Česká zemědělská univerzita v Praze

Technická fakulta

Katedra technologických zařízení staveb



Bakalářská práce

Návrh univerzální programové logiky pro vývoj her

Martin Novák

2022 ČZU v Praze





**Čestné prohlášení**

Prohlašuji, že svou bakalářskou práci "Návrh univerzální programové logiky pro vývoj her" jsem vypracoval(a) samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu použitých zdrojů na konci práce. Jako autor(ka) uvedené bakalářské práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne datum odevzdání \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Poděkování**

Rád(a) bych touto cestou poděkoval(a) jméno vedoucího, případně dalších osob, a informace, za co děkujete.

Návrh univerzální programové logiky pro vývoj her

**Abstrakt**

Souhrn práce (cca 15 řádek textu).

**Klíčová slova:** herní enginy, uživatelská rozhraní, programovací jazyky, objektový model, vývoj herDesign of universal program logic for game development

**Abstract**

Anglický překlad českého souhrnu

**Keywords**: game engines, user interface, programing languages, object model, game development

**Obsah**

[1. Úvod 1](#_Toc130556043)

[2. Cíl práce a metodika 1](#_Toc130556044)

[2.1 Cíl práce 1](#_Toc130556045)

[2.2 Metodika 1](#_Toc130556046)

[3. Výběr vhodných programovacích jazyků pro vývoj her 1](#_Toc130556047)

[3.1 C++ 3](#_Toc130556048)

[3.1.1 Kompilace a hardware 3](#_Toc130556049)

[3.1.2 Novinky oproti C 4](#_Toc130556050)

[3.1.3 Nevýhody 4](#_Toc130556051)

[3.2 Java 4](#_Toc130556052)

[3.2.1 JIT (Just In Time) 4](#_Toc130556053)

[3.2.2 Přístup k paměti a ovládání hardware 5](#_Toc130556054)

[3.2.3 Výhody 5](#_Toc130556055)

[3.2.4 Nevýhody 5](#_Toc130556056)

[3.3 C# 6](#_Toc130556057)

[3.3.1 Microsoft .NET 6](#_Toc130556058)

[3.3.2 Přístup k paměti a ovládání hardware 7](#_Toc130556059)

[3.3.3 Porovnání s Javou 7](#_Toc130556060)

[3.3.4 Podobnosti s C++ 8](#_Toc130556061)

[3.3.5 Modifikátory parametrů metod 8](#_Toc130556062)

[3.3.6 nové funkce 8](#_Toc130556063)

[3.4 výběr 9](#_Toc130556064)

[4. Výběr herních žánrů vhodných pro implementaci 10](#_Toc130556065)

[4.1 RPG 11](#_Toc130556066)

[4.2 akční 12](#_Toc130556067)

[4.3 strategie 12](#_Toc130556068)

[4.4 závodní 13](#_Toc130556069)

[4.5 shrnutí 13](#_Toc130556070)

[5. Grafické výstupy aplikací 14](#_Toc130556071)

[5.1 konzolová aplikace 14](#_Toc130556072)

[5.2 okenní aplikace 15](#_Toc130556073)

[5.2.1 WinForm 15](#_Toc130556074)

[5.2.2 WPF 15](#_Toc130556075)

[6. Návrh aplikačního modelu 15](#_Toc130556076)

[7. Návrh vzorového řešení 15](#_Toc130556077)

[8. Zhodnocení realizace aplikace 15](#_Toc130556078)

[9. Závěr 15](#_Toc130556079)

[Seznam použitých zdrojů i](#_Toc130556080)

[Přílohy x](#_Toc130556081)

**Seznam obrázků**

Odkazovaný seznam obrázků

**Seznam tabulek**

Odkazovaný seznam tabulek

**Seznam použitých zkratek**

Soupis a definování zkratek (vyskytuje-li se jich v textu velké množství)

# 1. Úvod

Text text text text text text text text text text text text text text text text text text text text text text text.

# 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.

Text text text

# 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 (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. Na Obr. 1 je porovnání sumy zapsané pomocí imperativního a funkcionálního jazyku. [1, 2]



Obr. imperativní vs. funkcionální jazyk [2]

Z popisu základních paradigmat je vidět, že jazyk bude vybírán z imperativních jazyků, které se dále dělí na dvě podskupiny. Procedurální (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. 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. [3, 4]



Obr. procedurální vs objektový jazyk-vlastní

## 3.1 C++

C++ je více paradigmatový jazyk 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, ale některé kódy možné napsat v C jsou v C++ neplatné. Se zpětnou kompatibilitou souvisí headery obsahující deklarace proměnných, struktur, tříd a jejich metod, které je potřeba používat i v jiných souborech, což sebou ale nese i nevýhodu, že přidání nových tříd a metod, či změny jejich hlaviček je nutno provádět na dvou místech. [5–8]

### 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), ale existuje také C++/CLI, který je součást Microsoft .NET a je kompilován na bytecode (viz kapitola 3.2.1), což umožňuje mít jednu verzi pro obě architektury a sestavit aplikaci z částí napsaných v různých .NET jazycích (viz Kap. 3.3.1). 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 [9–14].

### 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. C++ neobsahuje vlastní GUI a musíte proto použít některou z knihoven třetí strany [15, 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. 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 [17–19].

### 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, 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. [14, 20].

### 3.2.2 Přístup k paměti a ovládání hardware

Java neumožňuje pracovat s pointery, neboť správu paměti zajišťuje run time. 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. [18, 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. Doba potřebná ke smazání „mrtvých“ objektů zaleží na počtu „živých“ a velikosti paměti [23–25]. 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. 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. Java nemá datový typ pro bezznaménková celá čísla (*uint*), takže je k dispozici pouze polovina rozsahu a pokud je potřeba zapsat hodnotu nad dvě miliardy (231) musí se použít *long* (64bitový). [18, 28]

## 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. [29]

### 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 [13, 30, 31].

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. 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. 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. Další výhodou AOT je, že není potřeba, aby byl překlad co nejrychlejší, takže má dost času provést optimalizace. [13, 31–35]

Velkou výhodou je, že .NET runtime je od Windows Vista součást operačního systému, takže je aktualizován společně se systémem, díky čemuž uživatel nemusí nic instalovat. [36]

Prostředí .NET bylo původně určeno pouze pro platformy Microsoftu (Windows a Xbox), což se změnilo až v roce 2014 vydáním .NET Core, 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. [31, 37]

### 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 [38].

Ačkoliv C# oficiálně neumožňuje ovládání jednočipových počítačů, existují rozšíření třetích stran, jako například nanoFramework [39] nebo placené visualmicro [40], které podporují čipy založené na ARM architektuře.

### 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 [41]. C# dokáže primitivní datové typy (např. *int*) automaticky měnit na objekty [30]. Tak jako Java i C# má Garbage Collector, který za programátora uvolňuje paměť. K jeho spuštění dochází při nedostatku paměti, nebo překročení stanoveného limitu [42, 43]. 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 [38].

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. 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. [44–48]

### 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 [49].

### 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 [50].

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 časti třídy, aniž by ovlivnila programátorův soubor. Příkladem je Windows Form, jehož grafická část je generována Visual Studiem [51].

## 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 tři 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) [16] | ano | ano |
| Přetěžování operátorů | ano | ne | ano |
| Vlastní eventy | Ano [52] | 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 [53] | Unity, CRYENGINE |

Tab. 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). [54]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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. odlišné značení her[55–58][59, 60][61–64]

## 4.1 RPG

Tento žánr vychází z deskových her jako jsou Dungeons & Dragons [65] (zkráceně D&D či DnD) nebo české Dračí doupě [66], 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). [54, 67–70]

První a nejstarší kategorie je již zmíněná digitalizovaná verze deskových her (např. Baldur's Gate [71]), či jejich slovní podoby, 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. [54, 67, 69, 70]

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 [72]) nebo může přepnout ovládání ze své postavy na společníka (např. Star Wars: Knights of the Old Republic [73]). 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 [74]). [54, 67, 68, 70, 75]

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 [76]). [68]

Poslední jsou taktická RPG (TRPG), která kombinují vylepšování postav a strategii (např. XCOM [77]). Tento subžánr je často řazen mezi strategie. [54, 67–70, 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 [78]). [68, 70, 75]

## 4.2 akční

Akční hry je souhrnné označení bojových her, stříleček, plošinovek a ostatních her s rychlým tempem, které nemají vlastní žánr. V bojových hrách si hráč typicky volí postavu ze seznamu a následně s ní bojujete proti soupeři (např. Mortal Kombat [79]). 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 [80]) a TPS (Third Person Shooter), kde je v záběru kamery i postava (např. Mafia [81]). Pod pojmem plošinovka si většina lidí představí 2D hry jako Mario[82], kde se hráč pohybuje po platformách a musí se dostat na konec úrovně. Patří sem i akčnější tituly jako Tomb Raider [83] a Prince of Persia [84], které obsahují i soubojový systém se zbraněmi. [85–87]

## 4.3 strategie

Podobně jako RPG, mají i strategie předlohu ve stolních hrách (např. Warhammer 40 000 [88]). Existují dva způsoby členění. První je podle toho, zda je hra na kola (tahová neboli turn-based) a hráči se střídají (např. Civilization V [89]), nebo vše probíhá v reálném čase (RTS) jako například Age of Empires [90]. 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 [91] a RollerCoaster Tycoon [92]). [68, 70, 93]

## 4.4 závodní

Jedním z možných dělení závodních her je podle jízdního modelu. Realistický (např. DiRT Rally [94]) se snaží co nejvíce napodobit chování skutečných vozů. Arkádový (např. Need For Speed [95]) ubírá realističnost ve prospěch jednodušší hratelnosti. Dalším dělením je, zda se závodí v autech či jiných dopravních prostředcích (např. vesmírné lodě v Redout 2 [96]). Ačkoliv většina her hráče za narážení do soupeřů penalizuje, existují také série jako například FlatOut [97] a Asphalt [98], které za to naopak hráče odměňují a umožňují soupeře zpomalit či úplně vyřadit ze závodu. [70, 99]

## 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 [100]) 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, atd) 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. srovnání Windows UI[101]

## 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. [101]

## 5.2 okenní aplikace

Dnes se běžný uživatel nejčastěji setká s programy, které mají grafické rozhraní a k interakci se využívají 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. Oproti modernějším technologiím má menší spotřebu operační paměti, díky čemuž je vhodnější pro starší počítače. Vykreslování zde provádí pouze CPU, což má za následek pomalejší změny, projevující se problikáváním překreslovaných prvků či zanecháváním stopy při pohybu. Jelikož Windows Forms k vykreslování využívají GDI+ (Graphics Device Interface) je možné okno aplikace zobrazit i při použití vzdálené plochy, která neumožňuje využívat GPU. Výraznou nevýhodou jsou absolutní rozměry a poloha prvků, které zůstávají při změně velikosti okna neměnné a jejich přepočet musí programátor dělat sám. [101–106]

### 5.2.2 WPF

Windows Presentation Foundation zkráceně WPF je modernější typ .NET okenní aplikace využívající DirectX [107], 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.[101]

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.[108]



Obr. příklad reprezentace XAML[108]

## 5.3 Mobilní aplikace

Jelikož posledních pět let trh s mobilními aplikacemi překonává ten počítačový, 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 přistup k souborům). [109]

### 5.3.3 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). Aplikaci je možné vytvořit nativně pro konkrétní platformu nebo cross-platform pomocí *Xamarin.Forms*. API *Xamarin.Essentials* umožňuje pracovat s funkcemi jako jsou poloha a fotoaparát nezávisle na cílové platformě. Grafické rozhraní je možné místo v XAML vytvořit pomocí C#, 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)[110] a Stride (dříve Xenko)[111]. Kromě knihoven určených pro .NET je možné využít i knihovny napsané Objective-C, C/C++ nebo Java. 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) [112–116]

### 5.3.2 Android Studio

Android studio je oficiální IDE a SDK pro vývoj aplikací pro Android od Googlu. K dispozici jsou jazyky Java a Kotlin. Podobně jako v případě Xamarin se k definici grafického rozhraní využívá značkovací jazyk, kterým je XML. 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[117–119].

### 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. V případě potřeby je možné přidat platformě závislí kód. Tento kód se může provádět například spouštění aplikace, které je pro každou platformu odlišné. Další možnost je definovat rozhraní a pro každý systém udělat jinou definici, čímž je možné přistupovat k funkcím, které není možné zobecnit. [37]

Původně bylo datum vydání oznámeno na listopad 2021, ale kvůli technickým problémům bylo odloženo. Nakonec bylo oficiálně vydáno jako součást Visual Studio 2022 verze 17.3 v srpnu 2022. 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. [120–122]

## 5.4 Herní engine

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. 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. 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. [123, 124]

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. Kromě her je možné Unity využít i v průmyslu k vizualizaci modelů vytvořených v CAD programech ve virtuální nebo rozšířené realitě. Dále je Unity využíváno v architektuře, kde usnadňuje návrh budov nebo u tvorby filmů. [125–128]

Pokud zisk za předešlých dvanáct měsíců nepřekročil sto tisíc amerických dolarů, je licence zcela zdarma, poté se platí měsíční nebo roční poplatek za každý počítač, jehož výše se odvíjejí od zisku. 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. 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. Příkladem her vytvořených v Unity jsou Cuphead [129] a Hollow Knight [130]. Oproti konkurenci má výhodu v jednoduchosti, množství platforem a 2D hrách[131–135].

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. 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. 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). 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 (viz kap. 6.1.2)*, 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. [136–142]

### 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[143]. Pokud zisk produktu nepřekročí milion amerických dolarů je licence zdarma, poté se platí pěti procentní podíl z prodeje. Jako programovací jazyk se využívá C++ nebo nástroj Blueprint pro blokové programování. Příkladem her vytvořených v Unreal Engine jsou Fornite[144], který je stejně jako engine od Epic Games a série Borderlands[145]. [146–148]

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.[123]

### 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 dvou porovnávaných enginů není využíván v průmyslu a podporuje méně platforem. U prvních pěti tisíc amerických dolarů ze zisku za rok je poplatek odpuštěn ze zbytku se platí pěti procentní podíl. Jako programovací jazyk je možné použít C++ nebo C#. [149]

### 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 dvanáct 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. srovnání enginů[132, 149–153]

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.

7. Návrh vzorového řešení

Te

8. Zhodnocení realizace aplikace

Te

# 9. Závěr

Text…

# Seznam použitých zdrojů

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

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

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

[12] *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/

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

[14] *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

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

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

[17] *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/

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

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

[24] 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

[25] 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/

[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] . *NET 6 Desktop Dev Options: WPF, WinForms, UWP, .NET MAUI, Blazor ... -- Visual Studio Magazine* [online]. [vid. 2023-03-24]. Dostupné z: https://visualstudiomagazine.com/articles/2021/02/03/net-6-desktop.aspx

[30] 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/

[31] . *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

[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] *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

[36] 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

[37] 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

[38] 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

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

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

[42] 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

[43] 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/

[44] 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

[45] 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

[46] 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/

[47] 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/

[48] 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

[49] 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

[50] 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

[51] 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

[52] DOCS.MICROSOFT.COM. *Event handling in native C++* [online]. [vid. 2021-06-06]. Dostupné z: https://learn.microsoft.com/en-us/cpp/cpp/event-handling-in-native-cpp

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

[54] 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

[55] *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

[56] *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/

[57] *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

[58] *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

[59] *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/

[60] *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

[61] *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

[62] *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

[63] *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/

[64] *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

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

[66] *Dračí doupě* [online]. [vid. 2023-03-24]. Dostupné z: https://www.altar.cz/drd/

[67] 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/

[68] 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/

[69] 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/

[70] *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/

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

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

[73] *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

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

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

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

[78] *Star WarsTM 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

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

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

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

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

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

[84] *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

[85] 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

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

[87] 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/

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

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

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

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

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

[93] 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

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

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

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

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

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

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

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

[101] 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

# Přílohy

Odkazovaný seznam příloh