [online]. March 7, 2013 [cit. 2017-04-18]. Dostupné z: https://techterms.com/definition/framework

[16] MAJDA, David. Knihovny vs. frameworky. *David Majda* [online]. October 14, 2009, 12:14 SEČ [cit. 2017-04-18]. Dostupné z: https://majda.cz/blog/265

[17] MAGNO, Dennis. Metroframework-modern-ui. *GitHub* [online]. February 1, 2013 [cit. 2017-04-18]. Dostupné z: https://github.com/dennismagno/metroframework-modern-ui

[18] MAGNO, Dennis. MetroFramework - WinForms on steroids. *GitHub Pages* [online]. 2013 [cit. 2017-04-18]. Dostupné z: http://dennismagno.github.io/metroframework-modern-ui/

[19] THIEL, Jens. MetroFramework. *GitHub Pages* [online]. ©2013 Jens Thiel [cit. 2017-04-18]. Dostupné z: https://thielj.github.io/MetroFramework/

[20] Documentation for RPG Maker RGSS3. *RubyDoc* [online]. [b.r.] [cit. 2017-04-18]. Dostupné z: http://www.rubydoc.info/gems/rpg-maker-rgss3/1.02.0

[21] Game Library. *RPG-MAKER.FR* [online]. [b.r.] [cit. 2017-04-18]. Dostupné z: http://www.rpg-maker.fr/dl/monos/aide/vx/source/rgss/g_index.html

[22] KALDERON, Eyal. Game engines: What they are how they work. *Eyal Kalderon* [online]. May 9, 2011 [cit. 2017-04-18]. Dostupné z: https://nullpwd.wordpress.com/2011/05/09/game-engines-what-they-are-and-how-they-work/

[23] RPG Maker VX Ace. *RPG Maker Web* [online]. [cca. 2012] [cit. 2017-04-18]. Dostupné z: http://www.rpgmakerweb.com/products/programs/rpg-maker-vx-ace

[24] Run Time Packages. *RPG Maker Web* [online]. [cca. 2012] [cit. 2017-04-18]. Dostupné z: http://www.rpgmakerweb.com/download/additional/run-time-packages