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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |
| 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 |
| 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 |
| XAML | eXtensible Application Markup Language | Značkovací jazyk pro definici GUI |  |
|  |  |  |  |

Obsah

[1. Úvod 2](#_Toc82591971)

[2. Cíl práce a metodika 2](#_Toc82591972)

[2.1 cíl práce 2](#_Toc82591973)

[2.2 metodika 2](#_Toc82591974)

[3. Výběr vhodných programovacích jazyků pro vývoj her 3](#_Toc82591975)

[3.1 C++ 4](#_Toc82591976)

[3.2 Java 4](#_Toc82591977)

[3.3 C# 5](#_Toc82591978)

[3.4 výběr 6](#_Toc82591979)

[4. Výběr herních žánrů vhodných pro implementaci 7](#_Toc82591980)

[4.1 RPG 7](#_Toc82591981)

[4.2 akční 8](#_Toc82591982)

[4.3 strategie 9](#_Toc82591983)

[4.4 závodní 9](#_Toc82591984)

[5. Grafické výstupy aplikací 9](#_Toc82591985)

[5.1 konzolová aplikace 10](#_Toc82591986)

[5.2 okenní aplikace 10](#_Toc82591987)

[5.2.1 WinForm 10](#_Toc82591988)

[5.2.2 WPF 10](#_Toc82591989)

[5.3 herní engine 11](#_Toc82591990)

[5.3.1 Unity 11](#_Toc82591991)

[5.3.2 Unreal Engine 12](#_Toc82591992)

[5.3.3 CRYENGINE 12](#_Toc82591993)

[5.3.4 shrnutí 12](#_Toc82591994)

[5.4 mobilní aplikace 12](#_Toc82591995)

[5.4.1 Xamarin 12](#_Toc82591996)

[6. Návrh aplikačního modelu 12](#_Toc82591997)

[7. Návrh vzorového řešení 13](#_Toc82591998)

[8. Zhodnocení realizace aplikace 13](#_Toc82591999)

[9. Závěr 13](#_Toc82592000)

# 1. Úvod

// historie her (přelom 80s 90s se dostali na PC), vývoj je pomalý -> vznikají enginy

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

3. Výběr vhodných programovacích jazyků pro vývoj her

Programovací jazyky dělíme na 2 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]

Z popisu základních paradigmat je vidět, že jazyk bude vybírán z imperativních jazyků, které se dále dělí na 2 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 4 základní principy: zapouzdření, abstrakce, dědičnost a polymorfismus. Zapouzdření umožňuje omezit viditelnost proměnných 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.



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]. Stejně jako jazyk C je kompilován pro konkrétní architekturu procesoru a operační systém, takže musí rozlišovat 32bitovou(x86) a 64bitovou verzi operačního systému (x86 dokáže běžet na x64 obráceně ne)[8, 9], ale existuje také C++/CLI, který je součást Microsoft .NET a je kompilován na bytecode, což umožňuje sestavit aplikaci z částí napsaných v různých .NET jazycích (viz. Kap. 3.3)[10]. tak jako C je i C++ díky své schopnosti pracovat přímo s pamětí a registry vhodný pro psaní ovladačů, operačních systémů a řízení jednočipových počítačů[11, 12][13]. standarty C++ neobsahují Garbage Collector, takže se programátor musí starat o alokování a následné uvolňování paměti, ale je možné použít některý vytvořený třetí stranou[14]. Další vlastnost vycházející z C jsou headery obsahující deklarace proměnných, struktur, tříd a jejich metod, které potřebujete používat i v jiných souborech[15], což sebou ale nese i nevýhodu, že změny je nutno provádět na dvou místech. Jakožto objektový jazyk dovoluje přetěžování metod, ale oproti Javě a C# porovnává jen parametry, takže funkce s různým návratovým typem a stejnými parametry považuje za stejné a nepůjdou zkompilovat[15]. Další novinka oproti C jsou *namespace*, které umožňují kód organizovat do menších celků a je tak možné, aby se v projektu vyskytoval stejný název vícekrát[15]. 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[15]. C++ neobsahuje vlastní GUI a musíte proto použít některou z knihoven třetí strany[16].

// nahradit další a umožňuje

## 3.2 Java

Java je objektový jazyk, který oproti C++ není kompilován přímo do strojového kódu, ale do bytecodu, který je spouštěn ve virtuálním stroji (JVM), což umožňuje, aby stejný program bylo možné spustit na všech operačních systémech v 32bitové i 64bitové verzi[8], ale ke spuštění programu musí být na zařízení naistalována odpovídající verze. Nevýhodou bytecodu je jeho výpočetní náročnost, neboť je překládán do strojového kódu v momentě, kdy je spouštěn. Díky just in time (JIT) překladu je možné provést optimalizaci pro konkrétní CPU a tím dosáhnout vyšší rychlosti[17]. Pomocí Java ME Embedded je možné ovládat i jednočipové počítače, ale je podporováno pouze Raspberry Pi Model B a 2 čipy od STMicroelectronics[18]. Oproti C++ má Java Garbage Collector, který se stará o uvolňování paměti mazáním objektů bez reference[19], čímž usnadňuje programátorovi práci, ovšem za cenu občasného zastavení běhu aplikace, což je možné vyřešit přidáním dalšího vlákna[19]. Doba potřebná ke smazání „mrtvých“ objektů zaleží na počtu „živých“ a velikosti paměti[20, 21]. 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. Stejně jako C++ je zde možné využívat přetěžování metod, ale signaturu tvoří kromě parametrů i návratový typ. Oproti C++ neumí přetěžovat operátory[22]. Java nemá datový typ pro bezznaménková celá čísla (*uint*)[22], takže je k dispozici pouze polovina rozsahu a pokud je potřeba zapsat hodnotu nad 2 miliardy (231) musí se použít *long* (64bitový). Dále se zde nenachází pointery, neboť správu paměti zajišťuje run time[22]. 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[23]. Nenachází se zde funkce, ale jen metody, které jsou součástí objektu, nebo jsou statické[22, 24]. Další nevýhoda Javy je, že za generický typ, se kterým se můžeme setkat u *Collection* (např. *ArrayList*) není možné dosadit primitivní datový typ, takže například místo *int* musíme vytvořit objekt *Integer*[25]. Java má pro GUI 2 namespace a to *awt* a odlehčený *swing*[26, 27].

## 3.3 C#

C# je objektový jazyk z rodiny Microsoft .NET, což umožňuje používat knihovny napsané například ve Visual Basic, F#, C++/CLI nebo jiném z více než 20 CTS jazyků.[28] Velkou výhodou je, že .NET runtime je od Windows Vista součást operačního systému, takže je aktualizován společně se systémem[29], díky čemuž uživatel nemusí nic instalovat. Tyto jazyky jsou společně kompilovány do bytecodu označovaného jako MSIL (Microsoft Intermediate Language), který je podobně jako v případě Javy kompilován v momentě spouštění, ovšem ukládá si přeloženou verzi a v případě, že se sestava nezmění, se již nemusí znovu překládat[10]. Ačkoliv C# oficiálně neumožňuje ovládání jednočipových počítačů, existují rozšíření třetích stran, jako například nanoFramework nebo visualmicro (placený), které podporují čipy založené na ARM architektuře[30, 31]. Tak jako Java i C# má Garbage Collector, který za programátora uvolňuje paměť. K jeho spuštění dochází při nedostatku paměti, nebo překročení stanoveného limitu[32][33]. Stejně jako u Javy je zde viditelnost tříd řízena pomocí *namespace*. Při přetěžování metod je signatura dána typem a pořadím parametrů[34] a oproti Javě umí přetěžovat i operátory[34]. C# dokáže primitivní datové typy (např. *int*) automaticky měnit na objekty[28]. U parametru metody je možné použít klíčové slovo *out*, které ho změní na výstupní hodnotu[35], 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[35]. 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 označeného jako *unsafe*. Kód uvnitř toho bloku se podobá tomu, který by se napsal v C++ nebo C[36]. Další podobnost s C++ jsou 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[36]. 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[37]. 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[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[39–41]. Další jejich využití jsou eventy (např. kliknutí na tlačítko), kde metody, které na něj reagují, musí být typu *void* a mít parametry typu objekt a *EventArgs* nebo jeho potomka. První parametr říká, jaký objekt event vyvolal a druhý obsahuje podrobnosti, jako například jaká je poloha kurzoru[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. Prostředí .NET bylo původně určeno pouze pro platformy Microsoftu (Windows a Xbox), což se změnilo až v roce 2016 vydáním .NET Core[44] ovšem s GUI se vývojáři museli spoléhat na třetí strany. V roce 2021 bylo vydáno .NET MAUI umožňující vydat jednu aplikaci na všechny počítačové i mobilní systémy s minimálními zásahy do kódu.[45]

//

Jak již bylo řečeno, jazyky kompatibilní s .NET jsou kompilovány na bytecode označovaný jako MSIL, což je objektový nízko úrovňový jazyk, který umožňuje slučovat kódy napsané v různých CTS jazycích. Tento kód je po částech převáděn do strojového kódu za běhu v okamžiku, kdy je funkce volána. Jelikož překlad stojí procesorový čas, využívá MSIL řešení zvané Native Image Generator (NGen), který ukládá již přeložený obraz kompilovaný pro konkrétní procesor a není tak potřeba při každém spuštění programu volat JIT kompilátor. NGen 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. Obrazy jsou instalovány do Native Image Cache (NIC), kam jsou přidávány i závislosti, které je možno používat více obrazy, čímž se eliminuje duplicita. Nevýhodou NGen jsou vyšší potřeba místa na disku a administrátorského oprávnění pro vytvoření obrazu. [10, 46]

## 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.

|  |  |  |  |
| --- | --- | --- | --- |
|  | C++ | Java | C# |
| Byte code | Ne (C++/CLI ano) | ano | Ano (může být přeložen) |
| Headery | ano | ne | ne |
| Garbage Collector | Ne (možno použít 3. strany)[14] | ano | ano |
| Přetěžování operátorů | ano | ne | ano |
| Vlastní eventy | ano[47] | 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 |
|  |  |  |  |
| Příklad Herních enginů | Unreal, CRYENGINE | Greenfoot, libGDX[48] | 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). [49]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **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[50–53][54, 55][56–59]

//

4.1 RPG

//

Tento žánr vychází z deskových her jako jsou Dungeons & Dragons (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 má 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 rozlišují a z RPG samostatně vyčleňují jen některé. První a nejstarší kategorie je již zmíněná digitalizovaná verze deskových her, či jejich slovní podoby (např. Baldur's Gate), 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) nebo může přepnout ovládání ze své postavy na společníka (např. Star Wars: Knights of the Old Republic). 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). Další kategorie je MMORPG (Massively Multiplayer Online RPG), kde na velkých mapách hrají tisíce hráčů současně, kteří mohou navzájem komunikovat, pomáhat si, obchodovat, bojovat v PvP arénách a spojovat se do aliancí (např. World of Warcraft). Poslední jsou taktická RPG (TRPG), která kombinují vylepšování postav a strategii (např. XCOM), ale tento subžánr je často řazen mezi strategie. [49, 60–63][64] Především na mobilních zařízeních je možné se setkat s TRPG, kde hráč sbírá postavy s různými schopnostmi, které vylepšuje a následně v 5 členných týmech bojuje proti jiným týmům, kde v kampani jich je v rámci jedné úrovně více za sebou (např. Star Wars: Galaxy of Heroes).

Jelikož je tato práce zaměřena právě 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, nastavení hodnot a umělou inteligenci NPC.

4.2 akční

Akční hry je souhrnné označení bojových her, stříleček, plošinovek a ostatních her s rychlým tempem, které nemají vlastní žánr. [65] V bojových hrách si zvolíte postavu ze seznamu a následně s ní bojujete v ringu proti soupeři (např. Mortal Kombat). Ačkoliv se střílečky řadí mezi akční, většinou jsou jako samostatný žánr. Jak je z názvu zřejmé v těchto hrách jde o střílení. Střílečky dělíme podle umístění kamery na FPS (First Person Shooter), která je z pohledu postavy (např. Doom) a TPS (Third Person Shooter), kde je v záběru kamery i postava (např. Mafia). Pod pojmem plošinovka si většina představí 2D hry jako Mario, kde se hráč přeskakováním po platformách musí dostat na konec úrovně, ale patří sem i akčnější tituly jako Prince of Persia Warrior Within, které obsahují i souboje s mečem.[66]

//

Jelikož je tento žánr často kombinován s RPG ve většině případů zde lze využít téměř celá knihovna bez nutnosti výrazných úprav logiky.

//více rozvést

## 4.3 strategie

Podobně jako RPG mají i strategie předlohu ve stolních hrách (např. Warhammer 40 000). Existují dva způsoby členění. První je podle toho, zda je hra na kola (tahová/ turn-based) a hráči se střídají, nebo vše probíhá v reálném čase (RTS). Druhé je podle zaměření jako jsou válečná nebo budovatelská. Když se řekne strategie většina lidí si vybaví válečnou, kde hráč těží suroviny na stavbu budov a výrobu jednotek, kterými se snaží porazit nepřátelskou armádu.[67] V budovatelských je cílem postavit vlastní město nebo zábavní park, kde je třeba vyřešit logistiku, zajistit zisk na další rozvoj a starat se o spokojenost lidí.[61]

//

Z knihovny je možné využít třídu *Postava* na jednotky a budovy nebo *GameManager* na generování náhodné mapy.

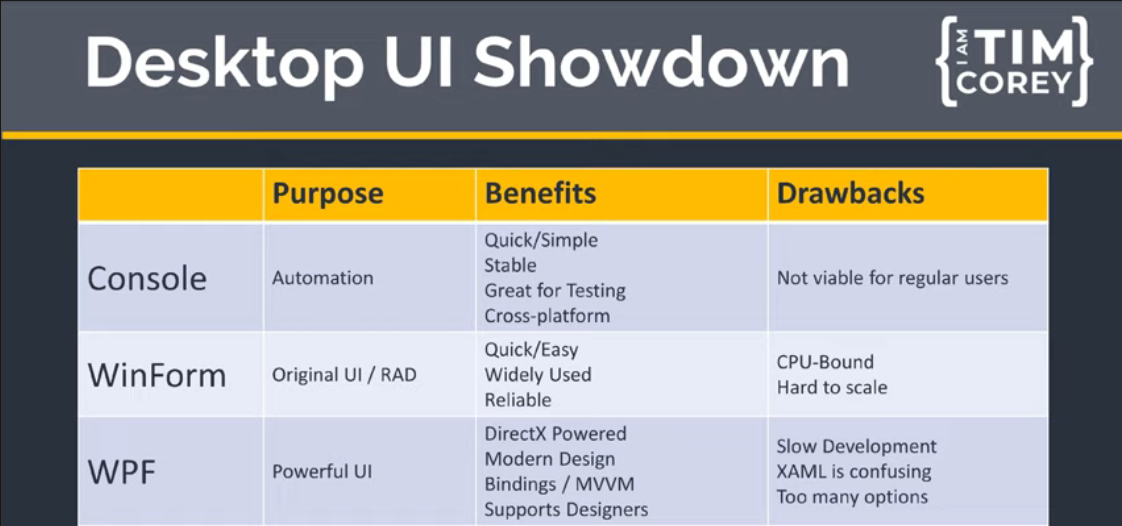
## 4.4 závodní

Jak již název napovídá tyto hry jsou o závodění v autech, či jim podobných strojích. Především u starších a dětských her si pouze vyberete vozidlo a všichni závodníci tak mají stejné podmínky, ale ve většině her za závod získáváte peníze, za které si svůj vůz vylepšujete nebo kupujete lepší, čímž se postupem hrou stávají vozidla rychlejší a hra náročnější. Dále zde jsou hry jako série FlatOut, kde je možné nárazy soupeřovo auto poškodit či úplně zničit a vyřadit ho tak ze závodu.

Z knihovny je možné 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. 2 jsou v několika bodech shrnuté rozdíly .NET UI, které jsou níže rozvinuty.



Obr. srovnání UI[68]

## 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. Je možné mít aplikaci s GUI, která spouští konzolové aplikace a následně zobrazuje jejich návratovou hodnotu.[68]

## 5.2 okenní aplikace

Dnes se většinou setkáme 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 probrány 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, ale 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[68].

### 5.2.2 WPF

Windows Presentation Foundation zkráceně WPF je modernější typ .NET okenní aplikace využívající DirectX, 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.[68]

## 5.3 herní engine

//lepší název?

Jelikož je vývoj hry dlouhý a náročný proces, je každé usnadnění vítané, a proto existují herní enginy, které za programátora řeší fyziku, světlo, vykreslování a další. Velké společnosti si často pro své hry vytváří vlastní, který je používán interně (např. Anvil od Ubisoftu), nebo dán k dispozici ostatním vývojářům (např. CRYENGINE od Crytek). Také existují společnosti, kde je engine hlavní produkt nikoliv vedlejší (např. Unity). Renderování je téměř vždy prováděno 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.

//zmínky o Anvil jsem našel jen na wiki

### 5.3.1 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[69]. Kromě her je možné Unity využít i v průmyslu k vizualizaci modelů vytvořených v CAD programech ve virtuální nebo rozšířené realitě[70]. Dále je Unity využíváno v architektuře, kde usnadňuje návrh budov[71] nebo u tvorby filmů[72]. 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[73]. 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[74]. 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[75]. Příkladem her vytvořených v Unity jsou Cuphead a Hollow Knight[76]. Oproti konkurenci má výhodu v jednoduchosti, množství platforem a 2D hrách[77].

//tyto odstavce jsou podle jednotlivých knih z doporučené literatury + rozvinutí informací a zatím slouží spíše jako výpisky, ze kterých následně budu skládat 1 výsledný odstavec (zatím jsou zde duplicity)

/\*V Unity jsou scény tvořeny *GameObjecty*, které mají vlastnost *Transform* obsahující informaci o poloze, natočení a měřítku (kolikrát je zvětšen původní model) objektu. Z objektu je možné vytvořit Prefab, který obsahuje všechny komponenty včetně nastavených hodnot a potomky (když je objekt tvořen více objekty, které se chovají jako jeden), což urychluje práci, jelikož ho stačí přidat do scény a nastavit umístění[78]. Logika se objektům přidává pomocí skriptů, které by mezi předky měli mít *MonoBehaviour*, čímž zdědí metody jako jsou *Start*, *Update*, *FixedUdate* a *LateUpdate*, které mají pevně dáno, kdy se budou za běhu hry zavolány. Start je zavolán při vytvoření objektu a obvykle se zde řeší načtení hodnot z managera nebo komponent. Update je volán při každém vykreslování nového snímku, což sebou přináší riziko v případě nestability FPS. Pro vypočet fyziky a ostatní úkony vyžadující konstantní interval je lepší využít *FixedUpdate*, který je defaultně volán padesátkrát za vteřinu[78]. LateUpdate je volán až po vykreslení snímku, kdy jsou dokončeny výpočty fyziky[79]. Unity má třídu *Time*, kde lze nastavit rychlost běhu hry a zjistit uplynulý čas. Pomocí statické třídy Gizmo je možné přikreslovat do obrazu pomocné útvary. [79] Na správu ostatních objektů se využívají manageři, což jsou prázdné GameObjecty obsahující jen skript, u kterého je vhodné využít návrhový vzor singleton, aby se zabránilo situaci, kdy je omylem vytvořeno více než jedna instance.[79]

Úrovně hry jsou nazývány scény. Každá scéna obsahuje seznam *GameObjectů*, které se v ní nachází. *GameObject* má své komponenty, které určují jeho vlastnosti (např. poloha, gravitace nebo logika). Komponenty mají své inspektory, kde je možné pohodlně měnit hodnoty. Všechny *GameObjecty* musí mít vlast *Transform*, která obsahuje informace o poloze. Rozměr je násobek velikosti modelu. Objektu je možné přiřadit potomky, jejichž poloha se nyní nevztahuje ke středu scény, ale k rodiči. Toto umožňuje vytvořit skupinu objektů, které se pohybují jako jeden celek. Tohoto lze využít například když má postava v ruce předmět. Z *GameObjectu* je možné vytvořit *prefab*, který obsahuje všechny jeho komponenty včetně nastavených hodnot a v případě potřeby provést změnu stačí upravit hodnotu jen na jednom místě. Pro usnadnění vývoje je možné využít statickou třídu *Debug*, která umožňuje v reálném čase psát do konzole nebo kreslit čáry do snímku[80]. Metoda *Update* je volána při vykreslení každého snímku, ale tato hodnota se může během hry měnit a pro časování je nespolehlivá. K určení času uplynulým mezi jednotlivými snímky slouží statická proměnná *Time.deltaTime*. Pro zvýšení přehlednosti inspektora je klíčových slov umístěných do hranatých závorek přidávat nadpisy či skrývat nepotřebné hodnoty. Pro výpočet natočení objektu Unity využívá *Quaternion* (4D vektor)[81]. K omezení objektů, které se budou vykreslovat, nebo u kterých se bude vyhodnocovat kolize je možné využít vlastnost *Layer*. Ve skriptech je možné načítat jak komponenty patřící objektu, kde se skript nachází, tak u jeho potomků a rodičů. Rychlost hry je možné nastavit pomocí *Time.timeScale,* která určuje kolika násobkem času hra poběží. Unity disponuje vlastní umělou inteligencí pro hledání cesty (pathfinding) využívající *NavMash*, což je mřížka určující kudy je možné projít.[82] \*/

Úrovně hry jsou 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[78]. 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[80][83]. 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) [81, 84]. 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.[79, 82]

### 5.3.2 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[85, 86]. 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[87].

### 5.3.3 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.[88] Jako programovací jazyk je možné použít C++ nebo C#.

### 5.3.4 shrnutí

V Tab. 2 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. srovnání enginů[89, 90][73, 91][88][73][92][87]

## 5.4 mobilní aplikace

### 5.4.1 Xamarin

6. Návrh aplikačního modelu

7. Návrh vzorového řešení

8. Zhodnocení realizace aplikace  
Při odhadu času potřebného k vypracování práce jsem podcenil hledání zdrojů pro rešeršní část.

//

# 9. Závěr

[1] imperative programming [online]. nedatováno [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] *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

[9] *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/

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

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

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

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

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

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

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

[17] *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

[18] 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

[19] 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/

[20] 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

[21] 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

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

[23] 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

[24] *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/

[25] 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

[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] 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/

[29] 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

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

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

[32] 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

[33] 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/

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

[35] 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

[36] 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

[37] 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

[38] 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

[39] 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

[40] 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

[41] 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/

[42] 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/

[43] 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

[44] *Download .NET Core 1.0 (Linux, macOS, and Windows)* [online]. [vid. 2021-09-23]. Dostupné z: https://dotnet.microsoft.com/download/dotnet/1.0

[45] *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

[46] 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

[47] 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

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

[49] 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

[50] *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

[51] *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/

[52] *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

[53] *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

[54] *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/

[55] *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

[56] *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

[57] *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

[58] *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/

[59] *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 Edition&source=detail

[60] 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/

[61] *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/

[62] 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/

[63] *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/

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

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

[66] *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

[67] 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

[68] 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

[69] 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

[70] UNITY TECHNOLOGIES. *Reimagine product design and development - YouTube* [online]. [vid. 2021-06-21]. Dostupné z: https://www.youtube.com/watch?v=j\_bQf0InYHM

[71] 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

[72] 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

[73] 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

[74] *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

[75] 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

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

[77] 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

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

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

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

[81] 3BLUE1BROWN. *Quaternions and 3d rotation, explained interactively - YouTube* [online]. 2018 [vid. 2021-09-20]. Dostupné z: https://www.youtube.com/watch?v=zjMuIxRvygQ

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

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

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

[85] 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

[86] 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

[87] *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

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

[89] 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-

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

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

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