Už zkopírováno do final změna od poslední verze

/*stará verze odstavce*/ //poznámka čistě autorský text nevycházející z žádného zdroje

Zapracování připomínek vedoucího zapracování připomínek ostatních

Trademark/copyright

zkratka	význam	vysvětlení	Viz. kap.
RPG	Role Playing Game	Herní žánr	4.1
OOP	Object Orienting Programming	Programovací paradigma	3
JVM	Java Virtual Machine	Virtuální prostředí pro Javu	3.2
JIT	Just In Time	Překlad v momentě spouštění	3.2 a 3.3
CTS	Common Type Specification	Jazyky kompilovatelné do MSIL	3.3
CLS	Common Language Specification		
CLR	Common Language Runtime		
CLI	Common Language Infrastructure		
MSIL	Microsoft Intermediate Language	.NET bytecode	3.3
NGen	Native Image Generator	Nástroj na správu nativních obrazů MSIL	3.3
NIC	Native Image Cache	Úložiště nativních obrazů MSIL	3.3
MAUI	Multi-platform App UI	Cross-platform .NET aplikace	
DRY	Don't Repeat Yourself	Minimalizace opakujících se částí kódu	
IDE	Integrated Development Environment	Vývojové prostředí (např. Visual Studio)	-
NPC	Non-Playable Character	Postava neovládaná hráčem	-
exp	Experience (point)	Body potřebné k zvýšení úrovně v RPG	-
JRPG	Japan RPG	RPG s předdefinovanou cestou	-
MMORPG	Massively Multiplayer Online RPG	Online RPG s tisíci hráči na stejném serveru	4.1
PvP	Player vs Player	Hráč proti hráči	-
PvE	Player vs Environment	Hráč proti monstrům ovládaným hrou	-
FPS	First Person Shooter	Herní žánr	4.2
TPS	Third Person Shooter	Střílečka z třetí osoby	4.2
RTS	Real Time Strategy	Herní žánr	4.3
UI	User Interface	Uživatelské rozhraní programu	-
GUI	Graphical User Interface	Grafické rozhraní programu (opak konzole)	-
CPU	Central Processing Unit	Procesor	-
GPU	Graphics Processing Unit	Grafický čip (externí i integrovaný)	-
WPF	Windows Presentation Foundation	Typ .NET okenní aplikace	5.2.2
UWP	Universal Windows Platform	Cross-platform aplikace pro Windows 10	5.3.1
XAML	eXtensible Application Markup Language	Značkovací jazyk pro definici GUI	5.2.2
MAUI	Multi-platform App UI	Cross-platform aplikace	5.3.3
DLL	Dynamic-link library	Knihovna obsahující již hotové třídy	
API	Application Programming Interface	Rozhraní usnadňující komunikaci aplikací a DLL	-
SDK	Software Development Kits	Sada nástrojů k vývoji pro určitou platformu	-

Obsah

1. Úvod

//Sem patří kromě historie i motivace autora k psaní práce, důležitost problému a představení strukturu práce.

Jelikož je vývoj her dlouhý a náročný proces je snaha ho co nejvíce urychlit, kvůli čemuž vznikají herní enginy, které za programátora řeší například fyziku a osvětlení. Ačkoliv je toto velká pomoc, stále je velká část kódu, která se s drobnými úpravami objevuje v každé RPG hře a programátor ji musí psát či kopírovat stále dokola.

2. Cíl práce a metodika

2.1 cíl práce

Cílem práce je popsat aktuální dostupné herní enginy, uživatelská rozhraní a programovací jazyky vhodné pro návrh vzorového řešení. Nejprve na návrhu aplikačního modelu popsat objektový model aplikace. Následně vytvořit návrh vzorového řešení, které bude univerzální v oblasti vývoje her typu RPG.

2.2 metodika

V první části budou zhodnoceny programovací jazyky, které je možné k realizaci využít a následně vybrán nejvhodnější. Dále bude následovat stručné seznámení s herními žánry a typy aplikací. U žánrů bude posouzeno, do jaké míry je možné pro jednotlivé žánry knihovnu využít a tím pádem, zda je třeba brát charakteristické prvky žánru v potaz při návrhu logiky.

V druhé části

//napíší až dám k praktické části něco dohromady

3. Výběr vhodných programovacích jazyků pro vývoj her

Programovací jazyky dělíme na dva základní skupiny. První jsou imperativní (např. C++), kam patří většina jazyků a jejich rysem je, že kód je sekvence instrukcí a je z něj čitelné co se v jaký okamžik bude provádět. Druhá skupina jsou deklarativní[1] (např. HTML), které říkají jen co se musí vyřešit, ale ne konkrétní instrukce potřebné k provedení a z toho důvodu často nejsou považovány za programovací jazyky, ale používá se pro ně označení kódovací. Další skupina jsou funkcionální (např. Haskell), které ačkoliv se řadí mezi deklarativní mají znaky obou skupin a je možné jejich přístup použít i v imperativních jazycích. Na rozdíl od imperativních nevyužívají žádné globální proměnné a vše je prováděno uvnitř funkcí[2]. Na Obr. 1 je porovnání sumy zapsané pomocí imperativního a funkcionálního jazyku.

Obr. 1 imperativní vs. funkcionální jazyk[2]

//vysvětlit paradigma?

Z popisu základních paradigmat je vidět, že jazyk bude vybírán z imperativních jazyků, které se dále dělí na dvě podskupiny. Procedurální[3] (např. C) pracují s funkcemi přijímajícími data pouze z parametrů nebo globálních proměnných. Pro svázání více souvisejících hodnot je možné použít strukturu, která je jako pole umožňující ukládat různé datové typy. Objektové (např. Java) mají třídy sloužící jako předlohy pro instance nazývané objekty, které stejně jako struktury mohou ukládat více hodnot různých typů, ale mají vlastní metody, a proto není potřeba všechna data předávat pomocí parametrů, protože si je může načíst z objektu kde se nachází. Objektově orientované programování (OOP) má čtyři základní principy: zapouzdření, abstrakce, dědičnost a polymorfismus. Zapouzdření umožňuje omezit viditelnost proměnných a metod mimo třídu, kontrolovat přístup k jejich hodnotám a ověřit, zda je zapisována platná hodnota. Abstrakce znamená, že pro práci s objektem není nutné znát vnitřní funkci jeho metod a při práci v týmu kolegovi stačí znát název, parametry a výstup metody. Použitím dědičnosti třída, která je potomek získá všechny proměnné a metody rodiče, ale je možné přidat nové, či změnit chování metody. Polymorfismus souvisí s dědičností, kde do proměnné typu rodič je možné vložit potomka, ale při volání metody se zavolá její přetížená verze, která má stejné jméno, typ a parametry, ale jiné tělo. Dále je možné přetěžovat metody změnou parametrů nebo návratové typu.[4] Na Obr. 2 je porovnání počítání obsahu čtverců a obdélníků napsané v procedurálním a objektovém jazyce (kvůli délce vynecháno zadávání hodnot). Je evidentní, že pro hry se nejvíce hodí objektové jazyky, a proto ty nejpoužívanější nyní budou probrány více do hloubky.

C

```c
int pocetCtvercu = 2;
Ctverec ctverce[2];
int pocetObdelniku = 2;
Obdelnik obdelniky[2];


for (int i = 0; i < pocetCtvercu; i++)
{
    printf("obsah ctverce=%d\n", obsah(&ctverce[i]));
}
for (int i = 0; i < pocetObdelniku; i++)
{
    printf("obsah obdelniku=%d\n", obsah2(&obdelniky[i]));
}
```

C++

```cpp
Ctverec *tvary[4];


for (size_t i = 0; i < size(tvary); i++)
{
    cout << tvary[i]->vypis()<<endl;
}
```

Obr. 2 procedurální vs objektový jazyk-vlastní

3.1 C++

C++ je více paradigmatový jazyk[5] rozšiřující jazyk C o objekty, nová klíčová slova a datové typy. Byla snaha zachovat co největší zpětnou kompatibilitu, pro usnadnění přechodu z C na C++ umožňující tvorbu komplexnějších programů[6], ale některé kódy možné napsat v C jsou v C++ neplatné[7]. Se zpětnou kompatibilitou souvisí headery obsahující deklarace proměnných, struktur, tříd a jejich metod, které je potřeba používat i v jiných souborech[8], což sebou ale nese i nevýhodu, že přidání nových tříd a metod, či změny jejich hlaviček je nutno provádět na dvou místech.

3.1.1 kompilace a hardware

Stejně jako jazyk C je kompilován pro konkrétní architekturu procesoru a operační systém, takže je nutno rozlišovat 32bitovou (označovanou jako x86) a 64bitovou verzi operačního systému (x86 dokáže běžet na x64 obráceně ne)[9, 10], ale existuje také C++/CLI, který je součást Microsoft .NET a je kompilován na bytecode (viz kap. 3.2), což umožňuje mít jednu verzi pro obě architektury a sestavit aplikaci z částí napsaných v různých .NET jazycích (viz Kap. 3.3)[11].

Tak jako C je i C++ díky své schopnosti pracovat přímo s pamětí a registry pomocí pointerů vhodný pro psaní ovladačů, operačních systémů a řízení jednočipových počítačů[12–14].

3.1.2 novinky oproti C

Mezi novinky, které C++ přináší patří *namespace*, které umožňují kód organizovat do menších celků a je tak možné, aby se v projektu vyskytoval stejný název vícekrát. Jakožto objektový jazyk dovoluje přetěžování metod, ale oproti Javě porovnává jen parametry, takže funkce s různým návratovým typem a stejnými parametry považuje za stejné a nepůjdou zkompilovat. Dále přibyli *Exceptions* sloužící jako zpráva o chybě ve volané metodě a umožňují tento problém vyřešit, aniž by došlo k pádu programu. Na rozdíl od Javy a C# se může jednat o libovolný datový typ[8].

3.1.3 nevýhody

standardy C++ neobsahují Garbage Collector, takže se programátor musí starat o alokování a následné uvolňování paměti sám, ale je možné použít některý vytvořený třetí stranou[15]. C++ neobsahuje vlastní GUI a musíte proto použít některou z knihoven třetí strany[16].

3.2 Java

Java je objektový jazyk, který byl vyvinut s myšlenkou, aby bylo možné jeden program spustit na všech systémech. Architektura vychází z jazyků jako Eiffel, SmallTalk a Objective C. Pro snazší přechod programátorů z C++ byla snaha zachovat co nejpodobnější syntaxi, ale jeho funkcionality použity nebyli.[17] oproti C a C++ se v Javě nenachází funkce, které existují samy o sobě a nenáleží žádné třídě, ale jen metody, které jsou součástí objektu, nebo jsou statické[18, 19].

3.2.1 JIT (Just In Time)

Oproti C++ není kód kompilován přímo do strojového kódu, ale do vysokoúrovňového platformě nezávislého kódu nazývaného bytecode, který je spouštěn ve virtuálním stroji (Java Virtual Machine neboli JVM), což umožňuje, aby stejný program bylo možné spustit na všech operačních systémech v 32bitové i 64bitové verzi[9], ale ke spuštění programu musí být na zařízení naistalována odpovídající verze JVM. Nevýhodou bytecodu je jeho výpočetní náročnost, neboť je překládán do strojového kódu v momentě, kdy je spouštěn. Díky just in time (JIT) překladu je ovšem možné provést optimalizaci pro konkrétní CPU a tím dosáhnout vyšší rychlosti, než jaké dosahují programy napsané například v C nebo C++ a zkompilované na počítači, který je starší než ten, kde je spouštěn.[20].

3.2.2 přístup k paměti a ovládání hardware

Java neumožňuje pracovat s pointery, neboť správu paměti zajišťuje run time[19]. Jelikož program nepřistupuje k paměti přímo je možné zajistit, že nebude zasahovat do paměti ostatních programů, což by mohlo způsobit pád systému či neoprávněný přístup k citlivým údajům[21]. Pomocí Java ME Embedded je možné ovládat i jednočipové počítače, ale je podporováno pouze Raspberry Pi Model B a dva čipy od STMicroelectronics[22].

3.2.3 výhody

Na rozdíl od C++ Java nepoužívá headery a pro použití třídy v jiném souboru stačí, aby se nacházely ve stejném *namespace*, nebo na příslušný namespace přidat referenci. Oproti C++ má Java Garbage Collector, který se stará o uvolňování paměti mazáním objektů bez reference, čímž usnadňuje programátorovi práci, ovšem za cenu občasného zastavení běhu aplikace, což je možné vyřešit přidáním dalšího vlákna[23]. Doba potřebná ke smazání „mrtvých“ objektů zaleží na počtu „živých“ a velikosti paměti[24, 25]. Java má pro GUI dvě knihovny, jimiž jsou *awt* a odlehčený *swing*[26, 27].

3.2.4 nevýhody

Stejně jako u C++ je zde možné využívat přetěžování metod, ale signaturu tvoří kromě parametrů i návratový typ, avšak oproti C++ a C# Java neumí přetěžovat operátory[19]. Další nevýhoda Javy je, že za generický typ, který se nejčastěji využívá u *Collection* (např. *ArrayList*) není možné dosadit primitivní datový typ, takže například pro přidání *int* do seznamu je třeba vytvořit nový objekt typu *Integer* s jeho hodnotou[28]. Java nemá datový typ pro bezznaménková celá čísla (*uint*)[19], takže je k dispozici pouze polovina rozsahu a pokud je potřeba zapsat hodnotu nad dvě miliardy (2^{31}) musí se použít *long* (64bitový).

3.3 C#

C# je plně objektový jazyk a hlavní zástupce rodiny Microsoft .NET, který spojuje to nejlepší z C++ a Javy. Ačkoliv vznikl původně pro Windows v posledních letech s přibývajícími frameworky postupně nahrazuje Javu ve vývoji mobilních aplikací (Xamarin a MAUI), PHP v back-endu (ASP .NET) a JavaScript na front-endu (Blazor) webových aplikací.

3.3.1 Microsoft .NET

Microsoft .NET je prostředí a rodina jazyků, které ho využívají. Tyto jazyky jsou vzájemně kompatibilní díky požadavkům na CTS (Common Type Specification), CLS (Common Language Specification), CLR (Common Language Runtime) a CLI (Common Language Infrastructure). Hlavní úlohou CLR je správa paměti a vláken. Mimo toho také kontroluje typovou bezpečnost. CTS zajišťují, že všechny jazyky mají stejnou definici datových typů a nemůže se tak stát, aby jednou byl *int* reprezentován třiceti dvěma bity a podruhé pouze šestnácti. Součástí těchto požadavků je, že veškeré referenční i hodnotové datové typy jsou potomky třídy *System.Object* a tím pádem jsou všechny .NET jazyky plně objektové. CLS zajišťuje, aby všechny jazyky byli kompilovatelné do bytecodu označovaného jako MSIL (Microsoft Intermediate Language), což umožňuje v jednom programu kombinovat knihovny napsané v C#, Visual Basic, F#, C++/CLI nebo jiném z více než dvaceti jazyků [11, 29, 30].

MSIL je objektový nízko úrovňový jazyk, který tak jako většinu bytecode je možné kompilovat v režimu JIT (Just In Time), ale navíc také podporuje AOT (Ahead Of Time), kdy se výsledný soubor chová podobně, jako v případě C++, a je tedy nutné ho sestavit pro každý systém a architekturu, kde chceme program spouštět[30]. Výhodou předem zkompilované aplikace je rychlejší start a pro složitější programy i výrazný nárůst výkonu, ovšem za cenu většího souboru, neboť obsahuje také MSIL, který je v některých případech potřeba[31]. při generování AOT jsou využívány nástroje NGen (Native Image Generator) pro .NET Framework a Crossgen2 pro .NET Core. Výstupy těchto nástrojů se nazývají nativní obrazy a jsou instalovány do NIC (Native Image Cache), kam jsou přidávány i závislosti, které je možno používat více obrazy, čímž se eliminuje duplicita. Kompilaci je možné spustit na počítači programátora, nebo až při instalaci programu. Vytvoření obrazu u uživatele má výhodu, že kód bude optimalizován pro jeho procesor a bude tak dosahovat nejvyššího možného výkonu. [11, 32, 33] Další výhodou AOT je, že není potřeba, aby byl překlad co nejrychlejší, takže má dost času provést optimalizace[34].

Velkou výhodou je, že .NET runtime je od Windows Vista součást operačního systému, takže je aktualizován společně se systémem[35], díky čemuž uživatel nemusí nic instalovat. Prostředí .NET bylo původně určeno pouze pro platformy Microsoftu (Windows a Xbox), což se změnilo až v roce 2014 vydáním .NET Core[30], ovšem s GUI pro ostatní systémy se vývojáři museli spoléhat na třetí strany. V roce 2022 bylo vydáno .NET MAUI umožňující vytvořit jednu aplikaci na Windows, Android, iOS a macOS s minimálními zásahy do kódu.[36]

3.3.2 přístup k paměti a ovládání hardware

Na rozdíl od Javy je v C# možné využívat i pointery a obcházet tak správce paměti, což může vylepšit výkon, ale současně vést k bezpečnostním problémům a nestabilitě, kvůli čemuž není možné ověřit bezpečnost a takovýto kód musí být umístěn do bloku vyznačeného pomocí preprocesorů *unsafe*. Kód uvnitř toho bloku se podobá tomu, který by se napsal v C++ nebo C[37]. Ačkoliv C# oficiálně neumožňuje ovládání jednočipových počítačů, existují rozšíření třetích stran, jako například nanoFramework nebo placené visualmicro, které podporují čipy založené na ARM architektuře[38, 39].

3.3.3 porovnání s Javou

Stejně jako u Javy je zde viditelnost tříd řízena pomocí *namespace*. Při přetěžování metod je signatura dána typem a pořadím parametrů, ale oproti Javě

umí přetěžovat i operátory[40]. C# dokáže primitivní datové typy (např. *int*) automaticky měnit na objekty[29]. Tak jako Java i C# má Garbage Collector, který za programátora uvolňuje paměť. K jeho spuštění dochází při nedostatku paměti, nebo překročení stanoveného limitu[41, 42]. Ačkoliv C# v některých situacích vyžaduje oproti Javě další klíčová slova, čímž působí jako pomalejší na psaní, snižuje se tím množství chyb a urychluje orientaci v kódu, protože je na první pohled vidět přetěžování při dědičnosti a použité modifikátory.

3.3.4 podobnosti s C++

Tak jako C++ má i C# struktury, které by se daly označit jako hodnotová verze objektu, ale mají omezené možnosti. Například nemohou mít hodnotu *null*, používat dědičnost a mít proměnné inicializované při deklaraci[37].

Podobně jako má C++ pointery na funkce, v C# jsou využíváni delegáti, kteří slouží k předávání metod v parametru, nebo umožňují dynamicky měnit volanou funkci. Delegáty je možné sloučit do *MulticastDelegate*, který obsahuje jejich seznam a při volání je postupně provádí[43–45]. Další jejich využití jsou eventy (např. kliknutí na tlačítko), kde metody, které na něj reagují, musí být typu *void* a mít parametry typu Object a *EventArgs* nebo jeho potomka. První parametr říká, jaký objekt event vyvolal a druhý obsahuje podrobnosti, jako například jaká je poloha kurzoru[46, 47].

3.3.5 modifikátory parametrů metod

U parametru metody je možné použít klíčové slovo *out*, které ho změní na výstupní hodnotu, což umožňuje vracet více než jednu hodnotu bez nutnosti použít pole objektů, ze kterého by se poté postupně přiřazovaly do příslušných proměnných, nebo vracet bool, pokud metoda proběhla úspěšně, a tuto hodnotu předávat výstupním parametrem. Dále je možné využít modifikátory *ref*, který mění hodnotovou proměnnou na referenční, a *in*, který brání úpravám hodnoty[48].

3.3.6 nové funkce

Mezi novinky, které C# přináší patří *properties*, umožňující zabalit *get* a *set* pod jeden název, se kterým se při volání pracuje jako by se jednalo o proměnnou[49]. Další nová funkce je modifikátor *partial* umožňující rozdělit definici třídy, struktury nebo interface na více částí, které mohou být i ve více souborech. Pomocí této funkce se dá zvýšit přehlednost velkých tříd rozdělením na menší logické celky a zjednodušuje tak práci u týmových projektů, kde každý programátor může pracovat na své části, aniž by omezoval kolegu. Dále se této možnosti využívá při generování části třídy, aniž by ovlivnila programátorův soubor. Příkladem je Windows Form, jehož grafická část je generována Visual Studiem[50].

3.4 výběr

V Tab. 1 je přehled vlastností porovnávaných jazyků, kde zeleně jsou označeny výhody, červeně nevýhody a žlutě body, kde záleží na situaci. Například když je potřeba maximální rychlost, může být Garbage Collector nevýhodou, ale zajišťuje že program nebude v paměti nechávat data bez reference a spotřebovávat tak zbytečně více paměti, než potřebuje. Z porovnání je vidět, že C# umožňuje snazší

implementaci knihovny, jelikož není potřeba importovat header pro každou použitou třídu a má snazší práci s eventy.

Jelikož jsou všechny 3 porovnávané jazyky objektové nebude mít volba na návrh logiky výrazný vliv a rozdíly budou jen v komunikaci objektů mezi sebou (event). Rozdíly se projeví při realizaci (např. práce s pamětí, přetěžování nebo generika) a implementaci, protože každá technologie podporuje jen některé jazyky.

Jelikož Java není podporována žádným z hlavních engynů, tak je pro univerzální knihovnu nevhodná.

Kvůli kompilaci do MSIL, který umožňuje sestavit program z kódů napsaných ve více jazycích, byl jako jazyk, pro který bude knihovna navrhována C#. Takto bude knihovna univerzálnější, neboť její implementace nebude omezena na jeden jazyk.

	C++	Java	C#
Byte code	Ne (C++/CLI ano)	ano	Ano (může být přeložen)
Headery	ano	ne	ne
Garbage Collector	Ne (možno použít třetí strany)[15]	ano	ano
Přetěžování operátorů	ano	ne	ano
Vlastní eventy	ano[51]	Vlastní řešení	ano
Properties	ne	ne	ano
In/out/ref	pointery	ne	ano
Maximální počet rodičovských tříd	neomezen	1	1
uint	ano	ne	ano
struct	ano	ne	ano
MSIL	jen C++/CLI	ne	ano
Příklad Herních enginů	Unreal, CRYENGINE	Greenfoot, libGDX[52]	Unity, CRYENGINE

Tab. 1 porovnání jazyků

4. Výběr herních žánrů vhodných pro implementaci

Jelikož je cílem praktické části navrhnout knihovnu pro tvorbu her, je třeba se podívat na základní žánry a zhodnotit, jak velkou část knihovny bude možné využít. Hry dělíme na několik žánrů (RPG, akční, strategie, závodní), podle jejich mechanik (např. vylepšování postavy, inventář, střelba, přeskakování mezi plošinami, stavba budov nebo řízení jednotek), což usnadňuje hráčům orientaci při výběru, jelikož mají základní představu, jaký zážitek od titulu očekávat. Často se stává, že hra spojuje více žánrů a je proto těžké ji jednoznačně zařadit. Další komplikací je nejednotné dělení subžánrů (především u RPG a akčních her), kvůli čemuž se můžete setkat s tím, že hra je na různých stránkách označena jinými štítky (viz Tab. 2). [53]

hra	steam	Epic games	Ubisoft	Alza
Assassins Creed Odyssey	S otevřeným světem, RPG, S asasíny, Akční	Akční, RPG	akční, adventura	akční, adventura, dobrodružná
Baldur's Gate II	RPG, Klasické, Fantasy, Dungeons & Dragons	-	-	RPG, Strategie
Heroes of Might & Magic III	Strategické, Klasické, Tahové strategie, Fantasy	RPG, strategie	strategie, RTS	Akční, Strategie

Tab. 2 odlišné značení her[54–57][58, 59][60–63]

//více technicky

4.1 RPG

Tento žánr vychází z deskových her jako jsou Dungeons & Dragons[64] (zkráceně D&D nebo DnD) nebo české Dračí doupě, kde hráč nebo skupina hráčů hraje za postavy, které mají různé rasy (např. člověk, elf nebo trpaslík), třídy (např. válečník, mág nebo zloděj), inventář, statistiky a schopnosti. Většinou plní úkoly zadané NPC, za což dostanou odměnu v podobě zkušeností, peněz a předmětů. Hlavním znakem je získávání zkušenostních bodů (exp) a vylepšování postav. Nejčastěji je hra zasazena do fantasy světa. Většinou mají RPG propracovanější a delší příběh než ostatní žánry. V deskové verzi se hází kostkami, zda se akce podařila, což je v digitální podobě nahrazeno generátorem náhodných čísel, případně je náhoda vynechána a výsledek záleží na schopnostech hráče a statistikách postavy (např. přesnost luku určuje hráčova práce s myší a jak dobře postava umí luk používat) nebo je akce vždy úspěšná. RPG má několik subžánrů, které ne všechny stránky a obchody rozlišují a z RPG samostatně vyčleňují jen některé (viz Tab. 2).

První a nejstarší kategorie je již zmíněná digitalizovaná verze deskových her, či jejich slovní podoby (např. Baldur's Gate[65]), která je na tahy a hráč má čas promyslet si své další kroky. Původně byly pouze textové, kde hráč vybíral z možností a později se začaly objevovat i grafické verze.

Druhá a dnes nejrozšířenější kategorie jsou akční RPG (ARPG) odehrávající se v reálném čase a hráč musí rychle reagovat na akce nepřátel. Hráč hraje za jednu postavu, která v některých titulech může mít společníky, nad kterými hráč nemá kontrolu (např. The Elder Scrolls V: Skyrim[66]) nebo může přepnout ovládání ze své postavy na společníka (např. Star Wars: Knights of the Old Republic[67]). Tyto hry mají často otevřený svět, což umožňuje hráči volně se pohybovat po celé mapě, prozkoumávat ji a plnit hlavní i vedlejší úkoly v libovolném pořadí. Většinou hráč svými rozhodnutími v dialozích ovlivní příběh, nebo jak na něj ostatní postavy reagují. Hry využívající tyto dva prvky se označují jako nelineární, nebo západní RPG, což je opak JRPG (Japan RPG), které jsou lineární a kladou důraz hlavně na vyprávění předem daného příběhu (např. Final Fantasy[68]).

Další kategorie je MMORPG (Massively Multiplayer Online RPG), kde na velkých mapách hraje současně tisíce hráčů, kteří mohou navzájem komunikovat, pomáhat si, obchodovat, bojovat v PvP (Player vs Player) arénách a spojovat se do aliancí (např. World of Warcraft[69]).

Poslední jsou taktická RPG (TRPG), která kombinují vylepšování postav a strategii (např. XCOM[70]). Tento subžánr je často řazen mezi strategie. [53, 71–74][75] Především na mobilních zařízeních je možné se setkat s TRPG, kde hráč sbírá postavy s různými schopnostmi, které vylepšuje a následně v pětičlenných týmech

bojuje proti jiným týmům, kde jich je v kampani v rámci jedné úrovně více za sebou (např. Star Wars: Galaxy of Heroes[76]).

4.2 akční

Akční hry je souhrnné označení bojových her, střílečekplošinovek a ostatních her s rychlým tempem, které nemají vlastní žánr. [77] V bojových hrách si hráč typicky volí postavu ze seznamu a následně s ní bojujete proti soupeři (např. Mortal Kombat[78]). Ačkoliv se střílečky řadí mezi akční hry, většinou jsou brány jako samostatný žánr. Střílečky dělíme podle umístění kamery na FPS (First Person Shooter), která je z pohledu postavy (např. Doom[79]) a TPS (Third Person Shooter), kde je v záběru kamery i postava (např. Mafia[80]). Pod pojmem plošinovka si většina lidí představí 2D hry jako Mario[81], kde se hráč pohybuje po platformách a musí se dostat na konec úrovně. Patří sem i akčnější tituly jako Tomb Raider[82] a Prince of Persia[83], které obsahují i soubojový systém se zbraněmi.[84, 85]

4.3 strategie

Podobně jako RPG, mají i strategie předlohu ve stolních hrách (např. Warhammer 40 000[86]). Existují dva způsoby členění. První je podle toho, zda je hra na kola (tahová neboli turn-based) a hráči se střídají (např. Civilization V[87]), nebo vše probíhá v reálném čase (RTS) jako například Age of Empires[88]. Druhé je podle zaměření, jako jsou například válečné či budovatelské. Když se řekne strategie většina lidí si vybaví válečnou, kde hráč těží suroviny na stavbu budov a výrobu jednotek, kterými se snaží porazit protivníka. V budovatelských se hráč stává starostou města či manažerem (např. zábavního parku, přepravní společnosti či nemocnice) a jeho úkolem je vyřešit logistiku, zajistit zisk na další rozvoj a starat se o spokojenost lidí (např. Cities: Skylines[89] a RollerCoaster Tycoon[90]).[72, 74, 91]

4.4 závodní

Jedním z možných dělení závodních her je podle jízdního modelu. Realistický (např. DiRT Rally[92]) se snaží co nejvíce napodobit chování skutečných vozů. Arkádový (např. Need For Speed[93]) ubírá realističnost ve prospěch jednodušší hratelnosti. Dalším dělením je, zda se závodí v autech či jiných dopravních prostředcích (např. vesmírné lodě v Redout 2[94]). Ačkoliv většina her hráče za narážení do soupeřů penalizuje, existují také série jako například FlatOut[95] a Asphalt[96], které za to naopak hráče odměňují a umožňují soupeře zpomalit či úplně vyřadit ze závodu [74, 97].

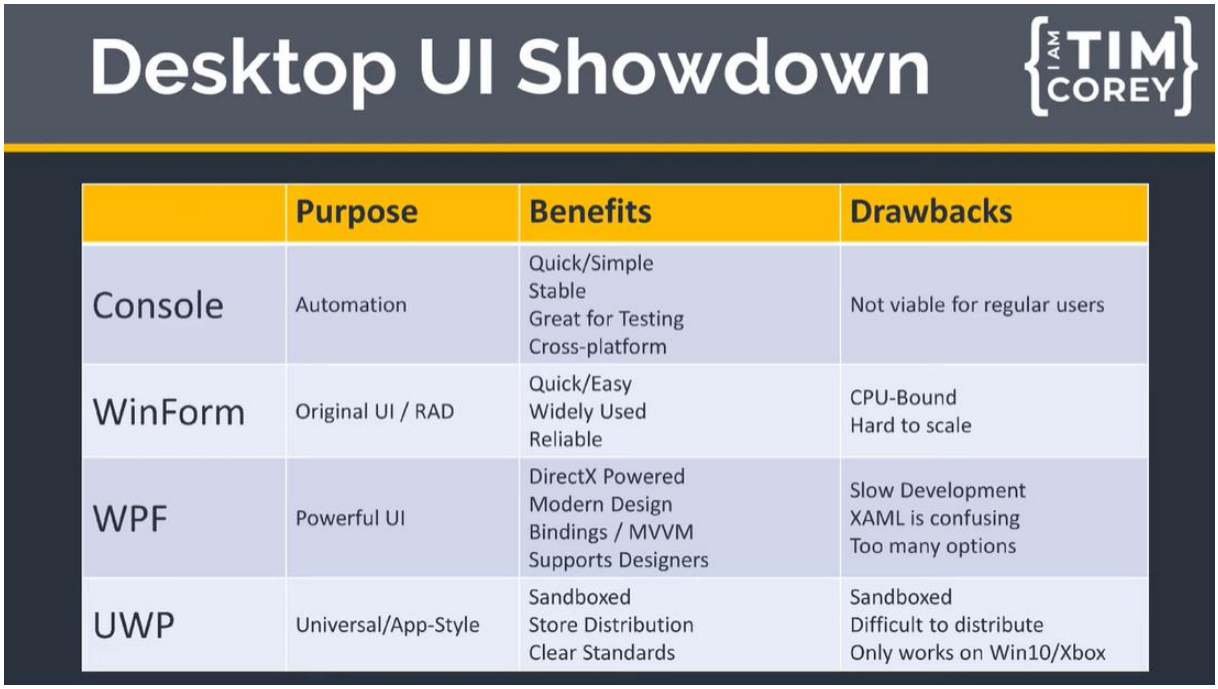
4.5 shrnutí

Toto bylo jen stručné seznámení s nejvíce zastoupenými žánry, ale existuje mnoho dalších, které vnikají například kombinací, či rozšiřováním těchto základních. Jelikož je tato práce zaměřena na RPG, je knihovna plně implementovatelná a neměla by být žádná část, která není pokryta a programátor musí dělat pouze grafickou část, reakce na Eventy a vstupy od hráče, nastavení hodnot a umělou inteligenci NPC. Jelikož většina RPG je současně akční hrou je i pro tento žánr možné využít téměř celou knihovnu bez výrazných úprav logiky. Výjimkou jsou akční hry mezi jejich mechaniky nepatří souboj (např. Blackhole[98]) nebo neobsahují předměty, protože v takovém případě zůstane velká část knihovny nevyužita. Pro strategie je z knihovny možné využít třídu *Postava* na jednotky a budovy nebo *GameManager* a *Chunk* na

generování náhodné mapy. U závodních her lze z knihovny využít *StatList* na vlastnosti vozidla (rychlost, akcelerace, …) a *Predmet* na vylepšení. V přídě her, kde se dá poškodit soupeřův vůz lze použít také *Postava*. Jelikož třídy nejsou určeny na toto využití, obsahují velké množství dat na víc, která budou zbytečně zabírat paměť a je proto lepší vytvořit si vlastní třídy.

5. Grafické výstupy aplikací

Programy ke komunikaci s uživateli potřebují uživatelské rozhraní. U nejjednodušších her se může jednat o konzoli, ale pro většinu her pouhé psaní nestačí a potřebují grafické prostředí neboli GUI. Na Obr. 3 jsou v několika bodech shrnuté rozdíly UI pro .NET, které jsou níže rozvinuty.



Obr. 3 srovnání Windows UI[99]

5.1 konzolová aplikace

Nejjednodušší UI je konzole, která jako vstup a výstup používá příkazový řádek. Dnes se s ní běžný uživatel obvykle nesetká, protože kvůli textovým vstupům není tak příjemná na ovládání jako GUI. Využívá se především pro automatizované úkoly a na programy s důrazem na co nejnižší zátěž hardwaru využívané administrátory. Je možné mít aplikaci s GUI, která spouští konzolové aplikace a následně zobrazuje jejich návratovou hodnotu.[99]

5.2 okenní aplikace

Dnes se většinou setkáváme s programy s GUI, které mají tlačítka, textová pole, rozbalovací menu a další grafické prvky. Takovéto programy nazýváme okenní aplikace. Jelikož byl jako jazyk zvolen C#, budou nyní blíže rozebrány typy okenních aplikací, které .NET nabízí.

5.2.1 WinForm

Windows Forms Application neboli WinForm je původní .NET GUI, které je dostupné pouze na Windows. Ačkoliv se jedná o zastaralou

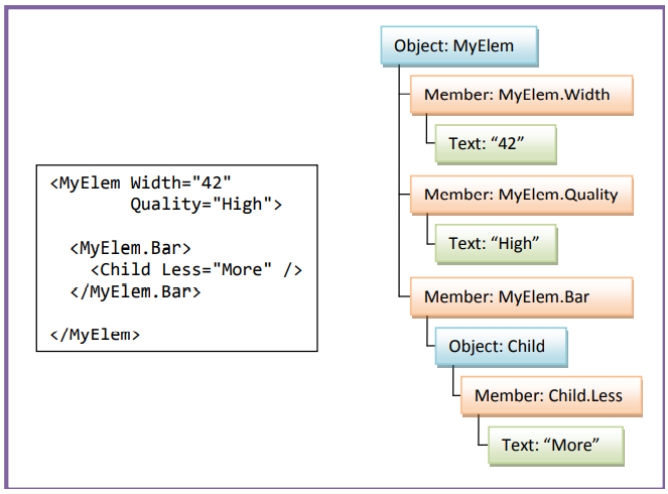
technologii, je stále využívána kvůli svému rychlému a jednoduchému vývoji. Vykreslování zde provádí pouze CPU, což má za následek pomalejší změny a vyšší výpočetní náročnost. Nepotřebuje grafické ovladače a okno je viditelné i při použití vzdálené plochy, kterou není možné přenášet obraz vykreslovaný pomocí GPU. Výraznou nevýhodou jsou absolutní rozměry a poloha prvků, které zůstávají při změně velikosti okna neměnné[99].

5.2.2 WPF

Windows Presentation Foundation zkráceně WPF je modernější typ .NET okenní aplikace využívající DirectX[100], což umožňuje okno vykreslovat pomocí GPU. GUI je definováno pomocí jazyku XAML, který nabízí větší možnosti a umožňuje designerovi vytvořit vzhled aplikace, aniž by uměl programovat. Nevýhoda WPF je nutnost naučit se 2 zcela odlišné jazyky a potřeba velkého množství řádků v XAML k nastavení vzhledu což zpomaluje vývoj.[99]

XAML (eXtensible Application Markup Language) je deklarativní jazyk vycházející z XML, ale je navržen k reprezentaci objektů v OOP. XAML je nejčastěji uložen jako XML dokument. Seznam klíčových slov není pevně daný, ale záleží na implementaci. Na rozdíl od XML, XAML nevyžaduje uložení jako formátovaný text, ale je možné použít jakýkoliv reprezentaci s odpovídající logikou struktury (např. binární soubor, nebo objekty v paměti). Na Obr. 4 je vidět příklad textové podoby XAML a její OOP reprezentace.[101]

//použito info ze specifikace str7-8



Obr. 4 příklad reprezentace XAML[101]

5.3 mobilní aplikace

Jelikož je trh s mobilními aplikacemi srovnatelný s těmi počítačovými, nesmí být opomenut. Oproti aplikacím určeným pro počítače, mobilní operační systémy vyžadují, aby se aplikace stahovaly pomocí obchodů, kde je možné ověřit jejich bezpečnost. Pro vývojáře má toto řešení výhodu snazších

aktualizací, neboť obchody umožňují automatické aktualizování. Nevýhodou je nutnost získat od uživatele oprávnění k některým funkcím (např. fotoaparát a přístup k souborům).

5.3.1 Xamarin

Xamarin je open source technologie od Microsoftu umožňující pomocí C# a XAML vytvořit aplikace pro zařízení se systémy Android, Windows a systémy od Apple (např. iOS a macOS)[102]. Aplikaci je možné vytvořit nativně pro konkrétní platformu nebo cross-platform pomocí *Xamarin.Forms*[102]. API *Xamarin.Essentials* umožňuje pracovat s funkcemi jako jsou poloha a fotoaparát nezávisle na cílové platformě[103]. Grafické rozhraní je možné místo v XAML vytvořit pomocí C#[104], ale takto vytvořené či upravené prvky se nezobrazí v náhledu a projeví se až po spuštění aplikace. Xamarin podporuje velké množství knihoven přímo od Microsoftu, nebo dalších společností zapojených do projektu .NET Foundation. Mezi tyto knihovny patří například SkiaSharp pro 2D grafiku nebo herní enginy MonoGames (založený na Microsoft XNA)[105] a Stride (dříve Xenko)[106]. Kromě knihoven určených pro .NET je možné využít i knihovny napsané Objective-C, C/C++ nebo Java[107]. součástí SDK je i emulátor umožňující otestovat aplikaci na různých konfiguracích a bez nutnosti neustálého nahrávání nové verze do fyzického zařízení. Mezi výstupy Xamarin patří UWP (Universal Windows Platform), což je aplikace spustitelná na všech zařízeních s Windows 10 (PC, Xbox, telefon a další)[108]

5.3.2 Android Studio

Android studio je oficiální IDE a SDK pro vývoj aplikací pro Android od Googlu[109]. K dispozici jsou jazyky Java a Kotlin. Podobně jako v případě Xamarin se k definici grafického rozhraní využívá značkovací jazyk, kterým je XML[110]. Hlavní nevýhodou Android Studia je možnost vývoje pouze pro Android a pokud chce vývojář vytvořit aplikaci i pro jinou platformu musí vytvořit projekt v jiném IDE, podporující vývoj pro tuto platformu a spravovat více projektů současně. Stejně jako Xamarin disponuje Android Studio emulátor umožňující testovat aplikaci na počítači[111].

5.3.3 .NET MAUI (Multi-platform App UI)

Jedná se o nástupce Xamarin umožňující vytvořit jeden projekt společný pro Windows, Android, iOS a macOS. Původně bylo datum vydání oznámeno na září 2021[36], ale kvůli technickým problémům bylo odloženo na leden 2022 a nakonec po dalších odkladech[112] bylo oficiálně vydáno jako součást Visual Studio 2022 verze 17.3 v srpnu 2022[113]. Avšak i po vydání trpělo mnoha nedostatky, které byli opravovány aktualizacemi vycházejících v krátkých intervalech. Dále v době psaní této práce mnoho knihoven pro Xamarin nemělo verzi pro MAUI, či podporovali jen některé platformy. Mezi nevýhody patří velikost, protože jednoduchá Android aplikace napsaná v Xamarin má kolem deseti megabytů, zatímco MAUI verze má kolem sta megabytů.

5.4 herní engine

//lepší název?

Oproti výše popisovaným typům aplikací, které jsou vytvořeny pro obecné aplikace, jsou herní enginy specializovány na hry a aplikace s podobnými požadavky (např. virtuální realita). Ačkoliv se od sebe jednotlivé enginy liší, všechny obsahují podobné podsystémy, které řeší fyziku, kolize, světlo a renderování výsledného obrazu[114]. Některé společnosti si často pro své hry vytváří vlastní engine, který je využíván pouze interně (např. Ubisoft a Naughty Dog), nebo je většinou za poplatek poskytnut i ostatním vývojářům (např. CRYENGINE od Crytek), čímž vzniká vedlejší zdroj příjmů[114]. Také existují společnosti, kde je engine hlavním produktem nikoliv vedlejším (např. Unity). Enginy jsou často cíleny pro konkrétní žánr a jeho využití pro

zcela odlišný žánr je obtížné či nemožné.[114] Renderování je téměř vždy prováděno pomocí GPU.

Při výběru enginu je nejprve potřeba zvolit si jaký jazyk bude použit a vzít v úvahu potencionální velikost zisků, kvůli limitu do kdy je licence zdarma a v případě Unity i její ceně. Nyní se podíváme na 3 nejpoužívanější enginy jejichž licence je zdarma, případně se odvíjí od zisku.

//

5.4.1 Unity

Unity 3D později přejmenováno jen na Unity od společnosti Unity Technologies je jedním ze dvou největších herních enginů, který pohání přes 50 % her na více než 20 platformách včetně webu. Na mobilních zařízeních z 1000 nejstahovanějších her je 71 % vytvořeno právě v Unity[115]. Kromě her je možné Unity využít i v průmyslu k vizualizaci modelů vytvořených v CAD programech ve virtuální nebo rozšířené realitě[116]. Dále je Unity využíváno v architektuře, kde usnadňuje návrh budov[117] nebo u tvorby filmů[118]. Pokud zisk za předešlých 12 měsíců nepřekročil 100 000 amerických dolarů, je licence zcela zdarma, poté se platí měsíční nebo roční poplatek za každý počítač, jehož výše se odvíjejí od zisku[119]. K programování v Unity se využívají jazyky C#, JavaScript nebo Boo (vycházející z Pythonu), případně je možné použít nástroj Bolt, který umožňuje programovat pomocí bloků[120]. Modely je možné vytvořit vlastní například v programu Blender, stáhnout z internetu, nebo použít Asset Store integrovaný přímo v prostředí Unity, kde jsou modely, efekty, scripty a nástroje, které jsou zdarma nebo placené. Pro usnadnění vývoje unity umožňuje vytvořit i vlastní nástroje, kterými můžete za běhu měnit hodnoty a v reálném čase vidíte změny[121]. Příkladem her vytvořených v Unity jsou Cuphead a Hollow Knight[122]. Oproti konkurenci má výhodu v jednoduchosti, množství platforem a 2D hrách[123].

Úrovně hry jsou v Unity nazývány scény. Každá scéna obsahuje seznam *GameObjectů*, které se v ní nachází. *GameObject* má své komponenty, které určují jeho vlastnosti (např. vzhled, umístění a gravitace). Povinný komponent je *Transform*, který obsahuje informaci o poloze, natočení a velikosti vůči původnímu modelu. Jednotlivé komponenty jsou reprezentovány inspektory, ve kterých je možné pohodlně měnit jejich hodnoty. *GameObjectu* je možné přidat potomky, jejichž poloha se nyní nevztahuje ke scéně, ale k rodiči. Toto umožňuje vytvářet skupiny objektů, které se pohybují jako jeden celek, čehož lze využít například když má postava v ruce předmět. Z *GameObjectu* je možné vytvořit *Prefab*, který obsahuje všechny jeho potomky a komponenty včetně nastavených hodnot. Takto je možné usnadnit vývoj, neboť již připravený objekt stačí pouze přidat do scény a nastavit polohu. Dále se změny provádí na jednom místě a samy se změní ve všech instancích. Logika se objektům přidává pomocí komponent script, což jsou zdrojové kódy obsahující třídu, která by mezi svými předky měla mít *MonoBehaviour*, čímž zdědí metody jako jsou *Start, Update a FixedUpdate*, které Unity volá v konkrétních situacích. *Start* je zavolán po vytvoření instance a obvykle se zde řeší načítání hodnot z managera či komponent. Update je volán při každém vykreslení nového snímku, jejichž počet za vteřinu není stabilní. Pro výpočet fyziky a ostatní úkony vyžadující konstantní intervaly se využívá *FixedUpdate*, který je defaultně volán padesátkrát za vteřinu[124]. Na práci s časem má Unity třídu Time, kde pomocí hodnoty *timeScale* je možné změnit kolikanásobkem standartní rychlosti hra poběží a pomocí *deltaTime* je možné zjistit kolik času uplynulo mezi jednotlivými snímky. Ve skriptech je možné využívat statické třídy *Debug* (konzole a čáry) a *Gizmo* (geometrické tvary), které umožňují do scény dokreslovat pomocnou grafiku, kterou hráč neuvidí[125][126]. Pro zvýšení přehlednosti inspektora skriptu je možné pomocí klíčových slov

umístěných do hranatých závorek možné přidat nadpisy nebo *slidery* a skrýt či zobrazit proměnné. Pro výpočet natočení objektu Unity využívá *Quaternion* (čtyřrozměrný vektor) [127, 128]. Objekty jsou umístěny do různých vrstev (*Layer*), podle kterých je možné omezit, zda bude vykreslen, se kterými objekty bude vyhodnocována kolize, či ho filtrovat ve skriptech. Skripty umožňují načítat komponenty jak objektu, ve kterém se nachází, tak i jeho potomků a rodičů. Na správu objektů se využívají skripty označované jako manageři, které se nachází v prázdných objektech (neobsahují nic jiného než skript) a je vhodné na ně využít návrhový vzor *singleton*, aby se zabránilo situaci, kde je omylem vytvořena více než jedna instance. Unity disponuje vlastní umělou inteligencí pro hledání cesty (pathfinding) využívající *NavMash*, což je mřížka určující kudy je možné projít.[129, 130]

5.4.2 Unreal Engine

Unreal Engine od společnost Epic Games je druhým největším enginem, který je na rozdíl od Unity orientován především na AAA tituly a dosahuje realističtější grafiky. Stejně jako Unity se využívá i v průmyslu architektuře a filmovém průmyslu. Unreal se používá například při natáčení Star Wars seriálu The Mandalorian[131, 132]. Pokud zisk produktu nepřekročí 1 000 000 amerických dolarů je licence zdarma, poté se platí 5% podíl z prodeje. Jako programovací jazyk se využívá C++ nebo nástroj Blueprint pro blokové programování. Příkladem her vytvořených v Unreal Engine jsou Fornite, který je stejně jako engine od Epic Games a série Borderlands[133].

V Unreal se o správu všech zdrojů (např. modely a textury) stará UnrealEd, který dokáže zobrazit model přesně tak, jak bude vidět ve hře a vše je unifikováno. Zdroje jsou přidávány do databáze, což usnadňuje vyhledávání a umožňuje ověřit validitu a zabránit tak chybám. UnrealEd data ukládá do balíčků reprezentovaných binárními soubory, což ovšem znamená, že není možné využít verzovací systémy (např. GitHub) a jeden balíček může současně editovat jen jeden uživatel.[114]

5.4.3 CRYENGINE

CRYENGINE je společnosti Crytek, kterou proslavila herní série Crysis na níž předvádí schopnosti svého enginu. Stejně jako Unreal Engine je orientován především na AAA tituly, ale není tolik využíván. Na rozdíl od zbylých 2 porovnávaných enginů není využíván v průmyslu a podporuje méně platforem. U prvních 5 000 amerických dolarů ze zisku za rok je poplatek odpuštěn ze zbytku se platí 5% podíl.[134] Jako programovací jazyk je možné použít C++ nebo C#.

5.4.4 shrnutí

V Tab. 3 je stručně sepsán obsah kapitol 5.3.1-5.3.3 a vypsány nejdůležitější platformy. Jelikož byl jako jazyk zvolen C#, je knihovnu možné použít pouze v Unity a CRYENGINE. Velkou výhodou Unity je počet podporovaných platforem. Pokud hra vydělá za posledních dvanáct měsíců sto tisíc amerických dolarů je cena licence Unity tři sta devadesát devět dolarů za každý počítač a čtyři tisíce sedm set padesát dolarů za CRYENGINE (z prvních pěti tisíc se neplatí). Z tohoto příkladu je vidět, že se Unity vyplatí především menším studiím, protože za cenu licence CRYENGINE budou mít licenci na 12 počítačů. Je však třeba vzít v úvahu, že v případě Unity se na rozdíl od CRYENGINE příjem nepočítá pro každou hru zvlášť, ale za všechny.

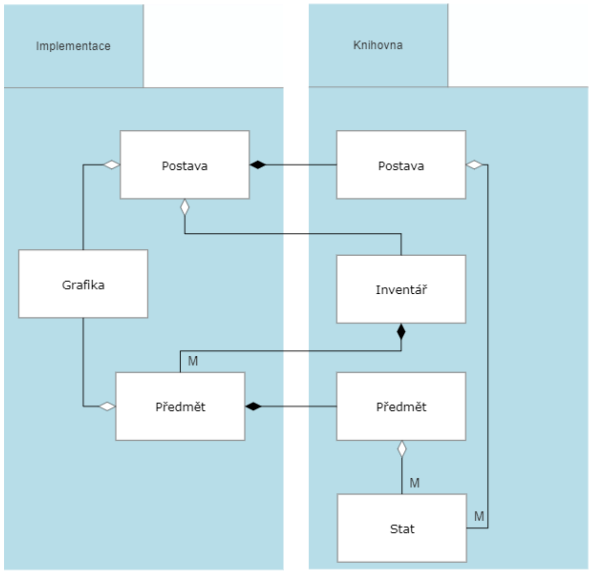
	Unity	Unreal Engine	CRYENGINE
Využíván převážně pro	indie	AAA	AAA
Windows, Linux	ano	ano	ano
MacOS	ano	ano	ne
PS4, Xbox One	ano	ano	ano
PS5, Xbox X	ano	ano	ne
iOS a Android.	ano	ano	ne
web	ano	ne	ne

Zdarma při zisku pod	$100 000/12 měsíců	$1 000 000/produkt	$5 000/rok
Cena licence	$399 (zisk pod $200 000) /$1 800 ročně za počítač	5% ze zisku	5% ze zisku
jazyky	C#, JavaScript, Boo	C++	C++, C#
Příklady her	Cuphead Hollow Knight	Fortnite Borderlands	Crysis Kingdom Come: Deliverance

Tab. 3 srovnání enginů[135, 136][119, 137][134][119][138][133]

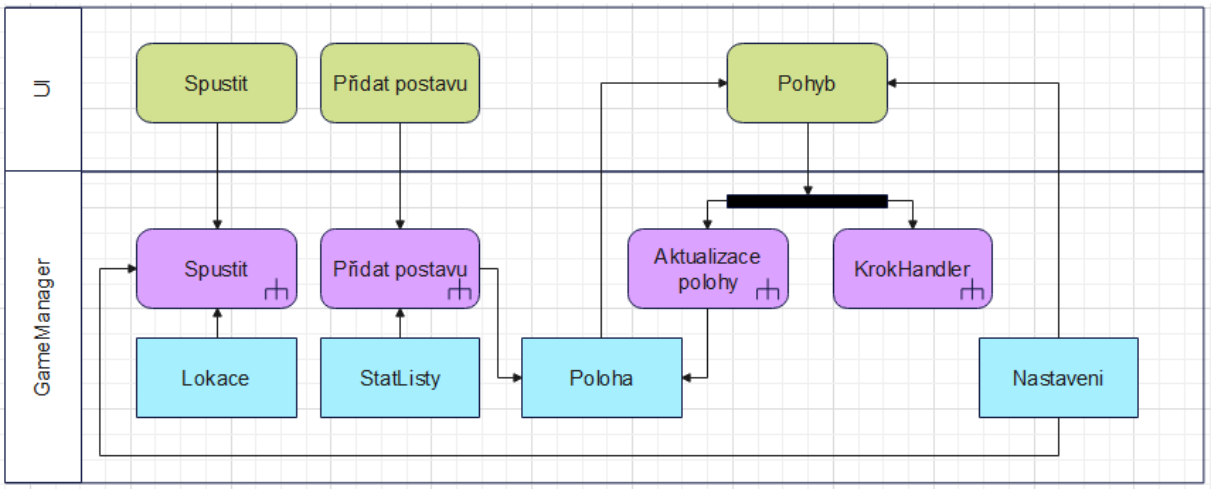
6. Návrh aplikačního modelu

Hry využívající DLL vytvořenou v této práci tvoří dvě části. První je obecná logika obsažená v této knihovně. Druhá část je uživatelské rozhraní a upřesnění chování, aby odpovídalo mechanikám konkrétní hry. Na Obr. 5 je vyobrazeno předpokládané použití knihovny. Objekty reprezentující postavy budou obsahovat instanci třídy obsahující logiku, svou reprezentaci, kterou uvidí hráč, a případně také inventář s předměty. Tyto předměty také budou obsahovat instanci třídy z knihovny a svou reprezentaci. Dále postavy i předměty obsahují list instancí třídy vyjadřující například útok a obranu.



Obr. 5 relační diagram postavy – vlastní

Za účelem snížení spotřeby operační paměti je využit GameManager obsahující veškerá data, která nejsou unikátní pro každou instanci nebo je možné, že budou potřeba i mimo svůj objekt (např. polohy postav). Toto má za následek, že objekt nemá více kopií, než je potřeba a veškeré metody pracují se stejnou hodnotou. Na Obr. 6 je nakreslen diagram znázorňující volání GameManagera. Zelenou barvou jsou označeny stavy, kdy je volání provedeno. Prováděné aktivity jsou vyobrazeny fialově. Modrá reprezentuje kolekce poskytující hodnoty potřebné k zpracování těchto požadavků.



Obr. 6 diagram GameManeger-vlastní

6.1 Návrhové vzory (design patterns)

Návrhové vzory jsou léty ověřené techniky pro řešení problémů v objektově orientovaných jazycích. Výhodou návrhových vzorů je, že je zná většina vývojářů po celém světě a nejsou limitovány konkrétní technologií, což usnadňuje komunikaci. Jsou děleny do tří skupin. První skupiny řeší vytváření nových instancí. Druhá skupina určuje strukturu objektů a komunikace mezi nimi. Třetí skupina umožňuje úpravu chování a v některých případech ho i měnit za běhu aplikace[139–143].

V návrhu je vidět několik situací, kde je vhodné použít návrhový vzor. Proto nyní budou vysvětleny vzory vhodné k realizaci tohoto návrhu.

6.1.1 Factory metoda

Při použití tohoto návrhového vzoru při vytváření nového objektu není volán přímo konstruktor, ale metoda, která ověří vstupy a případně rozhodne, jaký z konstruktorů bude zavolán. Tento postup je často využíván při dědičnosti, kdy potomci mají stejné metody, ale jiné chování. Dalším možným využitím je načítání z databáze, kdy objekt obsahuje již načtená data a pokud požadovaný údaj obsahuje, metoda vrátí již načtená data. V opačném případě je poslán dotaz do databáze a vrácena nová instance.[144]

6.1.2 Singleton

Účelem Singletonu je zajistit, aby třida měla pouze jednu jedinou instanci. Toho je většinou dosaženo tím, že konstruktor není možné zavolat mimo třídu a místo něj je volána statická *Factory* metoda. Ta vrací obsah *private static* proměnné obsahující instanci této třídy. Jeli proměnná prázdná, je předtím zavolán konstruktor a vrácený objekt je vložen do této proměnné. Singleton je využíván pro komunikaci s externím zdrojem dat, či pro uchování dat dostupných pro celou aplikaci.[145]

6.1.3 FlyWeight

Tento návrhový vzor slouží k snížení spotřeby operační paměti počítače. Aby toho docílil, jsou hodnoty, které jsou společné pro několik objektů, ale slouží pouze ke čtení, umístěny do samostatné třídy. Aplikace vytvoří instance jen pro unikátní kombinace a do původních objektů je vložena pouze reference. Toto řešení je často využíváno pro grafické elementy, jejichž velikost se často pohybuje od jednotek do desítek Megabytů. Oproti tomu reference má třicet dva nebo šedesát čtyři bitů.[146]

6.1.4 Observer

Observer je využíván v případech, kdy objekty potřebují dostat informaci o změně stavu jiného objektu. K realizaci se využívá kolekce odběratelů, ve které všechny prvky musí mít metodu, která

slouží k aktualizaci. V okamžiku, kdy nastane změna na sledovaném objektu, je na ni pomocí cyklu upozorněn každý odběratel. Některé jazyky (např. C#) mají tento mechanismus v sobě již zabudován, čímž šetří programátorův čas a zpřehledňují projekt, neboť není potřeba přidávat nové třídy.
Typickým použitím je interakce uživatele s grafickým rozhraním, kdy jsou volány metody uvnitř třídy okna[147]

7. Návrh vzorového řešení

Aplikaci tvoří dva projekty. První je DLL knihovna *KnihovnaRPG*, obsahující logiku a je výstupem praktické části této bakalářské práce. Druhý je Windows Forms aplikace *TestovaniCastiKnihovny*, sloužící jako příklad implementace a testování základní funkčnosti jednotlivých tříd. Testovací projekt má pouze ověřit funkčnost částí knihovny a nejedná se o plnohodnotnou hru. Tomuto účelu odpovídá i uživatelské rozhraní, které není určeno uživatelům, ale slouží programátorovi jako spouštěč jeho testovacích metod a grafická reprezentace jejich výsledků.

//kod

7.1 knihovna

Knihovna obsahuje třídy obsahující logiku, která je shodná v téměř všech RPG hrách. Při návrhu bylo bráno v úvahu, že hra může být v konzoli, okně s dvourozměrnou grafikou, či trojrozměrném prostředí herního enginu a k reprezentaci jednotlivých herních objektů musí být přistupováno abstraktně. Metody, u kterých se očekává že se v některých hrách mohou chovat jinak, jsou nastaveny jako *virtual*, což umožňuje jejich nahrazení v implementaci, bez výrazného ovlivnění chování zbytku knihovny. Tyto třídy by se dali rozdělit na několik skupin podle toho, jakou část logiky pokrývají.

7.1.1 atributy

První skupina se zabývá atributy či statistikami, které jsou častěji nazývány jako staty z anglického stats. Jedná se o vlastnosti postavy a vybavení, které určují například jak velké poškození způsobí, o kolik sníží poškození způsobené nepřítelem, či jak úspěšné bude vyjednávání. Jednotlivé atributy jsou reprezentovány objekty typu *Stat*, které obsahují název a zkratku, základní hodnoty a o kolik je navýšena. Navýšení je možné o konstantní hodnotu, či procenta. Tyto objekty jsou uloženy v objektu typu *StatList*, který umožňuje vyhledávání a kontroluje, že každý atribut se bude vyskytovat nejvýše jednou.

7.1.2 buffy

Tuto skupinu tvoří třída *Buff* a výčty (*Enum) BuffType* a *BuffZpusobZmeny*, které udávají kontext použití. *Buff* slouží ke krátkodobému vylepšení, či v případě *Debuff* zhoršení atributů. *Zraneni* je využito v případě přetrvávající poškození, jako je například zapálení či otrava. Pokud by v implementaci bylo třeba třídu využít i v jiné situaci je připravena hodnota *Jiny*, ale toto chování není definováno knihovnou. Vlastnost *Doba* určuje, jak dlouho efekt trvá (např. počet kol nebo vteřin). *Efekt* obsahuje hodnoty, o které se atribut postavy změní. *Obrana* je zkratka atributu, kterým je možné snížit negativní efekt. *Sesilatel* je využit v případě, že efekt způsobí smrt a hra mohla tuto informaci vypsat a přidělit odměnu. *Plati* slouží k zjednodušení určení, zda efekt již skončil a má být odstraněn ze seznamu. Metoda *Pouzij* má jako parametry atribut postavy, který bude měněn a znaménko (jedna pro aktivaci efektu a mínus jedna pro jeho zrušení). ZkratDobu krátí dobu zbývající

do konce trvání efektu o jedna. Snížením hodnoty pomocí metody místo přímou úpravou hodnoty je zabráněno nečekanému chování vlivem záporné hodnoty.

7.1.3 Postavy

Logiku kolem postav zajišťují třída *Postava* a její potomek *Hrac*, který má navíc zkušenosti a peníze. Postavy mají své jméno, aktuální stav životů (často označované jako HP z anglického health points) a maximální počet životů, který není možné překročit. Postavy je možné vytvořit nezranitelné a zabránit tak hráči ublížit NPC. Pokud se nejedná o prostého vesničana, který se ve hře nachází, aby svět nepůsobil prázdně, ale o obchodníka či nepřítele má postava StatList (viz kap. 6.1.1). K výpočtu hodnot je využívána úroveň postavy uložená ve vlastnosti *LV* (z anglického level). Během souboje je volána metoda *Zraneni*, která má jako parametry útočníka, velikost působeného poškození, a kterým atributem se toto poškození dá snížit. Po výpočtu nového stavu životů je vyvolána událost *Zranen*, který slouží k aktualizaci zobrazení stavu životu v uživatelském rozhraní. V případě že životy postavy klesnou na nulu, jsou vyvolány události *Zabil* a *Smrt*. *Zabil* slouží k výpisu a přidělení odměny útočníkovi. *Smrt* je určena k reakci grafické reprezentace postavy, jako je například přehrání animace. Pomocí metody PridejBuff jsou buffy (viz kap. 6.1.2) přidávány do listu, který není mimo objekt viditelný. Metoda *EfektyBuffu* je volána každé nové kolo, či v časových intervalech. Projde seznam a zjistí, zda je efekt stále platný. Pokud efekt vyprchal, je odstraněn ze seznamu a změněná hodnota je navrácena do původního stavu. Jedná-li se o přetrvávající zranění je zavolána metoda *Zraneni*, které jsou předány příslušné hodnoty.

7.1.4 Předměty

Skupinu předmětů tvoří rozhraní *IPredmet*, třída *Predmet* a její potomci *Konzumovatelne* a *Vybaveni*. Rozhraní umožňuje do inventáře (viz kap. 6.1.5) vkládat jak instance a potomky tříd knihovny (implementace typu je, která je v anglické literatuře označována jako is) z této skupiny, tak jejich grafické verze (implementace typu má, která je v anglické literatuře označována jako has). Každý objekt reprezentující předmět má jméno a základní cenu před tím, než se započtou atributy a bonusy ovlivňující vyjednávání, čímž vznikne výsledná cena. Dále mají předměty svou hmotnost, která se projeví u inventářů, jejichž kapacita je určena nosností. Vlastnost *Stackovatelne* vyjadřuje, zda je možné umístit na jednu pozici v inventáři více stejných předmětů, což se projeví při výpisu a u inventářů s omezeným počtem pozic. Třída *konzumovatelne* reprezentuje spotřební předměty, jakou jsou lektvary, jichž použití přidává bonusy. Třída *Vybaveni* je určena pro předměty, které si postava může nasadit (např. zbroj a zbraně).

7.1.5 inventáře

Logiku inventářů mají na starosti třída *Inventar* a její potomci *InventarHmotnost* a *InventarPocet*. *Inventar* slouží jako neomezený inventář, zatímco jeho potomci mají omezenou kapacitu. V inventářích je možné hledat předměty podle indexů, či naopak zjistit index předmětu. Metoda *Stav* vrací string kolik je již zaplněno, a jaká je kapacita. Tento údaj může být využít samostatně, či jako součást *ToString*, kde následuje obsah inventáře. Inventáře dále mají události sloužící ke grafickému zobrazení přidání a odebrání předmětu a zobrazení informace, že inventář je plný a není možné přidat další předmět. *InventarPocet* má svou kapacitu omezenou počtem předmětů, ale stejné předměty s vlastností *Stackovatelne* nastavenou na hodnotu *true* se počítají jako jeden. InventarHmotnost má počet předmětů omezený součtem jejich Hmotností.

7.1.6 kouzla

Jelikož je většina RPG her zasazena do fantasy světa, s vysokou pravděpodobností bude potřeba logika řešící kouzla. Základem je abstraktní třída Kouzlo, obsahující název, množství many potřebné k seslání kouzla, dobu nabíjení, která určuje, za jak dlouho bude možné kouzlo znovu seslat, a kolik

z tohoto času zbývá. Metoda *DalsiKolo* snižuje *ZbyvaDoNabiti* a brání dosáhnout záporné hodnoty. *KouzloUtok* slouží k zranění nepřítele a má DMG říkající hodnotu poškození, které způsobí. Obrana je zkratka *Statu*, kterým je možné tuto hodnotu snížit. Typ je využíván při výpisu, aby hráč věděl, jaký druh poškození působí. *KouzloBuff* přidá vybrané postavě *Buff* (viz Kap. 6.1.2) a *KouzloLeceni* po dobu efektu kouzla doplňuje vybrané postavě životy.

7.1.7 mapa

Během hry je třeba vědět, jaké postavy se kde v herním světě nachází. Tuto logiku zajišťují třídy *Mapa*, *Chunk* a *Lokace.* Základním stavebním kamenem jsou instance třídy *Lokace* reprezentující prostředí, která se ve hře mohou nacházet (např. les a vesnice). Každý objekt obsahuje seznam lokací, se kterými může sousedit, což je využíváno při náhodném generování mapy. Pro práci s lokacemi je využíván návrhový vzor fly weight, kdy není vytvářena nová instance pro každé použití, ale jsou vytvořeny jen instance unikátních objektů, na něž je odkazováno, čímž je výrazně sníženo potřebné místo v paměti. Dalším způsobem minimalizace potřebné paměti je rozdělení mapy na menší bloky zvané *chunky*. Ty umožňují uchovávat v paměti jen aktuálně vykreslovanou část mapy a její nejbližší okolí, kam hráč pravděpodobně brzy dorazí. Pro účely generování těchto bloků obsahuje třída několik metod schopných vygenerovat nový chunk na základě startovní lokace a její polohy, či podle sousedních chunků. Rozmístění chunků a informace o tom, které již byli vygenerovány se nachází v objektu *Mapa*, kde se také provádí volání příslušných metod. Pro předání potřebných souřadnic při načítání a uvolňování chunku z paměti slouží třída *LoadUnloadEventArg*, jejíž instance se předává jako argument události.

7.6.8 úkoly

K RPG hrám neodmyslitelně patří plnění úkolů. Ty jsou reprezentovány objekty typu Úkol, který má své jméno, popis a informaci, zda byl již hráčem splněn. Jednotlivé části úkolu, potřebné k jeho dokončení, jsou uloženy v listu objektů *UkolPolozka* a odměnu, kterou hráč získá reprezentuje třída implementující rozhraní *IUkolOdmena*. Po splnění je vyvolána událost *Dokoncen* určený k předání odměny hráči. *UkolPolozka* obsahuje výčet (enum) *UkolTyp* říkající jakou akci musí hráč provést s objektem ve vlastnosti *Polozka*, kde v závislosti na implementaci může být reference či hodnota jenž vyjadřuje, že se jedná o stejný objekt (například jméno nebo výpis). *Hodnota* Počet udává kolikrát je tato akce vyžadována a Hotovo je kolikrát se tak již stalo. Došlo-li k splnění požadavku je vyhodnocováno metodou *UpdateStav*. Pro odměny je kromě třídy *UkolOdmena* připraveno také rozhraní *IUkolOdmena* pro případ, že by toto řešení nebylo pro konkrétní hru vhodné.

7.1.9 GameManager

Aby bylo možné veškeré změny provádět na jednom místě a eliminovat tak nežádoucí situaci, že je hodnota zapsána na více místech, ale změna je provedena jen na některých a hra v některých situacích použije novou hodnotu, ale v ostatních případech stále využívá starou, je použita třída *GameManagerDLL*. Další výhodou je vyšší přehlednost a snazší balancování hodnot. Jelikož má každá hra jiné lokace, postavy, předměty a jejich statistiky není možné tyto údaje vytvořit univerzální pro všechny. Proto je třída vytvořena jako abstraktní, což znamená, že není možné vytvořit její instanci a je třeba vytvořit jejího potomka. Třída obsahuje abstraktní metody, které mají jen hlavičku a potomek musí obsahovat jejich přetížení. Jelikož je GameManage běžně využívaný název, byla třída v knihovně nazvána *GameManagerDLL*, aby nevznikl konflikt v názvech, jelikož třídy rozšiřující knihovnu mají odkaz na *namespace* knihovny. Na potomka třídu by měl použit návrhový vzor *singleton*, aby bylo zajištěno, že v programu existuje jen jedna instance. Toho je docíleno tak, že pokud je potřeba instance, je zavolána metoda či vlastnost, ověřující, zda v privátní statické

proměnné již není instance. Pokud ano, je vrácena. V opačném případě je zavolán privátní konstruktor, jehož výsledek je vložen do této proměnné a vrácen. *Staty* jsou rozděleny do skupin, což usnadňuje přidávání k postavám a předmětům, jelikož je možné například přiřadit obchodníkovi pouze atributy související s obchodem, nebo zbrani jen ty, co souvisí s poškozením. Dále objekt obsahuje seznam všech použitých zkratek, aby bylo zajištěno, že se ve hře neobjeví jeden stat s více zkratkami. Knihovna počítá s tím, že ve hře bude více než jeden hráč, ale pro jednodušší práci *GameManager* obsahuje kromě vlastnosti *PolohaHracu* vracející pole, také *PolohaHrac* vracející první prvek tohoto pole. Poloha hráčů a počítačem ovládaných postav je reprezentována instancemi třídy Point4D obsahující X a Y souřadnici uvnitř chunku a jeho polohu v mapě. Posouvání po mapě je řešeno metodou Pohyb, který má jako parametry změnu v souřadnicí a objekt *MapaConfig* (viz kap. 6.1.10) obsahující informaci o rozměrech. Je-li překročena hranice chunku, je *DalsiChunk* nastaven na true. Tato hodnota je kontrolována v *KrokHnadler* vyhodnocující, zda je třeba načíst nový chunk. GameManager také obsahuje seznam všech nastavení (viz kap. 6.1.10), což jsou třídy implementující rozhraní INastaveni.

7.1.10 nastavení

Aby bylo možné veškerá nastavení vkládat do jedné kolekce je třeba mít společného rodiče. Tuto roli má rozhraní INastaveni. Jediný povinný člen třídy implementující rozhraní je *vypis* sloužící k přehledu hodnot. V knihovně se nachází pouze třída MapaConfig obsahující informace o rozměrech mapy, do jaké vzdálenosti bude hra zobrazovat okolí hráče a startovací pozici hráče.

7.2 TestovaniCastiKnihovny

Tato část řešení slouží k ověření základní funkcionality knihovny. Je zde ukázáno vzorové použití knihovny. Nejedná se funkční hru, ale ukázku pro programátory, kteří by chtěli knihovnu použít ve svém projektu. Pro tento účel byla zvolena Windows Forms aplikace, neboť je rychlá a snadná na vývoj. Další výhoda je, že k pochopení činnosti této ukázky nemusí programátor, který tento projekt bude prohlížet, znát konkrétní technologii a jazyk či knihovnu použitou pro tvorbu uživatelského prostředí.

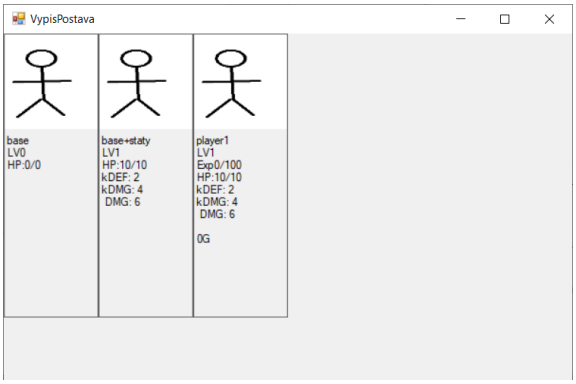
7.2.1 okno Form1

Na Obr. 7 je úvodní obrazovka odkazující na testy konkrétních částí. Tlačítka jsou rozdělena do sloupců podle oblastí, které pokrývají. Blok tlačítek vlevo při každém kliknutí vytvoří nová okna, jejichž data nejsou nikde uložena a po zavření okna jsou smazána. V prvním sloupci jsou funkce týkající se postav. Dále je zde test funkce inventáře, generování mapy a plnění úkolů. Blok vpravo testuje funkčnost GameManagera, jehož účelem je držet data po celou dobu běhu aplikace.

Obr. 7 okno Form1-vlastní

7.2.2 okno VypisPostava

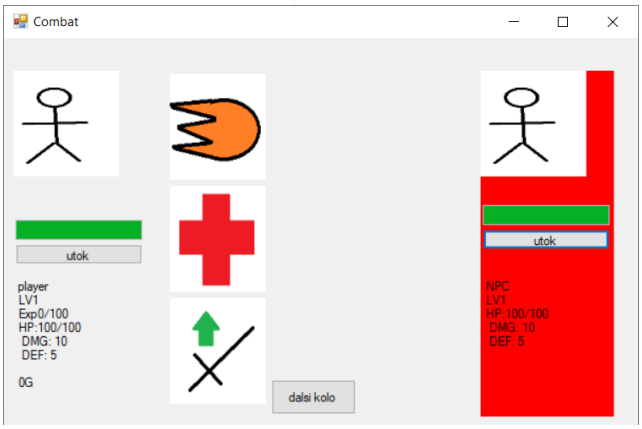
Obr. 8 ukazuje okno sloužící k výpisu informací o postavách jež jsou instance třídy *PostavaKomp* (komponentní neboli implementace typu má). Vlevo je nezranitelná postava, která nemá žádné atributy. Uprostřed je zranitelná postava, která reprezentuje nepřítele, s nímž bude hráč bojovat. Tento objekt má navíc seznam atributů, úroveň a počet životů. Vpravo je hráč, který má navíc počet zkušeností a peněz. Hráč je instancí *HracKomp*, což je potomek *PostavaKomp*. Ačkoliv zde oproti své rodičovské třídě nic nového nepřináší, v reálném použití by obsahoval ovládání postavy, které je zde nepotřebné.



Obr. 8 okno VypisPostava-vlastní

7.2.3 okno Combat

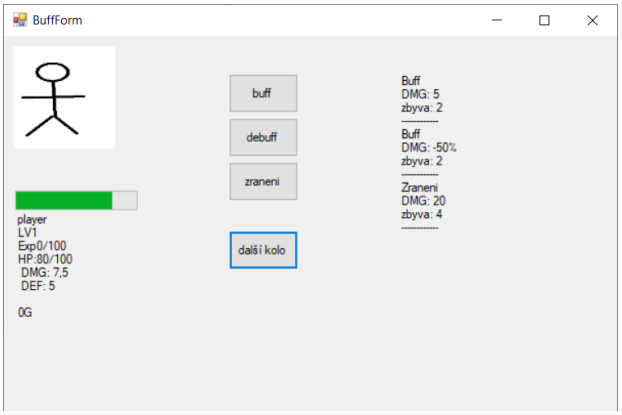
Okno Combat (Obr. 9) simuluje souboj hráče, jenž se nachází vlevo, s jinou postavou. Obě postavy mají tlačítko, kterým způsobí té druhé poškození. Stav životů je zobrazen ukazatelem upravovaným metodami naslouchající vyvolání událostí Zranen a *Uzdraven*. Hráč má navíc k dispozici kouzla, jejichž doba trvání je dána počtem kol. začátek dalšího kola je simulován tlačítkem. Prvním kouzlem je ohnivá koule způsobující zranění. Další je léčení obnovující hráčovi životy. Posledním kouzlem je zvýšení poškození.



Obr. 9 okno Combat-vlastní

7.2.4 okno BuffForm

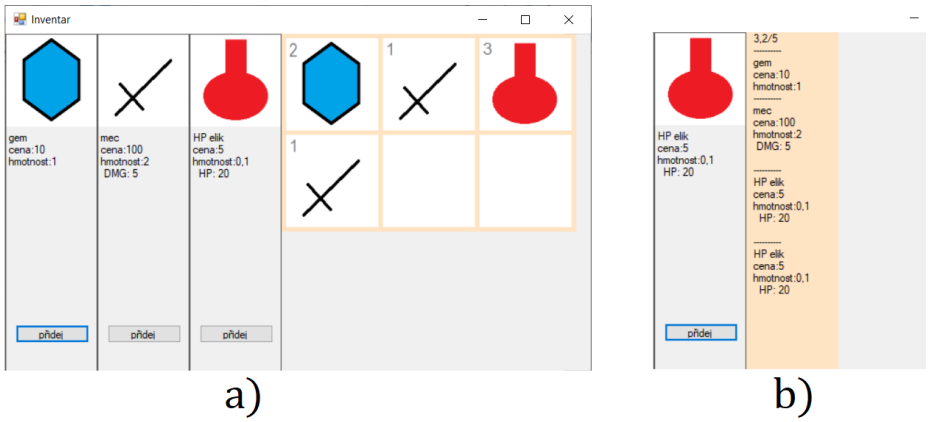
Okno BuffForm (Obr. 10) slouží k otestování přetrvávajících efektů. Stejně jako v případě Combat se vlevo nachází výpis postavy a další kolo je spouštěno tlačítkem. Vpravo je seznam aktuálně působících efektů, kde je u každého napsán jeho typ, ovlivňovaný atribut, hodnota změny a jak dlouho bude ještě působit.



Obr. 10 okno BuffForm-vlastní

7.2.5 okno InventarForm

Na Obr. 11 jsou ukázány dvě možné varianty okna InventarForm. V levé části okna se nachází tři předměty, které je možné do inventáře přidat, a v pravé se je vyobrazení těch, které již byli sebrány. Obrázek a) představuje variantu, kdy je kapacita inventáře omezena počtem předmětů. Některých předmětů je možné sebrat více, aniž by to mělo vliv zbývající kapacitu. Tyto předměty se přidávají do stejného políčka a velikost této hromady je ukázána číslem v levém horním rohu. Na obrázku b) je výsek obrazovky, jak se liší pravá část okna, v případě omezení kapacity pomocí celkové hmotnosti sebraných předmětů. Zaplněnost inventáře je napsána ve vrcholu panelu, kde se nachází výpis.



Obr. 11 okno InventarForm v početní a hmotnostní variantě-vlastní

7.2.6 okna ChunkForm a MapaForm

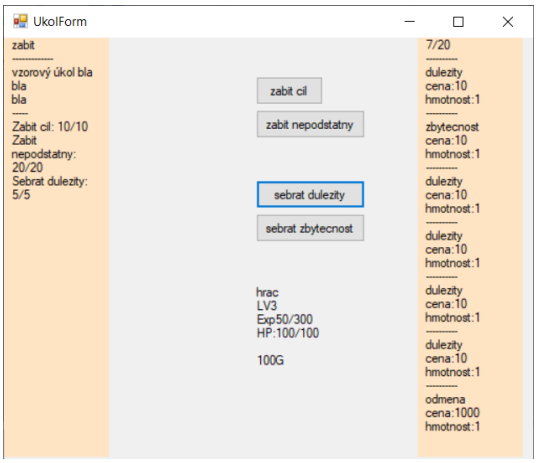
Obr. 12 ukazuje ChunkForm sloužící k otestování náhodného generování jednoho bloku mapy. Každá lokace má omezeny lokace, se kterými může sousedit. V tomto ukázkovém případě mohou les a vesnice sousedit jen s cestou, nebo lokací stejného typu.

Podobnou funkci má také MapaForm, avšak s tím rozdílem, že negeneruje pouze jeden blok, ale mřížku pět na pět, což znamená šest set dvacet pět lokací.

Obr. 12 okno ChunkForm-vlastní

7.2.7 okno UkolForm

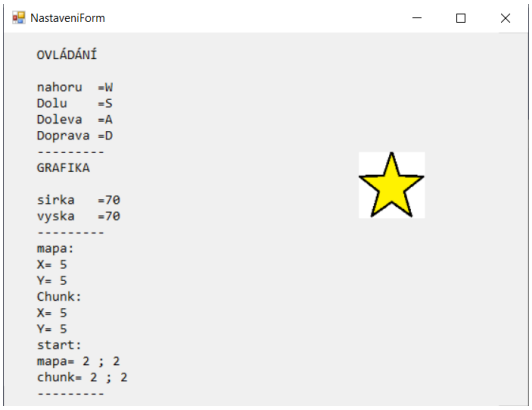
Obr. 13 vyobrazuje okno UkolForm sloužící k ověření funkčnosti logiky při plnění úkolů. V levé části je zadání úkolu, který má svůj název, popis a stav splnění. Napravo se nachází výpis inventáře. Uprostřed jsou tlačítka reprezentující vyhraný souboj a sebrané předměty. Pod tlačítky je umístěn výpis hráče, který po splnění úkolu dostane odměnu v podobě zkušeností, zlata a předmětu.



Obr. 13 okno UkolForm-vlastní

7.2.8 okno NastaveniForm

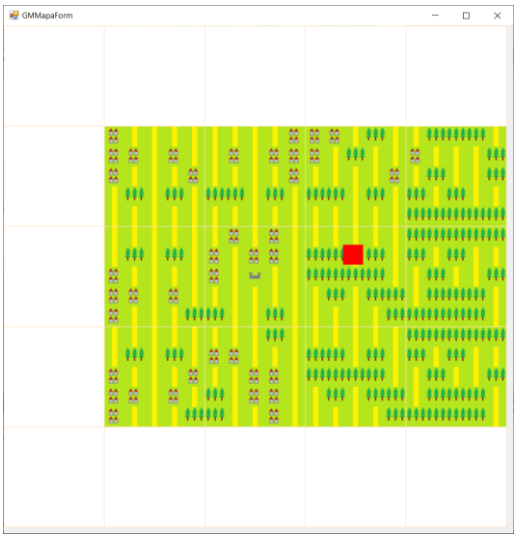
Okno NastaveniForm (Obr. 14) slouží k výpisu a otestování globálního nastavení obsaženém v sigleton instanci potomka třídy GameManager, které využívají všechna okna otevíraná pomocí tlačítek v pravém bloku úvodní obrazovky. Nastavení ovládání a grafiky je možné vyzkoušet na obrázku vpravo, který má stanovený rozměr a příslušnými klávesami je možné s ním pohybovat. Nastavení mapy je využito při generování v okně GMMapaForm a určuje rozměry mřížky bloků, které tvoří mapu, a lokací v každém tomto bloku. Dále určuje, do jaké polohy je umístěna startovní lokace, podle které je generován zbytek mapy.



Obr. 14 okno NastaveniForm-vlastní

7.2.9 okno GMMapaForm

Na Obr. 15 je ukázáno okno GMMapaForm, které obdobně jako MapaForm generuje mapu, avšak v tomto případě se více podobá skutečnému použití. Jak již bylo zmíněno v kapitole 7.2.8 rozměry nejsou dány napevno kódem, ale jsou nastaveny v GameManegerovi. Hlavním a na první pohled viditelným rozdílem je, že po otevření okna mapa není vygenerována celá, ale jsou zobrazeny jen bloky v okruhu startovní pozice. Na této pozice se také nachází červený čtverec reprezentující hráče. Ten se pohybuje pomocí kláves nastavených pro pohyb (viz kap. 7.2.8) a přejde-li do sousedního bloku, jsou vygenerovány sousední bloky.



Obr. 15 okno GMMapaForm-vlastní

7.2.10 okno GMPostavyForm

Ačkoliv na první pohled toto okno vypadá stejně jako *VypisPostava*, tak logika na pozadí je odlišná. Postavy nejsou vytvářeny přímo kódem na pozadí tohoto okna, ale je zavolána metoda v gamemanegeru, již je předán seznam skupin Statů, které má nově vytvořená postava mít.

7.2.11 okno GMUkladaniForm

Na Obr. 16 je vidět okno sloužící k otestování funkčnosti ukládání a načítání rozehrané hry. Při kliknutí na tlačítko *uloz* je vytvořena nová hra obsahující vygenerovanou mapu, dvě hráčovi a jednu nehratelnou postavu. Po uložení či načtení jsou tyto údaje zobrazeny, čímž je umožněno překontrolovat jejich shodnost. Aby šlo tuto operaci opakovat bez neustálého zavírání a znovu otevírání okna jsou pro každé uložení vytvořeny nové údaje místo využití již existujících.

Obr. 16 okno GMUkladaniForm – vlastní

8. Zhodnocení realizace aplikace

Při odhadu času potřebného k vypracování práce jsem podcenil hledání zdrojů pro rešeršní část. Do budoucna je v plánu udělat plnohodnotnou implementaci do různých technologií a odhalit případné chyby návrhu. Dále je plánováno přidat gramatiku, která umožní nastavit hodnoty pro *GameManager* pomocí čitelných souborů bez nutnosti zásahu do zdrojového kódu. Při návrhu bylo zapomenuto na trojrozměrnou mapu. jelikož tato chyba byla odhalena až na konci vývoje a její oprava by vyžadovala změnu velké části kódu, byla její oprava odložena na refaktorizaci, která bude provedena společně s opravou chyb, které budou nalezeny během implementace. Dále je třeba vymyslet praktičtější určení polohy na mapě.

//

9. Závěr

//

Použité zdroje

[1] *imperative programming* [online]. [vid. 2021-03-16]. Dostupné z: https://whatis.techtarget.com/definition/imperative-programming

[2] COMPUTERPHILE. *Programming Paradigms - Computerphile* [online]. 2013 [vid. 2021-03-29]. Dostupné z: https://www.youtube.com/watch?v=sqV3pL5x8PI

[3] *procedural and object oriented programming* [online]. [vid. 2021-03-29]. Dostupné z: https://www.geeksforgeeks.org/differences-between-procedural-and-object-oriented-programming/

[4] FREECODECAMP.ORG. *Intro to Object Oriented Programming - Crash Course - YouTube* [online]. 2020 [vid. 2021-07-04]. Dostupné z: https://www.youtube.com/watch?v=SiBw7os-_zI

[5] STROUSTRUP, Bjarne. *Stroustrup: FAQ-multiparadigm* [online]. [vid. 2021-07-22]. Dostupné z: https://www.stroustrup.com/bs_faq.html#multiparadigm

[6] STROUSTRUP, Bjarne. From The Handbook of Object Technology (Editor: Saba Zamir) [online]. 1999 [vid. 2021-07-18]. Dostupné z: https://www.stroustrup.com/crc.pdf

[7] STROUSTRUP, Bjarne. *Stroustrup: FAQ-C subset of C++* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/bs_faq.html#C-is-subset

[8] PRATA, Stephen. *Mistrovství v C++*. 1. vyd. Praha: Computer Press, 2001. ISBN 80-7226-339-0.

[9] *Why Java is Platform Independent? | by Neil Wilston | Medium* [online]. [vid. 2021-07-21]. Dostupné z: https://medium.com/@neil.wilston123/why-java-is-platform-independent-1d82c2249a69

[10] *What is x86 Architecture and its difference between x64? - Latest open tech from seeed studio* [online]. [vid. 2021-07-21]. Dostupné z: https://www.seeedstudio.com/blog/2020/02/24/what-is-x86-architecture-and-its-difference-between-x64/

[11] HANÁK, Ján. *C++/CLI začínáme programovat*. Brno: artax a.s., 2009. ISBN 978-80-87017-04-3.

[12] STROUSTRUP, Bjarne. *C++ Applications* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/applications.html

[13] STROUSTRUP, Bjarne. *Stroustrup: FAQ* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/bs_faq.html#true

[14] STROUSTRUP, Bjarne. *Stroustrup: FAQ-unsafe* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/bs_faq.html#unsafe

[15] STROUSTRUP, Bjarne. *Stroustrup: FAQ-garbage-collection* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/bs_faq.html#garbage-collection

[16] STROUSTRUP, Bjarne. *Stroustrup: FAQ-GUI* [online]. [vid. 2021-07-20]. Dostupné z: https://www.stroustrup.com/bs_faq.html#gui

[17] ORACLE. *The Java Language Environment* [online]. [vid. 2023-02-10]. Dostupné z: https://www.oracle.com/java/technologies/introduction-to-java.html

[18] *Difference between Methods and Functions in JavaScript - GeeksforGeeks* [online]. [vid. 2021-07-22]. Dostupné z: https://www.geeksforgeeks.org/difference-between-methods-and-functions-in-javascript/

[19] ORACLE. *The Java Language Environment* [online]. [vid. 2021-07-22]. Dostupné z: https://www.oracle.com/java/technologies/simple-familiar.html

[20] *Is Java slow? Compared to C++, it's faster than you think* [online]. [vid. 2021-07-22]. Dostupné z: https://www.theserverside.com/opinion/Is-Java-slow-Compared-to-C-its-faster-than-you-think

[21] EGGES, Arjan, Jeroen D. FOKKER a Mark H. OVERMARS. *Learning C# by Programming Games* [online]. 2013. ISBN 3642365795. Dostupné z: doi:10.1007/978-3-642-36580-5

[22] ORACLE. *Oracle Java ME Embedded Getting Started* [online]. [vid. 2021-07-22]. Dostupné z: https://www.oracle.com/java/technologies/javame-embedded/javame-embedded-getstarted.html

[23] FREECODECAMP.ORG. *Garbage Collection in Java – What is GC and How it Works in the JVM* [online]. [vid. 2021-07-21]. Dostupné z: https://www.freecodecamp.org/news/garbage-collection-in-java-what-is-gc-and-how-it-works-in-the-jvm/

[24] IBM. *Garbage collection impacts to Java performance - IBM Documentation* [online]. [vid. 2021-07-21]. Dostupné z: https://www.ibm.com/docs/en/aix/7.1?topic=monitoring-garbage-collection-impacts-java-performance

[25] ORACLE. *Java SE 6 HotSpot[tm] Virtual Machine Garbage Collection Tuning* [online]. [vid. 2021-07-22]. Dostupné z: https://www.oracle.com/java/technologies/javase/gc-tuning-6.html

[26] DOCS.ORACLE.COM. *java.awt (Java Platform SE 7)* [online]. [vid. 2021-07-26]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/java/awt/package-summary.html

[27] DOCS.ORACLE.COM. *javax.swing (Java Platform SE 7)* [online]. [vid. 2021-07-26]. Dostupné z: https://docs.oracle.com/javase/7/docs/api/javax/swing/package-summary.html

[28] ITNETWORK.CZ. *Lekce 3 - Seznam (List) pomocí pole v Javě* [online]. [vid. 2021-07-22]. Dostupné z: https://www.itnetwork.cz/java/kolekce-a-proudy/java-tutorial-seznamy-kolekce-list

[29] DOCS.MICROSOFT.COM. *A Tour of C# - C# Guide | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/

[30] . *NET (and .NET Core) - introduction and overview | Microsoft Learn* [online]. [vid. 2023-02-05]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/core/introduction

[31] *ReadyToRun deployment overview - .NET | Microsoft Learn* [online]. [vid. 2023-02-07]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/core/deploying/ready-to-run

[32] RICHARD LANDER. Conversation about crossgen2 - .NET Blog. *.NET Blog* [online]. 2021 [vid. 2023-02-07]. Dostupné z: https://devblogs.microsoft.com/dotnet/conversation-about-crossgen2/

[33] DOCS.MICROSOFT.COM. *Ngen.exe (Native Image Generator) | Microsoft Docs* [online]. [vid. 2021-08-06]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/tools/ngen-exe-native-image-generator

[34] MICROSOFT. *.NET Glossary | Microsoft Learn* [online]. [vid. 2023-01-29]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/standard/glossary#implementation-of-net

[35] DOCS.MICROSOFT.COM. *.NET Framework versions and dependencies* [online]. [vid. 2021-06-05]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies?redirectedfrom=MSDN#net-framework-30

[36] DOCS.MICROSOFT.COM. *What is .NET MAUI? - .NET MAUI | Microsoft Docs* [online]. [vid. 2021-09-23]. Dostupné z: https://docs.microsoft.com/cs-cz/dotnet/maui/what-is-maui

[37] DOCS.MICROSOFT.COM. *Unsafe code, pointers to data, and function pointers | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/unsafe-code

[38] *VisualMicro - Arduino IDE For Visual Studio* [online]. [vid. 2021-07-23]. Dostupné z: https://www.visualmicro.com/#

[39] . *NET nanoFramework VS2019 Extension - Visual Studio Marketplace* [online]. [vid. 2021-07-23]. Dostupné z: https://marketplace.visualstudio.com/items?itemName=nanoframework.nanoFramework-VS2019-Extension

[40] *C# | Method Overloading - GeeksforGeeks* [online]. [vid. 2021-07-23]. Dostupné z: https://www.geeksforgeeks.org/c-sharp-method-overloading/

[41] DOCS.MICROSOFT.COM. *Fundamentals of garbage collection | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/fundamentals

[42] DOCS.MICROSOFT.COM. *.NET garbage collection | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/garbage-collection/

[43] DOCS.MICROSOFT.COM. *MulticastDelegate Class (System) | Microsoft Docs* [online]. [vid. 2021-07-25]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.multicastdelegate?view=net-5.0#code-try-3

[44] ŽIVĚ.CZ. *Poznáváme C# a Microsoft.NET 15. díl – delegáty – Živě.cz* [online]. [vid. 2021-07-25]. Dostupné z: https://www.zive.cz/clanky/poznavame-c-a-microsoftnet-15-dil--delegaty/sc-3-a-123479/default.aspx

[45] DOCS.MICROSOFT.COM. *Delegates - C# Programming Guide | Microsoft Docs* [online]. [vid. 2021-07-25]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/delegates/

[46] DOCS.MICROSOFT.COM. *Handling and Raising Events | Microsoft Docs* [online]. [vid. 2021-07-25]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/standard/events/

[47] DOCS.MICROSOFT.COM. *EventHandler Delegate (System) | Microsoft Docs* [online]. [vid. 2021-07-25]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/api/system.eventhandler?view=net-5.0

[48] DOCS.MICROSOFT.COM. *out parameter modifier - C# Reference | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/out-parameter-modifier

[49] DOCS.MICROSOFT.COM. *Properties - C# Programming Guide | Microsoft Docs* [online]. [vid. 2021-07-23]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/properties

[50] DOCS.MICROSOFT.COM. *Partial Classes and Methods - C# Programming Guide | Microsoft Docs* [online]. [vid. 2021-07-24]. Dostupné z: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/partial-classes-and-methods

[51] DOCS.MICROSOFT.COM. *Event handling in native C++* [online]. [vid. 2021-06-06]. Dostupné z: https://docs.microsoft.com/en-us/cpp/cpp/event-handling-in-native-cpp?view=msvc-160#:~:text= Event handling in native C%2B%2B 1,you use the intrinsic function __hook... More

[52] LIBGDX. *libGDX features* [online]. [vid. 2021-06-05]. Dostupné z: https://libgdx.com/features/

[53] SLÁMA, David. Průvodce herními žánry – dungeony a rpg – Doupě.cz. *Computer* [online]. 2010 [vid. 2021-08-14]. Dostupné z: https://doupe.zive.cz/clanek/pruvodce-hernimi-zanry--dungeony-a-rpg

[54] *Hra na PC Assassins Creed Odyssey - PC DIGITAL | Hra na PC na Alza.cz* [online]. [vid. 2021-08-24]. Dostupné z: https://www.alza.cz/media/assassins-creed-odyssey-pc-digital-d6222907.htm?o=3

[55] *Ušetřete 75% na produktu Assassin's Creed® Odyssey ve službě Steam* [online]. [vid. 2021-08-24]. Dostupné z: https://store.steampowered.com/app/812140/Assassins_Creed_Odyssey/

[56] *Assassin's Creed Odyssey | Download and Buy Today - Epic Games Store* [online]. [vid. 2021-08-24]. Dostupné z: https://www.epicgames.com/store/en-US/p/assassins-creed-odyssey

[57] *Assassin's Creed Odyssey on PS4, Xbox One, PC | Ubisoft (UK)* [online]. [vid. 2021-08-24]. Dostupné z: https://www.ubisoft.com/en-gb/game/assassins-creed/odyssey

[58] *Baldur's Gate II: Enhanced Edition ve službě Steam* [online]. [vid. 2021-08-26]. Dostupné z: https://store.steampowered.com/app/257350/Baldurs_Gate_II_Enhanced_Edition/

[59] *Hra na PC Baldur's Gate II Enhanced Edition - PC DIGITAL | Hra na PC na Alza.cz* [online]. [vid. 2021-08-26]. Dostupné z: https://www.alza.cz/media/baldurs-gate-ii-enhanced-edition-pc-digital-d5866684.htm

[60] *Hra na PC Heroes of Might & Magic III - HD Edtion (PC) DIGITAL | Hra na PC na Alza.cz* [online]. [vid. 2021-08-26]. Dostupné z: https://www.alza.cz/media/heroes-of-might-magic-iii-hd-edtion-pc-digital-d5346604.htm

[61] *Might & Magic Heroes 3 | Download and Buy Today - Epic Games Store* [online]. [vid. 2021-08-26]. Dostupné z: https://www.epicgames.com/store/en-US/p/might-and-magic-heroes-3

[62] *Heroes® of Might & Magic® III - HD Edition ve službě Steam* [online]. [vid. 2021-08-26]. Dostupné z: https://store.steampowered.com/app/297000/Heroes_of_Might__Magic_III__HD_Edition/

[63] *Buy Heroes of Might and Magic III: Complete PC (Download)* [online]. [vid. 2021-08-26]. Dostupné z: https://store.ubi.com/uk/game?pid=575ffd9ba3be1633568b4d8c&dwvar_575ffd9ba3be1633568b4d8c_Platform=pcdl&edition=Complete%20Edition&source=detail

[64] *D&D Official Homepage | Dungeons & Dragons* [online]. [vid. 2021-11-12]. Dostupné z: https://dnd.wizards.com/

[65] *Baldur's Gate: Enhanced Edition* [online]. [vid. 2021-11-12]. Dostupné z: https://www.baldursgate.com/

[66] *The Elder Scrolls | Skyrim* [online]. [vid. 2021-11-12]. Dostupné z: https://elderscrolls.bethesda.net/en/skyrim

[67] *Knights of the Old Republic | StarWars.com* [online]. [vid. 2021-12-20]. Dostupné z: https://www.starwars.com/games-apps/knights-of-the-old-republic

[68] *FINAL FANTASY PORTAL SITE | SQUARE ENIX* [online]. [vid. 2021-11-12]. Dostupné z: https://na.finalfantasy.com/

[69] *World of Warcraft* [online]. [vid. 2021-11-12]. Dostupné z: https://worldofwarcraft.com/en-gb/

[70] *XCOM 2* [online]. [vid. 2021-12-20]. Dostupné z: https://xcom.com/

[71] Fantasy světy – historie počítačových her na hrdiny díl I. *Fantasymag.cz* [online]. 2017. Dostupné z: https://www.fantasymag.cz/fantasy-svety-historie-pocitacovych-her-hrdiny-dil-i/

[72] GAMEDESIGNING.ORG. *34 Popular Types of Video Games, Explained (With Examples and Fun Graphics)* [online]. [vid. 2021-07-27]. Dostupné z: https://www.gamedesigning.org/gaming/video-game-genres/

[73] Fantasy světy díl II. - čtverečkové dungeony na PC | Fantasymag.cz. *Fantasymag.cz* [online]. 2018 [vid. 2021-07-29]. Dostupné z: https://www.fantasymag.cz/fantasy-svety-dil-ii-ctvereckove-dungeony-pc/

[74] *Herní žánry na Databázi her – Nápověda – Databáze-her.cz* [online]. [vid. 2021-08-14]. Dostupné z: https://www.databaze-her.cz/napoveda/herni-zanry-na-databazi-her/

[75] TECHRAPTOR.NET. *Playing Roles: On Tactical-RPGs | TechRaptor* [online]. [vid. 2021-08-16]. Dostupné z: https://techraptor.net/originals/playing-roles-on-tactical-rpgs

[76] *Star Wars™ Galaxy of Heroes - Free Mobile Game - EA Official Site* [online]. [vid. 2021-11-12]. Dostupné z: https://www.ea.com/games/starwars/galaxy-of-heroes

[77] What is an Action/Adventure Game? - Gameranx. *Gameranx* [online]. 2011 [vid. 2021-08-30]. Dostupné z: https://gameranx.com/features/id/3350/article/what-is-an-action-adventure-game/

[78] *Mortal Kombat 11 Ultimate* [online]. [vid. 2021-11-12]. Dostupné z: https://www.mortalkombat.com/

[79] *DOOM Eternal | Bethesda.net* [online]. [vid. 2021-11-12]. Dostupné z: https://bethesda.net/en/game/doom

[80] *Mafia: Trilogy - Home* [online]. [vid. 2021-12-20]. Dostupné z: https://mafiagame.com/cs-CZ/

[81] *The official home of Super Mario™ – History* [online]. [vid. 2021-12-20]. Dostupné z: https://mario.nintendo.com/history/

[82] *Shadow Of The Tomb Raider | SQUARE ENIX* [online]. [vid. 2021-12-20]. Dostupné z: https://tombraider.square-enix-games.com/en-us

[83] *Prince of Persia | Ubisoft (US)* [online]. [vid. 2021-12-20]. Dostupné z: https://www.ubisoft.com/en-us/game/prince-of-persia/prince-of-persia

[84] IDTECH.COM. *Ultimate List of Different Types of Video Games | 49 Genres & Subcategories* [online]. [vid. 2021-08-31]. Dostupné z: https://www.idtech.com/blog/different-types-of-video-game-genres

[85] DESPAIN, Wendy. *Writing for Video Game Genres* [online]. B.m.: A K Peters/CRC Press, 2009. ISBN 9780429063343. Dostupné z: doi:10.1201/b10641

[86] *Warhammer 40,000 - Warhammer 40,000* [online]. [vid. 2021-11-12]. Dostupné z: https://warhammer40000.com/

[87] *Civilization V | Homepage* [online]. [vid. 2023-03-17]. Dostupné z: https://civilization.com/civilization-5/

[88] *Age of Empires Franchise - Official Web Site* [online]. [vid. 2023-03-17]. Dostupné z: https://www.ageofempires.com/

[89] *Cities: Skylines - Paradox Interactive* [online]. [vid. 2023-03-17]. Dostupné z: https://www.paradoxinteractive.com/games/cities-skylines/about

[90] *RollerCoaster Tycoon: Deluxe - RollerCoaster Tycoon - The Ultimate Theme park Sim* [online]. [vid. 2023-03-17]. Dostupné z: https://www.rollercoastertycoon.com/rollercoaster-tycoon-deluxe/

[91] KOŠŤÁL, Filip. Průvodce herními žánry - válečné strategie – Doupě.cz. *Computer* [online]. 2011 [vid. 2021-08-23]. Dostupné z: https://doupe.zive.cz/clanek/pruvodce-hernimi-zanry---valecne-strategie

[92] *DiRT Rally - The official game site* [online]. [vid. 2023-03-18]. Dostupné z: https://dirtgame.com/dirtrally/us/home

[93] *Need for Speed Video Games - Official EA Site* [online]. [vid. 2023-03-18]. Dostupné z: https://www.ea.com/games/need-for-speed

[94] *Home - Redout 2 - The Fastest Racing Game in the Universe* [online]. [vid. 2023-03-18]. Dostupné z: https://redout.games/redout2/

[95] *Bugbear Entertainment | Drive hard* [online]. [vid. 2023-03-18]. Dostupné z: https://bugbeargames.com/

[96] *Asphalt 9: Legends - Arcade Racing | Asphalt Legends* [online]. [vid. 2023-03-18]. Dostupné z: https://asphaltlegends.com/

[97] INDIAN. *Wreckfest - Recenze - YouTube* [online]. [vid. 2023-03-18]. Dostupné z: https://www.youtube.com/watch?v=3_3nvi1vsZ4

[98] *O HŘE | BLACKHOLE :: PC, MAC, LINUX :: 2D Platfomer* [online]. [vid. 2021-12-03]. Dostupné z: https://blackhole-game.com/cs/o-hre

[99] IAMTIMCOREY. *WinForm vs WPF vs UWP vs Console - The C# Desktop UI Showdown (and the future with .NET 5)* [online]. 2019 [vid. 2021-05-03]. Dostupné z: https://www.youtube.com/watch?v=yq0dSkA1vpM

[100] *DirectX graphics and gaming - Win32 apps | Microsoft Docs* [online]. [vid. 2021-11-12]. Dostupné z: https://docs.microsoft.com/en-us/windows/win32/directx

[101] *[MS-XAML]: Xaml Object Mapping Specification 2006 Intellectual Property Rights Notice for Open Specifications Documentation* [online]. 2008 [vid. 2021-10-16]. Dostupné z: http://download.microsoft.com/download/0/A/6/0A6F7755-9AF5-448B-907D-13985ACCF53E/%5BMS-XAML%5D.pdf

[102] MICROSOFT. *Xamarin | Open-source mobile app platform for .NET* [online]. [vid. 2021-10-03]. Dostupné z: https://dotnet.microsoft.com/apps/xamarin

[103] DOCS.MICROSOFT.COM. *Xamarin.Essentials - Xamarin | Microsoft Docs* [online]. [vid. 2021-10-05]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/essentials/?WT.mc_id=dotnet-35129-website

[104] DOCS.MICROSOFT.COM. *Xamarin.Forms Views - Xamarin | Microsoft Docs* [online]. [vid. 2021-10-04]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/xamarin-forms/user-interface/controls/views?WT.mc_id=dotnet-35129-website

[105] MONOGAMES. *About | MonoGame* [online]. [vid. 2021-10-05]. Dostupné z: https://www.monogame.net/about/

[106] STRIDE. *Stride Game Engine* [online]. [vid. 2021-10-08]. Dostupné z: https://www.stride3d.net/

[107] DOCS.MICROSOFT.COM. *What is Xamarin? - Xamarin | Microsoft Docs* [online]. [vid. 2021-10-07]. Dostupné z: https://docs.microsoft.com/en-us/xamarin/get-started/what-is-xamarin

[108] *What's a Universal Windows Platform (UWP) app? - UWP applications | Microsoft Docs* [online]. [vid. 2021-12-03]. Dostupné z: https://docs.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide

[109] DEVELOPER.ANDROID.COM. *Meet Android Studio | Android Developers* [online]. [vid. 2021-10-16]. Dostupné z: https://developer.android.com/studio/intro

[110] DEVELOPER.ANDROID.COM. *Create an Android project | Android Developers* [online]. [vid. 2021-10-16]. Dostupné z: https://developer.android.com/training/basics/firstapp/creating-project

[111] DEVELOPER.ANDROID.COM. *Run apps on the Android Emulator | Android Developers* [online]. [vid. 2021-10-16]. Dostupné z: https://developer.android.com/studio/run/emulator

[112] . *NET MAUI Release Candidate - Ready for cross-platform app development - .NET Blog* [online]. [vid. 2022-10-14]. Dostupné z: https://devblogs.microsoft.com/dotnet/dotnet-maui-rc-1/

[113] *Visual Studio 2022 version 17.3 Release Notes | Microsoft Learn* [online]. [vid. 2023-01-04]. Dostupné z: https://learn.microsoft.com/en-us/visualstudio/releases/2022/release-notes-v17.3

[114] GREGORY, Jason. *Game Engine Architecture, Third Edition* [online]. 2018. ISBN 9781138035454. Dostupné z: doi:10.1201/9781315267845

[115] UNITY TECHNOLOGIES. *Wondering what Unity is? Find out who we are, where we've been and where we're going | Unity* [online]. [vid. 2021-06-21]. Dostupné z: https://unity.com/our-company

[116] UNITY TECHNOLOGIES. *Reimagine product design and development - YouTube* [online]. [vid. 2021-06-21]. Dostupné z: https://www.youtube.com/watch?v=j_bQf0InYHM

[117] UNITY TECHNOLOGIES. *Create immersive experiences for real-world applications at scale | Unity - YouTube* [online]. [vid. 2021-06-21]. Dostupné z: https://www.youtube.com/watch?v=5VRxVVOIoJs

[118] UNITY TECHNOLOGIES. *Real Time Animation: Unity for Look Development - YouTube* [online]. [vid. 2021-06-21]. Dostupné z: https://www.youtube.com/watch?v=urew479-Wlw

[119] UNITY TECHNOLOGIE. *Compare Unity plans: Pro vs Plus vs Free. Choose the best 2D - 3D engine for your project! - Unity Store* [online]. [vid. 2021-06-23]. Dostupné z: https://store.unity.com/compare-plans?currency=USD

[120] *What is the best game engine: is Unity right for you? | GamesIndustry.biz* [online]. [vid. 2021-06-20]. Dostupné z: https://www.gamesindustry.biz/articles/2020-01-16-what-is-the-best-game-engine-is-unity-the-right-game-engine-for-you

[121] BRACKEYS. *How to make a CUSTOM INSPECTOR in Unity - YouTube* [online]. [vid. 2021-06-21]. Dostupné z: https://www.youtube.com/watch?v=RInUu1_8aGw

[122] UNITY TECHNOLOGIES. *Made With Unity | Unity* [online]. [vid. 2021-06-21]. Dostupné z: https://unity.com/madewith

[123] DICKINSON, Brendan. *Unity VS Unreal Engine in 2021 | What is the best Game Engine?* [online]. 2021. Dostupné z: https://www.youtube.com/watch?v=jjUsSL4T3ig

[124] DOCS.UNITY3D.COM. *Unity - Scripting API: MonoBehaviour.FixedUpdate()* [online]. [vid. 2021-09-04]. Dostupné z: https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html

[125] DOCS.UNITY3D.COM. *Unity - Scripting API: Debug* [online]. [vid. 2021-09-19]. Dostupné z: https://docs.unity3d.com/ScriptReference/Debug.html

[126] DOCS.UNITY3D.COM. *Unity - Manual: Prefabs* [online]. [vid. 2021-09-04]. Dostupné z: https://docs.unity3d.com/Manual/Prefabs.html

[127] SOCIAMIX. *10 mins GameDev tips - Quaternions - YouTube* [online]. [vid. 2021-09-20]. Dostupné z: https://www.youtube.com/watch?v=1yoFjjJRnLY

[128] 3BLUE1BROWN. *Quaternions and 3d rotation, explained interactively - YouTube* [online]. 2018 [vid. 2021-09-20]. Dostupné z: https://www.youtube.com/watch?v=zjMuIxRvygQ

[129] MURRAY, Jeff W. *C# Game Programming Cookbook for Unity 3D* [online]. 2. vyd. 2014. ISBN 9781466581401. Dostupné z: doi:10.1201/b17100

[130] HARDMAN, Casey. *Game Programming with Unity and C#* [online]. 2020. ISBN 1484256557. Dostupné z: doi:10.1007/978-1-4842-5656-5

[131] INSIDER. *Why „The Mandalorian" Uses Virtual Sets Over Green Screen | Movies Insider - YouTube* [online]. [vid. 2021-06-24]. Dostupné z: https://www.youtube.com/watch?v=Ufp8weYYDE8

[132] EPIC GAMES. *Real-Time In-Camera VFX for Next-Gen Filmmaking | Project Spotlight | Unreal Engine - YouTube* [online]. [vid. 2021-06-24]. Dostupné z: https://www.youtube.com/watch?v=bErPsq5kPzE

[133] *Gearbox Software's Borderlands More Awesome Than Ever with Unreal Engine 3* [online]. [vid. 2021-07-01]. Dostupné z: https://www.unrealengine.com/en-US/blog/borderlands

[134] *CRYENGINE | Support: Licensing* [online]. [vid. 2021-06-24]. Dostupné z: https://www.cryengine.com/support/view/licensing

[135] TECHNOLOGIES, Unity. *What platforms are supported by Unity? – Unity* [online]. [vid. 2021-06-23]. Dostupné z: https://support.unity.com/hc/en-us/articles/206336795-What-platforms-are-supported-by-Unity-

[136] EPIC GAMES. *Multi-platform development* [online]. [vid. 2021-06-23]. Dostupné z: https://www.unrealengine.com/en-US/features/multi-platform-development

[137] EPIC GAMES. *Download - Unreal Engine* [online]. [vid. 2021-06-22]. Dostupné z: https://www.unrealengine.com/en-US/download

[138] *CRYENGINE | Support: General* [online]. [vid. 2021-06-24]. Dostupné z: https://www.cryengine.com/support/view/general#platform-support

[139] *Design Patterns* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns

[140] *History of patterns* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/history

[141] *Creational Design Patterns* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/creational-patterns

[142] *Structural Design Patterns* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/structural-patterns

[143] *Behavioral Design Patterns* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/behavioral-patterns

[144] *Factory Method* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/factory-method

[145] *Singleton* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/singleton

[146] *Flyweight* [online]. [vid. 2023-03-20]. Dostupné z: https://refactoring.guru/design-patterns/flyweight

[147] *Observer* [online]. [vid. 2023-03-21]. Dostupné z: https://refactoring.guru/design-patterns/observer