Už zkopírováno do final změna od poslední verze

|  |  |  |
| --- | --- | --- |
| zkratka | význam | vysvětlení |
| RPG | Role Playing Game | Herní žánr (viz. Kap. 4.1) |
| OOP | Object Orienting Programming | Programovací paradigma (viz. Kap 3) |
| JVM | Java Virtual Machine | Virtuální prostředí pro Javu (viz. Kap 3.2) |
| JIT | Just In Time | Překlad v momentě spouštění (viz. Kap. 3.2 a 3.3) |
| CTS | Common Type Specification | Jazyky kompilovatelné do MSIL (viz. Kap. 3.3) |
| MSIL | Microsoft Intermediate Language | .NET bytecode (viz Kap. 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 |
| FPS | First Person Shooter | Herní žánr (viz. Kap. 4.2) |
| TPS | Third Person Shooter | Střílečka z třetí osoby (viz. Kap. 4.2) |
| RTS | Real Time Strategy | Herní žánr (viz. Kap. 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 (viz. Kap. 5.2.2) |
| XAML | eXtensible Application Markup Language | Značkovací jazyk pro definici GUI |
|  |  |  |

Obsah

[1. Úvod 2](#_Toc76037277)

[2. Cíl práce a metodika 2](#_Toc76037278)

[2.1 cíl práce 2](#_Toc76037279)

[2.2 metodika 2](#_Toc76037280)

[3. Výběr vhodných programovacích jazyků pro vývoj her 2](#_Toc76037281)

[3.1 C++ 4](#_Toc76037282)

[3.2 Java 5](#_Toc76037283)

[3.3 C# 6](#_Toc76037284)

[3.4 výběr 6](#_Toc76037285)

[4. Výběr herních žánrů vhodných pro implementaci 7](#_Toc76037286)

[4.1 RPG 7](#_Toc76037287)

[4.2 akční 8](#_Toc76037288)

[4.3 strategie 8](#_Toc76037289)

[4.4 závodní 8](#_Toc76037290)

[5. Grafické výstupy aplikací 8](#_Toc76037291)

[5.1 konzolová aplikace 8](#_Toc76037292)

[5.2 okenní aplikace 9](#_Toc76037293)

[5.2.1 WinForm 9](#_Toc76037294)

[5.2.2 WPF 9](#_Toc76037295)

[5.3 herní engine 9](#_Toc76037296)

[5.3.1 Unity 9](#_Toc76037297)

[5.3.2 Unreal Engine 10](#_Toc76037298)

[5.3.3 CryEngine 10](#_Toc76037299)

[5.3.4 shrnutí 10](#_Toc76037300)

[5.4 mobilní aplikace 10](#_Toc76037301)

[5.4.1 Xamarin 10](#_Toc76037302)

[6. Návrh aplikačního modelu 11](#_Toc76037303)

[7. Návrh vzorového řešení 11](#_Toc76037304)

[8. Zhodnocení realizace aplikace 11](#_Toc76037305)

[9. Závěr 11](#_Toc76037306)

# 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 budeme vybírat 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. Objektové (např. Java) mají třídy sloužící jako předlohy pro instance nazývané objekty, které mají proměnné a metody svázané dohromady a metody tak nepotřebují předávat všechna data parametry, ale může si je načíst z objektu, ve kterém se nachází. Další výhoda je možnost omezit co bude vidět mimo objekt, nebo pomocí zapouzdření kontrolovat přístup a ověřit správnost nově ukládané hodnoty. Dále je možné využít dědičnost, kdy potomek má všechny proměnné a metody rodiče a může přidat další, nebo upravit chování, což umožňuje do listu vkládat rodiče i potomky. Je evidentní, že pro hry se nejvíce hodí objektové jazyky, a proto se na ty nejpoužívanější nyní podíváme blíže. \*/

Z popisu základních paradigmat je vidět, že budeme vybírat 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 dědičnost třída, která je potomek získá všechny proměnné a metody rodiče, ale je možné přidat nové. Polymorfismus souvisí s dědičností, kde do proměnné typu rodiče je možné vložit potomka, ale při volání metody se zavolá přetížená, 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 se na ty nejpoužívanější nyní podíváme blíže.



Obr. procedurální vs objektový jazyk-vlastní

## 3.1 C++

/\*C++ je rozšířením jazyka C o objekty a je proto zpětně kompatibilní, ale některé funkce mají v C++ jinou syntaxi. Stejně jako C je kód kompilován pro konkrétní hardware a operační systém, takže program proto musí mít verze pro x86 i x64, ale je vhodný pro ovladače, operační systémy nebo řízení jednočipových počítačů. Další vlastnost vycházející z C jsou headery obsahující deklarace, které chcete použít v jiných souborech, kvůli čemuž ale musíte změny provádět na 2 místech. Oproti C přináší namespace, které umožňují kód organizovat do menších celků a je možné, aby se v projektu vyskytovali 2 třidy, funkce nebo globální proměnné se stejným jménem. Další novinka je přetěžování funkcí, což znamená, že v 1 souboru mohou být 2 funkce se stejným jménem lišící se počtem parametrů. Programátor si musí dávat pozor na alokování a následné uvolňování paměti. \*/

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 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[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 chcete používat i v jiných souborech[15], což sebou ale nese i nevýhodu, že změny musíte provádět na 2 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 čistě objektový jazyk, ~~což znamená,~~ že všechny metody jsou součást objektu, nebo jsou statické a volají se pomocí třídy, kde se nachází. Oproti C++ nevyužívá headery a pro použití třídy v jiné stačí, aby byli ve stejném namespace, nebo na příslušný namespace přidat referenci. Kód je kompilován na byte code, který běží ve virtuálním stroji, což umožňuje, aby stejný program bylo možné spustit na x86 i x64 nebo i jiném operačním systému, ale k spuštění musí mít uživatel nainstalovanou odpovídající verzi Javy. Další rozdíl je Garbage Collector, který průběžně maže objekty bez reference, což je ale výpočetně náročné, zpomaluje běh programu a je třeba dvojnásobné množství paměti. Java neumí vytvářet listy hodnotových datových typů jako je například int a pro hodnoty musíte vytvořit objekty typu Integer, které do listu následně vložíte. \*/

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 k 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, ale 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, ale za cenu občasného zastavení běhu aplikace, což je možné vyřešit přidáním dalšího vlákna[19]. doba potřebná k smazání „mrtvých“ objektů zaleží na počtu „živých“ a velikosti paměti[20, 21]. Na rozdíl od C++ Java nepoužívá headery a pro použití třídy v jiné stačí, aby se nacházeli ve stejném namespace, nebo na příslušný namespace přidat referenci. Stejně jako C++ je zde možné využívat přetěžování metod, ale signaturu tvoří kromě parametrů a návratový typ. Oproti C++ neumí přetěžovat operátory[22]. Java nemá datový typ pro bezznaménková celá čísla (uint)[22], takže máte k dispozici pouze polovinu rozsahu a pokud potřebujete zapsat hodnotu nad 2 miliardy (231) musíte použít long (64bitový). Dále se zde nenachází pointery a o správu paměti zajišťuje run time[22]. Nenachází se zde funkce, ale jen metody, které jsou součástí objektu, nebo jsou statické[22, 23]. 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[24]. Java má pro GUI 2 namespace a to awt a odlehčený swing[25, 27].

## 3.3 C#

/\*C# je kombinace C++ a Javy, ale přináší i vlastní funkce. Stejně jako Java je čistě objektový, nepoužívá headery, je kompilován na byte code a má Garbage Collector. Z C++ má přetěžování operátorů a klíčová slova. C# patří do rodiny Microsoft .NET, což umožňuje používat knihovny napsané pro Visual Basic nebo F# a .NET runtime je od Windows Vista součást systému[26], takže je aktualizován společně se systémem, díky čemuž uživatel nemusí nic instalovat. Ačkoliv se v některých situacích může zdát zdlouhavější na psaní, než Java má to své výhody, protože díky slovům navíc je možné se v kódu rychleji zorientovat a poznáte přetěžování metody rodičovské třídy.

~~Další výhoda oproti Javě je, že situace, které by neměli nastat a v Javě způsobují logické chyby, jako například 2 vlákna současně přistupující ke stejné proměnné, hází Exception.~~

Oproti Javě má klíčové slovo out u parametru, umožňující metodě vracet bool, zda proběhla v pořádku a výsledek vložit do výstupní proměnné, nebo vracet více hodnot současně bez použití pole objektů, ze kterého se poté musí vložit do příslušných proměnných.

Pomocí ref se dá hodnotový typu místo hodnotou předávat referencí. V C++ se pro tyto účely používají pointery.

Na rozdíl od C++ a Javy má C# properties, umožňující zabalit get a set pod 1 název, se kterým se při volání pracuje jako s proměnnou.

Další novinka jsou patrial class umožňující rozdělit třídu na více souborů, což se nejčastěji využívá u WinForm, kde je grafická část generovaná IDE oddělena od logické části tvořené programátorem. Pomocí této funkce se dá zvýšit přehlednost velkých tříd rozdělením na menší logické celky a zjednodušuje práci u týmových projektů, kde každý programátor může pracovat na své části, aniž by omezoval kolegu. \*/

C# je objektový jazyk z rodiny Microsoft .NET, což umožňuje požívat knihovny napsané například ve Visual Basic, F#, C++/CLI nebo jiném z více než 20 CTS jazyků[28] a .NET runtime je od Windows Vista součást systému, takže je aktualizován společně se systémem[26], 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, ale ukládá si přeloženou verzi a v případě, že se sestava nezmění, již se nemusí znovu překládat[10]. Ačkoliv C# oficiálně neumožňuje ovládání jednočipových počítačů, existují rozšíření třetích stran, jako například nanoFramework nebo visualmicro (placený), které čipy založené na ARM[29, 30]. Tak jako Java i C# má Garbage Collector, který za programátora uvolňuje paměť. Jako spouštěče jsou nedostatek paměti, nebo překročení stanoveného limitu[31][32]. Stejně jako Java je viditelnost tříd dána namespace. Při přetěžování metod je signatura dána návratovým typem a parametry[33] a oproti Javě umí přetěžovat i operátory[33]. 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[34], což umožňuje vracet více než 1 hodnotu bez nutnosti použít pole objektů, ze kterého by se poté postupně přiřazovali do příslušných proměnných, nebo vracet bool, zda metoda proběhla úspěšně, a hodnotu předávat výstupním parametrem. Dále je možné využít modifikátory ref, který mění hodnotovou proměnnou na referenční, a in, který brání úpravám hodnoty[34]. na rozdíl od Javy je v C# možné využívat i pointery a obcházet tak správce paměti, což může vylepšit výkon a 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[35]. Další podobnost s C++ jsou struktury, které by se dali označit jako hodnotová verze objektu, ale mají omezené možnosti. Například nemohou mít hodnotu null, používat dědičnost a mít proměnné inicializované při deklaraci[35]. Mezi novinky, které C# přináší patří properties, umožňující zabalit get a set pod 1 název, se kterým se při volání pracuje jako by se jednalo o proměnnou[36]. 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 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 váš soubor. Příkladem je Windows Form, jehož grafická část je generována Visual Studiem[37]. podobně jako má C++ pointery na funkce v C# jsou delegáti, kteří slouží k předávání metod v parametru, nebo dynamicky měnit volanou funkci. Delegáty je možné sloučit do MulticastDelegate, který obsahuje jejich seznam a při volání je postupně provádí[38–40]. Další 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[41, 42]. Ačkoliv C# v některých situacích vyžaduje oproti Javě další klíčová slova, čímž působí jako pomalejší na psaní, snižuje to množství chyb a urychluje orientaci v kódu, protože je na první pohled vidět přetěžování při dědičnosti a použité modifikátory.

//má C# cross platform GUI?

//MSIL + JIT/NGen (C++/CLI strana 28)

J

## 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 je Garbage Collector nevýhoda, 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 jednoduchou 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[43] | 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 |
|  |  |  |  |
| Herní enginy | Unreal, CryEngine | Greenfoot, libGDX[44] | Unity, CryEngine |

Tab. 1 porovnání jazyků

4. Výběr herních žánrů vhodných pro implementaci

Hry dělíme na několik žánrů, podle základních mechanik. Některé hry spadají do více žánrů, což se nejčastěji stává u RPG, které bývají současně akční.

//stručné představení žánru + mechaniky a zhodnotit, zda jde moje knihovna použít

4.1 RPG

Tento žánr vychází z deskových her jako jsou Dungeons & Dragons nebo české Dračí doupě, kde hráč nebo skupina hráčů hraje za postavy, které mají různé třídy (např. mág nebo válečník), inventář, statistiky a schopnosti, a plní úkoly. Hra se nejčastěji odehrává ve fantasy světě. V deskové verzi se hází kostkami na náhodu, což je v digitální podobě nahrazeno generátorem náhodných čísel, nebo je vynecháno a šance je vždy 100 %. RPG má 2 hlavní podkategorie. První a nejstarší kategorie je již zmíněná digitalizovaná verze deskových her, které jsou na tahy a hráč má čas promyslet si své další kroky. Původně byli pouze textové, kde hráč vybíral z možností a později se začali objevovat i grafické verze[45]. Druhá a větší kategorie jsou akční RPG, kterou jsou v reálném čase a hráč musí rychle reagovat na akce nepřátel. Hráč hraje za 1 postavu, která v některých titulech má společníky, nad kterými hráč nemá kontrolu (např. The Elder Scrolls V: Skyrim) nebo jim může zadávat rozkazy (např. Star Wars: Knights of the Old Republic). Často mají otevřený svět, což umožňuje hráči volně se pohybovat po celé herní mapě, prozkoumávat ji a plnit úkoly v libovolném pořadí. Další výrazně menší kategorie vyskytující se především na mobilních zařízeních je kombinace RPG a tahové strategie, kde hráč sbírá postavy, které vylepšuje a následně v 5členných týmech bojuje proti jiným týmům (např. Star Wars: Galaxy of Heroes).

//najít nějaký článek ať to není osobní způsob členění

//JRPG + MMORPG

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 a reakce na Eventy.

4.2 akční

Akční hry je souhrnné označení bojových her a stříleček.

//bojové + střílečky (je to spisovné?)

//jednak jsou často dávány pod 1 žánr a implementace je téměř stejná

## 4.3 strategie

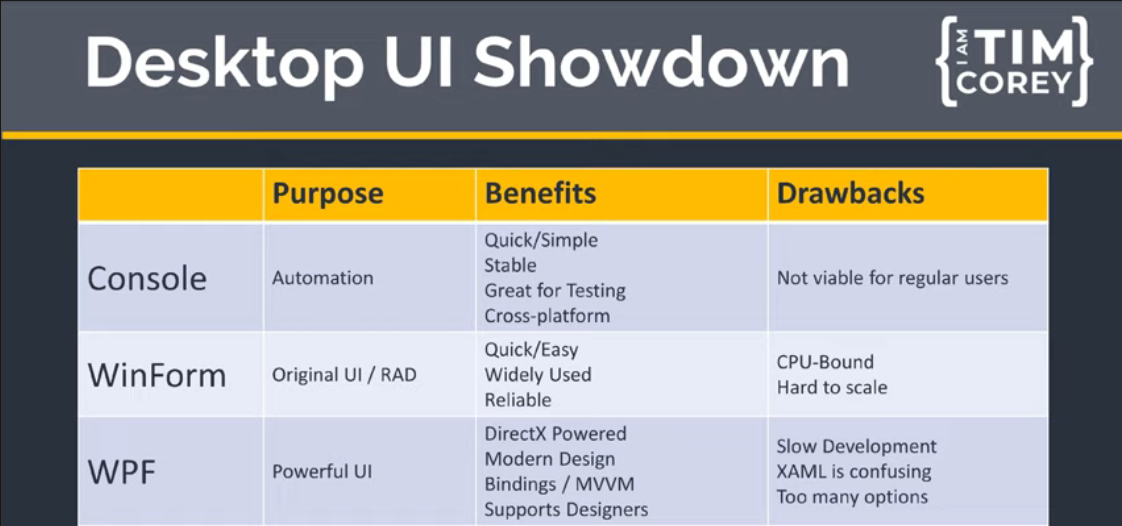
//RTS, tahové, budovatelské

## 4.4 závodní

//jdou požít StatList (max. rychlost, akcelerace, …), Predmet (upgrade) a u her typu FlatOut Postava (poškozování soupeřů), ale bude spoustu nevyužitého -> lepší udělat si třídy na míru

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. 3 srovnání UI[46]

//obrázek=screen z videa kde bylo ještě UWP, které zaniklo stejně rychle jako vzniklo (pouze přes MS store)

## 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 pouští konzolové aplikace a následně zobrazuje jejich návratovou hodnotu.[46]

## 5.2 okenní aplikace

Dnes se většinou setkáme s programy s GUI, které mají tlačítka, textová pole, sbalovací menu a další grafické prvky. Takovéto programy nazýváme okenní aplikace. Jelikož jsme jako jazyk zvolili C# podíváme se blíže na 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 technologie je stále využíván kvůli svému rychlému a jednoduchému vývoji. Je vykreslován čistě pomocí CPU, což má za následek pomalejší změny a vyšší výpočetní náročnost, ale nepotřebuje grafické ovladače a je viditelný 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[46].

### 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.[46]

## 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 si nejprve musíte rozmyslet jaký jazyk chcete použít a jak velký očekáváte zisk. 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[47]. Kromě her je možné Unity využít i v průmyslu k vizualizaci modelů vytvořených v CAD programech ve virtuální nebo rozšířené realitě[48]. Dále je Unity využíváno v architektuře, kde usnadňuje návrh budov[49] nebo u tvorby filmů[50]. Pokud váš zisk za předešlých 12 měsíců nepřekročil 100 000 dolarů je licence zcela zdarma poté se platí měsíční nebo roční poplatek za každý počítač odvíjející se od zisku[51]. 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[52]. Modely si můžete vytvořit samy například v Blenderu, stáhnout z internetu, nebo použít Asset Store integrovaný přímo v prostředí Unity, kde najdete modely, efekty, scripty a nástroje, které jsou zdarma nebo placené. Pro usnadnění vývoje si můžete vytvořit i vlastní nástroje, kterými můžete za běhu měnit hodnoty a v reálném čase vidíte změny[53]. Příkladem her vytvořených v Unity jsou Cuphead a Hollow Knight[54]. Oproti konkurenci má výhodu v jednoduchosti, množství platforem a 2D hrách[55].

### 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[56, 57]. Pokud zisk produktu nepřekročí 1 000 000 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[58].

### 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 dolarů ze zisku za rok je poplatek odpuštěn ze zbytku se platí 5% podíl.[59] 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.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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[60, 61][51, 62][59][51][63][58]

## 5.4 mobilní aplikace

### 5.4.1 Xamarin

6. Návrh aplikačního modelu

7. Návrh vzorového řešení

8. Zhodnocení realizace aplikace

# 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] *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/

[24] 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

[25] 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

[26] 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

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

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

[31] 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

[32] 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/

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

[34] 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

[35] 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

[36] 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

[37] 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

[38] 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

[39] 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

[40] 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/

[41] 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/

[42] 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

[43] 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

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

[45] 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/

[46] 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

[47] 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

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

[49] 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

[50] 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

[51] 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

[52] *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

[53] 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

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

[55] 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

[56] 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

[57] 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

[58] *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

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

[60] 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-

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

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

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