VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VIRTUÁLNÍ SVĚT
VIRTUAL WORLD

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. Robert Kováč
AUTHOR

VEDOUCÍ PRÁCE — Ing. Jan Roupec, Ph.D.
SUPERVISOR

BRNO 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Robert Kováč**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**Ing. Jan Roupec, Ph.D.**
Akademický rok:	2015/16

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Virtuální svět

Stručná charakteristika problematiky úkolu:

Cílem projektu je vytvořit aplikaci, spadající do kategorie online hry pro více hráčů s využitím herního enginu Unity3d. Aplikace bude rozdělena na 3 hlavní části: server, svět určený pro pohyb hráčů a závodní hra, odehrávající se v tomto světě. Za základ světa bude použit areál FSI v Brně a okolí.

Cíle diplomové práce:

1. Seznamte se s problematikou tvorby počítačových her.
2. Seznamte se s herním enginem Unity3d.
3. Navrhněte motiv hry a herní logiku.
4. Naprogramujte navrženou hru pro připojené hráče z herního světa.

Seznam literatury:

Blackman, S.: Beginning 3D Game Development with Unity 4. Springer, New York, 2013.

Jirkovský, J. a kol.: Game Industry : Vývoj počítačových her a kapitoly z herního průmyslu. D.A.M.O., 2011.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Cílem této práce je popsat problematiku tvorby počítačových her a následně pomocí herního enginu Unity 3D navrhnout a vytvořit závodní hru pro hráče připojené ve virtuálním světě. První část práce se věnuje historii videoher, platformám, popisuje postup vývoje a vývojové nástroje. Další část pojednává o enginu Unity 3D, který byl použit při vývoji. Výsledná aplikace je popsaná v poslední části práce.

ABSTRACT

The goal of this thesis is to describe the issue of computer game creation and to develop a race game for people connected to the Virtual World using Unity 3D game engine. The first part of this thesis is dedicated to the history of video games, platforms and to describing of development processes and development tools. The next part is about Unity 3D game engine, which was used for creating the game. The resulting application is described in the last part.

KLÍČOVÁ SLOVA

Počítačové hry, Unity 3D, Tvorba PC her, závodní hra

KEY WORDS

Computer games, Unity 3D, PC games creation, racing game

PROHLÁŠENÍ O ORIGINALITĚ

Prohlašuji, že jsem tuto práci vypracoval samostatně, pod vedením pana Ing. Jana Roupce, Ph.D.

……
Robert Kováč
24. května 2016

BIBLIOGRAFICKÁ CITACE

KOVÁČ, R. *Virtuální svět.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2016. 62 s. Vedoucí diplomové práce Ing. Jan Roupec, Ph.D.

Poděkování

Děkuji vedoucímu diplomové práce panu Ing. Janu Roupcovi, Ph.D. za odbornou pomoc a cenné rady při zpracování mé diplomové práce.

Obsah

1 Úvod

Hlavním cílem této práce je vytvořit pomocí moderních technologií online počítačovou hru, která bude propagovat fakultu strojního inženýrství VUT. Jedná se o rozsáhlejší projekt, který byl rozdělen na 3 části pro 3 různé diplomové práce:

- Virtuální svět
- **Závodní hra** (předmět této diplomové práce)
- Server a komunikace

Tyto části by měly tvořit funkční celek a vývojáři tedy musí pracovat společně jako tým a své části od začátku vyvíjet a připravovat pro vzájemnou komunikaci. Konkrétně bylo nutné si předem stanovit jaká data, v jaké situaci a jakým způsobem si budou vývojáři vzájemně sdílet. Většina komunikace se odehrává prostřednictvím serveru, ale některá data putují i mezi světem a hrou.

Jako celek by měl projekt vytvořit aplikaci, která komunikuje přes server. Hráč se zaregistruje a získá přístup k přihlášení do virtuálního světa. Ten je reprezentován modelem areálu VUT FSI k propagaci školy, kde se hráč může volně pohybovat a komunikovat s ostatními připojenými hráči. Ve světě pak existuje panel pro přihlášení do závodní hry. Je navržena pro čtyři osoby přihlášené ve světě, a jakmile se kapacita naplní, spustí se závod, který se opět odehrává v areálu školy. Hra se svým typem řadí do online závodních her s arkádovými prvky.

Samotnému vývoji předchází několik fází. S pomocí našich zkušeností s hraním počítačových her a společných diskuzí, jsme si vytvořili představu výsledného konceptu, který byl následně sepsán v game design dokumentu (viz 2.3). Další fází bylo vybrat vývojové nástroje a po průzkumu dostupných možností jsme jako hlavní nástroj vybrali herní engin Unity3D a pro modelování grafický program Blender. Unity3D byl převážně vybrán pro svoji oblíbenost u vývojářů a kvalitní dokumentaci. V neposlední řadě i pro skutečnost, že je nabízen zdarma.

2 Počítačové hry

Počítačové hry patří do skupiny softwaru určeného pro zábavu. Tento pojem konkrétně zahrnuje hry pro PC platformu, konzole (specializované počítače), tablety, mobilní telefony a podobná zařízení. Obecně se tyto hry označují jako videohry.

Existuje mnoho různých reprezentací, ale všechny mají podobný základ v kombinaci zobrazovacího a ovládacího zařízení. Hráč hraje hru, jedná se tedy o druh interakce mezi člověkem a strojem, kde člověk komunikuje pomocí ovládacího zařízení a stroj pomocí zobrazení textu, grafiky, nebo zvuky.

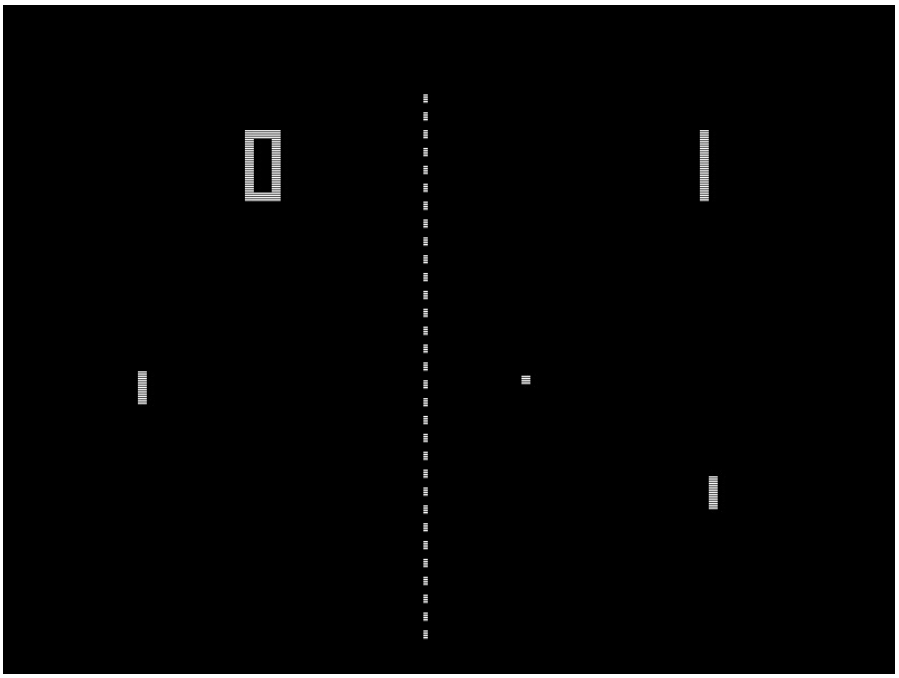
Existuje mnoho typů her. Nejčastěji se dělí podle žánru:

- **Adventury –** používá se styl zaměřit a kliknout (většinou myší), velice jednoduché ovládání, důraz kladen na obsáhlý příběh a řešení rébusů.
- **Akční hry –** zde jde hlavně o akci a dynamiku. Hráč zpravidla prochází úrovně, které dohromady tvoří příběh, ale hlavním bodem jsou akční prvky, jako například střílení po nepřátelích. Rozdělují se na FPS (First person shooter) a TPS (Third person shooter), podle pozice kamery. U FPS kamera zobrazuje pohled z první osoby, jako by člověk viděl vlastníma očima. Naopak u TPS je kamera za hráčem, který vidí svojí postavu.
- **RPG –** jsou hry, kde hráč má svého hrdinu (hrdiny), který se nám postupem hry vyvíjí a zlepšuje. Typickými znaky jsou rozlehlé herní prostory a spousta možností, jak hru hrát a kam směřovat.
- **Strategie –** základem strategií je plánování postupu a budování. Nejčastěji se buduje armáda k boji, města, infrastruktura a hráč rozhoduje, jakým způsobem bude budovat.
- **Online hry –** tyto hry nejsou samotným žánrem, ale spíše úpravou jiného žánru pro společné hraní po síti.
- **Sportovní hry –** využívají sportovního motivu, například automobilové závody, fotbal, hokej, atletika, střelba z luku apod.
- **Arkády –** u arkád jde převážně o jednoduchost a nápaditost. Patří sem klasické bojové hry, plošinové nebo logické hry atd.

2.1 Historie [2] [3]

Za počátek videoher je považován rok 1952 [1], kdy britský univerzitní profesor Alexander S. Douglas vytvořil první hru, která používala digitální display. Program byl simulací známé hry „Piškvorky“ zobrazené na osciloskopu, kterou Douglas vytvořil jako součást své doktorské práce na téma interakce člověk-počítač.

První videohra, která zaznamenala větší komerční úspěch, byl Pong od firmy Atari. Vytvořil ji Allan Alcorn, který jako nově příchozí zaměstnanec firmy neměl s vytvářením videoher žádné zkušenosti, a tak mu spoluzakladatel Atari Nolan Bushnell zadal úkol vytvořit Pong, jako trénovací cvičení, z důvodu aklimatizace na nový druh vývoje. Majitelé Atari však byli velmi překvapeni kvalitou, že se rozhodli hru vyrobit. V oblasti automatových her na mince zažíval Pong veliký úspěch, kdy konstantně vydělával 4x více než ostatní automaty, a to vedlo k vytvoření množství napodobenin. Atari se nemohlo bránit, jelikož neměli patenty na technologie použité ve hře, a tak se snažili porazit konkurenci vydáváním inovativních verzí a nových produktů.



Obr.1 Videohra Pong

U dalšího historicky významného milníku ve vývoji videoher byla opět firma Atari s produktem Atari 2600. Jednalo se o herní konzoli pro domácí použití, která se připojovala k televiznímu zařízení a velkou měrou přispěla k popularizaci mikroprocesorového hardwaru. Hlavní inovací nového herního systému byla možnost hrát hry uložené na ROM cartridges, neboli výměnných modulech. Díky těmto modulům se majitel herní konzole nemusel spokojit s jednou nebo pár hrami, které byly v konzoli napevno, ale mohl si dokupovat nové. Atari konzoli s původním názvem Atari Video Console System (později přejmenováno na Atari 2600) vydalo v září roku 1977, ale to již byl na trhu několik měsíců podobný produkt s názvem Channel F.

Následovalo zlevnění zastaralých herních systémů a hlavně hry Pong a jejích klonů, jejíž výrobou se zabývalo množství menších firem. Tyto společnosti povětšinou ukončily své působení na trhu, a tak se oběma konzolím naskytla příležitost ukázat veřejnosti, že lze hrát i jiné hry než Pong. Atari 2600 bylo úspěšnější. Roku 1979 prodali 1 milion konzolí a finanční zisky firmy stále rostly až k roku 1982, kdy se jich prodalo na 10 miliónů.

Celkově bylo pro konzoli vytvořeno od Atari a dalších firem okolo 1000 různých titulů.

Obr.2 Videohra Packman pro konzoli Atari 2600

Tento velký rozmach domácích konzolí opět přivedl do průmyslu množství nových firem. Jejich produkty brzo zahltily celý trh. Produkovalo se takové množství her a konzolí, že ani nebyl čas je hrát, a to byl jeden z hlavních důvodů, který v Severní Americe v letech 1983 – 1984 vedl ke krachu videoherního průmyslu. Během této doby natolik poklesla poptávka po herních konzolích, že se prakticky přestaly vyrábět, a to vedlo k úpadku velkého množství malých firem.

Pokles poptávky byl způsoben více faktory. Jedním z nich byl i zvyšující se prodej osobních počítačů, které se postupně stávaly finančně dostupnější a dávaly lidem možnost nejen hrát hry, ale i zjednodušovaly práci.

Samotná firma Atari přispěla ke krizi velkou měrou, a to především uspěchaným publikováním nekvalitních her. Tím se sama dostala do velkých problémů. Například u hry s názvem E.T (motiv známého filmu o mimozemšťanovi E.T) zaplatila přes 20 milionů dolarů za autorská práva a vydala ji už po šesti týdnech vývoje, aby stihla předvánoční prodeje. Ze 4 milionů vydaných kopií bylo 3,5 vráceno zpátky. Další chybou bylo opět unáhlené vydání nové verze velmi očekávané hry Pac-Man, která se také setkala s kritikou kvality. Atari sebevědomě objednalo 12 milionů kopií, ale prodáno bylo pouze 7 milionů.

Roku 1985 skončila sestupná tendence herního průmyslu, a to především díky vydání herního zařízení od japonské firmy Nintendo. Kvůli atmosféře, která panovala po krizi, se firma rozhodla raději nepoužít označení „konzole“, a nazvala svůj produkt Nintendo Entertainment Systém. Zavedla také nový model licencování pro vývojáře ze třetích stran, který účinně omezoval množství produkovaných her pro konzoli.

Obr.3 Nintendo Entertainment System

Na začátku devadesátých let začíná nová éra videoherní zábavy, s využitím osobních počítačů. Osobní počítače se staly finančně dostupnými veřejnosti a na rozdíl od herních konzolí nabízely uživateli širší využití, než jenom pro hraní her. V krátké době se tak stali velice populárními. Nejúspěšnějším osobním počítačem se stal Commodore 64, který byl na trh uveden roku 1982 a stal se později nejprodávanějším osobním počítačem všech dob s prodeji odhadovanými na 17 až 25 milionů prodaných kusů. Mezi dalšími výrobci byli Apple, IBM a jiné firmy, které mnohdy existují dodnes.

2.2 Typy videoher dle platformy

Videohry lze rozdělit podle platformy, pro kterou jsou vytvořeny. Platformou se rozumí kombinace hardwarových a softwarových komponent, které společně vytváří prostředí pro fungování aplikace. Je to důležité hledisko, které zásadně ovlivní vývoj od financování až po závěrečný marketing. Základem trhu jsou hry pro PC a konzoli. Jejich vývoj může trvat roky, stát miliony dolarů a můžou na nich pracovat stovky vývojářů. Spousta titulů se vytváří jako multiplatformní, tedy pro PC i konzole zároveň, nebo v rámci platformy PC pro různé operační systémy jako Windows, Linux a další. Rychleji a levněji se vyvíjí hry pro telefony, tablety a podobná přenosná zařízení. Tento trh zažívá veliký růst v posledních letech, a to především díky stále se zvyšující výkonnosti mobilních zařízení, kvalitní grafice, připojení k internetu a dalším možnostem. Další oblastí jsou hry pro webové prohlížeče, kde již dnes pomocí HTML5 jazyka, JavaScriptu a dalších nástrojů lze také vytvářet plnohodnotné a náročné hry.

2.2.1 Platforma PC a konzole

Vyvinout klasickou hru pro PC nebo konzoli je ze všech platforem finančně i časově nejnáročnější. Nejedná se jen o samotný vývojový proces, ale mnohdy důležitější částí je marketing. I v herním průmyslu, stejně jako v jiných oblastech, platí, že produkty se bez promyšlené a rozsáhlé reklamy a distribuce neprodávají snadno.

Srovnávání herních konzolí s PC platformou je velice diskutovaným tématem od dob prvních konzolí využívající 3D technologie. Hráči se stále nemohou shodnout, ze které platformy lze pocítit lepší herní zážitek.

Srovnání má více aspektů, které se v průběhu vývoje mění. Například porovnání výkonu platforem. Základním rozdílem je, že u PC lze dokoupit výkonnější hardwarovou součástku a vyměnit starší model, ale u konzole toto provést nelze (až na výjimky). Dříve konzole výkonově odolávaly počítačům díky odlišnému hardwaru a především optimalizací výkonu a operačního systému přímo na multimediální zábavu, ale s příchodem 8. generace herních konzolí se dva největší hráči na trhu s konzolemi (PlayStation od firmy Sony a Xbox od firmy Microsoft) rozhodli použít hardware, jaký se používá v osobních počítačích, tedy PC platformy [4]. Při současné rychlosti vývoje hardwaru pro počítače netrvalo dlouho a na trhu se objevily komponenty s vyšším výkonem, než měly konzole zabudované v sobě a platforma PC tak překonala výkon konzolí. V současnosti tedy, po grafické stránce, by měly být hry lepší na počítači, ale u konzole zase bývá hardware lépe optimalizovaný. Mnohdy také záleží na hře samotné.

Dalším aspektem je ovladatelnost, která je mnohdy spíše subjektivní a záleží, na co je daný hráč zvyklý. Záleží také na žánru hry, kdy například strategie a first-person-shooter hry bývají snadněji ovladatelné na počítačích, a naopak závodní a bojové hry na konzolích.

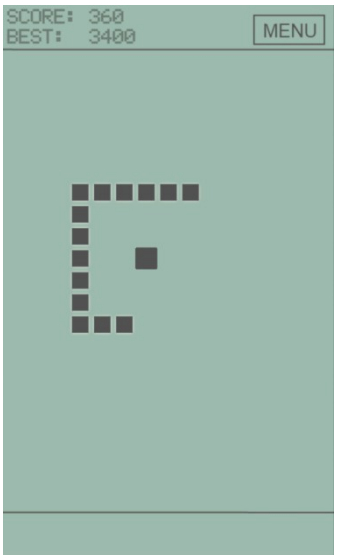
Jak PC tak konzole mají své přednosti i nedostatky. Mezi výhody PC patří hlavně všestrannost, má mnohem více způsobů využití než konzole a existuje obrovské množství softwaru, který přináší spoustu různých funkcí a možností využití. Další výhodou je modularita, tedy schopnost vyměnit komponentu v počítači a zvýšit výkon. Ta ovšem přináší i jistou nevýhodu plynoucí z velkého množství různých kombinací hardwaru, která především vývojářům značně komplikuje optimalizaci. Různé druhy použitých ovladačů, nebo často nestabilní operační systém Windows (nejrozšířenější), který se snadno destabilizuje působením virů, spyware a dalších škodlivých softwarů, mohou způsobovat nepříjemnosti.

Velkou výhodou konzolové platformy je především jednoduchost. Přestože nové konzole jsou podstatně složitější než ty dřívější, stále u většiny her stačí vložit médium do přístroje a spustit. Hráč nemusí řešit, jestli mu hra bude fungovat, nebo jakkoliv nastavovat či optimalizovat grafiku pro plynulý běh, jako je to obvyklé u PC. Z hlediska vývoje je programátor ovšem limitován výkonem konzole a jeho hra nemůže překročit určitou úroveň náročnosti. Toto omezení pro PC teoreticky neplatí. Mezi další výhodu konzolí patří pořizovací cena. V současnosti lze nejvýkonnější konzole zakoupit za cenu do 10 000 Kč, naproti tomu výkonné PC sestavy bývají minimálně dvakrát dražší. To ovšem do jisté míry kompenzuje cena her, která je vyšší u konzolí.

2.2.2 Mobilní platforma

Mobilní platforma zahrnuje široké množství zařízení, mezi které paří především chytré telefony, tablety, PDA a další přenosná zařízení.

První hra pro mobilní telefon vyšla roku 1994. Byla to variace známé hry Tetris. Ovšem popularitu si hry začaly získávat až s příchodem hry Snake, kterou firma Nokia předinstalovala na svá zařízení, a tak se Snake vyskytl na více jak 350 miliónech zařízeních.



Obr.4 Hra pro mobilní zařízení Snake

Od dob hry Snake udělala tato platforma obrovský pokrok a dnes si můžeme na našich přenosných zařízeních zahrát hry s nádhernou a propracovanou grafikou (obr. 5), přes internet hrát s hráči z celého světa, nebo případně si sami vytvořit vlastní mobilní aplikaci.

Obr.5 MODERN COMBAT 5: BLACKOUT

Vývoj her pro tuto platformu můžeme rozdělit podle OS (operační systém), pro který je hra určena. V současné době máme na trhu 3 hlavní systémy: Android, iOS a Windows Phone.

Nejrozšířenějším operačním systémem je Android, který zabírá 59% trhu (graf 1).

Graf. 1 Podíl operačních systémů mobilních zařízení (1.leden, 2016)[6]

Tento OS je vyvíjen společností Google a jedná se o otevřený operační systém. Je postaven na Linuxovém jádře a aplikace se pro něj převážně vyvíjí v programovacím jazyku Java. Otevřenost systému má spoustu výhod i pro vývojáře, kteří mohou využít široké množství vývojových nástrojů a nejsou tolik omezováni a kontrolováni firmou Google, jako je to například u iOS. Nejvíce používaným vývojovým prostředím byl dlouhou dobu Eclipse IDE s pluginem Android Developer Tools, ale možností je víc. Google vydal své vlastní IDE nazvané Android Studio a podporu pro Eclipse ukončil na konci roku 2015 [7]. Otevřenost systému má i svá negativa. Pro vývojáře aplikací je hlavním problémem roztříštěnost systému. Ta je způsobena obrovským množstvím různých telefonů s různým hardwarem (rozdíly ve výkonech, velikosti displaye, atd.). Především také několik různých verzí systému, které jsou stále aktivní, a pro které mnoho výrobců telefonů vytváří své vlastní grafické, či jiné nástavby. Tyto nástavby mohou systém zpomalovat nebo jinak ovlivnit a tím se stává velmi složité aplikaci optimalizovat [8].

Druhým nejrozšířenějším systémem je iOS (33%), tento software distribuje firma Apple striktně pouze na svých mobilních zařízeních. Vývoj aplikací probíhá v jazyku Objective-C. Jedná se o nástavbu jazyka C. Vývojové prostředí XCode je program přímo od společnosti Apple, který je oficiálně možný stáhnout pouze přes AppStore a tedy pro vývoj je potřeba počítač od Apple. To je typické pro uzavřenou politiku firmy a úplný opak konkurenčního Androidu. Existuje i možnost vyvíjet pro iOS v ostatních počítačových OS, a to například s použitím vývojových nástrojů Unity, WinChain, nebo Tersus.

Třetím operačním systémem s podílem 3% na trhu je Windows Phone, v poslední verzi Windows 10, u které firma Microsoft vynechala označení Phone v názvu. I Microsoft poskytuje vývojové nástroje pro svůj OS. Jedná se o Windows Phone SDK, který lze nainstalovat do Visual Studia a pomocí jazyka C#, Visual Basic, C++ a dalších vytvářet aplikace.

2.2.3 Web hry

Webové prohlížeče nejsou platformou v pravém slova smyslu jako počítače či mobilní zařízení, ale díky velkým inovacím posledních let a především příchodem nového standardu HTML5, získala tato platforma nový rozměr. S použitím kombinace JavaScriptu, HTML a CSS lze vytvářet i náročné a obsáhlé hry.

Mezi nejnáročnější hry zpravidla patří ty s 3D grafikou. I tyto hry lze vytvořit pro webové prohlížeče a to díky WebGL. Je to JavaScriptové API pro nativní zobrazování interaktivní 3D grafiky, které používá grafickou kartu počítače. Lze vytvářet i online hry, vytvořit server a ukládat data o hráčích (poslední pozice, nasbírané předměty, atd.). Možnosti jsou velké a tím se hry pro webové prohlížeče již dají srovnávat s hrami na ostatní platformy [9].

Obr. 6 Webová hra MAGEREALM: RISE OF CHAOS

Jednou z velkých výhod jsou možnosti distribuce her. Webové hry stačí jednoduše dát na internet. Lze i vytvořit hru cílenou pro jinou platformu, poté vložit do HTML kódu a hra se chová jako webová. Toto umožní vývojářům distribuci her pomocí webového prohlížeče, a zároveň mohou prostřednictvím aplikace využívat funkce platformy, například u mobilních zařízení přístup ke kontaktům a podobně [10].

2.3 Vývoj videoher

Hry jsou distribuovány vydavatelem, podobně jako kniha nebo film. Ten financuje vývoj hry, se stará o marketing, výrobu, distribuci, reklamu a vše s tím spojené. Větší vydavatelské firmy, jako Nintendo, nebo ElectronicArts, mají svá vlastní vývojová studia a mohou tedy samy celou hru vytvořit a následně i vydat. Menší vydavatelské firmy si na vývoj najímají externí studia, financují je po dobu vývoje a obvykle pak vlastní veškerá autorská a distribuční práva.

Vydání hry pro PC nebo konzoli je nesmírně nákladné. V roce 2000 se průměrné náklady spojené s vydáním hry pohybovaly od 1 do 4 milionů dolarů, v roce 2006 byl průměr přes 5 milionů a kolem roku 2010 to bylo již víc jak 20 milionů dolarů. Vysoké náklady s sebou nesou velké riziko při neúspěchu, které může vydavatelskou firmu dostat do potíží [11] [12].

Jako nejdražší hra je v současnosti označována online hra Destiny z roku 2014, která stála firmu Activision 500 milionů dolarů (pro srovnání, nejdražší film, Piráti z Karibiku: Na konci světa stál okolo 300 milionů dolarů). Z této částky šlo ovšem na vývoj hry jen 140 milionů a zbytek byl použit na marketing [13].

Obr. 7 Nejdražší hra Destiny

Stává se ovšem, že hra nenabude očekávané popularity, jako například online hra Star Wars: The Old Republic. Hra byla postavena na známém motivu z filmů Hvězdných válek, na vývoj a marketing použito minimálně 200 milionů dolarů, ale po počátečních slibných prodejích zájem hráčů upadl [14].

Délka vývoje hry se odvíjí především od rozsahu projektu a množství lidí, které studio na vývoj nasadí. U největších projektů to bývají i stovky, například na jedné z nejrozsáhlejších her současnosti, GTA 5 od firmy RockstarGames, se podílelo přes 1000 lidí [14].

Strukturně se vývojový tým dělí na několik základních pozic [15] [5]:

- **Producent –** řídí celý vývojový team, dohlíží na dodržování smluv a licenčních dohod, stará se o financování, podává zprávy o průběhu vývoje, atd. Producenti mohou být interní, neboli zaměstnaní vývojovým studiem, nebo externí, kteří pracují pro vydavatele.

- **Designer –** herní designer je člověk nebo skupina lidí, kteří vytváří celkový obraz hry, navrhují strukturu a vymýšlí obsah. Mezi jejich práci patří sepsání příběhu, dialogů, nápověd, návrh videí, uživatelského rozhraní a další podobné úkony.

- **Artist –** neboli také grafik má zodpovědnost za vytváření textur, obrázků, 3D modelů a prakticky všech částí, které ovlivní, jak hra vypadá. Grafiků bývají celé skupiny, které se mohou dělit podle zaměření na 2D a 3D objekty.

- **Programátor** – skupiny programátorů vytvářejí software a potřebné množství skriptů, které umožní ve hře vše spojit a dovést do funkční podoby. V závislosti na použitých nástrojích při vývoji patří mezi úlohy programátorů:

    - *Fyzika* – simulace fyzikálních zákonů (reálných, nebo smyšlených, podle žánru hry), pohyb a kolize objektů a další. Tyto komponenty bývají předprogramovány v herních enginech, a tedy stačí je upravit podle potřeby vytvářené hry.
    - *AI* – umělá inteligence je vytvářena za pomocí speciálních technik, například na základě plánování, nebo rozhodování podle daných pravidel.
    - *Grafika* – vytvoření, nebo přizpůsobení grafického enginu, aby modely a textury pracovaly efektivně s pamětí a fyzickým enginem.
    - *Zvuky* – integrace zvuků a hudby.
    - *Hratelnost* – implementace herních rysů a pravidel.
    - *Skriptování* – vytváření příkazů pro úkony přímo ve hře.
    - *UI* – neboli elementy uživatelského rozhraní jako menu, help systém a další.
    - *Input* – zajištění kompatibility možných vstupních zařízení, jako jsou klávesnice, myš a gamepady.
    - *Síťové propojení* – nastavení lokálního a internetového hraní.
    - *Vývojové nástroje* – vytvoření nástrojů pro vývoj komponent do hry, například pro skriptování, nebo level designéry.

- **Level designer** – vytváří nové stupně hry, jako levely, nebo mise. Obvykle pracuje s level editorem, který mu poskytnou programátoři, aby nemusel pracovat se základním kódem, ale mohl používat skriptovací jazyky.

- **Zvukař** – osoba zodpovědná za hudbu a zvukové efekty.

- **Tester** – provádí analýzy funkčních částí hry a následně i výsledné aplikace. Je součástí kontroly kvality a podává informace o tom, zda je hra v souladu s původním návrhem, důkladně testuje funkčnost a hratelnost a hledá možné chyby.

Podobně jako vývojový tým má svoji strukturu a hierarchii, tak i vývojový proces je rozčleněn. Rozděluje se na několik fází [16]:

- **Koncept** – základem všeho je koncept, neboli myšlenka, nápad. Zpočátku musí být vymyšleno, o čem hra bude, a jaký bude mít motiv. Jestli to bude logická hra, nebo akční first-person střílecí hra, případně bývají hry inspirovány filmy, komiksy, historií, sportem a spoustou dalších možností, které jsou limitovány pouze fantazií.

- **Předprodukční fáze** – v této fázi se předprodukční tým, který bývá složený z producentů, jejich asistentů, designérů, programátorů a umělců, zabývají vytvořením příběhové linie hry. Sepisují design dokument, ve kterém nastaví cíle hry, její průběh a celkovou mechaniku hry. Vývojáři musí dbát na limity dané motivem hry, technologií a hardwarem, pro který bude hra určena. Následuje vizualizace příběhové linie pomocí nákresů a popisků vysvětlujících, co se stane v různých fázích hry. Výstupem této fáze je Game design document, který by měl detailně popisovat hru, a obsahovat zmíněné mapy a nákresy. Měl by vysvětlit, jak se hra bude hrát a ovládat, co bude které menu obsahovat, a jak budou vypadat jednotlivé stupně hry, modely, celková grafika, fyzika a zvuky.

- **Produkce** – po vytvoření Game design document přichází samotný vývoj hry. Je důležité, aby každý člen týmu vyvíjel svoji část v souladu s cíly projektu. O to se stará game designer. Začíná se vytvořením prototypu, neboli zjednodušené verze hry se základními modely (např. krychle) a funkčností, která se postupně vylepšuje. Velice důležitou součástí je testování. Nejefektivnější je začít testovat co nejdříve, jakmile je projekt v hratelné fázi, třeba první level, nebo první trať závodu, a pokusit se o odstranění veškerých chyb, protože jakákoliv změna v programu může přinést nečekané následky. Projekt pak prochází několika fázemi:

    - **alpha verze** – neboli první hratelná verze obsahující všechny důležité funkce. V této verzi bývá stále spousta chyb a slouží především k testování a posledním úpravám designu a hratelnosti.

- **beta verze** – po odstranění většiny chyb přechází hra do beta verze. Ta by měla být již dostatečně vyladěná pro testování nejen najatými testery v rámci týmu, ale vývojáři někdy nechávají tuto verzi testovat veřejností.

- **finální verze** – dokončená a vyladěná verze hry určená k distribuci a prodeji. Měla by být odzkoušená na všech možných platformách, pro které je cílena, a testovaná ve všech možných situacích, které lze předvídat. Následné ladění a odstranění dalších chyb probíhá pomocí aktualizací.

2.4 Výběr nástrojů pro vývoj

Vyvíjet počítačové hry lze mnohými způsoby. Je důležité si daný způsob vybrat a dopodrobna promyslet před začátkem vývoje, a na základě toho zvolit vývojové nástroje.

Jednou z možností je vyvinout vše potřebné pro hru vlastní silou. To znamená navrhnout a vytvořit celé jádro hry, tedy herní engine. Tahle varianta je ze všech nejnáročnější, nejnákladnější a mohou si ji dovolit velké firmy vydávající nejznámější herní tituly, pro které vytvoření vlastního enginu bývá investice do budoucích projektů, nebo ho využijí jako samostatný produkt. Také se používá u malých projektů a jednodušších aplikací s malou časovou náročností. Velkou výhodou je v tomhle případě, že vývojáři si mohou všechny části nastavit přesně pro potřeby svého projektu, a nic neomezuje jejich flexibilitu. Na druhou stranu v dnešní době existuje velké množství velice kvalitních enginů a vývojových nástrojů, a vytváření všeho pro vývoj hry jako grafika, fyzika, zvuky, propojení, umělá inteligence, je velice neefektivní a zdlouhavé.

Běžně tedy vývojářské týmy používají pro tvorbu vývojové nástroje. Podle typu a množství je můžeme rozdělit na 3 skupiny [17]:

- **Lowest level**: Do této skupiny patří vývojové týmy, které vytváří herní engine za pomocí veřejně dostupných API (Application Programming Interface).
  V podstatě použijí volně stažitelné, nebo komerční knihovny jako například XNA, DirectX, OpenGL, SDL a další, a implementují je do svého projektu. Jedná se o časově nejnáročnější způsob vývoje, jelikož vyžaduje velké množství programování a optimalizování použitých knihoven, které musí společně fungovat. Výhodou je velká flexibilita. Programátoři si mohou vybrat, které knihovny a funkce jim vyhovují, a použít je způsobem, který potřebují.

- **Mid level**: Střední cestou je použití enginů typu Genesis3D nebo ORGE, které poskytují důležité funkce pro grafiku, rendering, vstup/výstup, fyziku a další, ale pro vytvoření fungující hry a využití funkcí je stále potřeba programovat. V porovnání s první skupinou ztrácí lehce na flexibilitě, ale na druhou stranu je mnohem jednodušší funkce navzájem optimalizovat.

- **Highest level**: Nejjednodušší cestou je použití programů, které v sobě již obsahují všechny důležité nástroje pro vývoj a pro jejich použití většinou stačí kliknout a nastavit. Jedná se přímo o aplikace, které mají uživatelské rozhraní, snaží se poskytnout přehledný přístup ke všem funkcím, a co nejvíce zjednodušit vývoj, bez potřeby vše programovat. I zde se využívá programování a především skriptování. Programátor ušetří mnoho času, protože funkce poskytované enginem jsou většinou kompatibilní a optimalizované. Vývoj v těchto aplikacích je nejméně flexibilní a má nejvíce omezení. Často jsou přímo zaměřené na určitý typ her, ale v dnešní době najdeme velice výkonné programy, které používají i profesionálové, a díky nim si může vytvořit počítačovou hru téměř každý.

2.4.1 Herní engine

Označení herní engine je nejčastěji používáno pro interaktivní vývojové prostředí, které slouží převážně k vytváření videoher na počítače, konzole nebo mobilní zařízení.
U dnešních enginů bývá obvyklá podpora multiplatformního vývoje, tedy vytvořenou hru lze většinou s malými úpravami vydat pro více platforem. Hlavní myšlenkou je poskytnout uživateli opakovaně použitelné a nastavitelné nástroje potřebné pro vytvoření funkční aplikace a tím ušetřit co nejvíce času, který může vývojář soustředit na hru samotnou.

Většinou herní enginy obsahují uživatelské rozhraní a funkce pro práci se vstupy. Dále obsahují fyziku, knihovny pro vykreslování 2D a 3D grafiky, detekce kolize, zvuky, nástroje pro skriptování, animace, sítové funkce, funkce umělé inteligence a mnoho dalších. Mezi základní funkce tedy patří [18]:

- **Input** (vstup) – jednou z nejdůležitějších částí pro vývoj hry je možnost jí ovládat. K tomu je potřebná podpora základních vstupních zařízení, jako myš, klávesnice, gamepad, a případně dalších, méně tradičních, jako joystick, volant, atd. Vstupní signály se dělí na dvě skupiny, na eventy a dotazování. Eventy se spustí na základě určité formy vstupu, jako je zmáčknutí klávesy nebo tlačítka na myši, a tím se spustí funkce v kódu. Jednoduše se tedy nastaví, co se má stát při zmáčknutí dané klávesy (například vystřelit, skočit atd.). Dotazování se používá, pokud je potřeba sledovat neustále se měnící hodnoty. Nejčastěji to bývají x/y hodnoty pozice kurzoru myši. Pro tento vstup nastavujeme, co se stane při změnách těchto hodnot.

- **Grafika** – další velmi důležitou součástí videoher je grafika. Enginy často poskytují možnost vytvářet modely přímo v programu. To se ale většinou používá pouze pro méně důležité a jednoduché modely. Pro složitější modelování využívají vývojáři možnosti importovat z grafických programů typu Maya, 3ds Max, Blender. V enginu se potom nastavují vlastnosti materiálů, osvětlení, stíny a další grafické efekty.

- **Zvuk** – ozvučení a hudba je další nezbytnou částí pro vytvoření dobrého herního zážitku. Různé zvukové formáty lze importovat a zpracovat, a pomocí množství efektů vytvořit reálný (případně nereálný) dojem. Například ve 3D hrách se zvuk nastavuje tak, aby bylo poznat, odkud z prostoru vychází, například že za dveřmi vpravo je slyšet nepřítel, nebo že ve vedlejší místnosti něco spadlo.

- **Networking** (síťové propojení) – většina dnešních videoher podporuje online propojení s dalšími hráči, ať už za účelem porovnávání výsledků nebo společného hraní. To vyžaduje vytvoření komunikace s různými servery nebo dalšími počítači, a to bývá velice náročná a zdlouhavá práce. K tomuto účelu však moderní enginy poskytují speciální komponenty, a nápomocné skripty, které velice ulehčí práci.

- **Fyzika** – fyzické enginy jsou navrhovány, aby interpretovaly fyzické zákony, a chování reálného světa. Jsou implementovány v herních enginech. Mezi nejznámější patří například Havok, nebo PhysX. Hlavní funkcí je provádění různě komplikovaných výpočtů, které zatěžují výkon procesoru. Někdy je tedy nutné použít metody, které sníží nároky hry na výkon (např. zamezení přístupu do určitých částí mapy, aby se nemuselo vypočítávat kolize se stromy a podobně).

- **Skriptování** – skripty jsou komponenty, které určují chování objektů. Enginy většinou obsahují několik předvytvořených skriptů. Uživatel si může naprogramovat svoje vlastní skripty použitím programovacích jazyků typu JavaScript, C++, C#, nebo například Unreal Engine 3 má svůj vlastní jazyk UnrealScript.

2.4.2 Srovnání herních enginů

Herních enginů existuje celá řada a v mnohých věcech se liší. Jako každý produkt mají své výhody a nevýhody a obvykle bývají zaměřené na určitý typ her. Před začátkem vývoje je vhodné si zmapovat, které aspekty jsou důležité pro náš projekt a podle toho vybrat vhodný engine. Jedním ze zmíněných aspektů je určitě i cena enginu. Dříve stály tyto produkty obrovské částky peněz a menší vývojová studia, případně jednotlivci, je tedy nemohli využívat. Dnes je situace jiná. Firmy vyvíjející herní enginy změnily svoji finanční politiku, a poskytují své produkty zdarma (většinou pro nekomerční či akademické účely), nebo za přijatelné ceny.

2.4.2.1 CryEngine

Jedním z nejpropracovanějších a z funkčního hlediska nejvyspělejších je CryEngine od německé firmy Crytek, která ho používá pro všechny své hry. Většina z nich spadá pod žánr FPS (first person shooter) a tedy i samotný CryEngine je nejvhodnější právě pro vývoj podobných žánrů, ve kterých je důraz na ztvárnění realistického okolí (zvláště přírody) [19]. Vhodný je především pro obsáhlé projekty s důrazem na grafiku. Pokud plánujeme vytvářet jednodušší hru (hlavně graficky), CryEngine je pro ně patrně zbytečně složitý. To ovšem není pravidlem a lze v něm vytvořit prakticky cokoliv.

Při vytváření skriptů lze použít programovacího jazyka C++, nebo speciálního skriptovacího jazyka Lua [19].

Engine podporuje multiplatfomní vývoj [20]. Lze v něm vytvářet hry pro PC, pro operační systém Windows i Linux, dále podporuje nejnovější konzole Playstation 4 a Xbox One, a nově lze vyvíjet hry i pro Oculus Rift (herní systém virtuální reality).

Crytek poskytuje několik licencí. Pro školy zdarma na výukové účely. Za cenu 9.90 dolarů měsíčně verzi, ve které lze vyvíjet pouze na platformu PC a Oculus Rift. Cena plné licence není oficiálně známá.

Příklady her vytvořených v tomto enginu jsou například: serie Far Cry a Crysis přímo od Cryteku a Homefront: The Revolution vydaný firmou Deep Silver.

Obr. 8 Ukázka ze hry Homefront: The Revolution

2.4.2.2 Unreal Engine 4

Dalším mocným nástrojem na poli vývoje her je Unreal Engine od americké firmy Epic Games. Podobně jako u CryEnginu, většina her od Epic Games, postavených na jádře Unreal Enginu, byly žánru FPS. Tyto hry jsou ale převážně zasazené do moderní doby a sci-fi prostředí, a tedy i samotný engine je vhodný spíše pro tento druh her [20]. Engine se již také osvědčil při vytváření her žánru Stelth, RPG, MMORPG a dalších. Stejně jako CryEngine má technologie pro vytvoření jakékoli hry.

Skripty se v dřívějších verzích enginu vytvářeli převážně ve speciálním jazyku UnrealScript. Od verze 4 byl tento jazyk vypuštěn, a přešlo se ke klasickému C++ [21].

Podporované jsou téměř všechny moderní platformy jako PC (Windows i Linux), nejnovější konzole, mobilní zařízení (Android, iOS), i platformy virtuální reality.

Licence na Unreal Engine je pro všechny zdarma. Epic Games poskytuje volně všechny nástroje a funkce a účtuje si 5% z výdělku, pokud produkt vytvořený v Unreal Enginu vydělá přes 3000 dolarů za kvartál. Firma dále uvolnila 5 miliónů dolarů, ze kterých dotuje nadějné projekty. Vývojáři mohou požádat o příspěvek od 5 do 50 tisíc dolarů [22].

Mezi známé hry používající Unreal Engine patří: série Unreal a Gears of War, nebo tituly jako Tekken 7, Dead Island 2 a Crackdown 3.

Obr. 9 Ukázka ze hry Gears of War

2.4.2.3 Unity 3D

Unity je v současnosti nejrozšířenější herní engine, který je poskytován pro nekomerční účely zcela zdarma. Jedná se o plnohodnotný, multiplatformní nástroj pro vývoj her, a s podporou C# jazyka. Pro tyto vlastnosti byl námi vybrán pro vývoj aplikace Virtual World v praktické části této práce. Enginu je věnována následující kapitola.

3 Unity 3D

Unity je multiplatformní herní engine vyvíjený společností Unity Technologies. Jde o vývojovou platformu především pro vytváření her, ale i jiných interaktivních 2D a 3D zážitků. Využívá se i pro tréninkové simulace a lékařské nebo architektonické vizualizace. V Unity lze vyvíjet pro mobilní, desktopové, konzolové, webové a různé další platformy. Mezi firmy používající tento software patří například Cartoon Network, Coca-Cola, Disney, Electronic Arts, LEGO, Microsoft, NASA, Nexon, Nickelodeon, Square, Ubisoft a Warner Bros, tedy i spousta společností a organizací mimo herní průmysl [23].

První verze 1.0 vyšla 8.června 2005. Byla zaměřená pouze na vývoj pro operační systém OS X, tedy pro zařízení od firmy Apple. Poslední verze z letošního roku je označena 5.3.4. Během deseti let vývoje a rozšiřování enginu byly přidány další desítky podporovaných platforem (tab.1).

Desktop	Windows	Mac	Linux/Steam			
Mobilní zařízení	iOS	Android	Windows Phone	Tizen		
Konzole	PS4	PS Vita	Xbox One	Xbox 360	Wii U	Nintendo 3DS
TV	Android TV	Samsung SMART TV	TV OS			
Web	WebGL					
Virtuální realita	Oculus Rift	Google Cardboard	Steam VR	Gear VR	Microsoft HoloLens	PlayStation VR

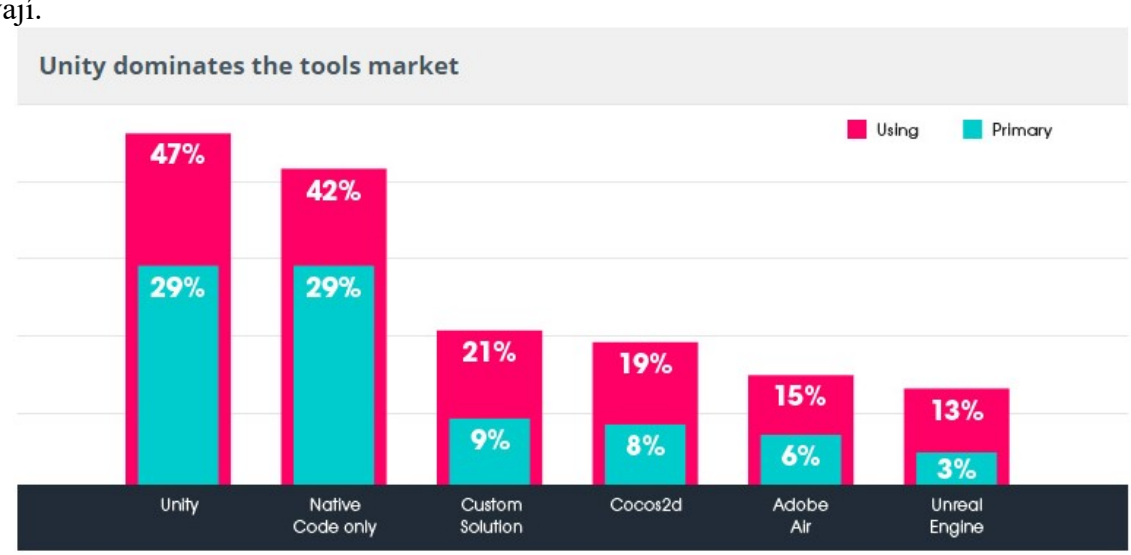
Tab.1 Přehled podporovaných platforem [24]

Roku 2010 byl spuštěn Unity Asset Store, ve kterém lze koupit, nebo i zdarma stáhnout velké množství modelů, skriptů, audio souborů a dalších komponent k použití v Unity.

Engin je poskytován ve dvou základních verzích Personal a Professional. Verze Personal je zcela zdarma a poskytuje téměř všechny vlastnosti a funkce jako Professional verze. Ta cenově začíná na 75 dolarech za měsíc, nebo jednorázově 1500 amerických dolarů. Mezi rozdíly těchto verzí patří například spouštěcí obrazovka s logem Unity, která se objeví při spuštění aplikace vytvořené v Personal verzi. Dále v této verzi chybí některé funkce pro analyzování dat, pro týmovou spolupráci, a také další (speciální) funkce pro vytváření mobilních aplikací pro Andoid a iOS. Personal verze je ale dostatečná pro vývoj, a i následný prodej aplikace. Zisky ovšem nemohou překročit částku 100 000 dolarů za fiskální rok. Pokud překročí, každý vývojář v týmu si musí zakoupit Professional licenci [25].

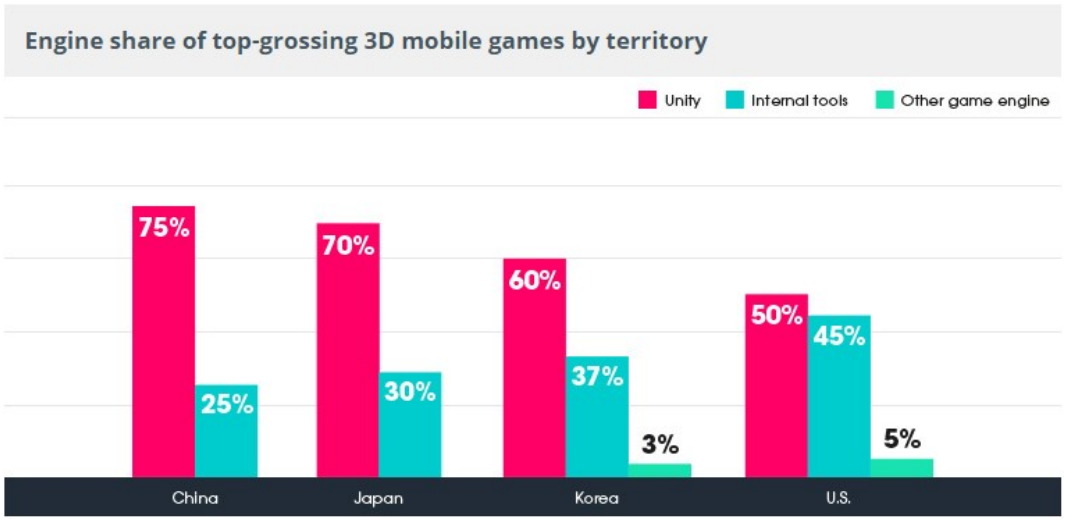
Unity 3D je v současné době nejrozšířenějším enginem na trhu herního průmyslu s podílem 47% [23]. Využívá se dokonce častěji, než vyvíjení nativním programováním.

V grafu (Graf.2) zastupuje modrý sloupec počet vývojářů, pro které je Unity hlavním nástrojem, a červené sloupce vyjadřují procentu vývojářů, kteří Unity aspoň částečně používají.



Graf.2 Procentuální zobrazení využití nástrojů při vývoji videoher [23]

Unity ještě více dominuje v mobilním sektoru. Pro vývoj 3D her je používán většinou vývojářů. V grafu (Graf.3) je srovnání ve vybraných lokacích. Je zřejmé, že jiné herní enginy v této sekci mají minimální podíl.



Graf.3 Podíl využívání Unity pro vývoj 3D her na mobilní zařízení [23]

V celkovém měřítku patří Unity 45% celkového trhu herního průmyslu. V číslech se jedná o 4.5 miliónu registrovaných vývojářů a 600 miliónů hráčů, kteří hrají nebo, jinak používají produkt postavený na jádře Unity [23].

3.1 IDE Unity

Vývojové prostředí Unity patří mezi moderní nástroje s intuitivním prostředím, a příjemným grafickým zpracováním. Snaží se být co nejvíce uživatelsky přívětivé a komfortně a přehledně dávat přístup ke všem funkcím a vlastnostem.

3.1.1 Hlavní okno

Hlavní okno editoru je složeno z několika částí. Jejich počet, umístění a velikost si může uživatel nastavit podle potřeby. Skládá se z:

- **Scene** – v podokně scene jsou zobrazeny všechny modely a objekty použité v daném levelu hry (pokud se nepřidávají pomocí skriptu). Slouží k přemísťování a uspořádání objektů a k úpravám velikosti a natočení.

- **Game** – zde se zobrazuje pohled, který vidí hráč při hraní, tedy pohled z kamery.

- **Inspector** – záložka inspector zobrazuje komponenty a jejich nastavitelné hodnoty, které jsou přiřazeny k označenému objektu. Například je zde základní komponenta Transform, která slouží k nastavení pozice, velikosti a rotace.

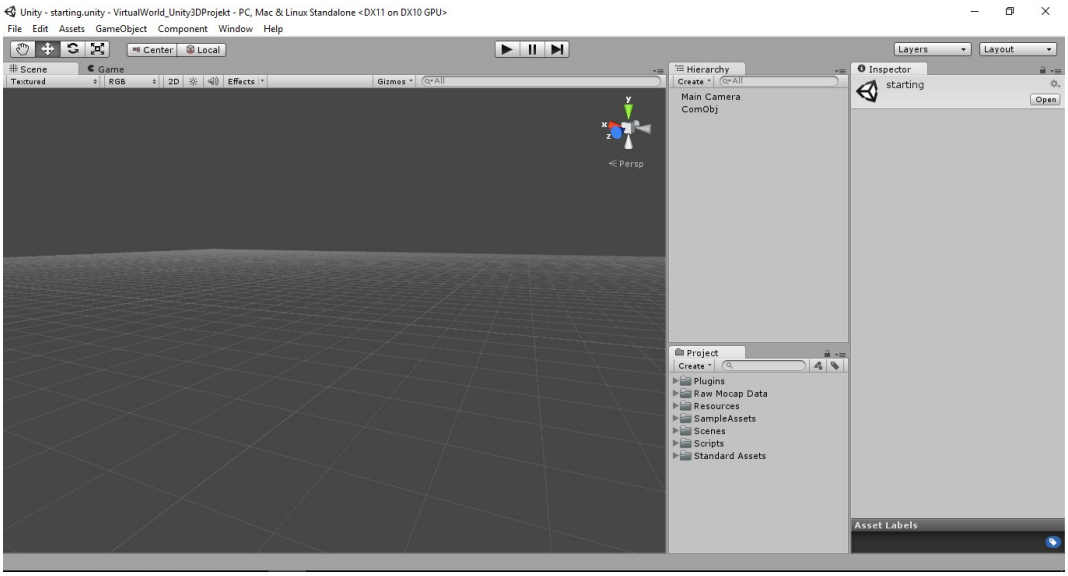
- **Hierarchy** – zobrazuje seznam objektů přítomných ve scéně. Slouží ke snadnějšímu přístupu k objektu a k jeho označení. Také slouží k vytváření rodičovských a dědičných objektů.

- **Project** – v tomto podokně je seznam složek a všech souborů k projektu. Nacházejí se zde scény, modely, skripty, audio soubory a další. Slouží k vytváření a organizaci těchto souborů.

- **Profiler** – toto okno lze využívat pouze s Professional licencí. Slouží k optimalizování hry a zobrazuje, kolik času trvají určité části hry (renderování, animace, atd.).

- **Animation** – slouží k zobrazení a editaci animací ve hře.

- **Console** – vypisuje chybové hlášky a odkazuje, kde se chyby, nebo warningy nacházejí. Vypisuje také ladící (debug) hlášky volané v kódu skriptů.



Obr.10 Unity vývojové prostředí

3.1.2 Play a Pause mód

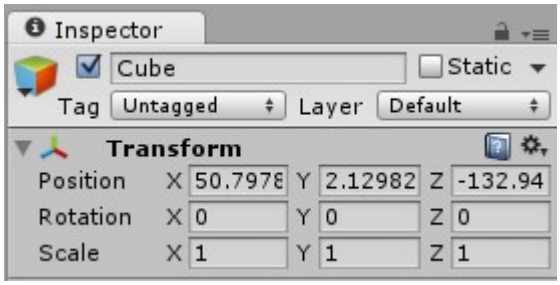
Velice užitečnou funkcí v Unity enginu je play a pause mód. Slouží ke spuštění a testování hry přímo ve vývojovém prostředí, bez nutnosti opakovaného vytváření aplikace. V pause módu lze během spuštěné hry měnit objekty ve scéně a jejich vlastnosti v inspektoru. To je velmi užitečné pro optimalizaci hry.

Aplikace, která je součástí praktické části této práce využívá síťového připojení na server a je spustitelná až po připojení určitého počtu hráčů. Při testování na jednom počítači je tedy nutné, aby hra byla spuštěná vícekrát. Tuto funkci však Unity neposkytuje. Bylo nutné opakovaně vytvářet aplikaci (exe soubor), a spouštět jednu v Unity a zbytek samostatně.

3.1.3 Komponenty

Komponenty jsou jednou z nejdůležitějších součástí enginu. S jejich pomocí nastavujeme chování objektů. Pomocí řady přednastavených a optimalizovaných komponentů lze ušetřit velké množství času, který by jinak vývojáři strávili při programování a skriptování. Nejčastěji se chování komponenty přidané k objektu nastavuje pomocí skriptu, který je k objektu také připojený. Tento skript má přístup k funkcím použitých komponent a k nastavení veškerých hodnot. V Unity jsou komponenty rozděleny do skupin:

- **Transform** – nejzákladnější komponenta, pomocí které se nastavuje pozice, rotace a velikost. Je součástí každého objektu od jeho vytvoření nebo importování do Unity.



Obr.11 Komponenta Transform

- **Mesh** – tyto komponenty slouží k vykreslování základní grafiky modelů v Unity. Patří sem:

  - *Mesh Filter* – vybere síť vybraného modelu a pošle ji do Mesh Renderer komponenty pro vykreslení.
  - *Text Mesh* – generuje 3D geometrii, která vykresluje text na display.
  - *Mesh Renderer* – pomocí sítě z Mesh Filter a pozice z Transform vykresluje objekty na display. Slouží také k nastavení renderovaného materiálu (textury) a k nastavení stínů objektu.

- **Effects** – skupina komponentů s různými grafickými efekty.

  - *Particle system* – slouží k simulaci látek jako tekutina, oheň, nebo mlha, generováním malých 2D obrázků.
  - *Trail a Line Renderer* – Line vykreslí spojnici mezi body v 3D prostoru a Trail znázorní dráhu pohybujícího se objektu.
  - *Lens Flare* – slouží k vykreslování odrazů světla na kameře.
  - *Halo* – vytváří světelné okolí kolem zdroje světla.

- **Physics** – komponenty simulující fyzikální vlastnosti.

  - *Rigidbody* – nejzákladnější fyzikální komponenta. Na objekt s Rigidbody působí gravitace, reaguje na další objekty, a pomocí skriptů lze přidat a nastavovat nejrůznější síly.
  - *Character Controller* – funkce pro ovládání, používaná ve hrách z pohledu první a třetí osoby.

- *Collider* – velice užitečná komponenta, která přidává neviditelné hranice objektům za účelem vytváření fyzikálních interakcí s ostatními objekty. Přednastavené collidery jako Box, Sphere, Capsule a Mesh Collider se aplikují podle tvaru objektu (krychle, koule, atd.). Wheel a Terrain Collider jsou specializované na kola a terén, ale pro funkční aplikaci vyžadují další nastavení.
- *Interactive Cloth* – funkce simulující chování látkového materiálu. Například ubrus na stole.
- *Skinned Cloth* – speciální funkce pro oblečení.
- *Hinge Joint* – komponenta používaná pro dva objekty s Rigidbody, aby tyto objekty byly spolu v závěsu. Použití například u dveří nebo řetězu.
- *Fixed Joint* – slouží ke spojení 2 objektů bez nutnosti spojování v hierarchii (rodič - dítě). Objekty se chovají jako jeden a lze nastavit velikost síly, která spojení přeruší.
- *Spring, Character, Configurable Joints* – další typy spojení pro různá specifická použití.
- *Constant Force* – přidá konstantní sílu k objektu.

Tyto výše zmíněné fyzikální komponenty jsou rozděleny na dvě skupiny, podle použití pro 3D nebo 2D objekty. Dalšími komponenty v Unity jsou:

- **Navigation** – navigační komponenty se používají pří hledání vhodné cesty. V některých případech při navigování hráče nebo NPC nelze jít přímou cestou z bodu A do bodu B, protože nám stojí v cestě překážky (budovy, les, apod.). Potřebujeme tedy nalézt jinou, pro nás vhodnou, cestu. K tomu navigační komponenty používají algoritmy umělé inteligence, kterým nastavujeme potřebný typ rozhodování.

- **Audio** – slouží ke správě a nastavení zvukových souborů.

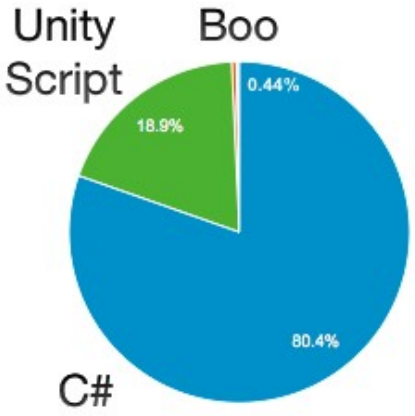
- **Rendering** – obsahuje skupinu komponentů, které ovládají vykreslování grafiky a veškeré elementy uživatelského rozhraní.

- **Miscellaneous** – tato skupina obsahuje komponenty různého druhu. Lze zde najít animační funkce, síťové funkce, Wind Zone, funkce pro simulaci větru, a další renderovací funkce.

3.2 Skriptování

Skript v Unity je považován za komponentu, která určuje chování objektů. Lze ji přidat k objektu stejně jako jakýkoliv jiný komponent. Programování skriptů je nedílnou součástí moderních enginů.

Skripty jsou soubory obsahující kód v některém z použitelných programovacích jazyků. Unity, od verze 5.0, podporuje dva jazyky. Jsou jimi jazyky C# a UnityScript (JavaScript pro Unity). V předchozích verzích ještě navíc obsahoval jazyk Boo, ale pro jeho malé využívání se rozhodli ukončit jeho podporu. Z Unity zmizelo tlačítko pro vytvoření Boo skriptu, a především se upustilo od vytváření dokumentace pro tento jazyk. Unity ovšem stále umí s těmito skripty pracovat.

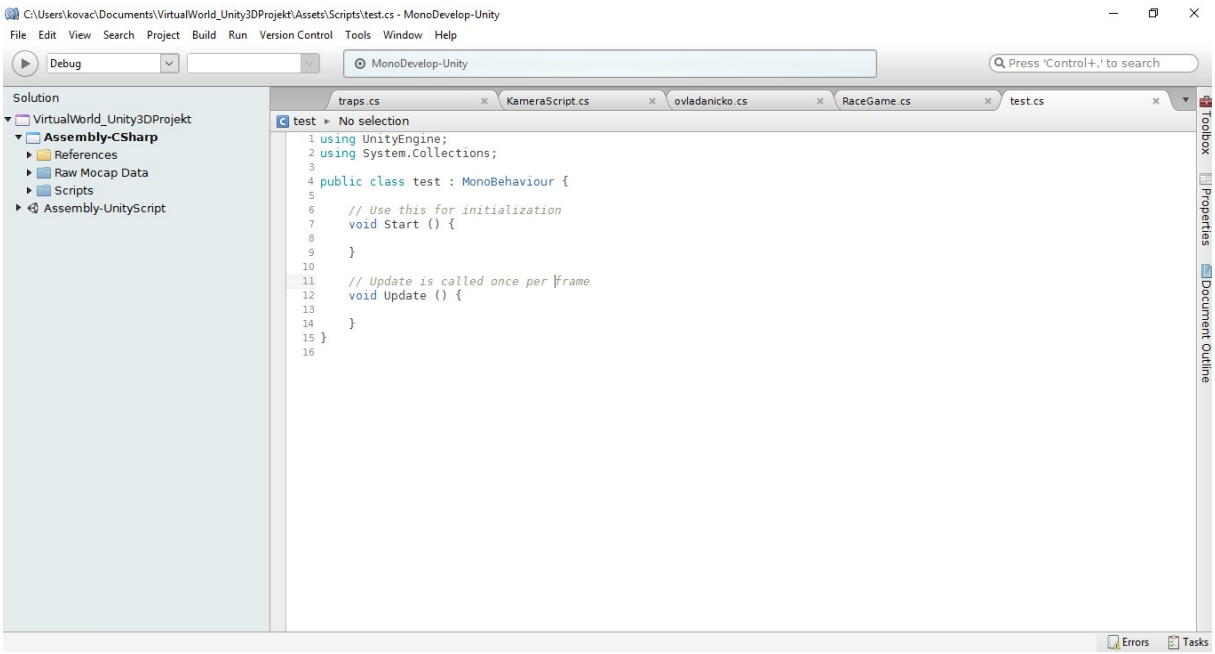


Graf.4 Procentuální využití skriptovacích jazyků v Unity (3. Září 2014)[26]

Při vytváření skriptů potřebuje mít programátor k dispozici dokumentaci s popisem veškerých funkcí a jejich vlastností, které jsou obsaženy v knihovnách enginu. Společnost Unity Technologies poskytuje na svých webových stránkách přehledný přístup k databázi veškerých funkcí s popisem, jak fungují, k čemu je lze použít a mnohdy jsou uvedeny i s konkrétními příklady a ukázkami kódů.

3.2.1 MonoDevelop

Pro samotné programování je k Unity poskytován editor MonoDevelop od firmy Xamarin (firmu vlastní Microsoft). Je to vývojové prostředí, ve kterém lze programovat celkem v 11 jazycích. Primárně se soustředí na projekty využívající Mono a .NET framework. MonoDevelop je propojené s Unity, takže pokud se v kódu vyskytuje chyba, Unity prostřednictvím okna Console vypíše chybovou hlášku, která přímo odkazuje na řádek, kde se chyba nachází.



Obr.12 Vývojové prostředí MonoDevelop

MonoDevelop není jedinou možností pro Unity. Lze například použít i Microsoft Visual Studio, ke kterému existuje plugin, pro propojení s Unity.

3.2.2 Struktura skriptu v jazyce C#

Nejpoužívanějším skriptovacím jazykem v Unity je podle grafu (Graf.4) jazyk C#. Struktura prázdného skriptu C# a UnityScriptu se liší.

```
1 using UnityEngine;
2 using System.Collections;
3
4 public class test : MonoBehaviour {
5
6     // Use this for initialization
7     void Start () {
8
9     }
10
11    // Update is called once per frame
12    void Update () {
13
14    }
15 }
```

Obr.13 Prázdný skript v jazyce C#

```
1 #pragma strict
2
3 function Start () {
4
5 }
6
7 function Update () {
8
9 }
```

Obr.14 Prázdný skript v jazyce UnityScript

Skript v jazyce C# obsahuje v základu reference na dvě knihovny z Unity, a také třídu, která dědí ze základní třídy MonoBehaviour. UnityScript skript tyto části také obsahuje, ale jsou skryté. Třída MonoBehaviour je základem skriptu, bez kterého by Unity neumělo skript spustit. Obsahuje funkce *Start()* a *Update(),*díky kterým Unity ví, kde začít a jak pokračovat.

Začíná se funkcí *Start(),* která se provádí pouze jednou, jakmile je skript spuštěný. Obvykle slouží k nastavení proměnných nebo přiřazování komponent objektům, apod. Následně se spustí funkce *Update()*. Tato funkce je nejčastěji využívanou funkcí pro implementaci chování objektů. Je to hlavně z důvodu, že se volá opakovaně každý snímek hry.

Třída MonoBehaviour obsahuje velké množství užitečných funkcí, které zjednodušují programování. Funkce *Start()* a *Update()* doplňuje funkce *Awake()*. Ta je jedenkrát zavolána, ještě před funkcí *Start()*, a poté, co jsou načteny všechny objekty ve scéně. Je vhodná pro navázání komunikace s objekty ve hře. Další funkcí je *FixedUpdate()*. Na rozdíl od *Update()*, která se volá každý snímek, tato funkce je volaná ve stejných intervalech nezávisle na výkonu hardwaru. Je tedy vhodná k použití v situacích, kdy rozdílný počet snímků za vteřinu u různých počítačů způsobuje nežádoucí problémy. Do skupiny update funkcí patří ještě *LateUpdate()*. Volá se každý snímek, ale až jako poslední po *Update()* a *FixedUpdate()*. Tuto vlastnost je vhodné využít například u kamery, která sleduje objekty, jejichž chování se mění v ostatních update funkcích.

Zbytek funkcí v základní třídě mají předponu “On” a většinou reagují na nějaký podnět nebo situaci. Patří sem například OnCollisionEnter(), která se spustí při interakci daného objektu s jiným, dále OnMouseDown(), která reaguje na kliknutí myší, nebo OnAplicationQuit(), která se spustí před vypnutím aplikace, a spousta dalších.

4 Vlastní aplikace

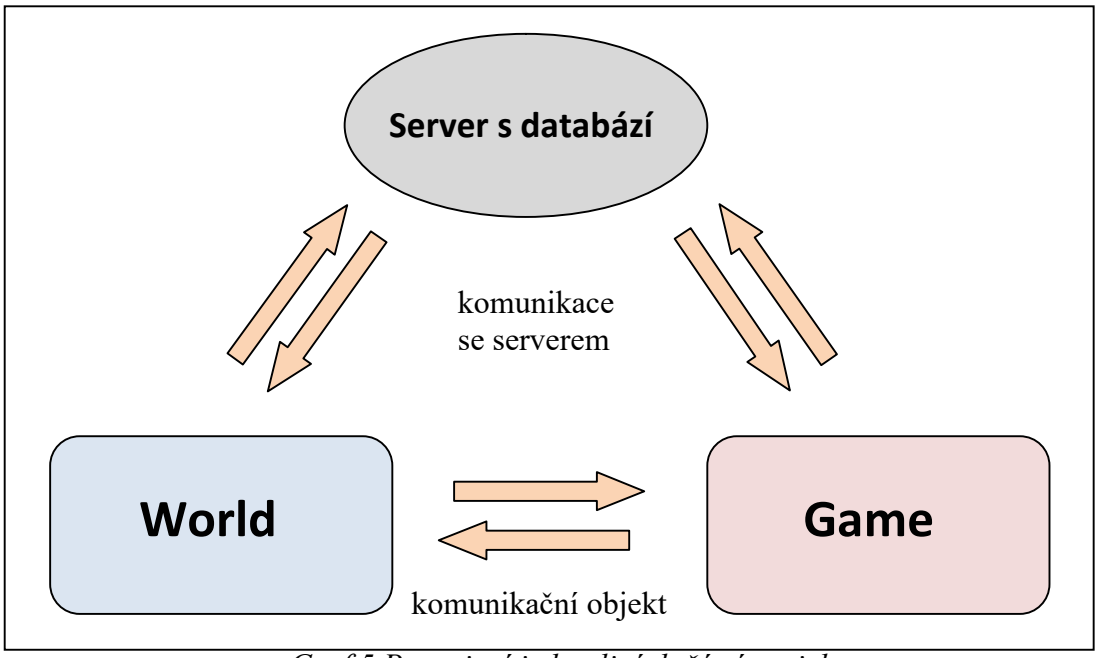
Součástí této práce je vytvoření aplikace s použitím technologie Unity 3D. Celkový projekt je rozdělen na 3 části, a každá je zpracovávaná v rámci jiné diplomové práce. Hlavním bodem, který má výsledná aplikace splňovat, je propagace fakulty strojního inženýrství.

#	Části projektu	Zpracoval
1.	Server a komunikace	Daniel Balcárek
2.	Virtuální svět	Jakub Hamal
3.	Závodní hra	Robert Kováč

Tab.2 Části projektu Virtual World

Veškeré nápady a návrhy vznikaly na základě společné diskuze všech tří zúčastněných a i při následném zpracování jednotlivých částí byla zapotřebí častá komunikace. Z celkového hlediska, práce v týmu byla bezproblémová a vždy se našlo společné řešení (k tomu napomohl především nízký počet členů). Ale i v našem týmu jsme se často dlouho nemohli shodnout, a kvůli absenci rozhodovacího článku, byla někdy cesta k dohodě velice dlouhá.

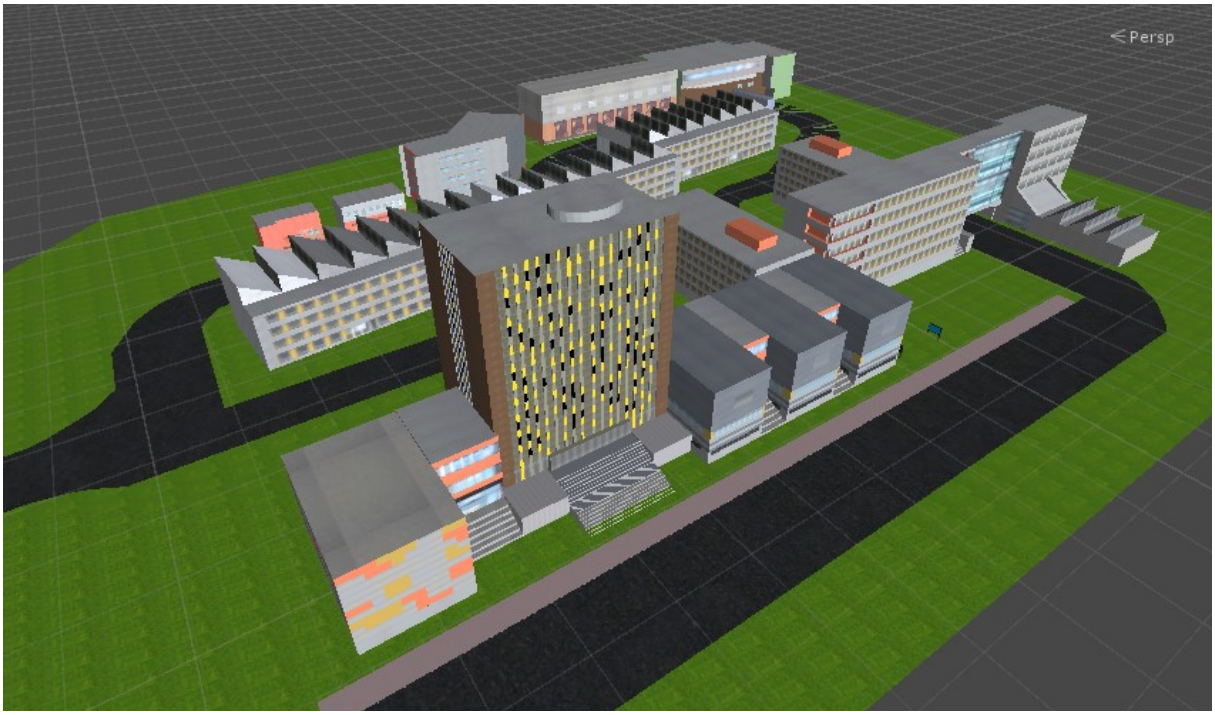
Části projektu (Tab.2) virtuální svět a závodní hra jsou propojené pomocí serveru s databází. Mezi světem a hrou probíhá komunikace pomocí komunikačního objektu, který si mezi sebou posílají.



Graf.5 Propojení jednotlivých částí projektu

4.1 Základní popis

Jak název Virtual World napovídá, jedná se o virtuální svět, neboli virtuální prostor, ve kterém se hráč může libovolně pohybovat. Tento prostor je z důvodu propagace fakulty modelovou kopií areálu VUT FSI. Modely byly vytvářeny v grafickém programu Blender a následně importovány do Unity. Textury na modelech jsou nafocené z reálných budov a opět pomocí Blenderu nanesené na modely.



Obr.15 Vymodelovaný areál Fakulty strojního inženýrství

Dalším charakterizujícím přívlastkem aplikace je „online“. Do virtuálního světa se lze připojit pomocí síťového připojení z počítačů připojených k dané síti. Každý připojený hráč musí být registrovaný a po přihlášení pomocí jména a hesla získá přístup do světa, ve kterém je zastoupený modelem postavy člověka (na výběr je muž a žena). Tyto postavy simulují chůzi reálné osoby a mohou se pohybovat kdekoliv ve světě, kromě budov, kterými nelze projít. Chůze se ovládá pomocí tlačítek W,A,S,D nebo šipek.

Kromě procházení po areálu školy může hráč i komunikovat s ostatními lidmi prostřednictvím chatového okna. Do okna se napíše text a hráč si může vybrat, jestli chce zprávu poslat všem připojeným hráčům, nebo soukromě určitému člověku.

Další funkcí ve světě jsou interaktivní panely, které reagují na kliknutí levým tlačítkem myši, pokud je hráč dostatečně blízko. Tyto panely se nacházejí před budovou A4. Jeden z nich, s označením „Informační panel“ zobrazí po kliknutí okno, ve kterém si může hráč zobrazit mapu areálu s popisy budov. Druhý panel označen „Panel hry“ se používá pro připojení k závodní hře. Každý připojený hráč ve světě si může zahrát hru s ostatními. Po kliknutí na panel se objeví tlačítko závodní hra (pro případ, kdyby v dalším vývoji byly přidány další hry), vybere si barvu auta a zmáčkne tlačítko připojit ke hře. Hra je nastavená pro 4 hráče a jakmile se tento počet lidí připojí, závod začíná.

4.2 Motiv a logika hry

Žánrem patří hra do skupiny arkádových závodů. Závodem je myšlen klasický systém start-cíl s tím, že všichni závodníci začínají ve stejný okamžik a výsledné pořadí závisí na rychlosti dojetí do cíle. První závodník v cíli je vítěz.

Pokud je hra vytvářena jako simulátor, vývojáři se zpravidla snaží s pomocí znalostí fyzikálních zákonů a pravidel, co nejpřesněji napodobovat reálné chování ve skutečném světě. Také ovládání vozidel bývá realistické. Hráč musí hlídat rychlost a otáčky a podle toho řadit stupně rychlosti, nebo může zajet do depa a vyměnit pneumatiky, atd.

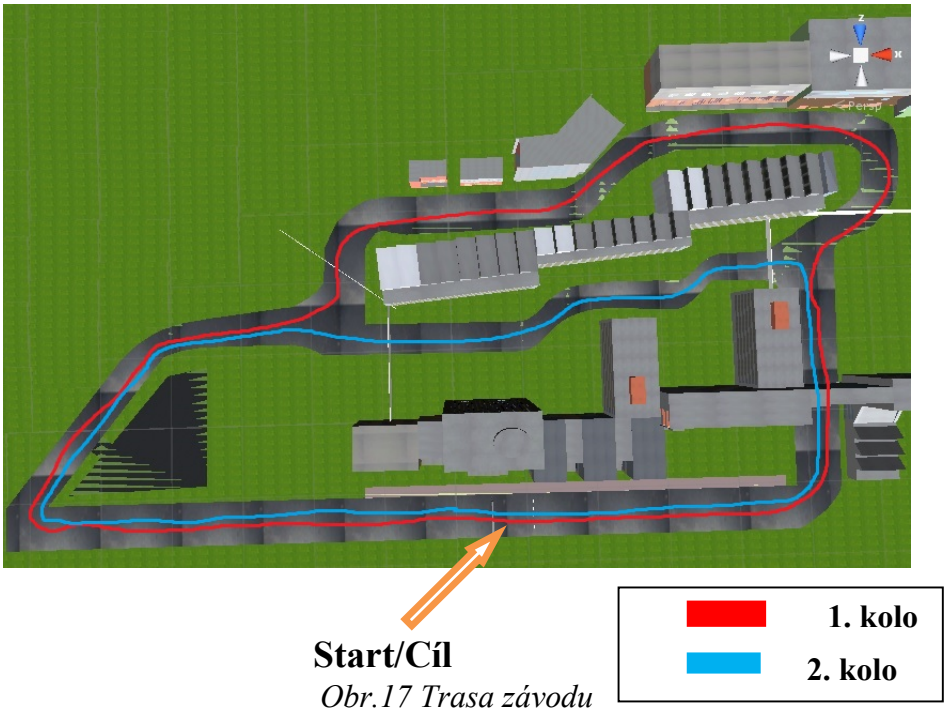
Jiným případem jsou hry s arkádovými rysy, kde se spolu s reálnými prvky objevují nereálné aspekty. Ty mají za úkol změnit hratelnost (většinou zlehčit) a zvýšit zábavný pocit ze hry. Jednou z populárnějších arkádových závodů je hra Mario Kart, která byla částečnou inspirací pro závod ve Virtual World. Tato hra je vydávána od roku 1992, až do dnešní doby. Typickým arkádovým prvkem v ovládání vozidla jsou nereálné skluzy, možnost skákat s autem, rotace kolem svislé osy ve středu vozidla nebo poškodit ostatní závodníky a spousta dalších.

Obr.16 Ukázka ze Mario Kart

Virtual World závod obsahuje spoustu arkádových prvků. Chování vozidla je částečně realistické. Napodobují se základní fyzikální vlastnosti, jako třecí síly mezi pneumatikami a závodní dráhou, schopnost zatáčení se zhoršuje při zvyšující se rychlosti, nebo napodobení brzdného smyku při použití ruční brzdy, a podobně. Ostatní aspekty jsou pak zjednodušené vůči realitě, nebo úplně vymyšlené a nereálné.

Po přihlášení do hry a naplnění potřebného počtu hráčů jsou závodníci seřazeni do jedné řady se stejnou vzdáleností od startovní čáry. Uskupení závodníků je určeno podle pořadového čísla, pod kterým se přihlásili do závodu. První přihlášený bude na levé straně a poslední na pravé straně řady ve směru závodu. Po uplynutí odpočítávací sekvence závod začíná.

Samotný závod se jede na 2 kola. První kolo je delší a vede kolem téměř celého areálu. Ve druhém kole jede závodník na začátku po stejné trati, ale pak musí odbočit, projet mezi budovami a následně se opět vrací na hlavní okruh.



Obr.17 Trasa závodu

Mimo klasické závodění se mohou hráči navzájem konfrontovat a ztěžovat si závod. Při hře každý závodník může projet objekty, které jsou rozmístěny na vozovce, a tím získat „zbraň“, kterou pak může kdykoliv použít. Nejsou to zbraně ve smyslu, že jimi lze někoho sestřelit, ale spíš působí jako překážky.

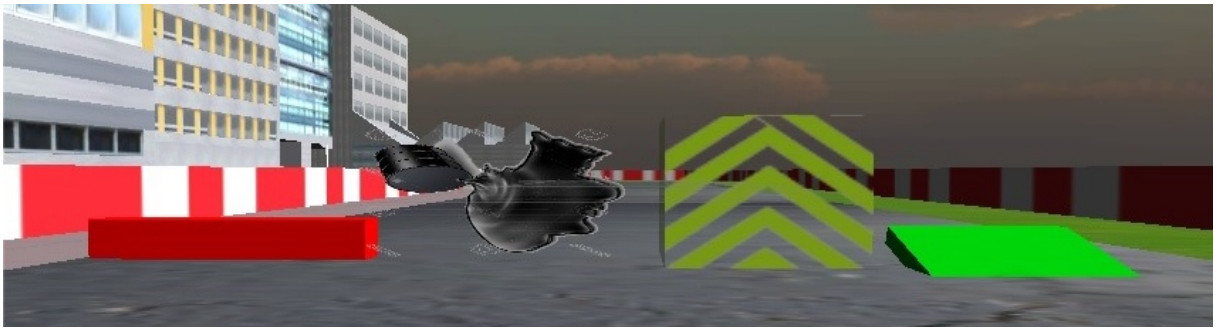
Tyto překážky jsou celkově 4:

- **Wall**: červená zeď, která donutí ostatní hráče k objíždění překážky, nebo v lepším případě k nárazu.

- **Fluid**: ve hře znázorněno obrázkem rozlitého oleje. Kdo jím projede, ztratí částečně kontrolu nad svým vozidlem na dobu pěti vteřin.

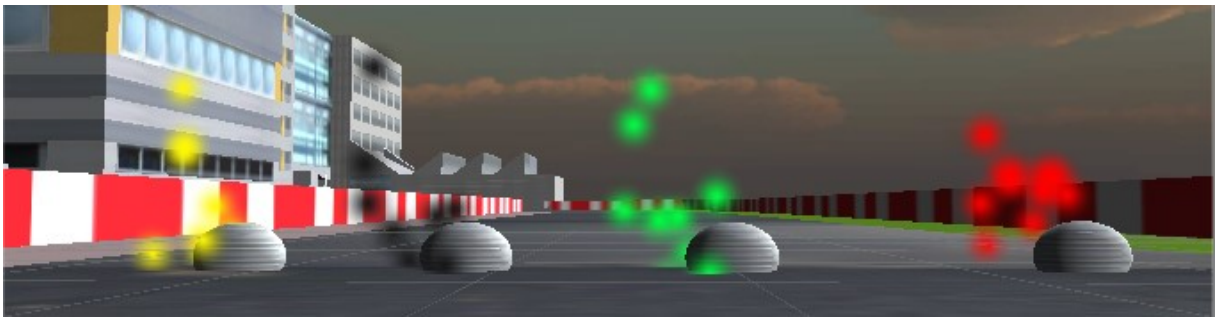
- **Speed**: žlutými šipkami označená překážka, která hráči zvýší o 10% okamžitou rychlost. Při špatném umístění může protihráčům spíše pomoc, než uškodit.

- **Jump**: zeleně označená rampa, která závodníka vymrští do vzduchu. Opět pro negativní ovlivnění ostatních je potřeba ideálně umístit, například za zatáčku.



Obr.18 Překážky v závodu

Barevné rozlišení překážek je stejné, jako u objektů, které se vyskytují na závodní dráze, a které zbraně aktivují. Tyto objekty jsou zploštělé koule, které emitují částice v požadované barvě, takže závodník se může rozhodnout, kterou zbraň si zvolí.



Obr.19 Objekty pro aktivaci zbraní

Po odjetí obou kol všemi hráči se závod ukončí a dojde k vyhodnocení. Výsledné pořadí hráčů je dáno pořadím dojetí do cíle.

Jelikož hra se hraje po síti a je součásti světa, který je navržen pro velké množství lidí, bylo nutné vytvořit systém hodnocení, který je dlouhodobější, aby bylo možné, na základě všech hráčových výsledků, porovnávat se nejen s účastníky jednoho závodu, ale i se všemi ostatními registrovanými uživateli. Dále, aby toto hodnocení případně podněcovalo hráče k opakovanému hraní.

K porovnání hráčů dochází na základě koeficientu, který se zvyšuje při úspěchu v závodě, a naopak snižuje při neúspěchu.

Umístění	Koeficient
1.	+ 0,3
2.	+ 0,1
3.	- 0,1
4.	- 0,3

Tab.3 Bodové ohodnocení celkového žebříčku hráčů

Body za umístění jsou následně násobeny pomocným koeficientem, který závisí na počtu uskutečněných závodů daného hráče. Se zvyšujícím se počtem závodů se pomocný koeficient snižuje. Tím postupně klesá i celkový přísun bodů na závod. Je to podobný styl jako v klasických RPG hrách, kde je hráč ohodnocen zvýšením stupně postavy, a čím vyšší stupeň hráč dosáhl, tím déle mu trvá dosáhnout dalšího stupně.

Počet závodů	Koeficient (pomocný)
0 - 10	1
11 - 25	0,8
26 - 75	0,6
76 - 200	0,5
201 - 500	0,5
501 - 1000	0,3
1001 a více	0,2

Tab.4 Pomocný koeficient závislí na počtu závodů

4.3 Komunikace se serverem

Pro vývoj serveru byl použit multiplatformní vývojový framework Photon Server, který je přímo určený pro vývoj her a obsahuje knihovny pro serverovskou i pro klientskou část.

Komunikace probíhá pomocí tzv. eventů, které jsou reprezentovány delegáty z importovaných knihoven. Tyto knihovny jsou umístěny přímo v projektu, ve složce Assets/Plugin, která slouží pro tyto účely. Celkovou komunikaci zajišťuje třída *MyRaceGamePeer*. Tato třída je obsažena ve zmíněných knihovnách a obsahuje potřebné eventy. Ve skriptu se vytvoří objekt (*myPeer*), který je datového typu *MyRaceGamePeer* a potřebné eventy pro hru zpřístupní.

K nastavení této třídy dochází před startem závodu, ve skriptu *PanelGame.cs*, který je komponentou panelu ve světě, sloužícího pro přihlášení do hry. V eventu *RaceGameEvents_OnStartGame*, který je součástí komunikační třídy pro virtuální svět se nastaví komunikační objekt *passObj*, se speciální komponentou *PassingObjekt*, neboli třídou, která je navržena pro uchování potřebných hodnot pro komunikaci. Tyto hodnoty se tedy nastaví a pomocí funkce *DontDestroyOnLoad()* se zajistí, aby tento komunikační objekt nebyl při ukončení scény zničen. Objekt *passObj* tedy slouží ke komunikaci mezi scénami. V našem případě pro komunikaci mezi světem a hrou, kde si posílají nastavené komunikační třídy, přístupové informace a data o připojených hráčích.

```
public class PasssingObj : MonoBehaviour
{
        public GameObject PassingObject { get; set; }
        public MyRaceGamePeer RacePeer { get; set; }
        public MyClientPeer ClientPeer { get; set; }
        public string PlayerName = "";
        public string PlayerLoginName = "";
        public string PlayerPassword = "";
        public Racer[] racers { get; set;}

        public PasssingObj()
        {

        }
}
```

Obr.20 Struktura třídy PassingObj

Následující scénou je samotná závodní hra. V té je vytvořený prázdný objekt (*game*), ke kterému je přiřazený skript *RaceGame.cs*. Tento skript zajišťuje hlavní funkce hry a komunikaci se serverem (i další skripty komunikují se serverem). Ve funkci *Start()* jsou nejdříve deklarovány všechny eventy a jejich funkce se následně volají, pokud přijde podnět ze serveru.

```
myPeer.RaceGameEvents.OnReturnCode += RaceGameEvents_OnReturnCode;
myPeer.RaceGameEvents.OnGameRacersLog +=RaceGameEvents_OnGameRacersLog;
myPeer.RaceGameEvents.OnRacerMove +=RaceGameEvents_OnRacerMove;
myPeer.RaceGameEvents.OnStartGame += RaceGameEvents_OnStartGame;
myPeer.RaceGameEvents.OnResults += RaceGameEvents_OnResults;
```

Obr.21 Deklarace eventů

Popis eventů:

- OnReturnCode – tento event slouží k zobrazení zprávy o stavu serveru.
- OnGameRacersLog – obsahuje operace pro přidání a smazání hráčů z pole, které uchovává data všech hráčích ve hře. Používá se před spuštěním hry, pro vytvoření daného pole, ale i při spuštěné hře, pokud dojde k odhlášení hráče během závodu.
- OnRacerMove – jeden z nejdůležitějších eventů, jelikož slouží k přijímání dat o pozici a rotaci ostatních hráčů. Tyto hodnoty se nastavují modelu auta, které zastupuje dalšího hráče, a my ho díky tomu vidíme na trati a víme, kolikáté místo nám průběžně patří.
- OnResults – vyhodnocovací event, který zachytává jednotlivé výsledné časy hráčů.

Dále třída *MyRaceGamePeer* obsahuje různé funkce pro správu komunikace. Funkci *Connect()* pro připojení k serveru, *Disconnet()* pro odpojení a funkci *Update()* pro aktualizaci serveru. Dále obsahuje funkci *SetMyRacer()*, pro vybrání barvy vozidla a důležitou funkci *SendMove()* pro posílání pozice a rotace na server. Tato funkce se volá každý snímek ve funkci *FixedUpdate()*. Pravděpodobně jde o funkci, která nejvíce zatěžuje server, jelikož všichni hráči, každý snímek posílají svá data a server je obratem rozesílá všem.

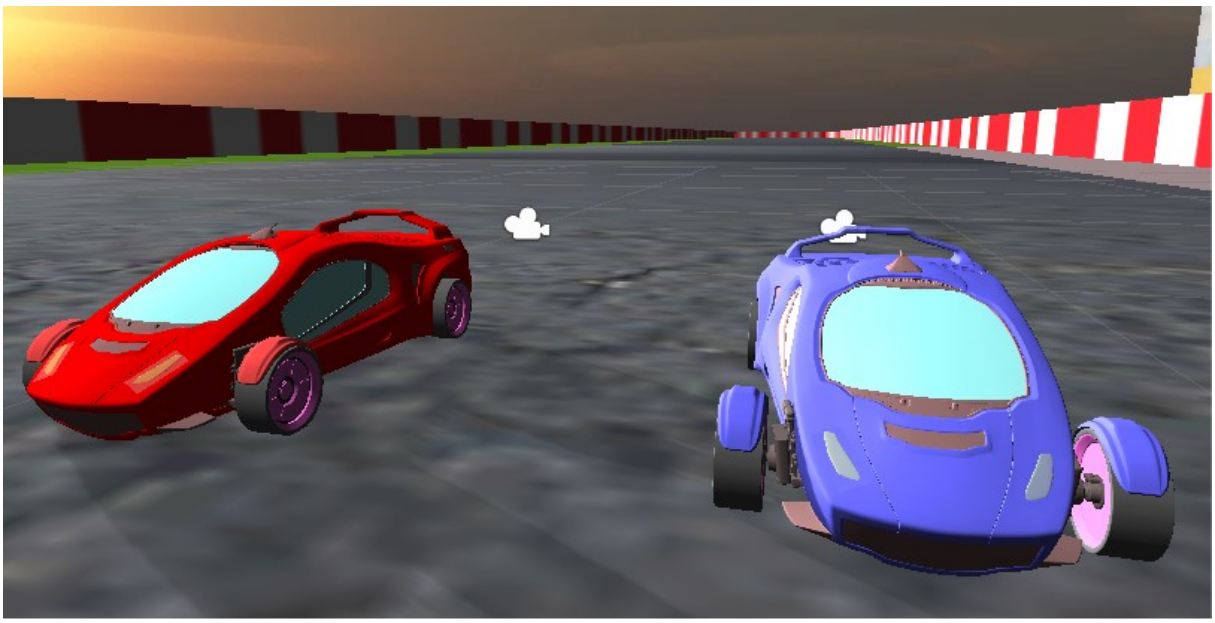
```
RacerMoveData moveData = new RacerMoveData(playerName, position, rotation);
myPeer.SendMove(moveData);
```

Obr. 22 Funkce posílající data o hráči na server

Po projetí cílem se volá funkce *SendFinish()*, která posílá výsledný čas na server a následně dochází k celkovému vyhodnocení.

4.4 Modely

Vytváření modelů je při malých zkušenostech časově velice náročné a výsledky jsou mnohdy nekvalitní. Z toho důvodu byly v některých případech použity zdarma přístupné modely z Unity Asset Sotre. Například model auta, který je v celé hře graficky nejsložitější.



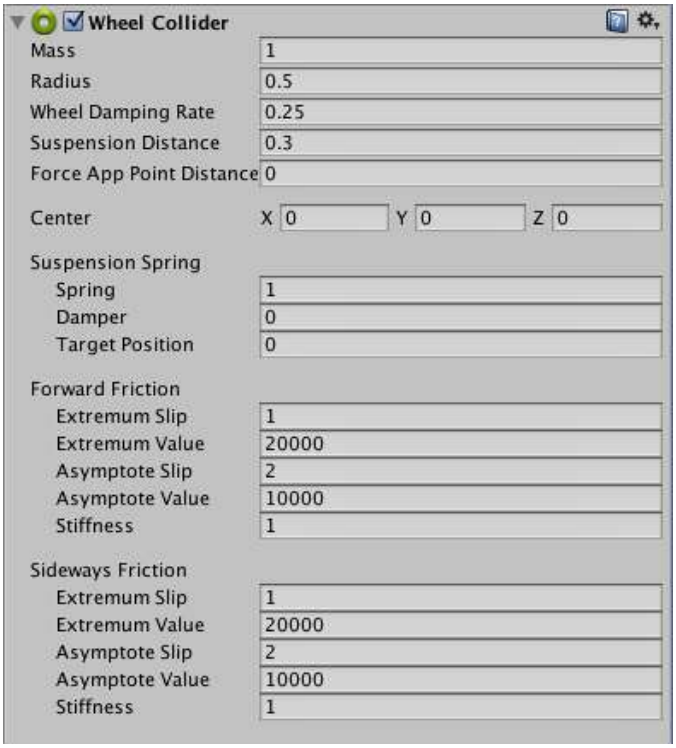
Obr. 23 Modely aut

Zbytek modelů napr. mantinely a směrové ukazatele na dráze nebo zdi, skokanské rampy, startovní cedule a další, byl vytvořen z jednoduchých základních tvarů typu kvádr, nebo koule, a byl i jednoduše zabarven. Při případné další práci na projektu by jedna z částí měla být, vytvoření propracovanějších a hezčích modelů, a následně nanesení hezčích textur.

4.5 Ovládání vozidla

U závodní hry je jednou ze základních částí, a mnohdy také nejsložitější, ovládání vozidla. K vytvoření arkádového, tedy z části nerealistického, ovládání se postupovalo tak, že se nejdřív napodobily základní síly, které reálně na auto působí a následně se upravily, aby působily nereálně.

Základem ovládání je komponenta *WheelCollider*. Ta je přidaná ke každému kolu zvlášť, aby bylo možné rozlišit přední a zadní kola, a rozdílně je nastavit. Komponenta je speciálně navržená pro kola vozidel jezdících po zemi. Obsahuje detekci kolizí, fyziku kol a model pro simulaci tření.



Obr.24 Komponenta WheelCollider

K ovládání vozidla ve hře slouží skript *Ovladani.cs*, který je přiřazený k prefab objektu vozidla. Prefaby jsou speciální objekty v Unity, které slouží převážně k snadnějším úpravám u objektů, které se ve hře opakují. Když se změní prefab, změní se i všechny ostatní objekty, které se z prefabu načítají.

Hnací síla

Síla, která simuluje výkon motoru a pohání vozidlo dopředu, je závislá na několika faktorech. Hlavním parametrem, který ovlivňuje absolutní výkon je maximální rychlost (*maxSpeed*). Tu lze nastavit přímo ve skriptu, nebo i v Unity inspektoru, jelikož je proměnná ve skriptu deklarovaná s přívlastkem public.

Následně ve *FixedUpdate()* funkci snímáme vstupy z klávesnice, pomocí funkce *GetAxis()* z třídy *Input*, která v závislosti na parametru určuje tlačítka na ovládání. Defalutně je nastaveno, pro parametr *„Vertical“* ,písmena W a S, případně šipky nahoru a dolů. Pro parametr *„Horizontal“* zase písmena A a D, a šipky doleva, doprava.

```
if (currentSpeed < maxSpeed && currentSpeed > -maxBackSpeed) {
    frontLeft.motorTorque = maxTortuge * Input.GetAxis ("Vertical");
    frontRight.motorTorque = maxTortuge * Input.GetAxis ("Vertical");

} else {
    frontLeft.motorTorque = 0f;
    frontRight.motorTorque = 0f;

}

if (Input.GetButton ("Vertical") == false) {

    rearLeft.brakeTorque = declarationSpeed;
    rearRight.brakeTorque = declarationSpeed;

} else {

    rearLeft.brakeTorque = 0f;
    rearRight.brakeTorque = 0f;
}

float speedFactor = rigidbody.velocity.magnitude / lowestSteerSpeed;
float currentSteerAngle = Mathf.Lerp (lowSpeedSteerAngle, heighSpeedSteerAngle, speedFactor);
currentSteerAngle *= Input.GetAxis ("Horizontal");
frontLeft.steerAngle = currentSteerAngle;
frontRight.steerAngle = currentSteerAngle;
```

Obr.25 Část kódu ovládání vozidla

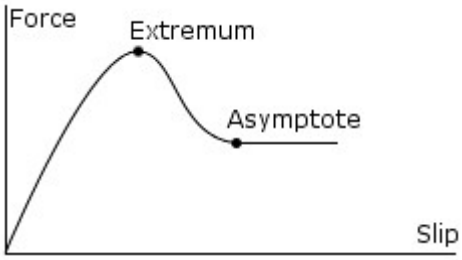
Samotná hnací síla je reprezentována pomocí funkce *motorTorque()* z třídy *WheelCollider*, která vytváří točivý moment motoru na nápravu kola v newton metrech. Pomocí podmínky se kontroluje, zda vozidlo nepřekročilo maximální rychlost. Dokud ji nepřekročí, snímají se vstupní tlačítka v závislosti na délce stlačení tlačítka. Snímá se v rozmezí od -1 do 1 a násobením koeficientem *maxTortuge* se vytváří hnací síla.

Pro zatáčení obsahuje třída *WheelCollider* funkci *steerAngle()*, která vrací hodnotu úhlu ve stupních. V této části se pomocí proměnných *lowSpeedSteerAngle* a *heighSpeedSteerAngle* přepočítává schopnost zatáčení v závislosti na rychlosti auta. Jde o napodobení reálného chování těles pohybujících se po zakřivené trajektorii, a působení setrvačných sil, které jsou závislé na rychlosti. Auto tedy lépe zatáčí při nízké rychlosti a naopak hůře při vysoké rychlosti.

Dalším ovládacím prvkem u auta je ruční brzda. Ta se používá zmáčknutím mezerníku na klávesnici. Při jízdě dopředu se projeví prudkým snížením rychlosti. Jiné to je u použití ruční brzdy při zatáčení. Pro napodobení smyku se při zmáčknutí ruční brzdy mění hodnoty tření mezi pneumatikou a vozovkou. Třecí síly se výrazně zmenší, a auto se lehce natočí a zastaví. Tato simulace smyku je spíše arkádová, než reálná. Pro reálnější napodobení by bylo zapotřebí použít jiný způsob řešení, na základě působení brzdných a setrvačních sil, které na auto působí při brždění v reálném světě.

Tření

Třída *WheelCollider* obsahuje i model pro tření mezi koly a vozovkou nazvaný *Wheel Friction Curves*. Vyhodnocuje zvlášť třecí síly v dopředném, a zvlášť v bočním směru, na základě natočení kol k vozovce a rychlostního rozdílu mezi koly a vozovkou.



Graf.6 Průběh třecí síly v závislosti na natočení vozidla (skluzu)

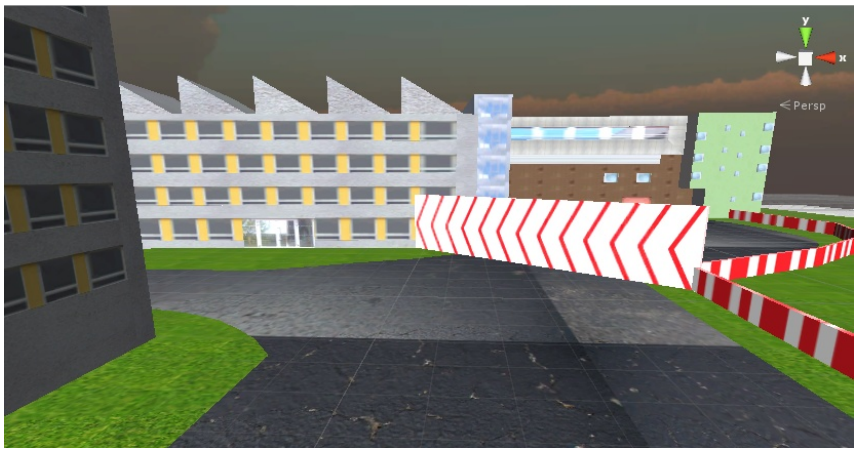
Průběh třecích sil je rozdělen na dvě části (Graf.6). První je z bodu (0,0) k (*ExtremumSlip, ExtremumValue*), kde tangenta je rovna 0. V tomto bodě působí největší třecí síly. V praxi se jedná o moment při zatáčení, kdy auto stále drží směr a je těsně před smykem. Druhá část grafu pokračuje k bodu (*AsymptoteSlip, AsymptoteValue*), kde je tangenta opět nulová. Zde se jedná o část za hranicí přilnavosti pneumatik, kdy vozidlo jde do smyku. Poslední nastavitelnou hodnotou u *Wheel Friction Curve* je hodnota Stiffness. Ta slouží jako násobitel ostatních hodnot. V základu je nastavená na hodnotu 1, a pokud ji nastavíme například na 0, zruší se veškeré třecí síly.

S využitím třecího modulu se tedy nastavuje základní ovladatelnost auta při jízdě rovně, i do zatáčky. Obdobný systém jako u ruční brzdy byl použit, při průjezdu překážkou fluid. Zde se také mění hodnoty tření, které simulují zhoršení ovladatelnosti vozidla.

4.6 Check point systém

Aby bylo možné závod spravedlivě vyhodnotit a zamezilo se podvádění, byl vytvořen check point systém. Ten funguje na bázi neviditelných objektů rozprostřených po dráze, kterými hráč musí projet, aby mohl pokračovat v závodě a aby došlo k započtení jeho výsledku. Na křižovatce, která rozděluje první a druhé kolo, se objevují neprůjezdné mantinely se směrovými šipkami, právě na základě toho, zda hráč projel všemi check pointy, kterými měl. Takže pokud by se nějakým způsobem chtěl vyhnout plánované trase, nemohl by odjet druhé kolo, nebo by nedošlo k započtení výsledného času.

Obr.26 Směrování na první kolo



Obr.27 Směrování na druhé kolo

Velice užitečnou funkcí, použitou v check point systému, je funkce *OnTriggerEnter()* z třídy *MonoBehaviour*. Ta detekuje interakci se všemi collider komponentami. Pokud tedy skript je komponentou objektu auta a auto narazí do jiného objektu, který má mezi komponenty jakýkoliv collider, zavolá se funkce *OnTriggerEnter()* s parametrem reprezentujícím detekovaný collider. S pomocí parametru a funkce *CompareTag()* se detekuje, jakým objektem auto projelo. Tag je datového typu string a slouží k identifikacím objektů. Lze ho nastavit pomocí skriptu, nebo i přímo v Unity inspektoru, ale musí být deklarovaný mezi ostatními tagy v Unity.

```
void OnTriggerEnter(Collider other)
{

    if (other.CompareTag("starting"))
    {
        Debug.Log("Start");
        Debug.Log (Lap);
        started = true;
        if (Lap == 1)
        {
            wallFirst1.transform.collider.isTrigger = true;
            wallFirst2.transform.collider.isTrigger = true;

            wallFirst1.renderer.enabled = false;
            wallFirst2.renderer.enabled = false;
            wallSecond1.renderer.enabled = true;
            wallSecond2.renderer.enabled = true;

            Debug.Log("Prvni kolo");

        }
```

Obr. 28 Funkce OnTriggerEnter

4.7 Kamera

Kamera je speciální objekt v Unity, pomocí kterého se zobrazuje dění ve hře pro hráče. Podokno Game v Unity IDE slouží k náhledu z kamery. Při spuštěném play módu přímo zobrazuje dění ve hře. Tento objekt, jako každý jiný, má mnoho speciálních funkcí a pomocí skriptu je možné libovolně nastavit chování podle potřeby pro daný typ hry. Například při 2D hrách obvykle bývá kamera statická nebo se pohybuje v jedné, či dvou osách. Naopak u first-person střílecí hry je kamera spojena s charakterem hráče a kopíruje jeho pohyb, jako by se hráč díval vlastníma očima. Kamer může být v jedné scéně i více než jedna. Mohou se střídat ve snímání z různých úhlů a míst nebo v různých vzdálenostech.

U závodní hry bylo důležité nastavit kameru, aby sledovala vozidlo hráče v závislosti na rychlosti a natočení. Je to důležité pro lepší herní zážitek. Pokud by kamera jenom suše kopírovala pohyb a natočení auta, hráč by neměl dobrý přehled o tom, kde se nachází a jestli dostatečně zatáčí. Bylo nastavené zpoždění, se kterým se kamera vrací do původní polohy za auto, případně před auto, pokud hráč couvá. V závislosti na rychlosti vozidla se kamera lehce oddaluje, anebo přibližuje.

Objekt kamery je připojený k vozidlu jako dědičný dobjekt. To ovšem není z důvodu propojení kamery s vozidlem, o to se stará skript *CarCamera.cs*, ale protože vozidlo hráče se do scény nahrává pomocí hlavního skriptu ze složky *Resources*, a pokud by nebyli objekty spojené, musela by se kamera nahrávat zvlášť. Skript *CarCamera.cs* je komponentou objektu kamery a nastavuje jeho celkové chování. Použita byla především funkce *LateUpdate()*, která je pro nastavení kamery nejvhodnější, protože se vyhodnocuje každý snímek, stejně jako ostatní upadte funkce, ale jako poslední z nich. Díky tomu se veškeré chování objektů v jednom snímku provede dřív, než se zaznamená na kameru.

```
void LateUpdate()
{
    float speedFactor = Mathf.Clamp01(target.root.GetComponent<Rigidbody>().velocity.magnitude / 70.0f);
    GetComponent<Camera>().fieldOfView = Mathf.Lerp(55, 72, speedFactor);
    float currentDistance = Mathf.Lerp(7.5f, 6.5f, speedFactor);

    currentVelocity = currentVelocity.normalized;

    Vector3 newTargetPosition = target.position + Vector3.up * height;
    Vector3 newPosition = newTargetPosition - (currentVelocity * currentDistance);
    newPosition.y = newTargetPosition.y;

    Vector3 targetDirection = newPosition - newTargetPosition;
    if(Physics.Raycast(newTargetPosition, targetDirection, out hit, currentDistance, raycastLayers))
        newPosition = hit.point;

        transform.position = newPosition;
        transform.LookAt (newTargetPosition);

}
```

Obr. 29 Hlavní funkce chování kamery

Pozice kamery se počítá pomocí hodnoty *speedFactor*, která je stanovena na základě rychlosti vozidla s rozsahem mezi hodnotami 0 až 1. Dále pomocí funkce *fieldOfView()*, která nastavuje úhel zobrazení kamery (úhel ve stupních kolem lokální osy y). Pomocí matematické funkce *Lerp()* se interpoluje mezi zadanými hodnotami na základě rychlosti a měnící se úhel zobrazení vytváří pocit, že kamera se při zrychlení oddálí a při zpomalení naopak přiblíží k autu.

Dále se nastavuje pozice kamery, ve výsledku spíše rotace, na základě vektorů *newPosition* (je datového typu Vector3, tedy trojrozměrného vektoru), *newTargetPosition* a především funkce *LookAt()*. Vektor *newPosition* nastavuje pozici kamery, aby byla vždy za autem díky vektoru rychlosti *currentVelocity.* Ten je normalizován na hodnotu 1, ale zachovává si směr. Funkce *LookAt()* vytváří rotaci kolem pozice auta a na základě rychlosti vytváří požadované zpoždění, které vylepšuje celkovou hratelnost a přehlednost.

4.8 Překážky (zbraně)

Pro zvýšení zábavního pocitu ze hry, byl vytvořen systém překážek, který byl obecně popsán v kapitole 4.2 Motiv a logika hry. O funkčnost se stará skript *Traps.cs*, který je připojený k vozidlu hráče.

Aby mohl hráč využít překážku a umístit ji na dráhu, musí nejprve projet jedním ze speciálních objektů, které jsou rozmístěné na třech různých místech okruhu. Tím že objektem projede, získá přístup k dané zbrani. Tento systém funguje podobně jako check-point systém. Objekty na dráze mají různé tagy podle zbraně, kterou zpřístupňují. Obsahují i collider komponentu. U vozidla se pomocí funkce *OnTriggerEnter()* kontroluje, zda došlo k interakci s collider komponentou v závislosti na tagu. Následně se jednoduchou kombinací boolovských hodnot a podmínek zpřístupní daná zbraň, kterou hráč může umístit za sebe zmáčknutím levého ctrl tlačítka.

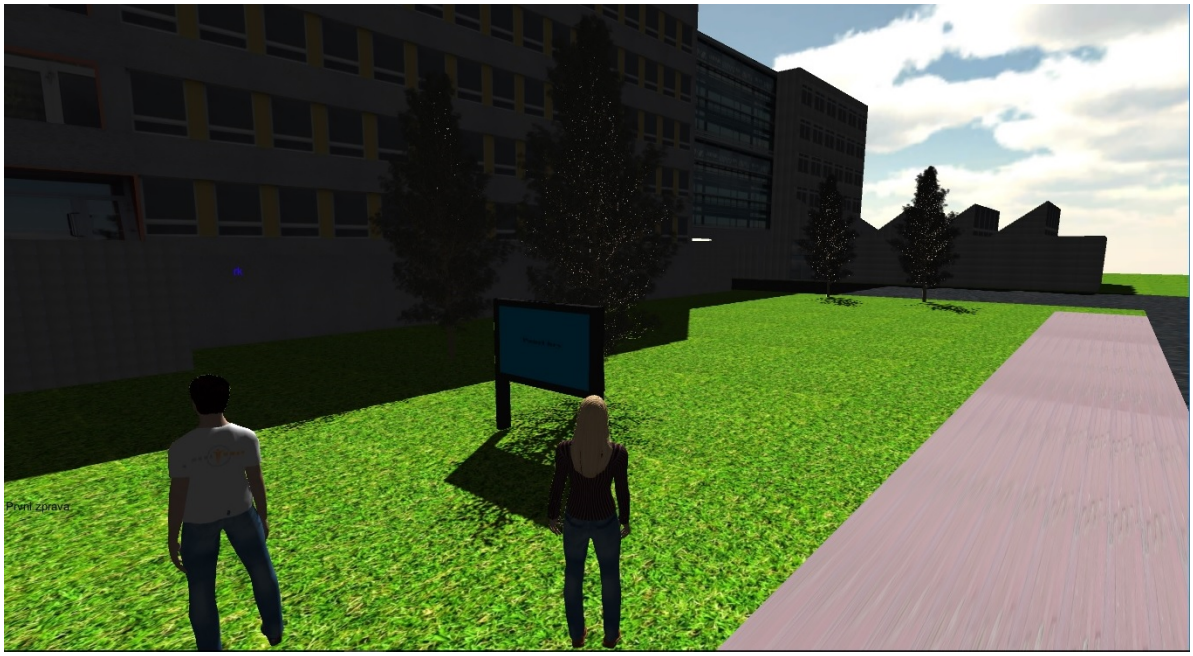
Po umístění překážky se volá funkce *AddTrap()*, z třídy pro komunikaci *MyRaceGamePeer*. Parametry funkce obsahují pozici, natočení a typ překážky a posílá tyto informace na server. Server obratem rozesílá tyto data všem hráčům ve hře, kteří je zachytávají pomocí eventu *RaceGameEvent_AddTrap,* a umísťují se na dráhu.

```
void RaceGameEvent_OnAddTrap (PlayerVector3 position, PlayerVector3 rotation, TrapType type){

    switch (type) {

    case TrapType.Acceleration:

        speedTrap.transform.position = new Vector3 (position.X, position.Y, position.Z);
        speedTrap.transform.localEulerAngles = new Vector3 (rotation.X, rotation.Y, rotation.Z);

        break;

    case TrapType.Fluid:

        fluidTrap.transform.position  = new Vector3(position.X, position.Y, position.Z);
        fluidTrap.transform.localEulerAngles = new Vector3 (rotation.X, rotation.Y, rotation.Z);

        break;

    case TrapType.Jump:

        wallTrap.transform.position  = new Vector3(position.X, position.Y, position.Z);
        wallTrap.transform.localEulerAngles = new Vector3 (rotation.X, rotation.Y, rotation.Z);

        break;

    case TrapType.Wall:

        jumpTrap.transform.position  = new Vector3(position.X, position.Y, position.Z);
        jumpTrap.transform.localEulerAngles = new Vector3 (rotation.X, rotation.Y, rotation.Z);

        break;
    }
}
```

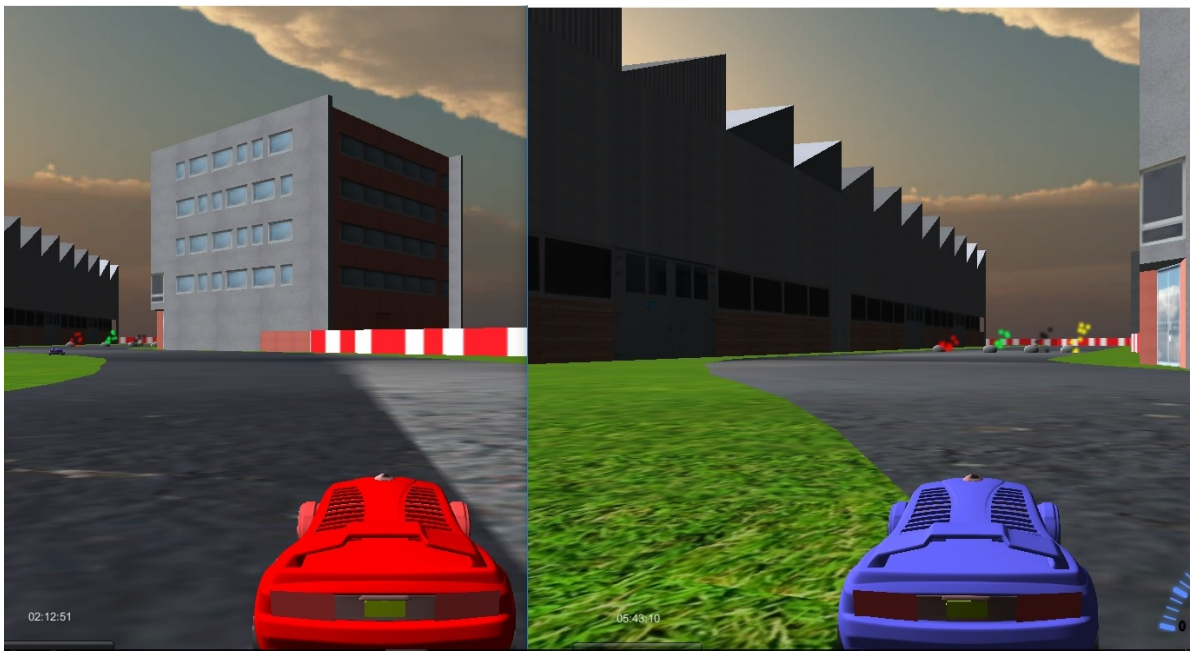
Obr. 30 Event zachytávající pozici, rotaci a typ překážky

4.9 Ukázky ze hry

Obr.31 Startovní sekvence



Obr.32 Panel pro přihlášení do hry ze světa



Obr.33 Aplikace spuštěná dvakrát na jednom počítači

Obr.34 Dojetí do cíle a výpis výsledků

Obr.35 Auto při skoku z rampy

Obr.36 Překážka fluid

Obr.37 Překážka wall

5 Závěr

Cílem této diplomové práce bylo zmapovat vývoj počítačových her a tyto znalosti následně využít při tvorbě vlastní aplikace, která propaguje fakultu VUT FSI.

První část je krátce věnovaná historii vývoje videoher a významným milníkům. Dále jsou popsány platformy, pro které se v současné době hry vyvíjejí, a následně je popisován samotný vývojový cyklus a struktura obvyklých vývojových týmů. V této kapitole jsou ještě rozebrány typy vývojových nástrojů a srovnání herních enginů.

Další kapitola se už věnuje přímo hernímu enginu Unity3D. Byl zvolen hlavním vývojovým nástrojem, a proto je detailně rozebrán od procentuálního zastoupení na trhu, přes vzhled IDE, až po veškeré použité funkce a komponenty. Unity je obsáhlý a komplexní nástroj se spoustou užitečných komponentů, které značně ušetřily čas při vývoji aplikace. Rozsáhlá základna uživatelů a dokumentační podpora dělá z Unity velmi výkonného a schopného pomocníka.

Poslední kapitola popisuje závodní hru, která je součástí praktické části této práce. Vývoj byl zdlouhavý a často bylo nutné řešit spoustu nečekaných situací, které ovlivnily následný postup. Aplikace byla několikrát testovaná přes síťové připojení a celkově je stabilní a hratelná. K vylepšení bych navrhl především grafickou část, propracovanější a hezčí modely a také přidat grafické a zvukové efekty.

Seznámil jsem se s tvorbou počítačových her a také s vývojovým programem Unity3D. Na základě těchto znalostí jsem následně navrhl a vytvořil závodní hru a získal základní zkušenosti s vývojem herních aplikací. Cíle diplomové práce jsou dle mého názoru splněny.

6 Seznam použité literatury

[1] Noughts And Crosses: The oldest graphical computer game. Pong Story [online]. David Winter, 2013 [cit. 2016-04-28]. Dostupné z: http://www.pong-story.com/1952.htm

[2] Video game development. Wikipedia [online]. Wikimedia Foundation, Inc., 2016 [cit. 2016-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Video_game_development

[3] History of video games. Wikipedia [online]. Wikimedia Foundation, Inc. [cit. 2016-04-28]. Dostupné z: https://en.wikipedia.org/wiki/History_of_video_games

[4] Why PC? *PC MASTER RACE* [online]. reddit a.s, 2016 [cit. 2016-04-28]. Dostupné z: https://www.reddit.com/r/pcmasterrace/wiki/guide

[5] VIDEO GAME DEVELOPMENT CYCLE. WILLIAM CLAY [online]. 2011 [cit. 2016-04-28]. Dostupné z: https://ladieslovekewlbowser.wordpress.com/2011/11/28/video-game-development-cycle/

[6] Mobile/Tablet Operating System Market Share. NET MARKET SHARE [online]. Net Applications, 2016 [cit. 2016-04-28]. Dostupné z: https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1

[7] An update on Eclipse Android Developer Tools. Android Developers Blog [online]. Android Developers, 2016 [cit. 2016-04-28]. Dostupné z: http://android-developers.blogspot.cz/2015/06/an-update-on-eclipse-android-developer.html

[8] Spása i prokletí: Android doplácí na roztříštěnost, odnášíte to i vy. Svět Androida [online]. SvetAndroida.cz, 2016 [cit. 2016-04-28]. Dostupné z: http://www.svetandroida.cz/android-roztristenost-ios-201602

[9] HTML5. MOZILLA DEVELOPER NETWORK [online]. Mozilla Developer Network, 2016 [cit. 2016-04-28]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5

[10] How to Make a HTML5 Game. GameDevAcademy [online]. Zenva Academy, 2013 [cit. 2016-04-28]. Dostupné z: https://gamedevacademy.org/how-to-make-a-html5-game/

[11] How Much Does It Cost To Make A Big Video Game? *KOTAKY* [online]. 2014 [cit. 2016-04-28]. Dostupné z: http://kotaku.com/how-much-does-it-cost-to-make-a-big-video-game-1501413649

[12] Video game development. Wikipedia [online]. Wikimedia Foundation, Inc., 2016 [cit. 2016-04-28]. Dostupné z: https://en.wikipedia.org/wiki/Video_game_development

[13] Nejdražší hra v historii Destiny. Reflex [online]. CZECH NEWS CENTER a.s, 2014 [cit. 2016-04-28]. Dostupné z: http://www.reflex.cz/clanek/zajimavosti/56371/nejdrazsi-hra-v-historii-destiny-bude-stat-500-milionu-dolaru-proboha-proc.html

[14] Nejdražší hry všech dob. BONUS WEB [online]. MAFRA, a. s., 2013 [cit. 2016-04-28]. Dostupné z: http://bonusweb.idnes.cz/nejdrazsi-hry-0d8-/Magazin.aspx?c=A130912_150615_bw-magazin_das

[15] How To Form A Solid Indie Game Development Team. NEW YORK FILM ACADEMY [online]. New York Film Academy, 2014 [cit. 2016-04-28]. Dostupné z: https://www.nyfa.edu/student-resources/forming-solid-indie-game-development-team/

[16] THE GAME PRODUCTION PIPELINE: CONCEPT TO COMPLETION. IGN [online]. ZIFF DAVIS, 2006 [cit. 2016-04-28]. Dostupné z: http://www.ign.com/articles/2006/03/16/the-game-production-pipeline-concept-to-completion

[17] What is a Game Engine? Gamecareerguide [online]. UBM, 2008 [cit. 2016-04-28]. Dostupné z: http://www.gamecareerguide.com/features/529/what_is_a_game_.php?page=2

[18] Game Engines: How do they work? GIANT BOMB [online]. CBS Interactive Inc, 2013 [cit. 2016-04-28]. Dostupné z: http://www.giantbomb.com/profile/michaelenger/blog/game-engines-how-do-they-work/101529/

[19] BITVA HERNÍCH ENGINŮ – UNREAL, CRYENGINE, UNITY NEBO SOURCE? VFXcz [online]. PRESSGRID DESIGN, 2015 [cit. 2016-04-28]. Dostupné z: http://vizualniefekty.cz/bitva-hernich-enginu-unreal-cryengine-unity-nebo-source/

[20] Platforms. CRYENGINE [online]. Crytek, 2016 [cit. 2016-04-28]. Dostupné z: https://www.cryengine.com/features/platforms

[21] Unreal Engine Features. UNREAL ENGINE [online]. EPIC GAMES, 2016 [cit. 2016-04-28]. Dostupné z: https://www.unrealengine.com/unreal-engine-4

[22] What is Unreal Engine4. UNREAL ENGINE [online]. EPIC GAMES, 2016 [cit. 2016-04-28]. Dostupné z: https://www.unrealengine.com/what-is-unreal-engine-4

[23] Unity Fast Fact. Unity [online]. Unity Technologies, 2016 [cit. 2016-04-28]. Dostupné z: https://unity3d.com/public-relations

[24] Unity Multiplatform. Unity [online]. Unity Technologies, 2016 [cit. 2016-04-28]. Dostupné z: https://unity3d.com/unity/multiplatform

[25] LICENSING & ACTIVATION. Unity [online]. Unity Technologies, 2016 [cit. 2016-04-28]. Dostupné z: https://unity3d.com/unity/faq

[26] DOCUMENTATION, UNITY SCRIPTING LANGUAGES AND YOU. Unity [online]. Unity Technologies, 2014 [cit. 2016-04-28]. Dostupné z: http://blogs.unity3d.com/2014/09/03/documentation-unity-scripting-languages-and-you/

[27] BLACKMAN, Sue. Beginning 3D Game Development with Unity 4: All-in-One, Multi-Platform Game Development. 1. New York: Springer, 2013. ISBN 978-1-4302-4899-6.

[28] JIRKOVSKÝ, Jan. Game Industry: Vývoj počítačových her a kapitoly z herního průmyslu. 1. Praha: D.A.M.O, 2011. ISBN 978-80-904387-1-2.

Obsah příloh (DVD)

- Elektronická verze diplomové práce
- Instalační soubor Unity3D 4.5.1f3
- Celý projekt Virtual World pro Unity3D
- Vyexportovaný projekt Virtual World
- Návod na zprovoznění aplikace