VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
ÚSTAV AUTOMATIZACE A INFORMATIKY
FACULTY OF MECHANICAL ENGINEERING
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

VIRTUAL WORLD

VIRTUAL WORLD

DIPLOMOVÁ PRÁCE
MASTER ‘S THESIS

AUTOR PRÁCE — Bc. JAKUB HAMAL
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN ROUPEC, Ph.D.
SUPERVISOR

BRNO 2015

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav automatizace a informatiky
Akademický rok: 2014/15

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Jakub Hamal

který/která studuje v **magisterském studijním programu**

obor: **Aplikovaná informatika a řízení (3902T001)**

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

v anglickém jazyce:

Virtual World

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 a realizujte herní svět, včetně jeho generování a zobrazování.

Seznam odborné literatury:

1. Blackman, S.: Beginning 3D Game Development with Unity 4. Springer, New York, 2013.
2. Jirkovský, J. a kol.: Game Industry : Vývoj počítačových her a kapitoly z herního průmyslu. D.A.M.O., 2011.

Vedoucí diplomové práce: Ing. Jan Roupec, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2014/15.

V Brně dne 24. 4. 2015

L.S.

Ing. Jan Roupec, Ph.D.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan

Abstrakt

Obsahem této práce je přiblížení problematiky procesu tvorby počítačových her a představení herního enginu Unity3D. Získané poznatky jsou poté uplatněny při vytvoření ukázkové aplikace pro PC. První část práce je věnována procesu tvorby počítačových her, ve druhé části je popsán engine Unity3D spolu s ostatním softwarem, použitým při vytváření aplikace. Poslední část práce se zabývá popisem vytvoření aplikace s využitím technik z předešlých kapitol.

Abstract

The content of this thesis is to describe process of developing computer game and presentation of Unity3D game engine. Acquisition of knowledge is used to create simple application for PC. The first chapter of this thesis deals with describing proces sof komputer games developing, Unity3D game egnine and its Integrated Development Enviroment, known as Unity Editor is described in the sekond charter with other programs, uset to creating sample application. Last chapter deals with describing creation of application via using techniques and software, which are described in first two chapters.

Klíčová slova

Unity3D, script, hráč, UI, virtuální svět, prostředí, postava

Key words

Unity3D, script, player, UI, virtual world, enviroment, character

Prohlášení o originalitě

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

.....

Jakub Hamal

27. května 2015

Bibliografická citace

HAMAL, J. *Virtuální svět.* Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2015. 71 s. Vedoucí diplomové práce Ing. Jan Roupec, Ph.D.

Poděkování

Chtěl bych poděkovat panu Ing. Janu Roupcovi, Ph.D. za poskytnuté rady a pomoc při psaní této práce.

Obsah

1 Úvod 11

1.1 Obsah práce 11

2 Problematika tvorby počítačových her 13

2.1 Klasifikace žánru hry 13

2.1.1 Textové hry 13
2.1.2 Adventura 14
2.1.3 Akční hry 14
2.1.4 RPG 16
2.1.5 Strategie 16
2.1.6 Online hry 18
2.1.7 Sportovní hry 19
2.1.8 Arkády 20

2.2 Volba platformy a technologie 21

2.2.1 Volba cílové platformy 21
2.2.2 Volba technologie 21

2.3 Game design dokument 22
2.4 Prototyp a testování 23
2.5 Před-finální verze 24

2.5.1 Alfa verze 24
2.5.2 Beta verze 24
2.5.3 Release Candidate 24

2.6 Finální verze 24

3 Herní engine Unity3D a další použitý software 25

3.1 Unity3D 25

3.1.1 Vývojové prostředí 26
3.1.2 Objekty scény 27
3.1.3 Práce se vstupními zařízeními 27
3.1.4 Osvětlení scény 28
3.1.5 Scriptování 28
3.1.6 Animace 29
3.1.7 Fyzika 30
3.1.8 Uživatelské rozhraní 31

3.2 MakeHuman 32
3.3 Blender 34

3.3.1 Nanesení textur 34

4 Vytvoření virtuálního světa 37

4.1 Charakteristika výsledného projektu 37
4.2 Game design dokument 37
4.3 Rozčlenění projektu 42

4.4 Přihlašovací klient 43
4.5 Herní svět 44
4.5.1 Osvětlení 46
4.5.2 Kamera 47
4.5.3 Postava hráče 48
4.5.4 Online hráči 51
4.5.5 Chat 54
4.5.6 Informační a herní panel 55
4.5.7 Funkčnost herního světa 57
4.6 Ukázky ze hry 60
5 Závěr 67

1 Úvod

V současnosti každý kdo pracuje s počítačem, ať už pravidelně každý den nebo jen příležitostně, se setkal s počítačovými hrami. Tyto hry mohou sloužit jako krátkodobé odreagování od vyčerpávající činnosti, vyplnění časových mezer v programu, ale také se dají praktikovat na profesionální úrovni. Rozdělit kategorii počítačových her dle jednoho kritéria je prakticky nemožné, protože většina novodobých her spadá do více skupin, ale je možné hru charakterizovat podle jejích základních vlastností. Mezi tyto vlastnosti patří např. žánr hry – „strategie“, „střílečka“, „agentura“, „tahová hra“, „závodní hra“, hra na hrdiny (dále jen RPG – Role Playing Game), „simulátor“ atd. Z hlediska hráče je jednou z rozhodujících vlastností hry to, jestli hra nabízí i mód hry pro více hráčů, nebo se jedná pouze o hru pro jednoho hráče, kterých je ale vydáváno stále méně. Hlavním důvodem tohoto vývoje je opadající zájem hráčů o tento typ her. Problém není v tom, že by tyto hry byly snad méně zábavné, ale po jejich úspěšném dokončení se zpřístupní pouze možnost opakování hry na vyšší obtížnosti se stejným obsahem a příběhem a počet lidí, kteří by za takovou hru zaplatili, klesá. Výjimku tvoří hry spadající do kategorie RPG. Tyto hry nahrazují absenci hry pro více hráčů rozsáhlostí obsahu, jeho zpracováním a tím také dobou potřebnou k dohrání hry. U těchto her je také možné je dokončit různými způsoby a někdy i s různým koncem, což se odvíjí od řešení jednotlivých situací ve hře.

Mnohem větší zastoupení mají tedy hry, nabízející možnost hry více hráčů. Tyto hry mohou být zpracovány různými způsoby. Může se jednat o hry s částí pro jednoho i více hráčů a ty mohou být na sobě buď nezávislé, nebo spolu mohou být propojeny např. tak, že část pro jednoho hráče lze absolvovat znovu v kooperativním módu ve hře pro více hráčů. Poslední možností jsou hry vytvořené pouze pro hru více hráčů.

1.1 Obsah práce

V této práci je přiblížena problematika tvorby virtuálního světa počítačové hry a získané poznatky jsou následně použity pro vytvoření ukázkové aplikace, která je použita jako základ pro skupinový projekt, jehož cílem bylo vytvořit funkční počítačovou hru ze žánru online hry pro více hráčů s použitím herního enginu Unity3D. V první kapitole je charakterizován možný postup při vývoji počítačové hry, který je následně aplikován při vytváření ukázkové aplikace. Druhá kapitola je věnována enginu Unity3D a ostatním programům, které byly použity při tvorbě aplikace. Vytvořená aplikace byla použita jako herní svět projektu Virtual World. Důvodem k založení tohoto projektu je záměr Ústavu Automatizace a Informatiky FSI VUT v Brně vytvořit počítačovou hru, prezentující studium na ústavu.

1 Úvod

2 Problematika tvorby počítačových her

Počítačová hra je druh interaktivního softwaru, který je určený výhradně pro zábavu uživatele. Komunikace s uživatelem je realizována textově, pomocí 2D a 3D grafiky a pomocí zvuků. Provozovat počítačovou hru je možné na jakékoliv platformě (soubor softwaru a hardwaru), pro kterou byla hra vyvinuta. Speciální platformou je herní konzole, která je určená výhradně pro hraní her a multimediální zábavu. Hry pro tuto platformu jsou často označovány jako videohry. Vytvoření počítačové hry je složitý a časově náročný proces, jehož postup je velmi podobný postupu při tvorbě jakéhokoliv softwaru a liší se pouze v detailech. Z hlediska náročnosti je prakticky nemožné vytvořit kvalitní konkurenceschopnou hru jako samotný programátor, platí zde přímá úměrnost: čím kvalitnější, složitější a propracovanější hra, tím více vývojářů je potřeba k jejímu vývoji [1]. Například na vývoji jedné z nejpropracovanějších her současnosti, GTA V, pracovalo více než jeden tisíc lidí. Vývojářské týmy jsou tvořeny např. grafiky, level designery, zvukaři, modeláři, programátory, testery aj.

Postup při tvorbě hry lze rozdělit na několik hlavních částí, které budou v této kapitole rozepsány.

- Klasifikace žánru hry
- Výběr platformy a technologie
- Vytvoření dokumentace – Game design dokument
- Vytvoření prototypu, jeho testování a rozšiřování
- Vydání před-finálních verzí hry
- Vydání finální verze

2.1 Klasifikace žánru hry

Zvolení žánru hry, kterou chceme vyvíjet, lze považovat jako první krok vývoje, protože od toho se vše odvíjí. Od tohoto kroku se poté odvíjí požadavky na technologii a vytvoření Design dokumentu, ke kterému se dostaneme. Každý žánr má své specifické vlastnosti a nelze tedy začít pracovat např. na adventuře a v polovině vývoje ji předělávat na strategickou hru. V této podkapitole tedy budou představeny různé žánry spolu s uvedením typického příkladu.

2.1.1 Textové hry

Přestože čistě textové hry prakticky vymizely, tak nelze tento žánr opomenout, jelikož právě z těchto her vznikl žánr adventura. Jak napovídá název, tak veškerá interpretace hry a interakce s uživatelem je realizována pomocí textu. Postupem času byly textové hry doplňovány o ilustrace lokací nebo nahrazením slovních příkazů pro hraní výběrovou nabídkou. Takto se žánr textových her postupně vyvinul v žánr adventura. Kouzlo textových her spočívá v tom, že velkou roli zde hraje fantazie uživatele, protože každý hráč si textem popsanou lokaci, postavu atd. vybaví jinak, což je věc, které u klasických grafických her nelze dosáhnout. V posledních letech je tento žánr oživován a na jeho základech jsou vytvářeny hry pro nevidomé, kde text je nahrazen mluveným slovem. Příkladem textové hry z české produkce je hra Belegost, inspirovaná světem Pána Prstenů od J. R. R. Tolkiena.

2.1.2 Adventura

Vývojem technologií počítačů a jejich využití pro hraní her vedlo k nahrazení textových her adventurami. Název vynikl jako odvození od anglického výrazu adventure (dobrodružství). Hlavní znak her tohoto typu je rozmanitý příběh, který se hráči postupně odkrývá řešením různých hádanek a používáním předmětů, které se ve hře vyskytují. Velký důraz je kladen také na dialogy, jež nezřídka kdy slouží jako nápověda k rébusům. Struktura adventury je tedy postavená na řešení hlavního příběhového úkolu, za doprovodu několika úkolů vedlejších. Splnění některých z nich může být povinné pro posunutí se v příběhu, naopak další může hráč ignorovat. Původní adventury byly založeny na ovládání 'Point and click'. Jedná se o vcelku jednoduché ovládání, ve kterém hlavní úlohu plní myš, pomocí které uživatel najede na objekt a kliknutím zahájí interakci. Tento žánr se hodí pro zpracování detektivních a hororových témat, kde hraje hlavní roli složitá zápletka.

Z českých adventur je nutné zmínit hru Tajemství Oslího ostrova, která byla v roce 1994 první celoplošně distribuovaná hra v České republice a vznikla jako parodie na sérii Monkey island. Známější a také úspěšnější české adventury byly ovšem dva díly hry Horké léto, na jejichž dabingu se podílel Zdeněk Izer a série Polda I-VI kde hlavního hrdinu nadaboval herec Luděk Sobota.

Obr. 1 Horké léto 2

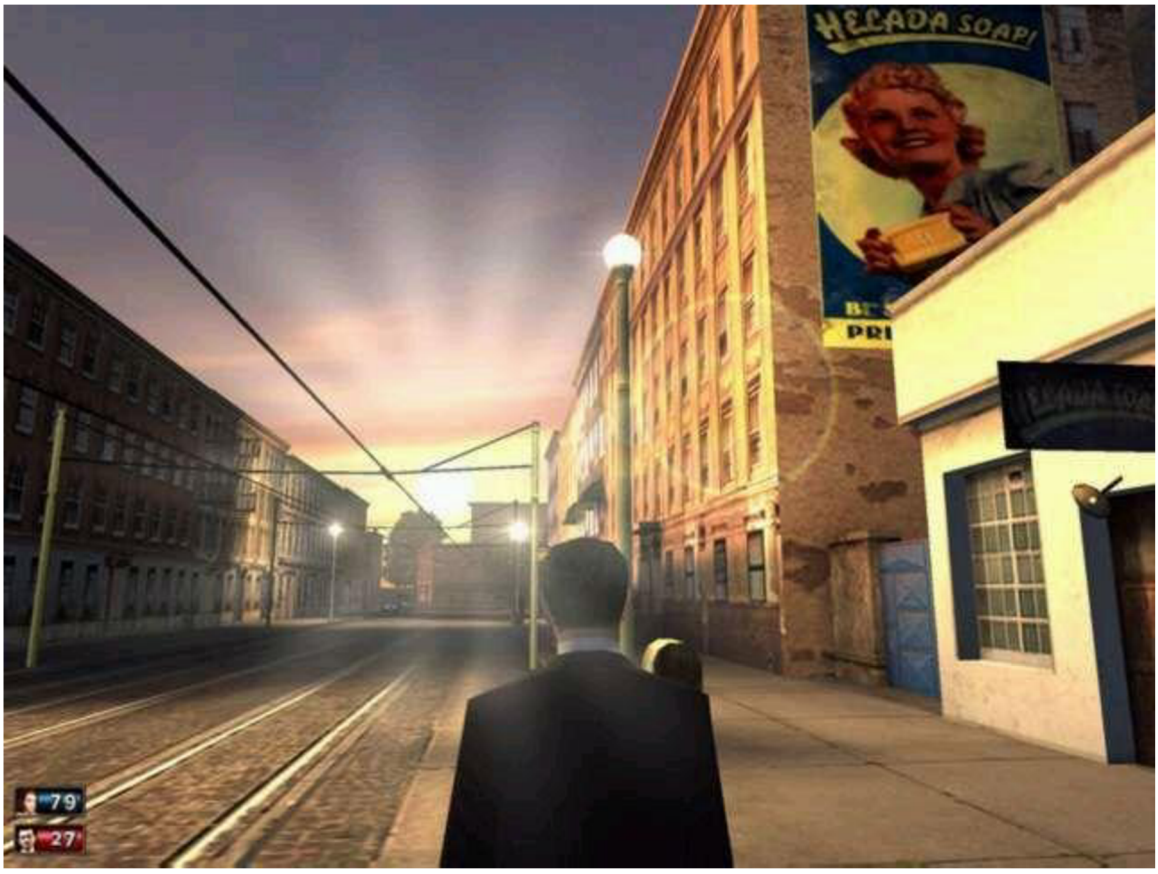
2.1.3 Akční hry

Jak napovídá název, tento žánr her sází hlavně na akci a dynamiku. Hra bývá rozdělena na několik úrovní a hráč postupným dokončením jednotlivých úrovní prochází příběhem. Ten ovšem v tomto žánru slouží až na několik vyjímek jako doprovodný prvek. Podmínkou pro úspěšné dokončení úrovně zpravidla bývá splnění několika podbodů, které mají hráče v úrovni směrovat a pomáhat mu k dokončení. Tyto vedlejší úkoly mohou být

různého druhu počínaje prostou eliminací všech nepřátel v lokaci přes vyzvednutí důležitého předmětu a donesení jej na určité místo až po zajištění přežití některé z postav. V některých hrách se tyto úkoly mohou postupným plněním měnit v závislosti na průběhu hry, avšak většinou je tato změna pevně naprogramovaná a nastane, ať se hráč zachová jakkoliv.

Akční hry lze rozdělit podle jejich grafického pojetí a to na FPS a TPS hry. FPS (First person shooter) jsou hry zobrazené z pohledu první osoby. Zjednodušeně hráč vidí svět očima svého avataru. Dalším znakem těchto her bývá týmová hra. V sólové hře je hráč doprovázen několika postavami umělé inteligence a je zde i podpora hry pro více hráčů přes internet. Úspěšným českým zástupcem FPS her byla Operace Flashpoint nebo herní série Armed Assault (ArmA) od studia Bohemia Interactive Studio.

Druhým typem akčních her jsou TPS (Third person shooter) hry, ve kterých je naopak kamera umístěna za avatarem hráče. Výhodou těchto her je větší přehled po okolí, ale pokud je pohyb kamery špatně zpracován tak to může způsobit problémy s ovládáním postavy. U některých her tohoto typu je možné kamerou posunout až do pohledu první osoby, proto může být těžké jednoznačně hru kategorizovat. Velmi úspěšným titulem z české produkce byla Mafia: The City of Lost Heaven z roku 2002 od studia Illusion Softworks (dnes 2KGames), jejíž velké přednosti byly poutavý příběh z mafiánského prostředí a vysoká úroveň grafická kvalita prostředí i animací, díky čemuž místy hra dosahovala filmových kvalit. Díky pozitivnímu ohlasu se hra dočkala v roce 2010 pokračovaní.



Obr. 2 Mafia

2.1.4 RPG

Dalším žánrem počítačových her jsou hry na hrdiny. V těchto hrách se hráč ujme ovládání postavy, kterou si mohou vytvořit pomocí různých vlastností, zvolí si povolání a za tuto postavu pak ve hře jednají. Rozdíl oproti akční hře je ve velmi rozlehlém a otevřeném herním světě a způsobu jakým hráč hrou prochází. Může si např. vytvářet vlastní zbraně nebo vykonávat jiné běžné činnosti. Kostrou hry je příběh, ale hráč není nucen se v něm posouvat dále. Naopak pokud hráč chce, může se jen ve světě pohybovat a objevovat všechna skrytá místa, plnit vedlejší úkoly atd. Velmi podstatným prvkem tohoto žánru je vývoj postavy. Za splnění každého úkolu hráč obdrží kromě odměny také zkušenosti a po získání určitého počtu postoupí hráčova postava na vyšší úroveň. Tím obdrží několik dovednostních bodů, jejichž rozdělením mezi jednotlivé vlastnosti se stává postava silnější a mohou se takto zpřístupnit některé její speciální schopnosti.

Z důvodu rozlehlosti a propracovanosti RPG her je jejich vývoj velice náročný a to jak časově tak i finančně. Možná i to je jeden z důvodů, proč českých her tohoto žánru příliš mnoho nevzniká. Za zmínku ovšem stojí Brány Skeldalu. Hra vyšla v roce 1994 a jednalo se o velmi úspěšný titul, který si i dnes najde své fanoušky. Nebyla to klasická RPG hra, nýbrž krokovací dungeon, kde se hráč po světě pohybuje pomocí kroků do určitého směru. Hra byla původně vydána pro operační systém MS-DOS, ale později byla předělána pro Windows 95 a v roce 2015 byla vydána verze, kompatibilní s platformou Windows 7.

Obr. 3 Brány Skeldalu

2.1.5 Strategie

Strategické hry se řadí mezi jeden z prvotních žánrů počítačových her. Jejich předlohou často bývá desková nebo karetní hra. Jsou založeny na budování vlastní základny, vytváření jednotek a jejich použití pro splnění cílů. Hra je rozdělena na úrovně, přičemž příběh je před první úrovní uveden většinou animační sekvencí a dále pak rozšiřován během a mezi úrovněmi. Pro dosažení vítězství v úrovních je často nutné postup

plánovat – z toho plyne název strategické hry. Velmi často obsahuje strategická hra několik souborů úrovní tzv. kampaní. Tyto kampaně poskytují hráči možnost hrát hru z pohledu všech stran, které jsou ve hře vytvořeny. Lze je rozdělit na RTS strategie, tahové a na budovatelské strategie.

V RTS (real time strategy) čas neustále plyne a všechny úkony hráče a následná počítačem řízená odezva je realizována okamžitě. Opakem jsou potom tahové strategie, ve kterých je postup úrovní rozdělen na jednotlivé tahy. Tah bývá omezen rozsahem pohybu a počtem budov, které je možné postavit na základně. Možností je i hra, která vzniká kombinací těchto dvou přístupů. Ta může být realizována např. tak, že globální část hry se odehrává v reálném čase, ale jakmile vstoupí do akce nepřítel, hra přechází do tahového systému. Českým příkladem této kombinace je trilogie UFO her od studia ALTAR Interactive, ve které se hráč musí vypořádat s mimozemskou invazí na Zemi.

Obr. 4 První díl série UFO

Specifickým druhem strategických her jsou budovatelské strategie. V nich je vynechán faktor síly a boje a naopak je kladen na promyšlené plánování a budování. Sem patří různé tycoon hry, ve kterých se na začátku hry ujmete malé firmy a v závislosti na druhu hry ji postupně rozvíjíte. Další možností může být stavba a řízení vývoje vlastního města, zábavního parku nebo zoologické zahrady. Základem je vždy řízení akcí na globální úrovni tzn., že hráč neovlivňuje přímo postavy ve hře, ale naopak reaguje na jejich pocity a požadavky a podle toho volí, které budovy dále stavět. Odnoží budovatelských strategií jsou Tower Defense (TD) hry. V nich se hráč snaží zabránit umělé inteligenci nebo jinému hráči ve zničení své základny budováním a vylepšováním obranných věží.

2.1.6 Online hry

Specifickým typem her jsou ty, které vycházejí pouze s módem hry více hráčů. Mezi těmito hrami je možné nalézt hry, spadající do ostatních žánrů, ale absencí možnosti sólové hry je lze zařadit mezi online hry. Velmi rozšířeným typem her tohoto žánru jsou MMO-RPG (massively multiplayer online role-playing game). Jak název napovídá, jedná se o hru typu RPG, která je upravena pro více hráčů. Princip spočívá v pohybu všech hráčů v jednom herním světě. Zřejmě nejznámější MMO-RPG na světě je hra World of Warcraft od studia Blizzard. Důvodem vzniku těchto her je interakce a spolupráce mezi hráči. Ti se sdružují do skupin (v různých hrách nazýváno jinak – guildy, cechy, klany) a každý hráč zde v závislosti na jeho povolání a schopnostech plní svoji úlohu. Samozřejmě je možné hrát hru samostatně, ale velmi se tím zpomaluje postup hrou a některé lokace přímo vyžadují spolupráci několika hráčů. Českých her tohoto typu je na trhu velmi málo, za zmínku stojí např. Chmatákov online, což byla vůbec první česká MMO-RPG.

Stejně tak zde můžeme najít hry MMO-RTS (massively multiplayer online real-time strategy) a MMO-FPS (massively multiplayer online first-person shooter). Oba tyto podžánry jsou stejně jako MMO-RPG velmi podobné žánrům, ze kterých vychází. Ve hrách MMO-RTS proti sobě stojí dva týmy, většinou tvořeny pěti hráči, a snaží se si navzájem zničit základnu. Hry MMO-FPS jsou založeny na podobném principu, ale obsahují několik různých modů např. deathmatch - který tým eliminuje více hráčů soupeře, tak vítězí, team-deathmatch - vítězí tým, kterému zůstane naživu alespoň jeden hráč, capture the flag – cílem je získat vlajku soupeře a dopravit ji na vlastní základnu, aj.

Speciálním typem online her jsou sandboxové hry. Ty se vykazují kompletně otevřeným světem a absencí jakéhokoliv cíle. Jedinou náplní těchto her je přetváření herního světa podle hráčovy představy – odtud název sandbox (pískoviště). Nejúspěšnější hrou tohoto typu je Mincraft, kde hráči pomocí kostiček vytvářejí stavby. Za vesmírný klon Minecraftu lze považovat český titul Space Engineers. V roce 2013 byla vydána první alpha verze a hra je stále ve vývoji.

Obr. 5 Space engineers

2.1.7 Sportovní hry

Jako námět sportovních her slouží jakýkoliv sport, bez ohledu na to zdali se jedná o míčový sport nebo např. automobilové závody. Tyto hry je možné rozdělit podle několika kriterií. Podle způsobu hraní se dělí na klasické a manažerské, a podle zpracování na simulace nebo arkády.

Klasické sportovní hry spočívají v přímém ovládání sportovce nebo závodníka. Na začátku hry se jedná o nezkušenou a ne moc zručnou postavu a podle toho je také využívána vedením týmu, ale postupem času získává zkušenosti a stává se klíčovou součástí týmu, na jejímž výkonu často závisí úspěch či neúspěch. Manažerské hry postaví hráče do přesně opačné pozice. Hráč se ujímá role ředitele klubu, řídí jeho vývoj, příchody a odchody jeho členů a snaží se zvolením vhodného postupu dosáhnout nejlepších výsledků. Může ale nastat i situace, kdy vedení klubu nebude s Vaší prací spokojeno a dostanete výpověď. V tom případě můžete čekat, zda o vaše služby projeví zájem jiný klub nebo začít novou hru. S rozšiřováním internetového připojení se stále větší oblíbenosti těší online manažery. Příkladem českého fotbalového manažeru je volně dostupný Czech Soccer Manager (CSM). První verze pod tímto názvem vyšla již v roce 1999, a hra je od té doby pravidelně aktualizována.

Obr. 6 CSM 2002

Z hlediska hratelnosti je možné sportovní hry rozdělit na simulátory a arkády. Simulátory kladou důraz na realistické zpracování. Např. u automobilových her si lze kompletně měnit nastavení auta a je tak možné dosáhnout lepších výsledků. Negativem těchto her ale je právě náročnost na pochopení všech vyskytujících se parametrů a také složité a náročné ovládání, a pokud si chce hráč užít opravdu realistický závod, doporučuje se opatření některých speciálních herních ovladačů např. volantu s pedály pro závodní simulátory nebo joysticku pro letecké simulátory protože s použitím klávesnice a myši nelze dosáhnout takové přesnosti v ovládání. Náročnost ovládání je možné snížit zapnutím

různých pomocných asistentů. Naopak arkádové zpracování upřednostňuje zábavnou složku před realistickým chováním.

2.1.8 Arkády

Jak bylo naznačeno v závodních hrách, označením hry jako arkádu znamená, že je založena na jednoduchém principu. Patří sem klasické bojové hry, plošinovky nebo třeba logické hry. U bojových her může být cílem probojovat se turnajem jeden na jednoho co nejdále, nejlépe zvítězit, nebo procházet lokaci a eliminovat nepřátele pomocí jednoduchého ovládání a použitím speciálních schopností, jejichž aktivace bývá podmíněna stisknutím správné kombinace tlačítek. U plošinovek hráč překonává překážky herního světa skákáním nebo posouváním objektů a eliminací nepřátel. Často je ovládána skupina postav s různými vlastnostmi a k dosažení konce úrovně je potřebná jejich spolupráce. Jak napovídá název, pointa hry byl pohyb v jedné rovině – 2D. S nástupem nových technologií se i tyto hry posunuly do třetího rozměru

Logické hry vykazují nároky na použití správného postupu (logiky) pro dosažení cíle. Může se jednat o správný pohyb, užití vhodných předmětů ve vhodnou dobu, správné skombinování předmětů za účelem vytvoření nového, spolupráce několika postav atd. Mezi logické hry je možné zařadit i deskové hry jako šachy, dáma aj. nebo klasického hada, piškvorky. Příkladem české logické hry je titul Fish Fillets z roku 1997, ve které je cílem hráče dostat dvě ryby z místnosti. Jedna ryba je malá a může se tak protáhnout menšími otvory, naopak druhá je větší a silnější. Může tak manipulovat s objekty úrovně. Titul se dočkal pokračování v roce 2007 a na obou dílech pracovalo studio ALTAR Games (vyniklo z ALTAR Interactive).

Obr. 7 Fish Fillets

2.2 Volba platformy a technologie

Po klasifikaci žánru hry, kterou chceme vyvíjet, je na řadě zvolení cílové platformy a technologie, která bude pro tvorbu použita. Je důležité postupovat přesně v tomto pořadí, protože ne všechny technologie podporují multiplatformní vývoj. Za hlavní faktor pro volbu platformy lze tedy považovat počet dostupných technologií a jejich případná cena.

2.2.1 Volba cílové platformy

Za nejotevřenější platformu lze považovat počítač (dále PC). PC hry je dnes možné vyvíjet bez nějakých speciálních vybavení, bez ohledu na to, pro jaký operační systém má hra být. Pro tuto platformu existuje mnoho jednoduchých technologií, ve kterých je vytvoření hry hračka. Samozřejmě se nejedná o hry, které by měly udávat trend nebo přinášet nové věci do žánru. Otevřenost této platformy s sebou ale přináší také negativní faktor, a sice složité prosazení se na trhu, který je doslova přehlcený množstvím her od nezávislých studií a je až na výjimky, nemožné vytvořit konkurence schopnou hru zdarma. Jednou z možností, jak tedy získat prostředky pro vývoj hry, jsou internetové stánky, založené na crowdfoundingu. Na tyto stránky jsou umísťovány různé projekty a lidé mají možnost přispět na jejich vývoj s příslibem určitého bonusu za příspěvek. U počítačových her tímto bonusem bývá např. sleva na zakoupení hry, možnost účasti na uzavřeném testování nebo obdržení speciálního předmětu ve hře.

Druhou, poměrně otevřenou platformou, jsou mobilní telefony. Zde je třeba rozdělení dle dvou nejrozšířenějších operačních systémů – open source Android od společnosti Google a iOS od firmy Apple. Zde má menší výhodu iOS v menším počtu aktivních verzí, kdežto verzí Androidu je poměrně mnoho a velmi podivující je fakt, že ačkoliv v červnu 2014 byla vydána verze 4.4, tak téměř 50% všech v té době nainstalovaných verzí tvořila verze 2.2. Tento fakt výrazně ovlivňoval a hlavně stěžoval práci vývojářům. Naopak výhodou Androidu je širší podpora zařízení, protože iOS je možné provozovat pouze na zařízeních firmy Apple.

Obecně za nejsložitější je možné označit nezávislý vývoj her pro herní konzole. Hlavními důvody je absence volně dostupných technologií a velice složitá distribuce hry.

2.2.2 Volba technologie

Jakmile máme určenou cílovou platformu, nastává čas na zvolení technologií pro vývoj hry. Nejedná se pouze o základní technologii hry – herní engine, ale také o další software, potřebný pro vytvoření hry. Sem patří modelovací programy, různé nástroje na úpravu grafiky a zvuku, scriptovací nástroje aj. Z hlediska nezávislého vývojáře je určující prvek při volbě cena, takže přednostně volí programy dostupné kompletně nebo alespoň pro nekomerční účely zdarma.

Zvolením vhodného herního enginu lze velmi usnadnit následný vývoj. Jedná se totiž o jádro celé hry, zabezpečující systémové záležitosti. Důležitá je také kompatibilita s ostatními použitými programy, proto je zvolení správné kombinace programů velmi důležitým krokem při vývoji hry. Pokud porovnáme množství herních engine a množství jiných potřebných programů tak pro nezávislého vývojáře se jeví nejlépe nejdříve zvolit herní engine a k němu poté vybrat další potřebné programy. Platí totiž, že kvalitnější engine výrazně ulehčuje celkový vývoj hry.

V závislosti na kvalitě obstarává herní engine některé (u nejkvalitnějších všechny) funkce:

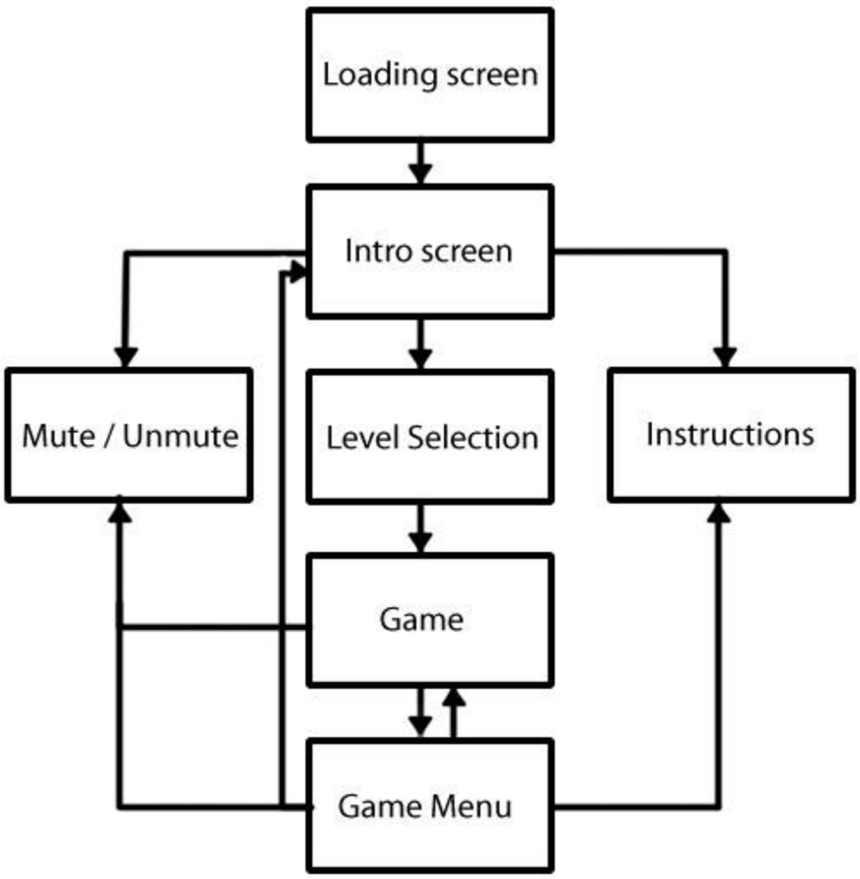
- Práce s grafikou – 2D i 3D
- Zpracování kolizí a fyziky
- Práce se vstupními a výstupními zařízeními
- Zpracování zvuku
- Scriptování
- Správa souborů
- Podpora vývoje pro více platforem

2.3 Game design dokument

Po určení žánru hry a zvolení cílové platformy a technologie nastává první etapa samotného vývoje hry. Ta spočívá ve vytvoření Game design dokumentu (dále jen GDD). V tomto dokumentu je představena kompletní představa hry, která je dále rozčleněna na jednotlivé body a podbody, které jsou dále specifikovány. Jsou zde popsány všechny části a vlastnosti hry, jednotlivé úrovně a objektů, což velice pomáhá v jejich následné realizaci. Nejedná se ovšem o statický popis, obsah se neustále mění, rozšiřuje nebo naopak zmenšuje v závislosti na samotné realizaci. Obsah dokumentu není nijak specifikován – může se v něm vyskytovat text, obrázky, grafy nebo konceptuální návrh – zobrazen na Obr. 8. U velkých projektů se zpravidla jedná o dokument, čítající stovky stran. Základní struktura GDD může vypadat takto [1]:

- Popis hry – úvodní část GDD, obsahuje popis očekávané výsledné podoby hry a představuje základní myšlenky a cíle hry
- Návrh ovládání a způsobu hraní – v této části je popsáno kompletní ovládání a nastíněna jeho aplikace ve hře
- Popis úrovní – charakteristika jednotlivých úrovní, podmínky pro jejich dokončení a návaznost mezi nimi – pokud si hráč přenáší některé získané vlastnosti a objekty z jedné úrovně do druhé
- Uživatelské rozhraní – popis zobrazení hry včetně volby a nastavení kamery, struktury hlavního menu a různých doplňkových menu
- Grafika, fyzika, zvuk – o tyto části se stará herní engine, pokud neobsahuje některou z těchto funkcí tak je nutné určit, jak bude řešena
- Popis objektů – rozdělení hry na jednotlivé objekty, definování všech jejich vlastností
- Návrhy a doplňky – část pro rozšiřující návrhy nebo popřípadě myšlenky ze hry vyřazené
- Popis týmu vývojářů – určení kdo bude pracovat na jaké části hry

I když se může část Návrhy a doplňky zdát v dokumentu zbytečná, tak tomu tak není. Zde jsou sepsány nápady, které v původním návrhu byly, ale z nějakého důvodu musely být odstraněny. Z hlediska budoucího vývoje je ale zbytečné tyto myšlenky vyřazovat z projektu a to hlavně z důvodu, že by se problém podařilo odstranit a myšlenka by tak mohla být vložena zpět do hry. Takto navržená struktura byla použita při tvorbě GDD pro ukázkovou aplikaci.



Obr. 8 Příklad konceptuálního návrhu hry

2.4 Prototyp a testování

Po vytvoření prvotního GDD nastává okamžik samotného vývoje. Prvním krokem je vytváření prototypu podle GDD. To samozřejmě neznamená vzít první bod a vytvořit ho. První prototyp bývá zpravidla oživení základního prvku hry. Například jedná-li se o akční hru, založenou na ovládání postavy pomocí klávesnice, tak prototypem bude přesně toto. A nemusí se jednat o postavu ve výsledné podobě, zpočátku postačí jednoduchá krychle, která bude reagovat tak, jak by měla ve výsledku postava uživatele. Po oživení základního prvku hry je prototyp dále rozšiřován podle GDD a neustále testován. Testování probíhá prakticky nepřetržitě a v podstatě lze říct, že se testuje každý řádek kódu. Nelze totiž realizovat např. celý bod z GDD a následně testovat, to by bylo velice neefektivní, protože čím větší celek testujeme, tím horší bývá následná lokalizace případného problému a navíc odstranění jednoho problému vede většinou ke vzniku nového a celý proces testování a debutování se nepříjemně a hlavně zbytečně natahuje. Z tohoto důvodu se testuje každá funkce samostatně ihned po napsání.

Jakmile prototyp obsahuje většinu z hlavních bodů GDD lze jej označit jako alfa verzi celé hry.

2.5 Před-finální verze

Jakmile jsou zpracovány všechny body z dokumentace, muže být vydána první z několika před-finálních verzí hry. Tato verze je označována jako Alfa verze. Po ní následuje Beta verze a Release Candidate.

2.5.1 Alfa verze

Tato verze je první verzí kompletního prototypu (pre-alfa verze), která obsahuje všechny důležité funkce, ale zároveň spoustu chyb a vykazuje velkou nestabilitu. Zpravidla je poskytována v rámci společnosti, která na hře pracuje a slouží hlavně k uzavřenému testováni vývojáři, kteří vědí, jak software funguje. Během tohoto testování dojde k nalezení a odstranění největšího množství chyb.

2.5.2 Beta verze

Jakmile dojde k odstranění závažných chyb v alfa verzi, hra přejde do stádia beta verze. To znamená, že je opravena většina chyb, ale hra je stále nestabilní, nespolehlivá a může být i destruktivní pro ostatní spuštěný software. V této fázi je na týmu vývojářů, zda se rozhodnou vydat uzavřenou či otevřenou beta verzi hry. Uzavřená beta verze slouží k testování omezeným počtem testerů a přístup k takovéto verzi je možný pouze obdržením např. pozvánky nebo přístupového klíče od vývojářského týmu. Naopak otevřená beta verze je přístupná všem uživatelům, kteří jsou ochotni se testování zúčastnit. Jelikož v této fázi hry se na testování nepodílí pouze a výhradně vývojáři, je velice důležitá zpětná vazba uživatelů.

2.5.3 Release Candidate

Po odstranění chyb z beta verze hra přechází do verze Release Candidate. Jedná se o poslední z předfinálních verzí hry a při vydávání této verze se používá spojení RCX, kde X se nahrazuje pořadovou číslicí aktuální Release Candidate verze.

2.6 Finální verze

Po schválení verze Release Candidate je hra připravena k distribuci - anglicky release to manifacturing – RTM. Tato verze může obsahovat digitální zámek, pomocí kterého lze ověřit, zda uživatel hru zakoupil nebo si opatřil nelegální kopii. Zároveň je oficiální kopie této verze odeslána k hromadné duplikaci pro distribuci. Další možností je distribuce hry online, která funguje tak, že si uživatel zakoupí možnost stáhnout si instalační soubory hry.

Vydáním finální verze hry a její uvedení na trh znamená pro část týmu, který se na jejím vývoji podílel, konec práce na této hře a možnost pracovat na jiném projektu. Zbývající vývojáři dostanou na starost podporu dokončeného produktu a přípravu prvního patche. Patch je někdy považován za aktualizaci, což ale není úplně správné. Aktualizací je myšlena instalace nové verze, která má sloužit jako vylepšení verze předchozí. Naopak patch je spíše soupis změn mezi soubory a slouží výhradně k opravě chyb stávající verze.

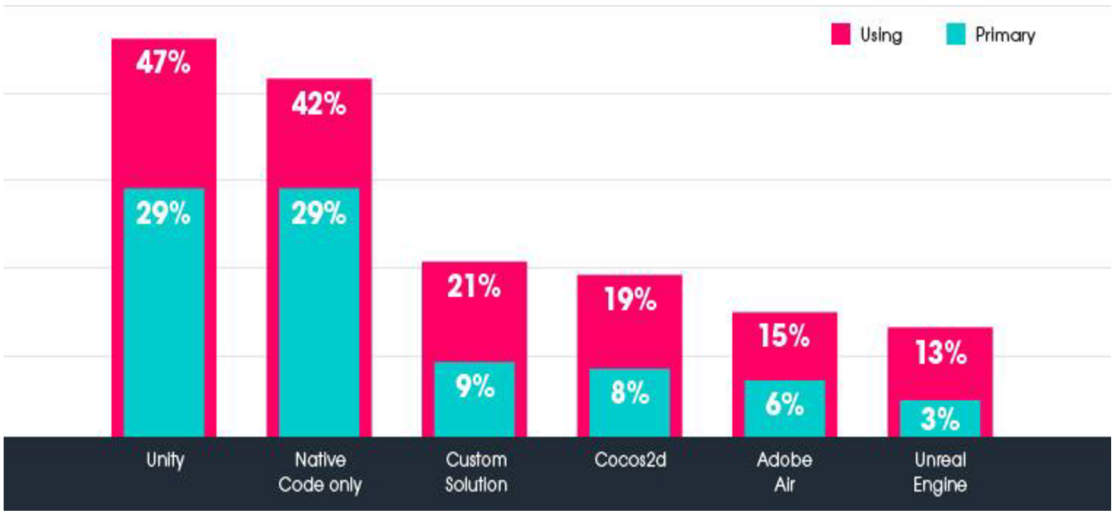
3 Herní engine Unity3D a další použitý software

Jedním z cílů celé práce bylo vytvořit virtuální svět pomocí herního enginu Unity3D. Tato kapitola je tedy věnována základnímu popisu tohoto enginu spolu s jeho funkcemi, které byly použity pro tvorbu virtuálního světa a programů, které byly použity pro vytváření online světa a jeho jednotlivých částí. Hlavní podmínka při výběru programů byla možnost pracovat s plnou verzí bez poplatku a po neomezeně dlouhou dobu. Pro modely postav byl zvolen software Makehuman, pro budovy modelovací program Blender.

3.1 Unity3D

Jedná se o multiplatformní herní engine vyvíjený firmou Unity Technologies. Verze 1.0 byla vydána 8. 7. 2005 původně pouze pro vývoj her s operačním systémem MacOS. Toto omezení bylo odstraněno s verzí 2.5, která jako první podporovala vývoj i pro operační systém Windows. Současná verze 5.0 z března roku 2015 umožňuje vývoj pro dvacet jedna platforem včetně vývoje aplikací pro televizní platformy Android TV a Samsung Smart TV. V té době čítala oficiální komunita vývojářů, pracujících s tímto enginem, hodnoty čtyř a půl milionu osob.

Engine Unity3D je k dispozici se dvěma různými licencemi, které od verze 5.0 pouze omezují užívání a distribuci her a aplikací s tímto enginem, protože na Game Developers Conference 2015 firma Unity Technologies, spolu s představením verze 5.0, oznámila zpřístupnění kompletního enginu včetně všech doplňků a uživatelské podpory zdarma. Licence Personal umožňuje tedy pro nekomerční účely používat tento engine zdarma, kdežto licence Professional pro komerční užívání je k dispozici za sedmdesát pět dolarů za každý měsíc užívání nebo jednorázově za jeden tisíc pět set dolarů. Na Obr. 9 je zobrazeno srovnání užití různých engine a je vidět že Unity3D je jednoznačně nejvyužívanější engine na trhu. Jedinou konkurencí jsou aplikace naprogramované v nativním kódu. Červené sloupce zastupují procento vývojářů, kteří pracují s Unity3D a modré vývojáře, kteří preferují Unity3D [2]. Zřejmě nejúspěšnější hrou, používající tento engine, je karetní hra Hearthstone: Heroes of Warcraft od firmy Blizzard, která v květnu evidovala více než třicet milionů uživatelských účtů.



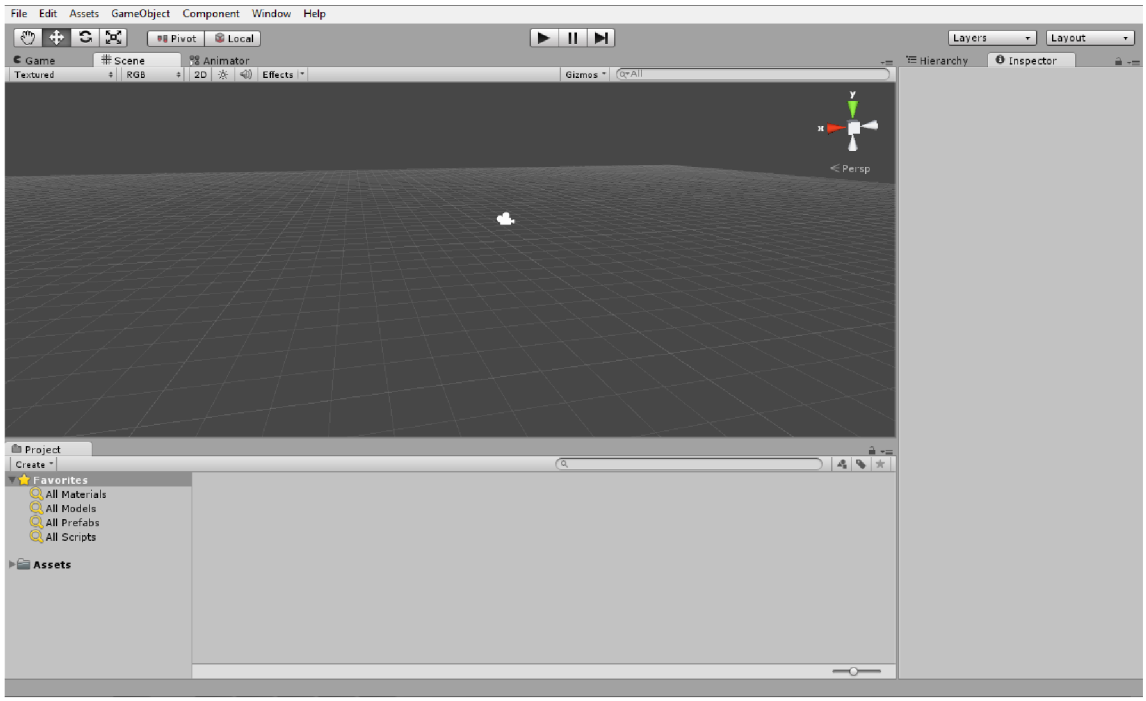
Obr. 9 Srovnání užití různých technologií pro tvorbu her

Pro tuto práci jsem zvolil verzi Unity3D 4.5.1f3 a to z toho důvodu, že se jednalo o poslední dostupnou stabilní verzi na počátku vývoje aplikace (podzim 2014).

3.1.1 Vývojové prostředí

Okno vývojového prostředí je z velké části tvořeno několika různými podokny, jejichž rozmístění je velice variabilní a závisí pouze na uživateli. Horní lišta obsahuje záložky pro: správu a nastavení projektu – Edit, import/export souboru doplňků popř. vytvoření speciální struktury (prefab, material atd.) – Assets, vytváření a editaci objektů scény – GameObject, záložky pro správu komponent objektů – Component, záložku pro editaci rozvržení – Window a záložku s nápovědami – Help.

Při práci s vývojovým prostředím Unity3D je důležité rozlišovat pojmy hra a scéna. Pojmem hra je označen výsledný projekt, který se skládá z několika scén (může být označeno jako úroveň).



Obr. 10 Vývojové prostředí Unity3D

Funkce podoken:

- Game – vyobrazení scény z pohledu herní kamery, tento pohled je automaticky aktivován při spuštění scény
- Scene – klasický náhled na scénu, slouží pro rozmísťování objektů
- Animator – prostředí pro vytváření a editaci animačních sekvencí
- Hierarchy – seznam a struktura všech objektů, vložených do scény
- Inspektor – detailní informace o označeném objektu s možností editace
- Project – kompletní struktura celého projektu, myšleno s umístěním importovaných objektů a scriptů ve složkách. Základní složkou je složka Assets, do které se následně vytváří podsložky a vkládají objekty.
- Console – výpis chyb, varování a informací při spouštění a běhu scény nebo importování objektů. Důležité pro debugování
- Asset store – speciální okno, ve kterém po přihlášení má uživatel přístup k různým balíčkům doplňků ať již oficiálních nebo uživatelských. Oba typy mohou být dostupné zdarma nebo za poplatek.

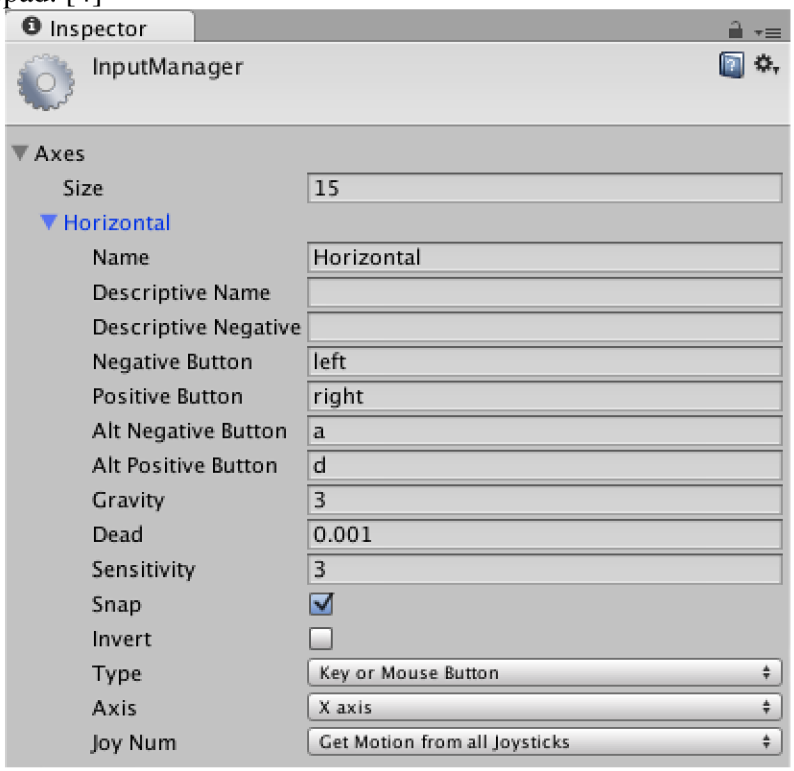
3.1.2 Objekty scény

Základem každé vytvořené scény jsou herní objekty a jejich komponenty. Lze je vytvářet přímo ve vývojovém prostředí, takové objekty se objeví okamžitě po spuštění scény a pokud nejsou odstraněny pomocí scriptu, tak v ní zůstávají až do ukončení scény. Druhou možností je vytváření pomocní scriptu. Vytvoření takového objektu se odvíjí v závislosti na struktuře scriptu. Mohou být také vytvořeny okamžitě po spuštění, ale mohou se také vytvořit až po splnění určitých podmínek. Ve scriptu se pro práci s herními objekty používá proměnná typu *GameObject*. Vytvoření objektu je realizováno funkcí *Instantiate(),* která vytvoří kopii objektu, který je součástí projektu a vloží ho do scény.

Každému objektu může být přiřazeno několik komponent, které slouží pro určení vlastností objektu. Příkladem komponenty může být určitý fyzikální model, script nebo animační sekvence [3]. Stejně jako v případě objektů tak i komponenty lze přiřazovat přímo v editoru, nebo pomocí scriptu.

3.1.3 Práce se vstupními zařízeními

Práci se vstupy obstarává třída Input. Pro PC hry se nastavují dvě základní osy pohybu a libovolné množství funkčních tlačítek, u her pro telefony se pracuje se čtením doteku obrazovky, místa doteku a akcelerometry. Nastavení os a tlačítek pro PC hry je realizováno přes záložku Edit>>Input Manager. Na Obr. 11 je ukázáno nastavení osy pojmenované jako horizontal. Pro následné použití jsou nutné zejména parametry Name - pojmenování, (Alt) Negative Button – tlačítka pro zápornou část osy a (Alt) Positive Button – nastavení kladné části osy. Položka Type určuje vstupní zařízení, ze kterého bude osa čtena. Unity3D podporuje čtení několika vstupních zařízení, konkrétně klávesnice, myš, joystick a gamepad. [4]



Obr. 11 Input manager

3.1.4 Osvětlení scény

Pro simulaci světelného zdroje je nutné do scény vložit Light objekt. Těch je v Unity3D několik typů, přičemž některé parametry mají společné, některé jsou specifické pouze pro daný druh. Pro výpočet nasvětlení a z toho plynoucí vrhání stínů objektu jsou důležité parametry Intensity, Direction a Color, udávající intenzitu směr a barvu nasvětlení. Druhy zdrojů světla a jejich použití: [5]

- Point Light – bodový zdroj, vyzařuje do všech stran do vzdálenosti omezené parametrem Range, intenzita se snižuje se vzdáleností, používá se pro simulaci lamp a jiných lokálních světelných zdrojů
- Spot Light – bodový zdroj, který ale nevyzařuje do všech směrů, nýbrž produkuje světelný kužel omezený úhlem při vrcholu – Angle, použití např. pro baterku
- Directional Light – plošný zdroj světla, nezáleží na umístění ve scéně, osvětluje všechny objekty ze stejného směru, používá se pro slunce

3.1.5 Scriptování

Jak již bylo zmíněno, scripty slouží pro vytváření objektů a přiřazení komponent. To je ale jedna z mnoha jejich funkcí. Obecně lze říci, že slouží pro zasahování do scény v reálném čase. Pro práci se scripty je spolu s enginem Unity3D poskytována upravená verze programu MonoDevelop. Jedná se o open-source vývojové prostředí pro programování, které podporuje celkem 12 programovacích jazyků.

Od verze 5.0 je engine Unity3D kompatibilní pouze se dvěma programovacími jazyky – klasický C# a UnityScript, což je speciální jazyk, vytvořený pro tento engine odvozením od JavaScriptu. K dispozici je kompletní online manuál s popisem všech funkcí. Níže je vidět automaticky vygenerovaný kód při vytvoření nového scriptu v jazyce C#.

```
using UnityEngine;
using System.Collections;

public class NewBehaviourScript : MonoBehaviour
{
  void Start ()
  {
  }
  void Update ()
  {
  }
}
```

Pro jazyk C# je důležité uvést odkaz na knihovnu *UnityEngine* a určit dědičnost nové třídy od třídy *MonoBehaviour*, která obsahuje funkce pro používání enginu. Prázdný script v jazyce UnityScript je ukázán níže.

```
#pragma strict
function Start ()
{
}
function Update ()
{
}
```

Jelikož je UnityScript upraven přímo pro engine Unity3D, tak není potřeba odkazovat na jeho knihovnu jako v C#. Další podstatný rozdíl je absence pojmenování nově vytvořené třídy. To je přiřazeno automaticky podle jména souboru, obsahujícího script.

Při vygenerování nového scriptu byly v obou jazycích vytvořeny dvě funkce, které mají přesně určenou dobu volání při běhu programu [6]:

- Start() – tato funkce se zavolá pouze jednou při začátku hry, před prvním voláním funkce *Update()*, hodí se zejména pro inicializaci proměnných
- Update() – volání této funkce je realizováno pravidelně s každým snímkem, používá se proto pro realizaci pohybu, spouštění dalších akcí nebo čtení vstupů z klávesnice

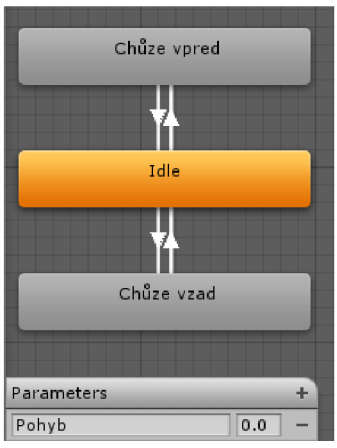
Při programování je možné použít další předem definované funkce jako např. *Awake(), LateUpdate()* nebo *FixedUpdate()*. Pro scriptovací část závěrečné práce byl zvolen jazyk C#.

3.1.6 Animace

O animování objektů se v Unity3D stará systém Mecanim. Nejedná se ovšem o vytváření animací, ale o jejich skládání do sekvencí – Animator Controller, nastavení podmínek pro spuštění jednotlivých animací a následné přiřazení celé sekvence k danému objektu. Velkou výhodou je podpora přepočtu animace na model postavy, takže animace původně vytvořená pro postavu s určitými parametry může být použita také pro jiné modely. Podmínkou pro tento přepočet je nastavení animovaní objektu na Humanoid [7].

Vytváření Animator Controlleru probíhá jako vytváření blokového schématu, zastupující kompletní animaci objektu, kde každý blok má přiřazenu jednu animaci. Jednotlivé bloky mohou zastupovat určitou situaci k animování (blok typu State) např. chůze vpřed nebo další rozvětvený systém bloků (New Blend Tree in State) a spojující šipky určují přechody mezi bloky a podmínky přechodu.

Příklad jednoduchého Animator Controlleru pro postavu je na Obr. 12. Skládá se ze tří bloků, kde blok Idle zastupuje klidovou fázi postavy, a bloky chůze vpřed a chůze vzad spouští animaci chůze v závislosti na hodnotě proměnné pohyb, která je typu float. Ta nabývá hodnot v intervalu (-1;1) v závislosti na čtení vstupu z klávesnice. Pokud bude hodnota větší než 0, spustí se blok Chůze vpřed, naopak při hodnotě menší než 0 je spuštěn blok Chůze vzad. Přiřazení této hodnoty je realizováno v ukázkovém scriptu pod Obr. 12.



Obr. 12 Ukázkový Animator Controller

```
public class AnimaceUkazka : MonoBehaviour
{
 Animator animator;
 float chuze;

 void Start ()
 {
      animator = GetComponent<Animator> ();
 }
 void Update ()
 {
      chuze = Input.GetAxis("Vertical");
      animator.SetFloat ("Pohyb", chuze);
 }
}
```

V ukázkovém kódu je deklarovaná proměnná typu Animator, které je ve funkci Start přiřazena komponenta stejného typu od objektu, ke kterému je script přiřazen. Hodnoty proměnné chuze se získají čtením hodnot osy Vertical a následně se přiřadí parametru Pohyb v Animator Controlleru z Obr. 12.

3.1.7 Fyzika

Velmi důležitou komponentou objektů je fyzikální model, určující kolize. Engine Unity3D podporuje několik kolizních modelů, které lze rozdělit na modely pro statické objekty a pro pohybující se objekty. Mezi ty první patří různé typy Colliderů v závislosti na tvaru objektu. Základním je klasický Box Collider, který vytvoří jednoduchý kvádr simulující kolizní model. Zřejmě nejsložitější je Mesh Collider, což je kolizní model kopírující přesný tvar objektu [8]. U tohoto typu může být problém s použitím Mesh Collideru pro importované objekty a to z důvodu různých souřadných systémů Unity Editoru a příslušného modelovacího programu. Ve výsledku tak získáme sice kolizní model správného tvaru ale třeba špatně natočený nebo naporcovaný. Pro odstranění této komplikace obsahuje Editor funkci pro přepočítání Mesh Collideru importovaného objektu na své souřadnice.

Druhým typem fyzikálních modelů jsou tedy modely pro pohybující se objekty. Do této kategorie patří RigidBody a Character Controller. První z nich je hlavní komponenta pro simulaci fyzikálního chování objektů. Po přiřazení k objektu je automaticky na objekt aplikována gravitační síla. Samozřejmostí je kolizní model kopírující tvar objektu. Při pohybování objektem s touto komponentou se nedoporučuje pohyb pomocí úpravy pozice a natočení, ale doporučuje se pohybovat s objektem pomocí vytvoření pohybové síly působící na daný objekt [9]. Character Controller se využívá hlavně pro postavy ve hrách z perspektivy první nebo třetí osoby. Jeho hlavní podstatou je, aby postava nepropadla podlahou a neprocházela zdmi. Slabinou tohoto fyzikálního modelu je ne zcela realistický pohyb objektu, což je způsobeno absencí momentu setrvačnosti [10].

Fyzikální modely nemusí být použity pouze pro kolize mezi objekty. Lze je použít také jako spouštěče určitých situací. Příkladem takového použití je vytvoření kolizní kapsle, která bude detekovat, zda hráč prošel daným místem a pokud ano spustí určitou akci. Toho lze docílit deaktivací kolize, ale zároveň zpracováním detekcí kolize. K tomu slouží funkce *OnTriggerEnter()*.

3.1.8 Uživatelské rozhraní

Součástí každé počítačové hry je uživatelské rozhraní (dále UI), jako je např. hlavní menu nebo uživatelské menu v průběhu hry. Od verze enginu 4.6 se s uživatelským rozhraním pracuje dvěma způsoby. Původní způsob je vytváření UI pomocí scriptu, konkrétně vytvoření funkce *OnGUI()*. Tato funkce může být volána několikrát za frame, v závislosti na počtu centů v ní. Možnou nevýhodou tohoto způsobu a zřejmě také důvod vytvoření nového, je pracnost při vytváření a také testování, jelikož pro každou sebemenší úpravu je nutné scénu spustit a zkontrolovat výsledný dopad.

Novým jednodušším způsobem je vytváření UI přímo v Editoru. Základem je vytvoření struktury Canvas, což je prázný GameObject s komponentou Canvas, který je vložen do scény. Canvas lze zjednodušeně chápat jako podklad pro vykreslování UI. Přidáváním prvků na tuto plochu se tedy docílí vytvoření UI a je hned patrné výrazné zjednodušení celého procesu vytváření a hlavně testování, protože vývojář vidí okamžitě výsledek bez nutnosti scénu spouštět.

Vzhledem k tomu, že jsem pracoval s verzí 4.5.1f3, ve které nebyl ještě systém Canvas nebyl zaveden, tak veškeré UI je realizováno scriptově přes funkci *OnGUI()* [11]. Níže je ukázkový kód jednoduchého UI, obsahující tlačítko, po jehož stisknutí bude hra ukončena.

```
using UnityEngine;
using System.Collections;
public class UkazkaUI : MonoBehaviour
{
 void Start()
 {
 }

void Update()
 {
 }
 void OnGUI() {
  if (GUI.Button(new Rect(100,120,100,30),"Konec Hry")
       {
       Application.Quit();
        }
 }
}
```

Pokud je tento script přiřazen např. kameře a ve scéně se žádný jiný objekt nevyskytuje, po spuštění se zobrazí pouze tlačítko na modrém pozadí na pozici, která mu byla nastavena. Pokud se ve scéně vyskytují další objekty, tak je nutné vytvořit pozadí a také určit podmínku, kdy se toto rozhraní zobrazí. Příkladem užití tohoto UI je zobrazení tlačítka pro vypnutí hry po stisknutí klávesy ESC. Upravena funkce *OnGUI()* z předešlé ukázky by potom vypadala takto:

```
void OnGUI() {
 if (UImenu){
  GUI.Box(new Rect(0,0,Screen.width,Screen.height),"UI");
   if (GUI.Button(new Rect(100,120,100,30),"Konec Hry"){
            Application.Quit();
       }
  }
```

```
}
```

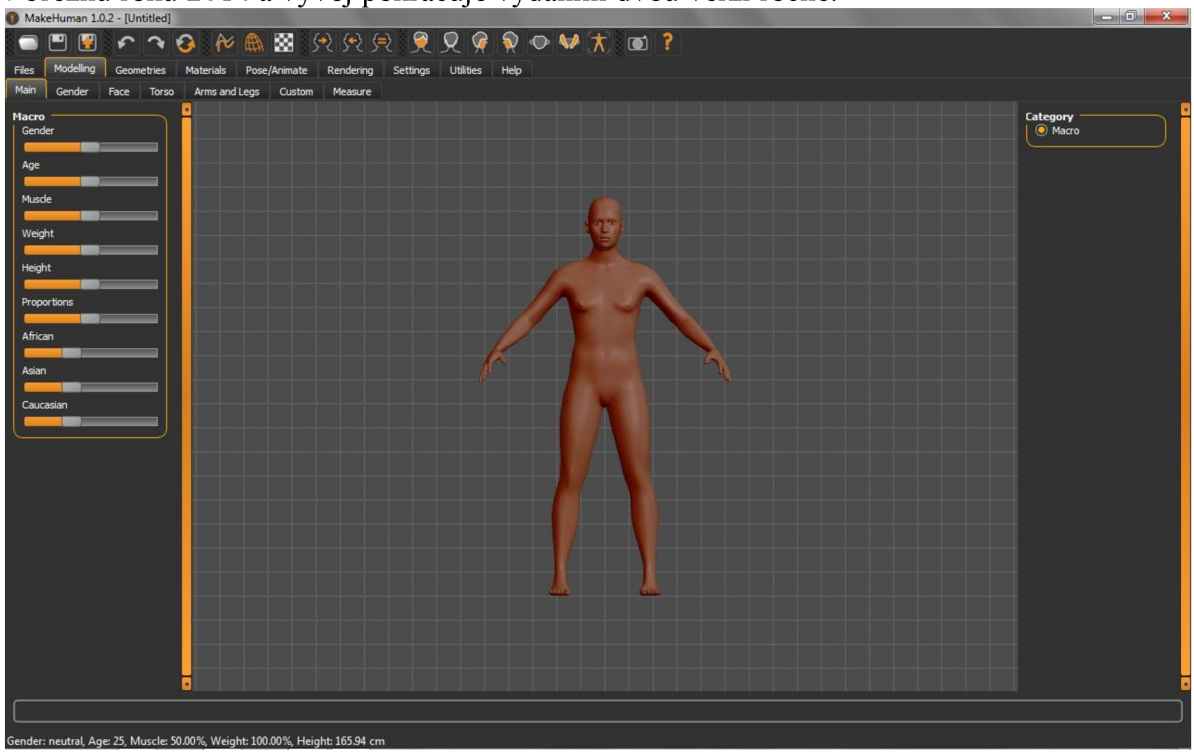
Při tomto použití se UI zobrazí, pouze pokud bude boolovská proměnná nastavena na hodnotu true. Aby bylo možné tuto proměnou nastavovat v libovolném čase, je potřeba ji globálně deklarovat a do funkce *Update()* umístit tento kód:

```
if(Input.GetKey("escape")){
 UImenu = true;
}
```

3.2 MakeHuman

MakeHuman je multiplatformní open source program pro vytváření 3D modelů postav. Vytvořila ho skupina programátorů ve spolupráci s umělci a vědci, kteří se zajímali o 3D modely lidské postavy. Modelování je založeno na 3D morph target technice – začíná se od neutrálního modelu lidské postavy a přes různé nástroje je model postupně upravován. První verze MakeHuman byla vydána v roce 2000 jako nástupce programu MakeHead, což byl nástroj pro modelovací program Blender napsaný v jazyce Python programátorem Manuelem Bastionini. Ten stál také u zrodu skupiny programátorů, kteří vytvořili MakeHuman, také jako nástroj pro Blender v jazyce Python. V roce 2004 byl vývoj tohoto nástroje zastaven z důvodu náročnosti programování v jazyce Python pouze pomocí BlenderAPI [12].

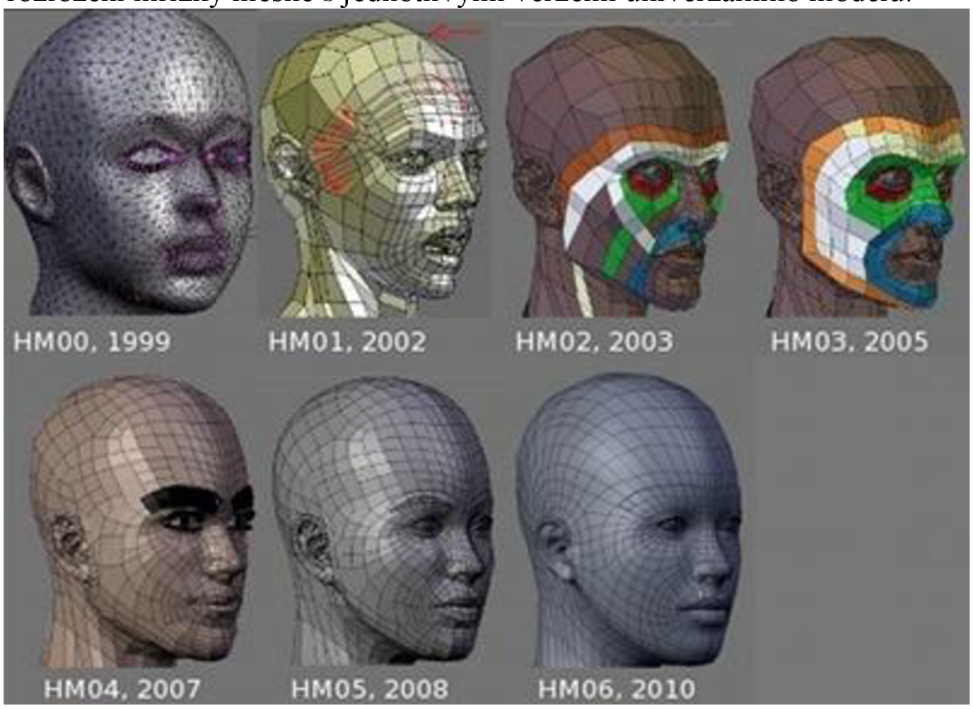
V roce 2005 byla vydána první verze MakeHuman jako samostatný program, kompletně přepracovaný z jazyku Python do jazyku C. V roce 2009 se tým vývojářů rozhodl vrátit k jazyku Python a byla to verze 1.0 pre-alpha. Verze 1.0.0 byla vydána v březnu roku 2014 a vývoj pokračuje vydáním dvou verzí ročně.



Obr. 13 Uživatelské rozhraní MakeHuman 1.0.2

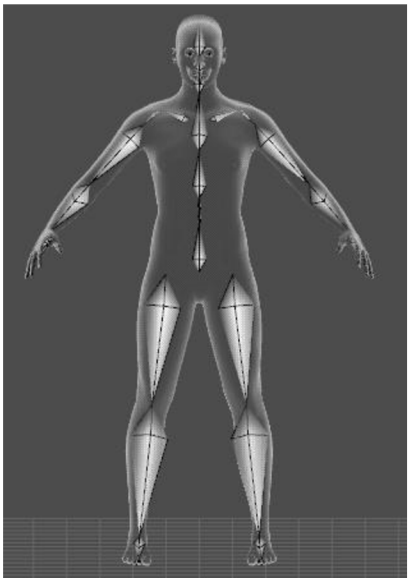
Cílem projektu je vytvořit aplikaci, ve které je možné vytvářet velkou škálu modelů lidských forem z jednoho univerzálního meshe. Vytvořit takto univerzální mesh byl hlavní

úkol od prvního vydání MakeHuman. První prototyp univerzálního meshe byl dokončen v roce 1999 pro program nástroj MakeHead a jednalo se pouze o mesh hlavy. První univerzální model celé postavy byl dokončen v roce 2002 pod označením HM01. Model používaný ve verzi 1.0.2 byl dokončen v roce 2013. Na Obr. 14 je zachycen vývoj topologie hlavy a rozložení mřížky meshe s jednotlivými verzemi univerzálního modelu.



Obr. 14 Vývoj topologie hlavy v MakeHuman

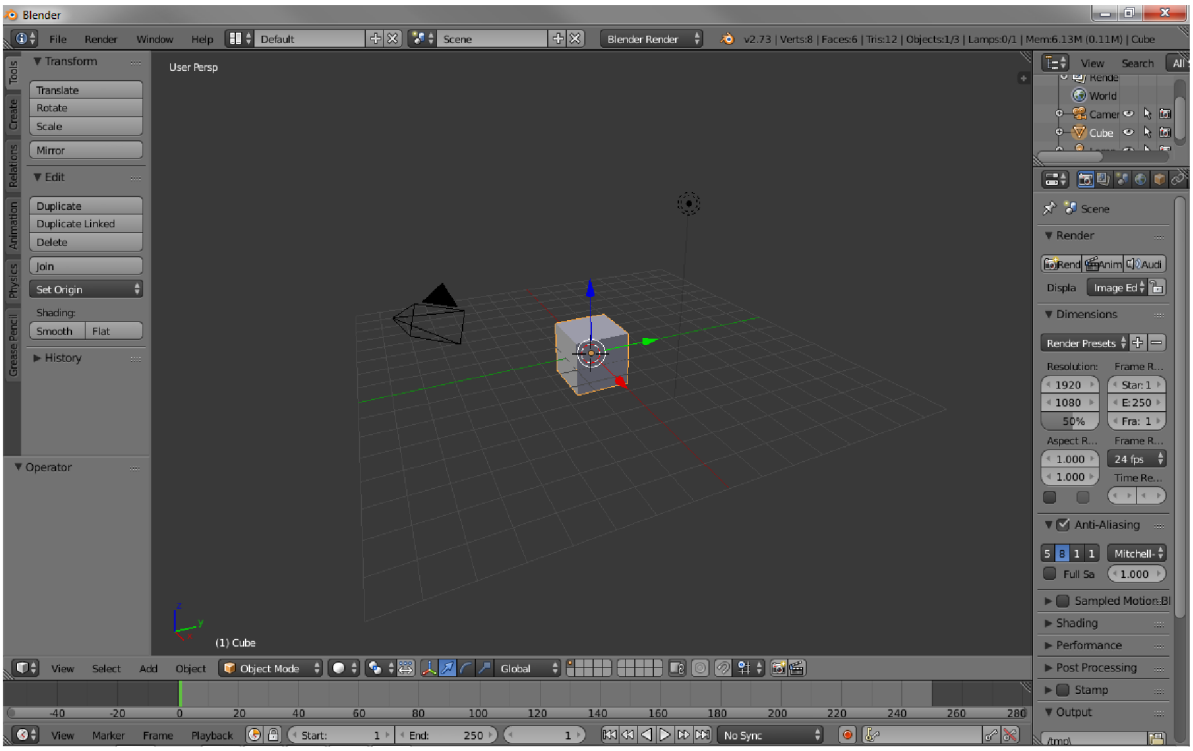
Podstatným nástrojem programu MakeHuman je vytvoření kostry modelu (dále Rig) pro následné animování. K dispozici je několik nastavení, které se liší v počtu kostí pro animování. Tento počet určuje výsledný rozsah animování modelu. Nechybí ani možnost vytvořit si vlastní Rig např. v programu MakeRig.



Obr. 15 Základní Rig pro animace složený ze 73 kostí

3.3 Blender

Blender je multiplatformní open source program od firmy Blender Foundation, určený k modelování a vykreslování 3D počítačové grafiky a práci s animacemi. Hlavní výhodou a také důvodem volby tohoto programu pro modelování je jeho kompletní dostupnost zdarma i pro komerční účely. Konkrétně jsem použil verzi 2.73. O veškeré vykreslování se stará grafická knihovna OpenGL, což také umožňuje jednoduchou přenositelnost vytvořených modelů na všechny podporované platformy. Velkou výhodou je volně nastavitelný Editor, což mnoho jiných i placených programů nenabízí. Uživatel si otevře a rozmístí nástrojová okna přesně tak, jak mu to vyhovuje – Obr. 16.



Obr. 16 Blender Editor

Modelování v Blenderu je na rozdíl od způsobu modelování v MakeHuman založeno na přidávání bodů (Vertex) a jejich spojování pomocí hran (Edge). Objekty jsou tedy reprezentovány takto vzniklými plochami – plošková reprezentace těles. Pro uživatele, kteří s tímto programem začínají, může být zpočátku problém ovládání pomocí klávesových zkratek. To je způsobeno množstvím nástrojů, které jsou k dispozici, protože není možné je všechny rozmístit v Editoru, tak má každý přiřazen klávesovou zkratku. Jakmile si na tento způsob uživatel zvykne a osvojí si zkratky, tak je modelování v tomto programu naprosto intuitivní a hlavně podstatně rychlejší než hledání nástrojů v jednotlivých podnabídkách [13].

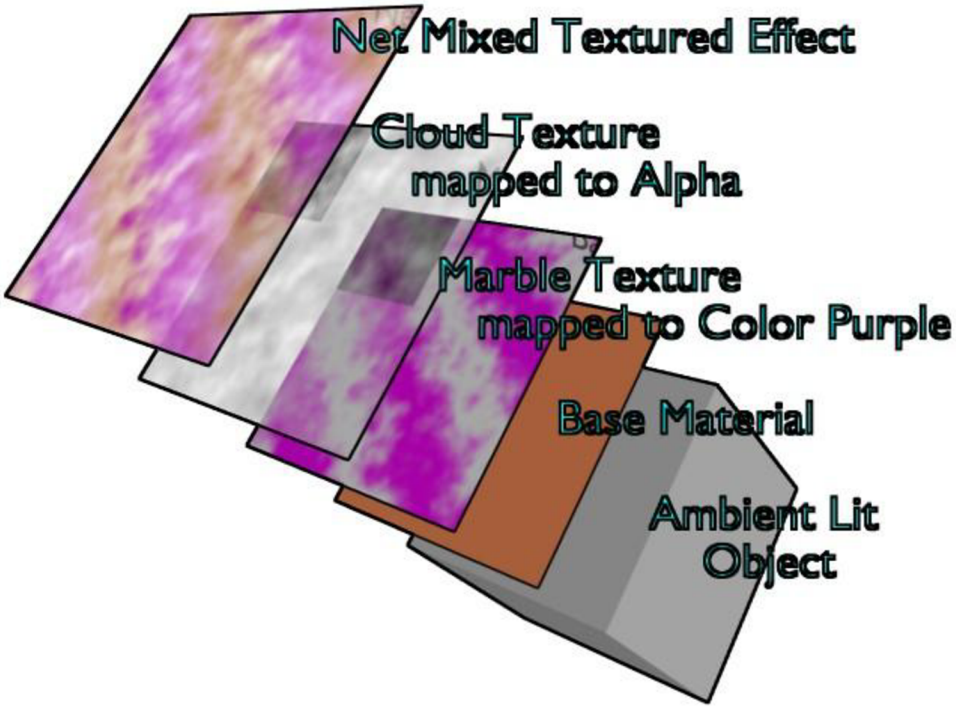
3.3.1 Nanesení textur

Práce s texturami je při vytváření modelu velice důležitá, protože určuje výsledný vzhled objektu. Navíc aplikací textur lze změnit také průhlednost, odrazivost nebo zrcadlení, to vše v závislosti na přidělení textury a jejím namapování – správné rozvržení plochy na zdroj textury v podobě obrázku. V Blenderu lze texturu přiřadit materiálu,

příchozímu světlu, pozadí objektu nebo částicovému systému. V této práci jsem pracoval pouze s použitím textury na materiál, proto je přiblížena jen tato možnost.

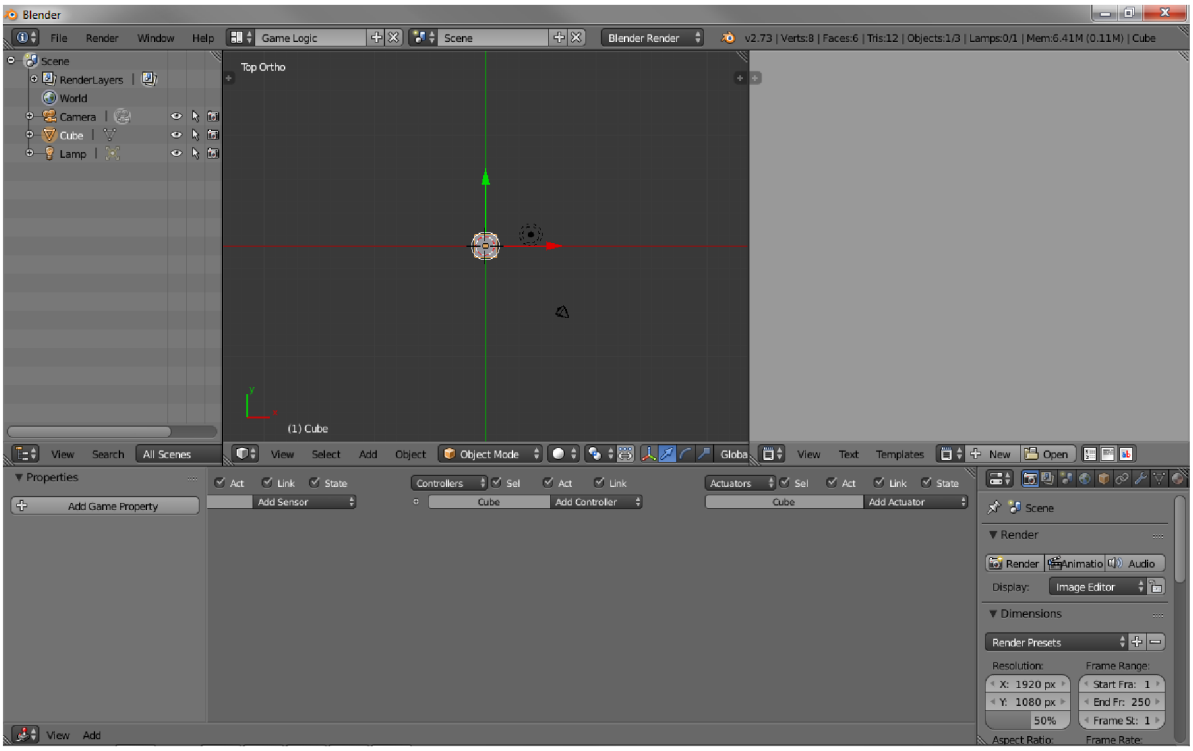
Při aplikaci materiálu na objekt vznikne poměrně jednotvárný a ne příliš reálný vzhled. Použitím textury lze tento aspekt odstranit např. tak, že se na materiál nanese několik různých textur do vrstev, jejichž smísením se docílí realistické odrazivosti a hloubky barev. Textury aplikované na materiál je možné rozdělit do tří kategorií [14]:

- Procedurální Textures – textury vytvořené podle určitých matematických vzorců, užití např. dřevo nebo mraky
- Obrázky nebo klipy – fotky a filmy, které jsou promítány na objekt, např. namapování mapy Země na kouli s cílem vytvoření glóbusu
- Mapy prostředí – fotky nanesené na objekt pro docílení realistické odrazivosti, např. fotky ulice okolí na okno



Obr. 17 Nanesení několika textur do vrstev

Zajímavým nástrojem je integrovaný herní engine (Obr. 18), pomocí kterého lze vytvářet i počítačové hry, ale jeho hlavní využití je spíše tvorba různých vizualizací. Pro práci s tímto enginem nejsou vyžadovány žádné programátorské dovednosti, avšak je zde i možnost využití scriptů v jazyce Python.



Obr. 18 Blender game engine

Tento engine byl společně s enginem Crystal Space použit při tvorbě open source hry Yo Frankie! z roku 2008 a také při vytváření tří krátkých filmů Elephants Dream (2006), Big Buck Bunny (2008) a Sintel (2010). Při tvorbě všech výše zmíněných projektů byl jako hlavní cíl stanoven použití pouze Open Source softwaru.

4 Vytvoření virtuálního světa

Poslední část této práce pojednává o popisu vytvoření ukázkového virtuálního světa pomocí herního enginu Unity3D. V předchozích kapitolách byl vytvořen a přiblížen možný postup tvory počítačové hry (kapitola 2) a představeny zvolené technologie, které byly použity při tvorbě ukázkové aplikace (kapitola 3). Je nutné zmínit, že tento virtuální svět byl vytvořen jako základ pro větší projekt a je tomu uzpůsoben. Nejedná se tedy o čistě ukázkový virtuální svět, který by bylo možné použít bez jakýchkoliv úprav pro jinou počítačovou hru.

4.1 Charakteristika výsledného projektu

Výsledným projektem, pro který byl tento virtuální svět vytvořen, je online hra pro více hráčů. Na projektu jsem spolupracoval ještě s dalšími dvěma spolužáky. Veškerá rozhodnutí, která měla vliv na celkovou podobu výsledné hry, byla učiněna kolektivně, ale rozhodnutí týkající se jednotlivých částí byla pouze na tom, kdo na ní pracoval. Hra byla pojmenována Virtual World.

Charakteristika výsledné hry:

- Online hra pro více hráčů
- Použití herního enginu Unity3D
- Rozdělení projektu na tři části
  - komunikace – Daniel Balcárek
  - virtuální svět pro pohyb hráčů – Jakub Hamal
  - závodní hra, odehrávající se ve virtuálním světě – Robert Kováč

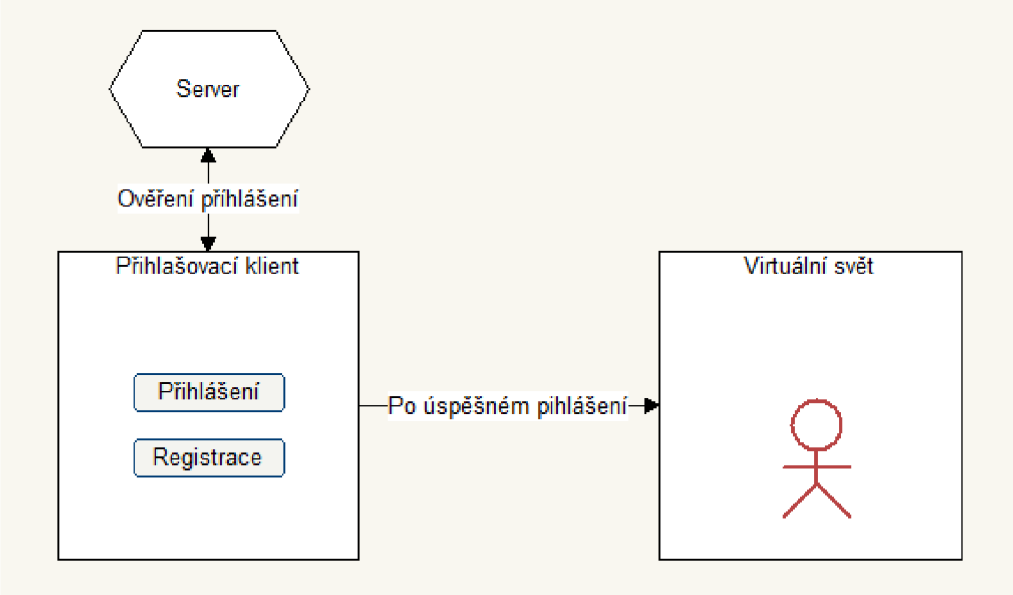
Při vytváření virtuálního světa byl použit postup popsaný v kapitole 2. Prvním krokem po určení žánru hry – online hra pro více hráčů a zvolení technologie – Unity3D herní engine, bylo vytvoření Game design dokumentu pro virtuální svět. Dokument byl upraven, aby sloužil částečně jako dokumentace a také pro zdůvodnění voleb řešení jednotlivých částí. Následně jsou postupně zpracovány jednotlivé body design dokumentu.

4.2 Game design dokument

Cílem je vytvořit virtuální svět pro pohyb hráčů. Každý hráč bude zastoupen svoji postavou, kterou bude moci ovládat a bude také vidět postavy ostatních hráčů. Hráči mezi sebou budou komunikovat pomocí chatu, přičemž bude možné odeslat buď veřejnou zprávu, kterou uvidí všichni připojení hráči, nebo soukromou zprávu zvolenému hráči. Ve světě se budou nacházet dva interaktivní panely. Jeden z nich bude obsahovat stručné informace o virtuálním světě, pomocí druhého se bude moci hráč přihlásit do závodní hry.

Základní logika

Pro vstup do světa je nutné se nejprve přihlásit popř. vytvořit nový účet. Tyto funkce bude realizovat přihlašovací klient. Po úspěšném přihlášení bude hráči umožněn vstup do virtuálního světa. Pro zobrazení světa byla zvolena kamera z pohledu třetí osoby. Návrh logiky je zobrazen na Obr. 19.



Obr. 19 Logika virtuálního světa

Ovládání

Pro ovládání byl zvolen běžně používaný systém, založený na dvou osách. Vertikální osa určuje pohyb dopředu (kladné hodnoty) a dozadu (záporné hodnoty). Tato osa je ovládána klávesami W a S nebo šipkami Nahoru a Dolů. Horizontální osou se ovládá natočení postavy, kde záporné hodnoty určují rotaci vlevo a kladné rotaci vpravo. Ovládání této osy zajišťují klávesy A a D nebo šipky Vlevo a Vpravo. Natočení postavy je možné ovládat i pomocí myši, konkrétně stisknutím a držením pravého tlačítka a pohybem myši. Dále jsou ve hře používána další dvě tlačítka, a sice klávesa Enter pro otevření okna chatu a odeslání zprávy a klávesa Escape pro zobrazení hlavního menu hry. Hlavním důvodem volby tohoto stylu ovládání byla jeho jednoduchost a všeobecná známost.

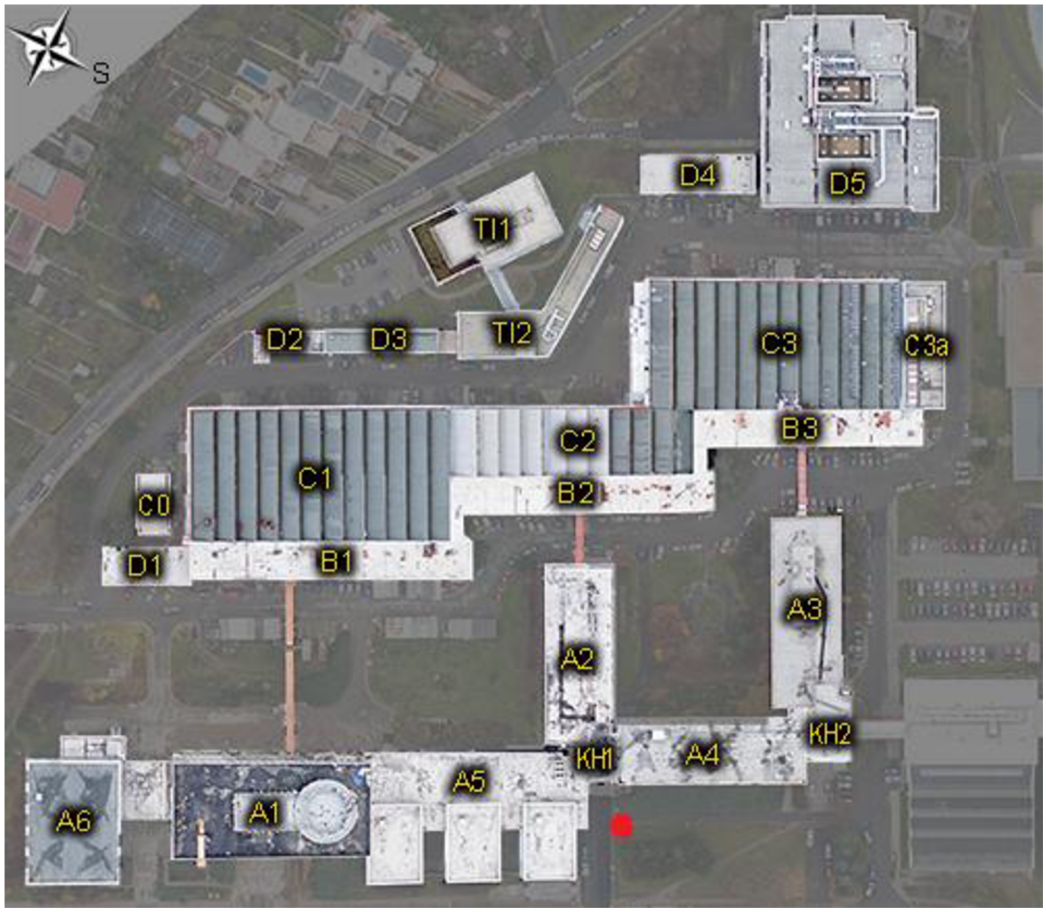
Kamera

Jako kamera byl zvolen pohled třetí osoby, kdy je kamera umístěna za postavu hráče. Důvodem této volby byl fakt, že kamera z pohledu první osoby se užívá výhradně pro akční hry, což není tento případ. Hráč si bude moci zvolit, zda chce kameru uzamknout v pohledu postavy hráče, nebo její natáčení nechat volitelné. Ve druhém případě jsou umožněny dva způsoby natáčení kamery. První z nich byl popsán v popisu ovládání, kdy se kamera natáčí pomocí stisknutí a držení pravého tlačítka myši. Kamera se poté natáčí stejně jako postava hráče. Druhý způsob využívá levé tlačítko myši. Při jeho stisknutí a držení je možno kameru libovolně natáčet bez vlivu na postavu hráče.

Prostředí virtuálního světa

Jako základ pro virtuální svět pro pohyb hráčů byl použit areál FSI VUT v Brně a to z důvodu propagace školy. Budovy byly jednotlivě vymodelovány a poté rozmístěny tak, jak je tomu ve skutečnosti. Na budovy byly umístěny skutečné textury, které byly nafoceny přímo z jednotlivých budov. Všechny jsou opatřeny kolizním modelem, aby jimi hráč nemohl procházet. Mapa areálu FSI je zobrazena na Obr. 20, přičemž červený bod zastupuje počáteční pozici ve virtuálním světě. Na tuto pozici je umístěna postava při prvním vstupu

do světa po registraci. V této lokaci jsou také umístěny dva interaktivní panely – jeden pro informace a druhý pro přihlášení do závodní hry.



Obr. 20 Mapa areálu FSI VUT v Brně

Střídání den/noc

Pro zvýšení realističnosti bude aplikován efekt střídání dne a noci v závislosti na skutečném čase dne. Tento čas je získáván z počítače, na kterém je hra provozována. Efekt počasí do hry aplikován nebyl, ale je to jedna z možností budoucího vývoje.

Postava hráče

Při registraci do hry si hráč bude moci zvolit, zda chce být zastoupen mužskou či ženskou postavou. Tato volba je definitivní, nelze ji později změnit. Hráč se může volně pohybovat ve světě a veškerý pohyb postavy bude animován. Jelikož tvorba animací nebyla cílem práce a jedná se o časově velmi náročný proces, byly použity animace pro postavy, které jsou zdarma dostupné jako doplňkový balíček k enginu Unity3D. Postava má definované tři druhy pohybu:

- Chůze vpřed a vzad
- Úkrok vlevo a vpravo
- Otáčení kolem svislé osy

Postava bude opatřena kolizním modelem pro zamezení procházení stěnami a postavami online hráčů.

Chat

Součástí světa je jednoduchý chat pro komunikaci mezi hráči. Komunikace může být veřejná – zobrazená všem hráčům, nebo soukromá – předána pouze cílovému hráči.

Postava online hráče

Ve světě se kromě hráčovy postavy budou vyskytovat také postavy online hráčů. Ty budou zastoupeny stejnou postavou, jakou si online hráč zvolil při registraci a budou kopírovat jeho pohyb. Ten bude animovaný stejně jako v případě postavy hráče, stejně jako bude totožný i kolizní model. Nad postavou každého online hráče bude zobrazena jeho herní přezdívka.

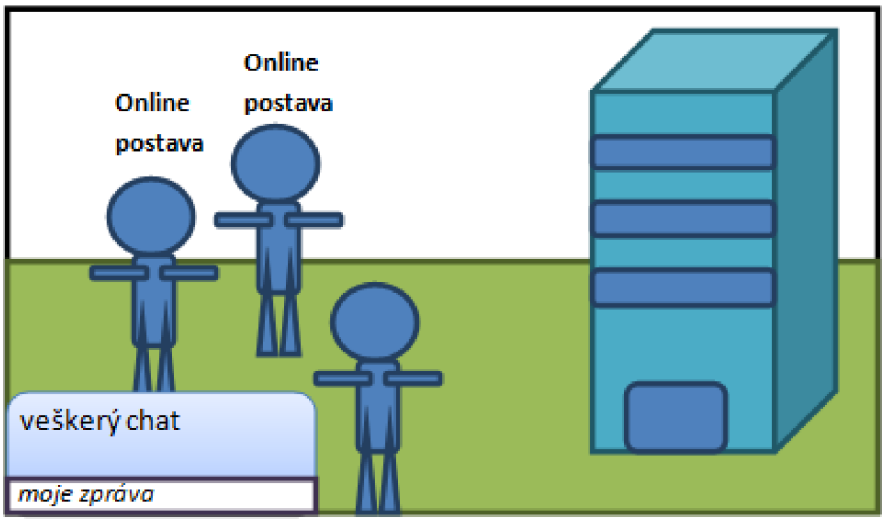
Interaktivní panely

Ve světě budou umístěny dva interaktivní panely. Pro jejich spuštění se musí hráč přemístit do jejich blízkosti a kliknout na ně pomocí levého tlačítka myši. Jeden panel slouží pro stručné informace o hře, přes druhý panel se hráč bude moci přihlásit do závodní hry. Oba panely budou opatřeny kolizním modelem.

Uživatelské rozhraní

Ve hře se bude vyskytovat celkem pět uživatelských rozhraní, konkrétně UI přihlašovacího klienta, UI herního světa, hlavní menu a UI pro oba panely. Rozhraní přihlašovacího klienta bude dále rozděleno na část pro přihlášení a část pro registraci. Toto UI se zobrazí jako první věc po spuštění hry. Při pohybu ve světě bude aktivní hlavní UI (Obr. 21). Většinu obrazovky bude pokrývat pohled na svět a v levém dolním rohu bude umístěno okno pro chat. Po stisknutí klávesy Escape bude zobrazeno UI hlavního menu. Zde bude umístěn popis ovládání, možnost ukončit hru a funkce pro přesunutí hráčovi postavy na počáteční pozici.

UI pro informativní panel bude tvořeno úvodní obrazovkou, ze které se hráč po stisknutí příslušného tlačítka přesune do jednotlivých podnabídek. UI panelu závodní hry bude zobrazena tabulka s nejlepšími výsledky a tlačítko pro přihlášení se do hry. Souřadnice pro rozmístění jednotlivých prvků jsou vztaženy vzhledem k velikosti okna, ve kterém je hra otevřena, čímž je docíleno stejného rozvržení UI pro různá rozlišení.



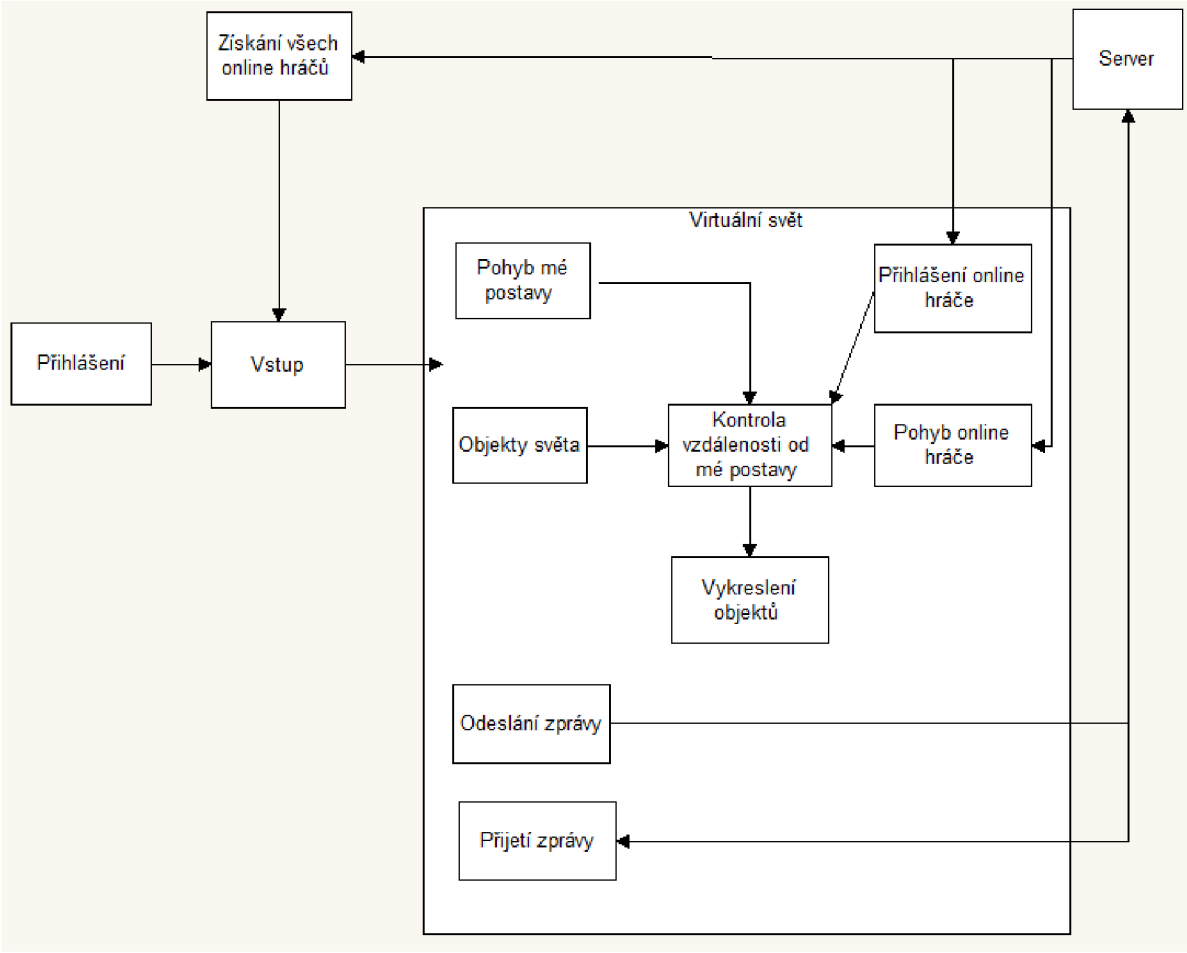
Obr. 21 Navržené UI herního světa

Ostatní objekty

Aby svět nepůsobil prázdně, bude v něm rozmístěno několik stromů a keřů. Jejich modely budou opatřeny kolizním modelem, aby jimi hráč nemohl projít.

Řízení vykreslování

Pro odlehčení výpočetní náročnosti bude vykreslování objektů světa omezeno vůči jejich vzdálenosti od postavy hráče. Toto omezení se vztahuje na všechny postavy online hráčů a stromy, nikoliv na budovy, protože pak by svět působil necelistvě a prázdně. Dalším důvodem vynechání budov z omezení vykreslování je prezentace celého areálu školy, čehož by nemohlo být dosaženo, kdyby nebyly vykreslovány všechny budovy. Celková funkčnost herního světa je zobrazena na Obr. 22.



Obr. 22 Kompletní funkčnost herního světa

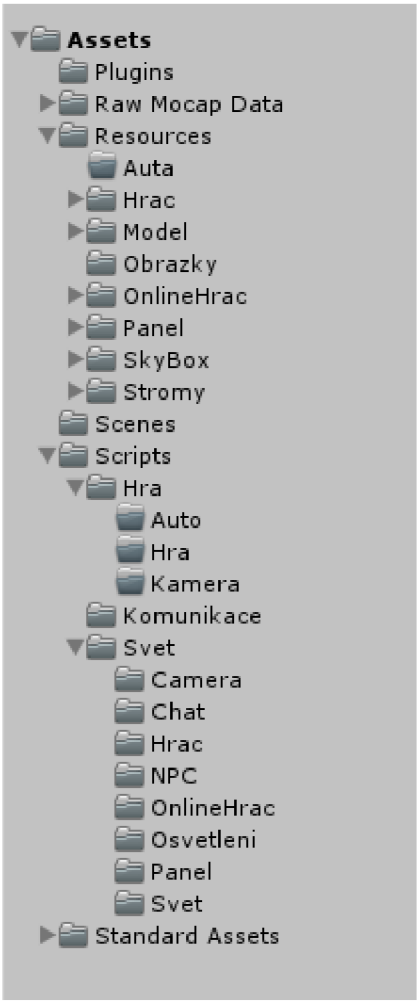
Chronologický postup vývoje

1. Vytvoření přihlašovacího klienta
2. Vytvoření prostředí herního světa včetně veškerých funkcí a propojení s přihlašovacím – vstup do světa, generování a vykreslování postav
3. Propojení panelu závodní hry se samotnou hrou

Propojením klienta se světem, resp. panelu s hrou, je nutno realizovat, protože každá z těchto částí je vytvořena ve vlastní scéně a je nutné tyto scény provázat.

4.3 Rozčlenění projektu

Pro lepší orientaci v souborech projektu je složka Assets, obsahující veškeré soubory projektu, rozdělena na další podsložky viz Obr. 23.



Obr. 23 Výsledné členění celého projektu

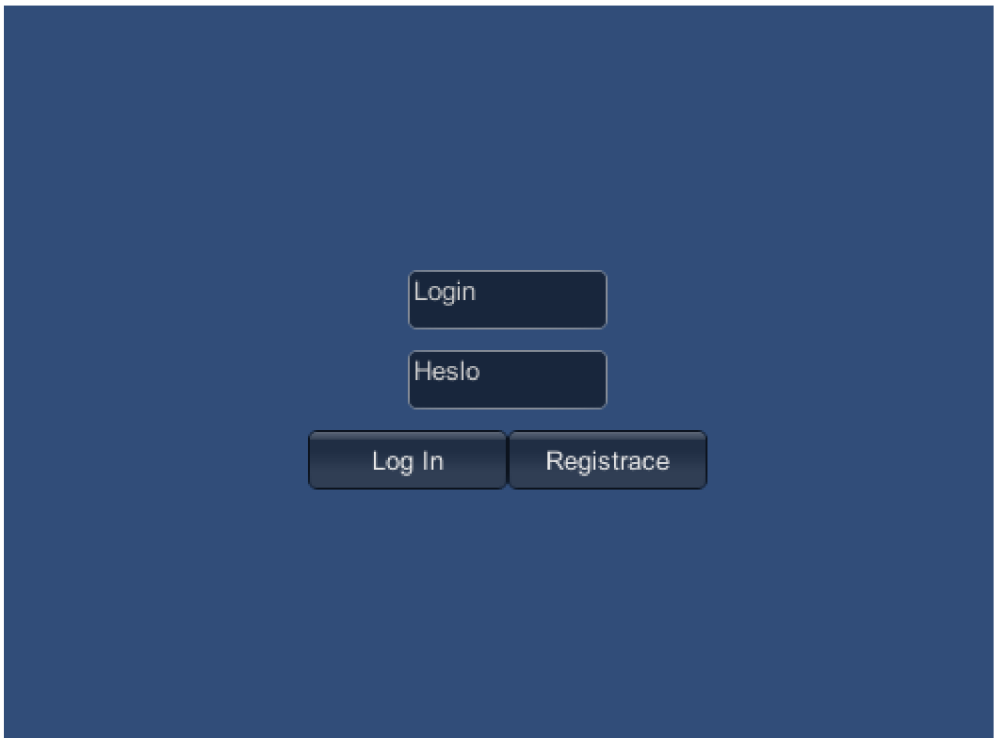
Všechny importované knihovny jsou umístěny ve složce Plugins a pro jejich správnou funkčnost nemohou být jinde. Tato podmínka je daná přímo Editorem enginu. Stejně tak je určený název složky Resources, ve které jsou veškeré objekty, vyskytující se ve scénách. K těmto objektům se pomocí scriptu přistupuje funkcí *Resources.Load(„*název objektu*")*. Složka Scripts obsahuje všechny použité scripty a je dále členěna na složky, ve kterých jsou umístěny scripty, které se vztahují k jednotlivým částím v závislosti na názvu podsložky. Ve složce Scenes jsou umístěny vytvořené scény.

Složka Raw Mocap Data obsahuje balíček animací postavy, který byl stažen zdarma z Unity Asset Store. Složka Standart Assets obsahuje balíček pro práci s osvětlením scény. Tyto soubory jsou součástí základního obsahu při stažení enginu Unity3D a pro jejich použití je stačí jen importovat do projektu.

4.4 Přihlašovací klient

Při spuštění se jako první načte scéna Login, která tvoří přihlašovacího klienta. Tato scéna je tvořena pouze prázdným objektem, ke kterému je přiřazen script, obsahující třídu *Login*, a kamerou. Jelikož prázdné objekty kamera nezobrazuje, tak po spuštění se automaticky zobrazí uživatelské rozhraní.

UI klienta je tvořeno částí pro přihlášení, částí pro registraci a částí pro vstup do herního světa. Pokud není hráč přihlášen, může se volně přepínat mezi registrační a přihlašovací částí. Pro úspěšnou registraci musí hráč vyplnit všechny položky a zvolit si postavu, přičemž položky k přihlášení Login a Heslo a položka Nickname je vyhrazena pro přezdívku v herním světě.



Obr. 24 Přihlašovací UI

Po úspěšném přihlášení se zobrazí část pro vstup do herního světa. Zde je umístěn obrázek postavy, kterou si uživatel vybral a tlačítko pro nastavení pozice na počátek. Třída *Login* slouží krom realizace UI také pro zajištění komunikace se serverem, a proto je nutné předat objekt této třídy i do scény herního světa. K tomu je třeba přesunout tento objekt do scény herního světa, protože při načítání nové scény jsou všechny objekty staré scény zničeny. Kód níže realizuje nahrání scény herního světa a předání objektu passingObject s třídou *Login*.

```
if(connected && estabilish && GUI.Button(new
    Rect(Screen.width / 2 + 125, Screen.height / 2 ,
    150, 30), "Vstup do sveta"))
{
     menu = false;
     DontDestroyOnLoad(passingObject);
     Application.LoadLevel("VirtualWorld");
}
```

Informace od serveru jsou zpracovány pomocí eventů (událostí), které jsou reprezentovány delegáty z importovaných knihoven. Eventy mají několik vstupních parametrů, které jsou v těle eventu zpracovány a předány příslušným funkcím. Komunikaci zajišťuje třída *MyClientPeer*, přičemž její objekt myPeer vytvořený ve třídě *Login* je deklarován jako public a to z důvodu přístupu z jiných tříd. Pro komunikaci byly vytvořeny eventy zajišťující tyto funkce:

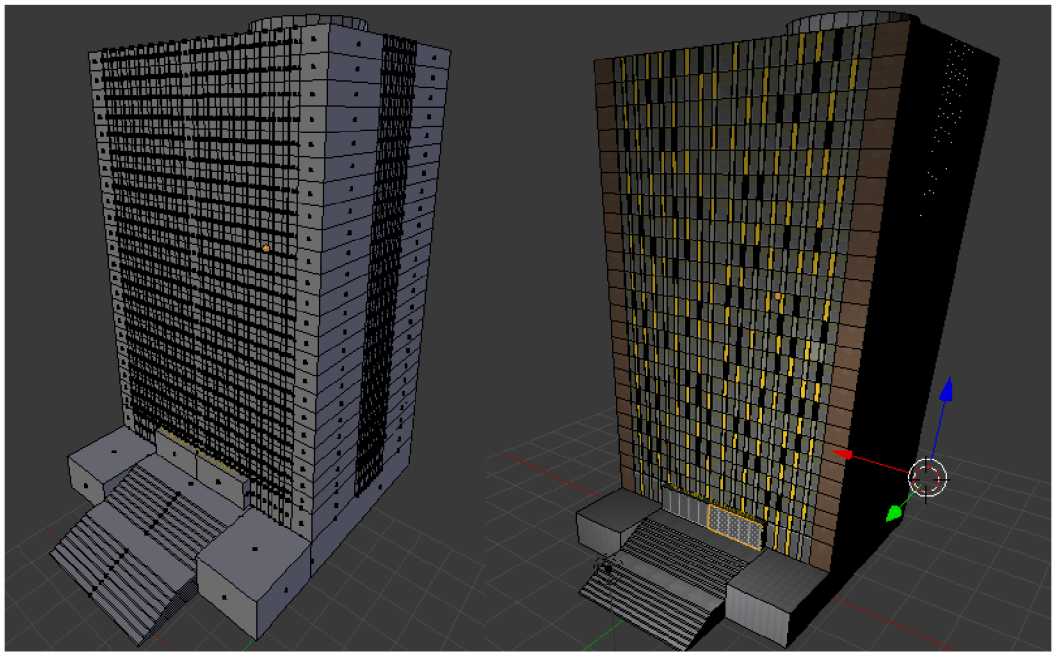
- Úspěšné přihlášení - *ClientEvents_Login*()
- Přihlašování a odhlašování online hráčů - *ClientEvents_LogPlayersEvent*()
- Pohyb online postav - *ClientEvents_OnMovingEvent()*
- Chat - *ClientEvents_ChatMessageEvent*()

Scéna Login s přihlašovacím klientem je spuštěna také po ukončení závodní hry. V tomto případě se přihlášení provede pomocí informací, uložených v objektu třídy *PassingObject* při spuštění závodní hry. Po odeslání metody pro přihlášení je spuštěn časovač bTimer po jehož uplynutí je načten herní svět.

```
private void StartFromGame()
{
  GameObject obj;
  GameObject[] arayOfGObj =
     GameObject.FindGameObjectsWithTag("passTag");
  PassingObj passObj;
  if (arayOfGObj.Length != 0)
   {
      obj = arayOfGObj[0];
      passObj = obj.GetComponent<PasssingObj>();
      myPeer.Login(passObj.PlayerLoginName,
          passObj.PlayerPassword);
      bTimer.Enabled = true;
    }
}
private void OnTimedEventB(object source, ElapsedEventArgs e)
    {
        DontDestroyOnLoad(passingObject);
        Application.LoadLevel("HerniSvet");
        bTimer.Enabled = false;
    }
```

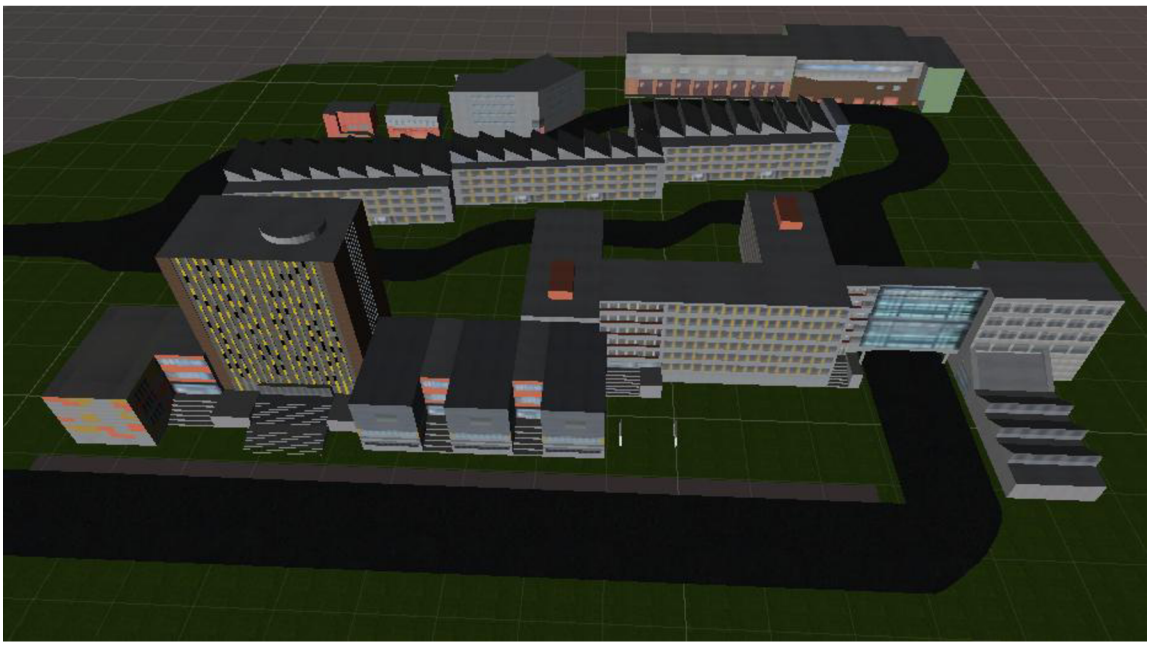
4.5 Herní svět

Herní svět se odehrává ve scéně HerniSvet. Jako předloha pro herní svět byl zvolen areál FSI VUT v Brně. Modely byly postupně vytvořeny v programu Blender, následně na ně byly namapovány textury a nakonec složeny do jednoho výsledného modelu. Na Obr. 25 je ukázka modelu budovy A1, kde levá část zachycuje model rozdělený na plochy pro textury a na pravé části obrázku je model otexturovaný.



Obr. 25 Model budovy A1

Po importování modelu do Editoru Unity3D je vytvořena složka Materials, ve které jsou umístěny všechny materiály, které se na modelu vyskytují. Pro jistotu použití správných textur na materiály jsou všechny textury importovány do složky Textures. Model lze do scény vložit přímo přetažením, nebo jako strukturu Prefab. Jedná se o strukturu používanou pro herní objekty, která umožňuje přidat objektu komponenty ještě před vložením do scény. Používá se např. pro objekty, které se ve scéně vyskytují několikrát a přiřazování stejných komponent by bylo zdlouhavé a složité. Jelikož samotný model má pouze komponentu MeshCollider, tak byl do scény vložen přímo. Jedinou úpravou bylo zaškrtnutí výpočtu nového kolizního modelu. Model nahraný do scény je na Obr. 26.



Obr. 26 Model areálu FSI

4.5.1 Osvětlení

Pro nasvětlení herního světa byl použit světelný zdroj typu Directional Light, na který byl aplikován efekt slunečních paprsků z balíčku Light Flares (konkrétně paprsky 50 mm Zoom) a dále bylo nastaveno jemné stínování Soft Shadows. Pro větší realističnost byl tento zdroj použit v kombinaci s metodou SkyBox. Podstatou SkyBoxu je vytvoření pozadí scény a tím i její optické zvětšení a simulace oblohy.

Obr. 27 Vytvořené slunce pro SkyBox určený jako soumrak a svítání

Pro herní svět byly použity tři různé SkyBoxy, které se mění spolu s intenzitou osvětlení v závislosti na denním čase. Rozdělení dne a přiřazení Skyboxu k jednotlivým částem je v Tab. 1. Realizace je provedena ve třídě *Lighting*, která je přiřazena světelnému zdroji.

Tab. 1

Čas dne	SkyBox
06:00 – 08:00	Rano_vecer
08:00 – 20:00	Den
20:00 – 22:00	Rano_vecer
22:00 – 06:00	Noc

Třída *Lighting* obsahuje funkci *Awake()* pro inicializaci a počáteční nastavení SkyBoxu, funkci *ChangeSkyBox()* pro nastavení příslušného SkyBoxu a změny intenzity osvětlení a funkci *Update()*, ve které je předešlá funkce neustále volána. Při přímém nastavení příslušného SkyBoxu podle času by se zbytečně zvýšily nároky na výpočet, proto se SkyBox nastaví pouze pokud je splněna podmínka období dne a pokud už tento SkyBox nastaven není. Tím je docíleno pouze přepnutí při přechodu mezi obdobími dne.

```
private void ChangeSkybox()
    {
      dayTime = DateTime.Now.ToString("HH:mm:ss");
      time = dayTime.Split(':');
      hours = Convert.ToInt32(time[0]);

      if( 8 <= hours && hours <= 19)
       {
        if (RenderSettings.skybox != day)
         {
          RenderSettings.skybox = day;
          lightSource.transform.eulerAngles = new
Vector3(46.0f, -122.0f, -135.0f);
          light.intensity = 0.8f;
          }
        }

        else if(hours >= 22 || hours <= 5)
        {
         if (RenderSettings.skybox != night)
          {
           RenderSettings.skybox = night;
           lightSource.transform.eulerAngles = new
Vector3(50.0f, -182.0f, -184.0f);
           light.intensity = 0.1f;
           }
         }

        else
        {
         if (RenderSettings.skybox != duskDawn)
          {
           RenderSettings.skybox = duskDawn;
           lightSource.transform.eulerAngles = new
Vector3(8.5f, -28.0f, -27.6f);
            light.intensity = 0.5f;
            }
        }
      }
```

4.5.2 Kamera

Realizace kamery z pohledu třetí osoby je zajištěna scriptem PlayerCamera, který je přiřazen k objektu kamery. Aby kamera sledovala postavu hráče, je nutné pracovat s jeho souřadnicemi. Pro přístup k nim slouží proměnná myPlayer, deklarovaná jako public aby ji mohla byt přiřazena hodnota z jiné třídy. Funkčnost kamery je realizována ve funkcích *CameraControl()* a *CameraZoomPosition()*, které jsou volány s každým framem funkcí *Update()*.

Funkce *CameraControl()* zpracovává otáčení kamery a její průběh je ovlivněn hodnotou boolovské proměnné camLock a zaznamenáním stisknutí tlačítka myši - struktura GUIUtility.hotControl. Pokud uživatel stiskne a drží tlačítko myši, testuje se, zda je povoleno otáčení kamery (camLock = false). Pokud ano, kamera se otáčí podle pohybu myši. Pokud ne, natočení kamery je určeno natočením postavy, stejně jako v případě že

proměnná camLock je nastavena na hodnotu true. Toto řešení je realizováno ve funkci *RotateBehind()*. Funkce *CorrectAngle()* zajišťuje, aby při volném natáčení kamery žádné natočení nepřesáhlo hodnotu 360°.

```
private void CameraControl()
    {
    if (GUIUtility.hotControl == 0)
     {
      if (Input.GetMouseButton(0) || Input.GetMouseButton(1))
       {
        if (!camLock)
         {
        xDeg += Input.GetAxis("Mouse X") * xRotation * 0.02f;
        yDeg += Input.GetAxis("Mouse Y") * yRotation * 0.02f;
          }
        else
         {
           RotateBehind();
          }
        }
       else if (Input.GetAxis("Vertical") != 0 ||
Input.GetAxis("Horizontal") != 0 || camLock)
            {
              RotateBehind();
            }
        }
        yDeg = CorrectAngle(yDeg, yMinLimit, yMaxLimit);
        rotation = Quaternion.Euler(yDeg, xDeg, 0);
       }
```

Funkce *CameraZoomPosition()* nastavuje polohu kamery ve vztahu k postavě hráče v závislosti na přiblížení (omezeno hodnotami minZoom a maxZoom). Také obstarává úpravu pozice kamery, pokud by její objekt kolidoval s jiným objektem (ukázka níže). V tom případě se objekt kamery oddálí od kolidovaného objektu o hodnotu camBorder. Přibližování a oddalování kamery je ovládáno kolečkem myši a hodnotou koeficientu zoomSpeed.

```
if (Physics.Linecast(playerPosition, camPosition, out
     colision, colisionDetect))
 {
  calculatedDistance = Vector3.Distance(playerPosition,
   colision.point) - camBorder;
  corrected = true;
 }
```

4.5.3 Postava hráče

Pro zastoupení hráče v herním světě byly vytvořeny modely postav muže a ženy, přičemž hráč si při registraci zvolí, kterou postavou chce být zastoupen. Tato volba se v dalším průběhu nedá změnit. Stejné modely byly použity také pro zastoupení online hráčů. Tyto modely byly naimportovány do projektu a následně z nich byly vytvořeny prefaby pro hráče i online hráče.

Obr. 28 Vytvořené modely postav

Prefabu postavy hráče byla přiřazena komponenta Animator pro animační sekvenci pohybu, Character Controller pro kontrolu pohybu a scripty PlayerAnimations a PlayerMove se stejnojmennými třídami. Pro vytvoření kolizního modelu byla použita možnost vypočítání nového Mesh Collideru na základě souřadnicového systému Unity Editoru.

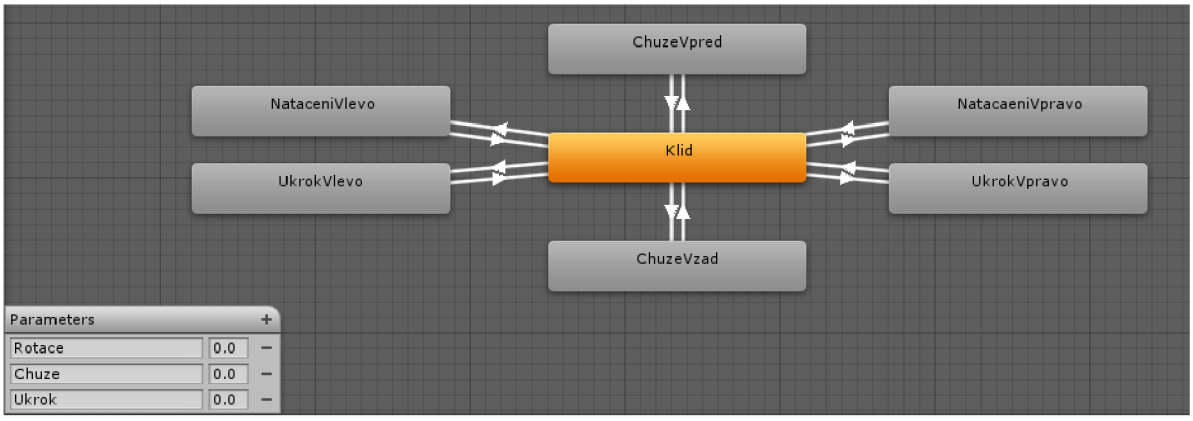
Třída *PlayerMove* se věnuje pohybu postavy a odeslání informace o pohybu ostatním online hráčům. Odeslání je realizováno předáním směrového vektoru pohybu postavy serveru, který jej rozešle všem připojeným hráčům. Vytvoření tohoto vektoru se provádí ve funkci *CharacterMove()*, která je volána funkcí *Update(),* pokud hráč nechatuje. Pohybový vektor je proměnná typu Vector3 a skládá se ze tří parametrů. První a poslední parametry jsou nastavovány podle vstupu z klávesnice a prostřední hodnota je vždy nulová, takže se postava pohybuje v rovině. Při kolizi s překážkou, v závislosti na velikosti překážky, na ni buď vystoupí nebo se o ni zastaví. Tento vektor je vynásoben koeficientem rychlosti pohybu movespeed a předán jako vstupní argument funkci *CharacterController.Move()*. Ta pracuje s přidělenou komponentou Character Controller.
Pro přístup ke komponentě je nejdříve nutné vytvořit proměnnou stejného typu a následně ji jako hodnotu přiřadit komponentu daného objektu. Níže uvedená struktura funkce *Start()* je pouze ukázka inicializace proměnné jako komponenty a nejedná se o přesnou strukturu funkce *Start()* ze třídy *PlayerMove*.

```
CharacterController controller;
void Start ()
     {
      controller = GetComponent <CharacterController>();
     }
private void CharacterMove()
 {
  myMoveVector = new Vector3((Input.GetMouseButton(1) ?
     Input.GetAxis("Horizontal") : 0), 0,
     Input.GetAxis("Vertical"));

  if (Input.GetMouseButton(1) && Input.GetAxis("Horizontal")
     != 0 && Input.GetAxis("Vertical") != 0)
   {
     myMoveVector *= 0.7f;
    }

  myMoveVector = transform.TransformDirection(myMoveVector);
  myMoveVector *= moveSpeed;

  if (Input.GetMouseButton(1))
   {
    transform.rotation = Quaternion.Euler(0,
     Camera.main.transform.eulerAngles.y, 0);
    }
  else
   {
    transform.Rotate(0, Input.GetAxis("Horizontal") *
     rotationSpeed * Time.deltaTime, 0);
    }

  controller.Move(myMoveVector * Time.deltaTime
 }
```

Pokud se uživatel pohybuje – myMoveVector není nulový, tak je volána funkce *MoveType()*, která v závislosti na vstupech určí, o jaký typ pohybu se jedná. Toto vyhodnocení je důležité pro animování vlastního pohybu na straně online hráčů a pro přeposlání typu je použita proměnná typu PlayerDirection. Tento typ je vytvořen v komunikační knihovně speciálně pro tento účel. Poté se zpracuje funkce *SendMove()* jejíž vstupní argument je okamžitá hodnota pohybového vektoru. Tato funkce převede vektor do proměnné pro komunikaci typu PlayerVector3 a spolu se současnou pozicí a typem pohybu odešle na server s kontrolní hodnotou true. Pokud se uživatel přestane pohybovat, je zavolána funkce *EndMove()*. V tomto zavolání se na server odešle informace o ukončení pohybu spolu s pozicí uživatele. Posílání pozice při ukončení pohybu bylo použito z důvodu odstranění možných nepřesností při zobrazení pohybu na straně online hráče přímým přesunutím na pozici, na které byl ukončen pohyb (popsáno v 4.5.4). Při ovládání natočení postavy pomocí myši je ve funkci *Update()* zajištěno skrytí kurzoru myši.

Pro animování postavy při pohybu byl stažen a importován zdarma dostupný balíček animací Raw Mocap Data z Unity Asset Store. V projektu byl ponechán celý balíček, i když byly použity jen čtyři animace. Jedna pro chůzi vpřed, další pro chůzi vzad a zbývající pro

otočení vlevo a úkrok vlevo. Animace úkroku a otočení vpravo je realizována zrcadlením levých animací. Na Obr. 29 je zobrazen Animator Controller pro hráčské postavy.



Obr. 29 Animator Controller postavy hráče

Spouštění jednotlivých bloků je řízeno hodnotami proměnných Rotace, Chuze a Ukrok, které jsou nastavovány ve třídě *PlayerAnimations*. Nastavování hodnot je realizováno funkcí *SetAnimation()*, která je volána ve funkci *Update()*, pokud hráč nechatuje. Jako hodnoty proměnných se nastavují hodnoty získané čtením vstupů od uživatele.

```
private void SetAnimation()
{
  rotation = Input.GetMouseButton(1) ? Input.GetAxis("Mouse
     X") : Input.GetAxis("Horizontal");
  walk = Input.GetAxis("Vertical");
  animator.SetFloat("Rotace", rotation);
  animator.SetFloat("Chuze", walk);

  if (Input.GetMouseButton(1) &&  Input.GetAxis("Horizontal")
     != 0)
   {
     sideStep = Input.GetAxis("Horizontal");
     animator.SetFloat("Ukrok", sideStep);
   }
  else
   {
     sideStep = 0;
     animator.SetFloat("Ukrok", sideStep);
   }
}
```

Pro animaci pohybu online postav byl použit stejný blokový model, ale s rozdílným spouštěním animací (viz podkapitola 4.5.4).

4.5.4 Online hráči

Pro zastoupení online hráčů byly použity prefaby, vytvořené ze stejných modelů jako prefaby pro vlastní postavu a byly opatřeny komponentami Animator, Character Controller a script se třídou *OnlinePlayerScript*.

Pohyb online postav je realizován pomocí pohybového vektoru, který je přijímán od serveru. K tomu slouží event *ClientEvents_OnMovingEvent()* ve třídě *Login*. V tomto

eventu jsou zpracována data, která hráč odesílá na server při pohybu. Pokud se jedná o data pohybujícího hráče, tak se přijatý pohybový vektor a typ pohybu předá objektu třídy *OnlinePlayerScript* a spustí se zpracování tohoto pohybu – nastavení proměnné moving na hodnotu true. Jestliže se jedná o data odeslaná při ukončení pohybu, porovná se pozice postavy online hráče s přijatou pozicí a pokud se tyto hodnoty neshodují, postava je přemístěna na přijatou pozici.

```
private void ClientEvents_OnMovingEvent(string player, bool
     move, ClientToServer.PlayerClass.PlayerVector3 position,
     ClientToServer.PlayerClass.PlayerVector3 movingVector,
     ClientToServer.Operations.PlayerDirection direction)
{
  movingPlayer = GameObject.Find (player);
  ops = movingPlayer.GetComponent<OnlinePlayerScript> ();
  switch (move)
   {
    case true:
      ops.moving = true;
      Vector3 smerPohybu = new Vector3
          (movingVector.X,movingVector.Y,
          movingVector.Z);
      ops.moveDirection = smerPohybu;
      ops.direction = direction;
       break;
    case false:
      ops.moving=false;
      Vector3 endMovePosition = new Vector3
          (position.X , position.Y, position.Z);
      if(endMovePosition != movingPlayer.transform.position)
       {
        movingPlayer.transform.position = endMovePosition;
        }
       break;
    }
}
```

Třída *OnlinePlayerScript* je tvořena funkcemi *Move()* a *ResetAnimation()*. Ve funkci *Update()* je volána funkce *ResetAnimation()*, která pokud se daná postava pohybuje, volá funkci *Move()*. Pokud se postava nepohybuje tak jsou resetovány všechny proměnné pro spouštění animací. Tyto proměnné nejsou typu float jako v případě postavy hráče, nýbrž se jedná o hodnoty typu bool. Funkce *Move()* obstarává pohyb postavy online hráče a spouštění animací v závislosti na hodnotách proměnných, které byly obdrženy z eventu *ClientEvents_OnMovingEvent*.

```
private void Move()
 {
   switch (direction)
     {
       case PlayerDirection.forward:
          animator.SetBool("ChuzeVpred",true);
          transform.LookAt(moveDirection);
          break;
```

```
        case PlayerDirection.backward:
           animator.SetBool("ChuzeVzad",true);
           break;
        case PlayerDirection.left:
           if(moveDirection.magnitude > 0)
            {
              animator.SetBool("UkrokL",true);
            }
           else
            {
              animator.SetBool("OtoceniL",true);
              transform.Rotate(0, -120.0f * Time.deltaTime,0);
            }
           break;
        case PlayerDirection.right:
           if(moveDirection.magnitude > 0)
            {
              animator.SetBool("UkrokP",true);
            }
           else
            {
              animator.SetBool("OtoceniP",true);
              transform.Rotate(0, 120.0f * Time.deltaTime,0);
            }
           break;
      }

   controller.Move (moveDirection * Time.deltaTime);
 }
```

Zobrazení přezdívky online hráče nad jeho postavou je realizováno pomocí objektu typu GUI Text. Jedná se objekt pro zastoupení textového pole ve scéně a je mu přiřazena komponenta stejného typu a script NickName se třídou *NickName*. V této třídě je zpracováno umístění objektu a nastavení obsahu textového pole. Poloha objektu kamery se odvíjí od souřadnic postavy, ke které patří, ale je přepočítána do souřadnicového systému, který používá kamera. Pokud se postava online hráče nachází v záběru kamery, textové pole se zobrazí nad ní, pokud ne tak je textové pole zobrazeno při okraji obrazovky ve směru umístění postavy. Přepočet pozice objektu je třeba provádět neustále, proto je umístěn ve funkci *Update()*.

```
void Update() {
  relativePosition = camTransform.InverseTransformPoint(
     targetOnlinePlayer.position);
  relativePosition.y = relativePosition.y + 2.5f;
  relativePosition.z =  Mathf.Max(relativePosition.z, 1.0f);
thisTransform.position =
cam.WorldToViewportPoint(camTransform.TransformPoint(
     relativePosition));
thisTransform.position = new
     Vector3(Mathf.Clamp(thisTransform.position.x,
clampBorderSize,    1.0f - clampBorderSize),
Mathf.Clamp(thisTransform.position.y,  clampBorderSize, 1.0f
- clampBorderSize),thisTransform.position.z);
}
```

4.5.5 Chat

Komunikace mezi hráči je zpracována třídou *ChatWindow*, přiřazenou k objektu kamery a eventem *ClientEvents_ChatMessageEvent()* ze třídy *Login*. V eventu jsou zpracovány zprávy od ostatních hráčů, které jsou rozděleny na veřejné a soukromé a zpřístupní je pomocí boolovských proměnných třídě *ChatWindow*.

```
private void ClientEvents_ChatMessageEvent(string
     StartPlayer, string message, string EndPlayer)
 {
  if (EndPlayer == "")
   {
     publicMessage = true;
     chat = StartPlayer + ": " + message;
    }
  else
   {
     privateMessage = true;
     chat = "Soukroma od :" + StartPlayer + ": " + message;
     }
 }
```

Ve třídě *ChatWindow* je zpracováno zobrazení odeslaných a přijatých zpráv do okna chatu, které je umístěno v levém dolním rohu obrazovky. Zobrazení je prováděno do oblasti GUI.TextField, pod kterou je umístěno pozadí pomocí GUI.Box. Otevření řádku pro odeslání zprávy a následné odeslání se provádí stisknutím klávesy Enter. Pokud je chat aktivní, tak je proměnná chatting nastavena na hodnotu true. Tato proměnná slouží k omezení čtení vstupů z klávesnice v ostatních třídách – postava se nebude pohybovat při chatování a nebudou se spouštět animace.

Odesílání zpráv je realizováno funkcí *SendChatMessage()*. Jako systém rozlišení veřejné zprávy od soukromé bylo použito porovnání textu zprávy při odeslání se jmény všech online hráčů a pokud je nalezena shoda, text zprávy je vynulován a očekává se nový text zprávy, který je s dalším stisknutím Enter odeslán soukromě danému online hráči. Pokud toto porovnání nenalezne shoda, je původní text odeslán všem online hráčům jako veřejná zpráva.

```
private void SendChatMessage()
 {
   if (chatting)
    {
     foreach (Player p in login.ListOfPlayers)
      {
        if (messageText == p.Name)
         {
           chatting = false;
           privateChatPlayer = p.Name;
           messageText = "";
           sendPrivate = true;
         }
      }
```

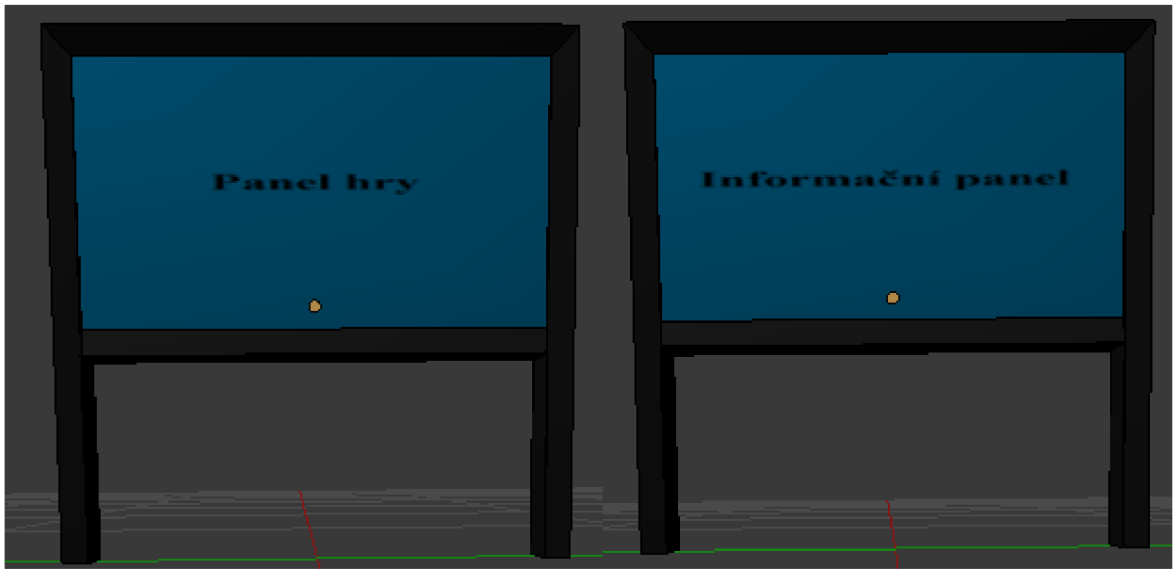
```
        if (privateChatPlayer == "")
         {
          messages.Add(login.playerName + ":" + messageText);
          login.myPeer.SendChatMessage(login.playerName,
             messageText);
          messageText = "";
          i = 0;
           }
        }
      if (sendPrivate)
       {
        if (i == 3)
         {
          messages.Add(privateChatPlayer + "=>:" + messageText);
          login.myPeer.SendChatMessage(login.playerName,
             messageText, privateChatPlayer);
          messageText = "";
          sendPrivate = false;
          privateChatPlayer = "";
          i = 0;
          }
         }
    }
```

Funkce *ReceiveMessage()* zajišťuje zpracování příchozích zpráv a jejich zobrazení v textovém poli chatu. Pracuje s hodnotami boolovských proměnných publicMessage a privateMessage ze třídy *Login*, které nabývají hodnoty true při přijetí veřejné nebo soukromé zprávy. Volání této funkce probíhá ve funkci *Update()*. Volání funkce *SendChatMessage()* je realizováné také z *Update()* ale provede se pouze při stisknutí Enter.

```
private void ReceiveMessage()
    {
        if (login.publicMessage)
        {
            messages.Add("" + login.chat);
            login.chat = "";
            login.publicMessage = false;
        }

        if (login.privateMessage)
        {
            messages.Add("" + login.chat);
            login.chat = "";
            login.privateMessage = false;
        }
 }
```

4.5.6 Informační a herní panel

Interaktivní panely jsou zastoupeny stejným objektem s rozdílnou texturou na jedné ploše (Obr. 30) a jsou umístěny poblíž počáteční pozice. Fungování informačního panelu zajišťuje třída *PanelInfo* a panelu pro hru třída *PanelGame*.



Obr. 30 Objekty pro interaktivní panely

Funkčnost obou panelů je řešena přes UI pro každý panel. Společnou vlastností je podmínka pro spuštění panelu – určitá vzdálenost postavy hráče od panelu. Tuto podmínku zpracovává funkce *PanelTurnOnOff()*, která je volána z funkce *Update()* a její vstupní parametr je vektor vzdálenosti hráče od panelu.

```
private void PanelTurnOnOff(Vector3 PlayerDistance)
  {
        if (PlayerDistance.magnitude < 4)
        {
            turnOn = true;
        }
        else
        {
            turnOn = false;
            visible = false;
        }
   }
```

Pokud se uživatel nachází v daném rozsahu vzdálenosti od panelu, tak jeho spuštění provede kliknutím na něj. Po spuštění panelu s informacemi se je zobrazeno jednoduché UI s tlačítkem pro ukončení činnosti panelu a tlačítkem pro přechod na obrazovku s informacemi o areálu FSI VUT v Brně. Text s informacemi je načítán ze souboru obsahPanelu.txt, který není přímou součástí projektu a je proto uložen v kořenové složce projektu, nikoliv ve složce Assets. Načítání správných dat je realizováno funkcí *ArealWindow()*, volané při spuštění herního světa pomocí funkce *Start()*.

```
private void ArealWindow()
 {
    string[] linesplit;

    foreach (string line in File.ReadAllLines(
     "obsahPanelu.txt"))
     {
        if (line.Contains("#areal"))
         {
            linesplit = line.Split(';');
          areal = areal + Environment.NewLine + linesplit[1];
          }
      }
    GUI.Label(new Rect(100, 290, 600, 50), areal);
 }
```

UI panelu závodní hry je tvořeno tlačítkem pro připojení se ke hře (pokud připojen jsem tak je toto tlačítko nahrazeno tlačítkem pro odpojení) a tlačítkem pro zavření. Při spuštění panelu se vytvoří komunikace se serverem závodní hry pomocí objektu třídy *MyRaceGamePeer*. Po stisknutí tlačítka pro připojení ke hře je hráč vložen do seznamu hráčů čekajících na hru. Jakmile budou připojeni čtyři hráči, aktivuje se event *RaceGameEvents_OnStartGame()*. V tomto eventu se uloží data připojeného hráče do objektu třídy *PassingObj*, uzavře se spojení se serverem herního světa a spustí se scéna závodní hry.

```
private void RaceGameEvents_OnStartGame(string serverTime)
    {
      obj = new GameObject();
      obj.transform.position = new Vector3(0.0f, 0.0f, 0.0f);
      obj.AddComponent("PasssingObj");
      obj.tag = "passTag";
      passObj = obj.GetComponent<PasssingObj>();
      passObj.RacePeer = rgPeer;
      passObj.PlayerName = loginScript.playerName;
      passObj.PlayerLoginName = loginScript.login;
      passObj.PlayerPassword = loginScript.password;
      passObj.racers = _racers.ToArray();
      DontDestroyOnLoad(obj);
      loginScript.myPeer.Disconnect();
      Application.LoadLevel("ZavodniHra");
     }
```

4.5.7 Funkčnost herního světa

Funkce ovlivňující herní svět, včetně UI herního menu, jsou umístěny ve třídě *ControlScript*, přiřazené objektu kamery (třída musí být přiřazena k objektu, který se ve scéně vyskytuje po celou dobu hraní). Ve funkci *Start()* je řešeno vytvoření postav všech hráčů po vstupu do herního světa volání funkce *CreateOnlinePlayer()* pro všechny připojené online hráče a funkce *CreateMyPlayer()* pro vytvoření postavy hráče.

Při vytvoření postavy hráče je objekt této postavy přiřazen jako cíl kamery a ovládací prvek pro oba interaktivní panely. Postava je umístěna na pozici, která byla uložena do databáze při posledním odhlášení ze hry.

```
private void CreateMyPlayer(Vector3 position,Quaternion
     rotation, byte gender )
{
  switch (loginScript.resetPosition)
   {
     case true:
        position = new Vector3(0, 0, 0);
        loginScript.resetPosition = false;
          break;
     case false:
          break;
    }
  switch (gender)
   {
     case 0:
        myPlayer = Instantiate(Resources.Load(
          "Hrac/MaleCharacter"), position,rotation) as
          GameObject;
           break;
     case 1:
        myPlayer = Instantiate(Resources.Load(
          "Hrac/FemaleCharacter"), position, rotation) as
          GameObject;
           break;
    }
  myPlayer.transform.name = loginScript.playerName;
  Transform transform = myPlayer.GetComponent<Transform> ();
  myPlayer.tag = "Player";
  cameraScript.myPlayer = transform;
  panelScript.controlPlayer = transform;
  panelGameScript.controlPlayer = transform;
}
```

Při zavolání funkce *CreateOnlinePlayer()* se vytvoří postava online hráče spolu s objektem textového pole, které zajišťuje zobrazení přezdívky daného hráče nad jeho postavou. Tato funkce je volána také z funkce *Update()* z důvodů vytvoření postav online hráčů, kteří se do hry přihlásí později než hráč. Postavy online hráčů jsou označeny jednotným tagem OnlinePlayer, čehož je poté využito ve funkci pro ulehčení vykreslování.

```
private void CreateOnlinePlayer(string nickName, Vector3
     position, Quaternion rotation, byte gender)
{
   switch (gender)
    {
      case 0:
        onlinePlayer = Instantiate (Resources.Load(
          "OnlineHrac/OnlineCharacterMale"),
          position,rotation)as GameObject;
        break;
      case 1:
        onlinePlayer = Instantiate(Resources.Load(
          "OnlineHrac/OnlineCharacterFemale"),
          position, rotation) as GameObject;
        break;
     }
```

```
GameObject playerNick = Instantiate (Resources.Load (
     "OnlineHrac/Nick"))as GameObject;
 Transform transform= onlinePlayer.GetComponent<Transform>();
 NickName nickScript = playerNick.GetComponent<NickName>();
 onlinePlayer.tag = "OnlinePlayer";
 nickScript.targetOnlinePlayer = transform;
 nickScript.nickName = nickName;
 onlinePlayer.transform.name = nickName;
 playerNick.transform.name = "hrac"+nickName;
}
```

Smazání postav online hráčů, kteří se ze hry odhlásili, je realizováno funkcí *RemovePlayer()*. Odhlášení uživatele je zpracováno v eventu pro práci se seznamem připojených hráčů *ClientEvents_LogPlayersEvent()*, pokud nastane operace Remove. Smázání spočívá v odstranění postavy online hráče a přiděleného objektu textového pole.

Ostatní objekty herního světa jsou do scény přidány pomocí třídy *GenerateObjects*. Pro snížení počtu vykreslovaných objektů obsahuje třída *ControlScript* funkci *SetRender()* volanou ve funkci *Update()*. V té jsou načteny všechny postavy online hráčů do pole objektů OnlinePlayers a ostatní objekty jako stromy a keře do pole TreesObjects. Následně je vypočítána vzdálenost každého objektů z obou polí od postavy hráče a při velké vzdálenosti se daný objekt nevykresluje. Protože model postavy a modely stromů jsou vytvořeny z velkého počtu vertexů, jsou rozděleny Unity Editorem na několik částí a je nutné vypnout vykreslování všech částí.

```
private void SetRender()
{
 Vector3 distance;
 GameObject myPlayerObject = GameObject.FindGameObjectWithTag
     ("Player");
GameObject[]OnlinePlayers = GameObject.FindGameObjectsWithTag
     ("OnlinePlayer");
 GameObject[]TreesObjects = GameObject.FindGameObjectsWithTag
     ("Tree");

 foreach(GameObject playerObject in OnlinePlayers)
   {
      distance = myPlayerObject.transform.position -
          playerObject.transform.position;
      MeshRenderer[] renderers =
playerObject.GetComponentsInChildren<MeshRenderer>();

      foreach (MeshRenderer renderer in renderers)
         {
           if (distance.magnitude > 35)
              {
                  renderer.enabled = false;
               }
           else
              {
                  renderer.enabled = true;
               }
         }
    }
```

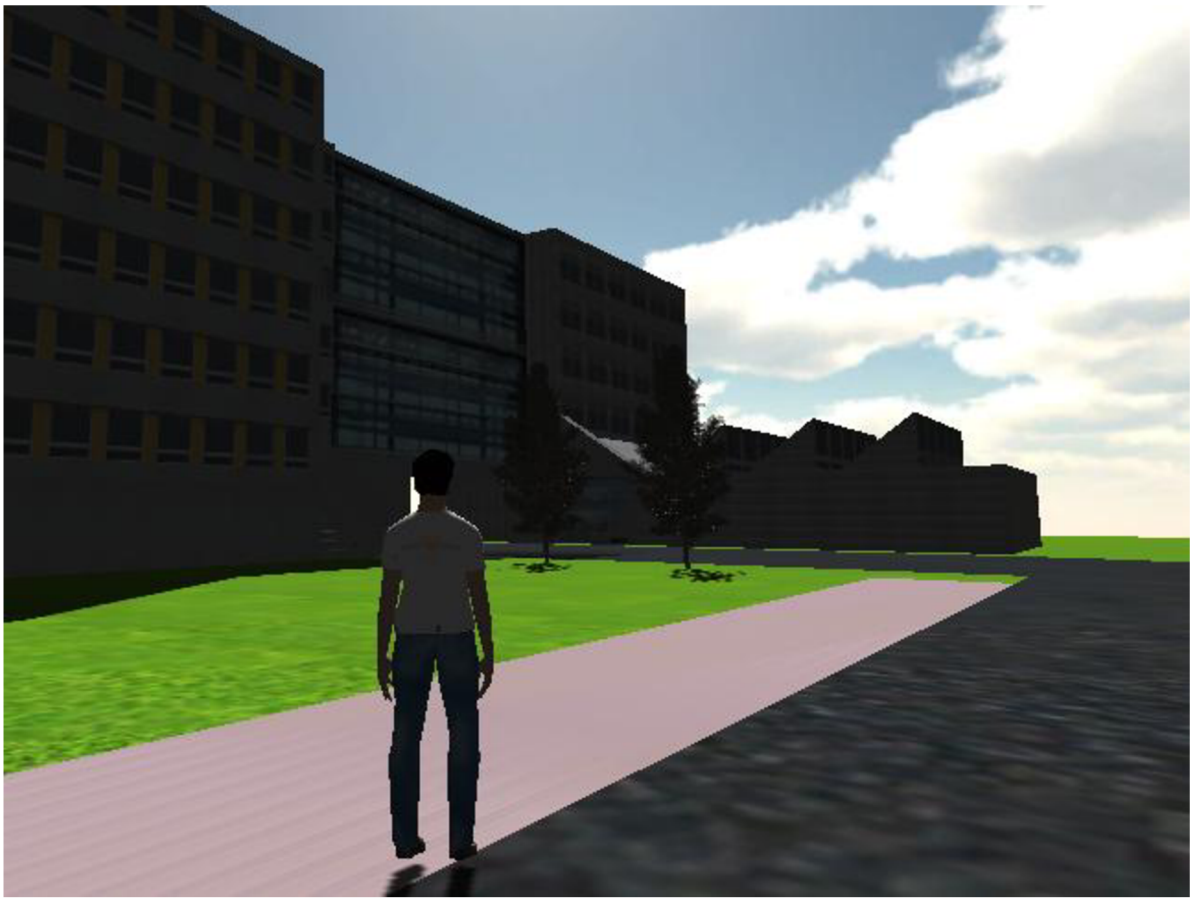
```
foreach (GameObject treeObject in TreesObjects)
  {
    distance = myPlayerObject.transform.position -
          treeObject.transform.position;

    MeshRenderer[] renderers =
     treeObject.GetComponentsInChildren<MeshRenderer>();

    foreach (MeshRenderer renderer in renderers)
       {
           if (distance.magnitude > 10)
            {
              renderer.enabled = false;
            }
           else
            {
              renderer.enabled = true;
            }
       }
   }
}
```

UI herního menu je vytvořené pomocí funkce *OnGUI()* a jeho aktivace se provede stisknutím klávesy Escape. Součástí herního menu je obrazovka s popisem ovládání a možností zamknout/odemknout kameru a možnost přesunutí postavy na počáteční pozici.

4.6 Ukázky ze hry

Obr. 31 Registrační formulář přihlašovacího klienta

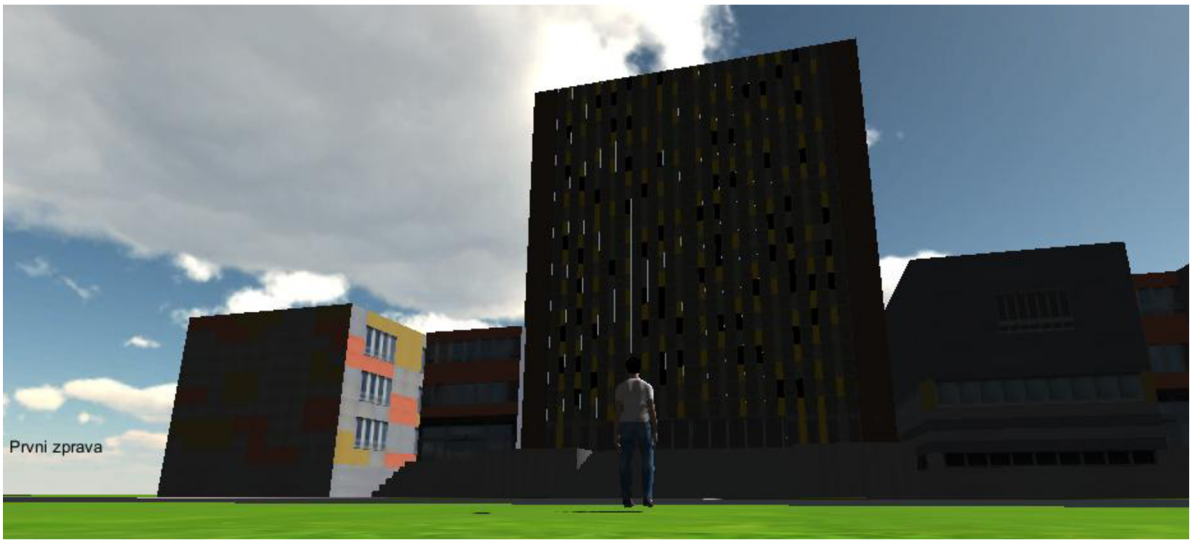
Obr. 32 Ukázka herního světa



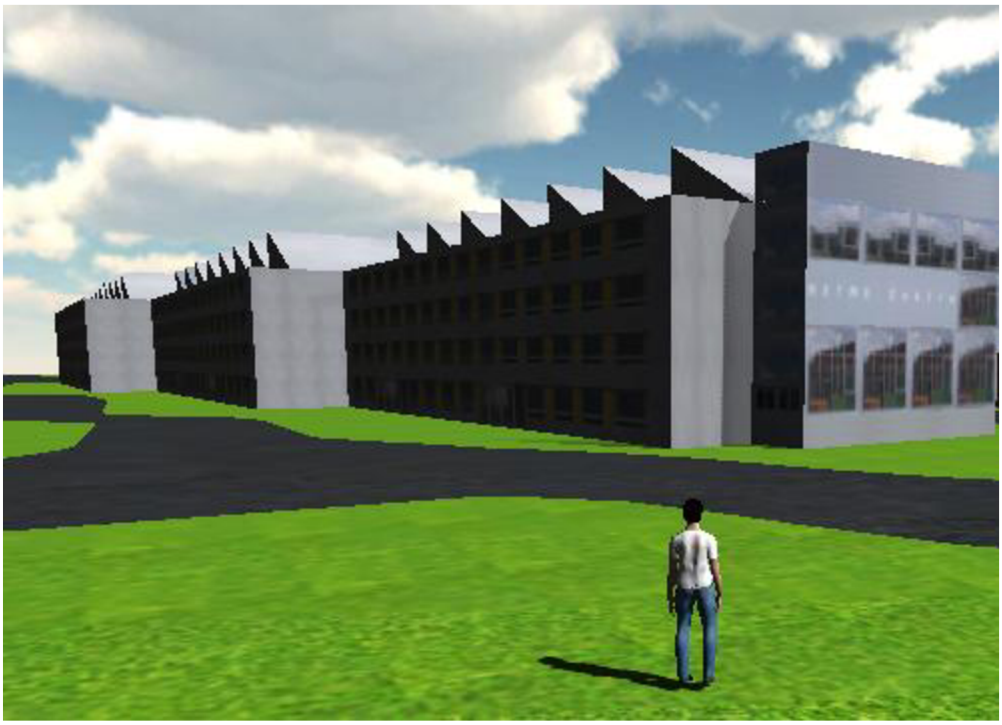
Obr. 33 Ukázka herního světa

4 Vytvoření virtuálního světa

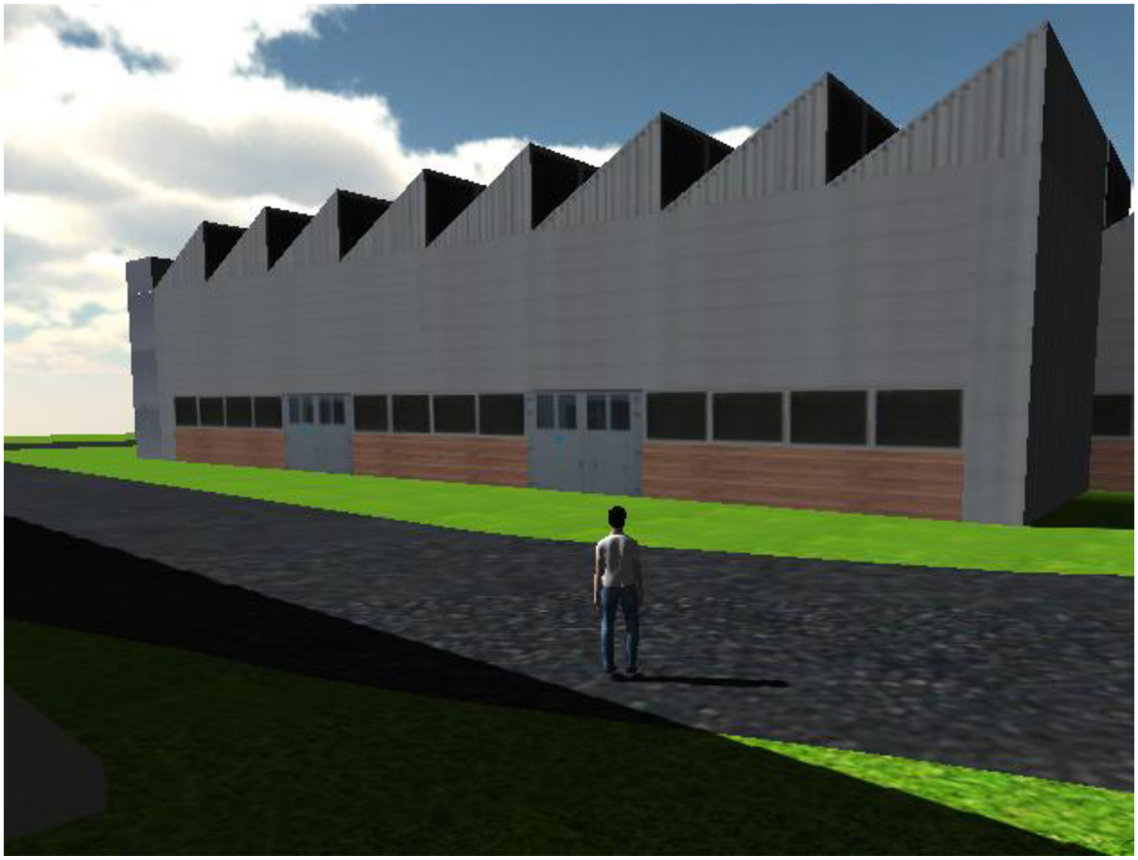
Obr. 34 Ukázka herního světa



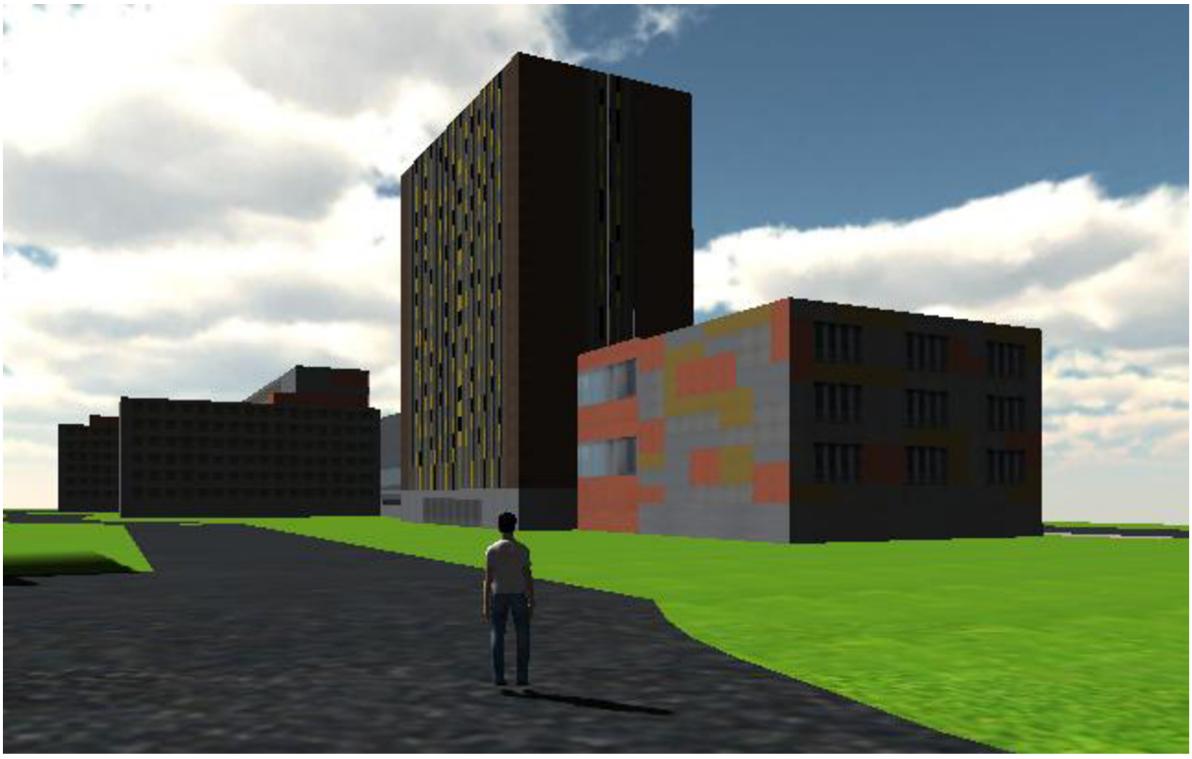
Obr. 35 Ukázka herního světa



Obr. 36 Ukázka herního světa

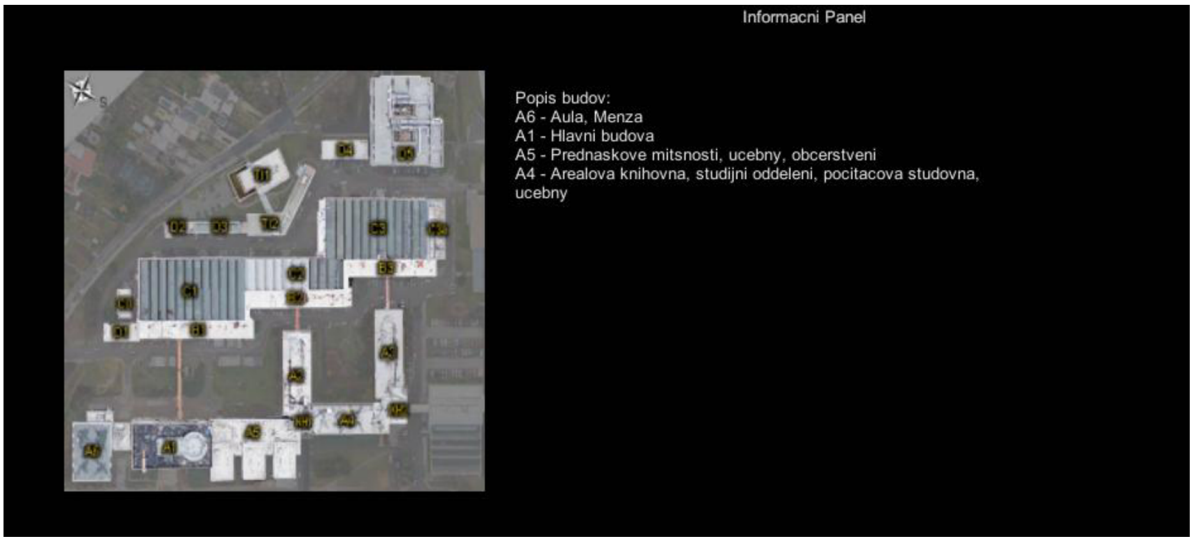


Obr. 37 Ukázka herního světa



Obr. 38 Ukázka herního světa

Obr. 39 UI popisu ovládání



Obr. 40 UI popisu herního světa

4 Vytvoření virtuálního světa

5 Závěr

Podstatou této práce je přiblížení problematiky tvorby počítačových her a získané poznatky využít při vytvoření ukázkové aplikace. V první kapitole je tato problematika objasněna na jednoduchém postupu tvorby počítačové hry. Druhá kapitola je věnována hernímu enginu Unity3D a dalšímu softwaru, který byl použit při vytváření aplikace. Ve třetí kapitole je popsáno vytvoření virtuálního prostředí počítačové hry s použitím postupu tvorby počítačové hry z kapitoly 2. Vytvořené prostředí bylo použito jako základní herní svět pro projekt, jehož cílem bylo vytvořit online počítačovou hru pro více hráčů s názvem Virtual World, na které jsem spolupracoval se studenty Danielem Balcárkem a Robertem Kováčem. Z hlediska budoucího vývoje je možné rozvíjet vytvořený herní svět pro projekt Virtual World, nebo při odstranění online prvků je možné použít vytvořenou aplikaci jako virtuální prostředí pro jinou počítačovou hru. Vzhledem k funkčnosti virtuálního prostředí v projektu Virtual World lze považovat cíle práce za splněné.

Seznam použité literatury

[1] AUTOR NEUVEDEN. The Process of Game Creation & the Game Design Document. *Digital Worlds – Interactive Media and Game Design* [online]. 2010 [cit. 2015-03-22]. Dostupné z: https://digitalworlds.wordpress.com/2008/04/10/the-process-of-game-creation-the-game-design-document/

[2] AUTOR NEUVEDEN. Unity - Fast Facts. *Unity - Game Engine* [online]. © 2005- [cit. 2015-04-03]. Dostupné z: http://unity3d.com/public-relations

[3] AUTOR NEUVEDEN. Unity - Manual: Creating Scenes. *Unity - Documentation* [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://docs.unity3d.com/Manual/CreatingScenes.html

[4] AUTOR NEUVEDEN. Unity - Manual: Input. *Unity - Documentation* [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://docs.unity3d.com/Manual/Input.html

[5] AUTOR NEUVEDEN. Unity - Manual: Lighting Overview. *Unity - Documentation* [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://docs.unity3d.com/Manual/Lighting.html

[6] AUTOR NEUVEDEN. Unity - Manual: Creating and Using Scripts. *Unity - Documentation* [online]. 2015 [cit. 2015-04-10]. Dostupné z: http://docs.unity3d.com/Manual/CreatingAndUsingScripts.html

[7] AUTOR NEUVEDEN. Unity - Manual: Unity's Animation System. *Unity - Documentation* [online]. 2015 [cit. 2015-04-11]. Dostupné z: http://docs.unity3d.com/Manual/AnimationOverview.html

[8] AUTOR NEUVEDEN. Unity - Manual: Colliders. *Unity - Documentation* [online]. 2015 [cit. 2015-04-12]. Dostupné z: http://docs.unity3d.com/Manual/CollidersOverview.html

[9] AUTOR NEUVEDEN. Unity - Manual: Rigidbody. *Unity - Documentation* [online]. 2015 [cit. 2015-04-13]. Dostupné z: http://docs.unity3d.com/Manual/class-Rigidbody.html

[10] AUTOR NEUVEDEN. Unity - Manual: Character Controller. *Unity - Documentation* [online]. 2015 [cit. 2015-04-13]. Dostupné z: http://docs.unity3d.com/Manual/class-CharacterController.html

[11] AUTOR NEUVEDEN. Unity - GUI, *Unity - Documentation* [online]. 2015 [cit. 2015-04-16]. Dostupné z: http://docs.unity3d.com/Manual/gui-Basics.html

[12] AUTOR NEUVEDEN. MakeHuman. *Wikipedia* [online]. 2007 [cit. 2015-04-22]. Dostupné z: http://en.wikipedia.org/wiki/MakeHuman

[13] AUTOR NEUVEDEN. Blender. *Wikipedia* [online]. 2006 [cit. 2015-04-23]. Dostupné z: http://cs.wikipedia.org/wiki/Blender

[14] AUTOR NEUVEDEN. Textures. *Blender Reference Manual* [online]. 2015 [cit. 2015-04-23]. Dostupné z: https://www.blender.org/manual/render/blender_render/textures/index.html

[15] BLACKMAN, Sue. *Beginning 3D Game Development with Unity 4*. New York: Apress, 2013. 2. ISBN 978-1-4302-4899-6.

[16] THORN, Alan. *Learn Unity for 2D Game Development*. New York: Apress, 2013. 2. ISBN 978-1-4302-6229-9.

Obsah přiloženého DVD

- Elektronická verze této práce
- Instalační soubor editoru Unity3D 4.5.1f3
- Virtual World jako projekt pro Unity3D Editor
- Vyexportovaný projekt Virtual World
- Návod na zprovoznění aplikace