UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

BAKALÁŘSKÁ PRÁCE

2020 Jan Kubík

Univerzita Pardubice
Fakulta elektrotechniky a informatiky

Software pro podporu výuky BOZP v chemickém průmyslu

Jan Kubík

Bakalářská práce

2020

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2018/2019

ZADÁNÍ BAKALÁŘSKÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Jan Kubík**
Osobní číslo:	**I16109**
Studijní program:	**B2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Software pro podporu výuky BOZP v chemickém průmyslu**
Zadávající katedra:	**Katedra informačních technologií**

Zásady pro vypracování

Bude vypracována softwarová pomůcka, která interaktivním způsobem podpoří výuku BOZP na pracovištích chemických provozů. Cílem je vytvořit takový koncept, který bude věcně korektně reflektovat požadavky na výuku BOZP, ale zároveň přinese prvky interaktivity a entuziasmu do výuky.
V rámci teoretické práce bude realizován přehled existujících software a jejich srovnání. Zároveň bude provedena rozprava, která vymezí vhodné softwarové pojetí.
V rámci praktické realizace bude proveden sběr a zpracování požadavků, vytvořena koncepce díla a bude proveden výběr vhodných komplementů pro software. Součástí práce bude též základní popis použitých nástrojů, použitých postupů a výčet použitých komplementů. Nutným požadavkem práce je hratelné demo, které bude rozebráno přímo v práci.

Rozsah pracovní zprávy: **30**
Rozsah grafických prací:
Forma zpracování bakalářské práce: **tištěná**

Seznam doporučené literatury:

SCHELL, Jesse, 2015. The art of game design: a book of lenses. Second edition. Boca Raton: CRC Press. ISBN 978-146-6598-645.
Univerzita Karlova, 2018. 7. Počítačová grafika a vývoj počítačových her [online]. 2013?2018 [cit. 2018-08-25]. Dostupné z: https://www.mff.cuni.cz/studium/bcmgr/ok/i3b27.htm
JIRKOVSKÝ, Jan, 2012. Game industry 2. Praha: D.A.M.O. 240s. ISBN 978-80-904387-3-6.

Vedoucí bakalářské práce: **Ing. Josef Brožek**
Katedra informačních technologií

Datum zadání bakalářské práce: **31. října 2018**
Termín odevzdání bakalářské práce: **12. května 2019**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

Ing. Lukáš Čegan, Ph.D.
pověřený vedením katedry

V Pardubicích dne 14. prosince 2018

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 9/2012, bude práce zveřejněna v Univerzitní knihovně a prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 17. 5. 2020

Jan Kubík

PODĚKOVÁNÍ

Na prvním místě bych chtěl poděkovat své rodině za nedocenitelnou a neutuchající podporu během zpracovávání této práce i během celého bakalářského studia. Taktéž bych velice rád poděkoval Josefu Brožkovi za odborné vedení a cenné rady při zpracování této bakalářské práce.

ANOTACE

Tato bakalářská práce se zabývá vývojem edukativní videohry, jenž je zaměřena na dodržování bezpečnostních pravidel na chemických pracovištích. Začátek této práce je věnován základním pojmům a charakteristice herního enginu Unity, za jehož pomoci byl tento software zhotoven. Následně se práce zaměřuje na popis vývoje videoher, vytváření assetů a posléze na samotnou implementaci. V úplném závěru je taktéž věnována pozornost exportu hry na různé platformy.

KLÍČOVÁ SLOVA

herní vývoj, modelování, herní engine, asset, animace

TITLE

Chemical Safety Education Support Tool

ANNOTATION

This bachelor thesis deals with the development of an educational videogame which is focused on the observance of safety principles in chemical departments. The beginning of this thesis contains a list of basic terms and a characteristic of game engine Unity, which was used for the development of this software. Afterwards, the work is focused on the description of game development, creating game assets and on the very implementation. Attention dedicated to export to multiple platforms is included in the final part.

KEYWORDS

game development, modelling, game engine, asset, animation

OBSAH

SEZNAM ILUSTRACÍ

SEZNAM ZKRATEK

2D	Dvoudimenzionální
3D	Trojdimenzionální
PC	Personal Computer
FPS	First-person shooter
PHP	PHP: Hypertext Preprocessor
OOP	Object Oriented Programming
IDE	Integrated Development Environment
HUD	Head-up display
GCPS	Global Congress on Process Safety
APSAC	4th International Conference on Applied Physics, Simulation and Computing

ÚVOD

Vývoj videoher bývá náročný a lidé mají často malé povědomí o průběhu tohoto procesu, a jaké činnosti s ním souvisí. Jedním z hlavních cílů této bakalářské práce je proto přiblížit postupný vývoj hry a její následný export na další platformy za použití herního enginu *Unity*.

Teoretická část této bakalářské práce si klade za úkol provést čtenáře procesem a jednotlivými fázemi vývoje videoher. Dále zde budou představeny různé nástroje a technologie, které jsou nezbytné pro vývoj herních titulů.

Praktická část práce se zaměřuje na vývoj edukativní hry, která se odehrává v prostředí chemické laboratoře a zaměřuje se na dodržování bezpečnosti. Tato část si klade za cíl demonstrovat nezbytnost dodržování bezpečnostních pravidel a opatření při práci v chemických laboratořích, a to poutavějším a zábavnějším způsobem, než je u této problematiky standardem v případě, že je použita prostá edukace prostřednictvím konvenčního školení.

Čtenář je na začátku práce obeznámen se základními pojmy, které jsou nezbytné pro hlubší proniknutí do problematiky vývoje her. Také zde budou uvedeny podrobnější informace týkající se softwaru Unity 3D, který byl využit pro realizaci praktické části práce. Dále zde bude také detailně rozebrán samotný vývoj her.

Následně práce pojednává o tvorbě praktického software, ve kterém je blíže charakterizována vyvíjená hra. Jsou zde také rozebrány požadavky a další náležitosti, které by zmíněná hra měla splňovat. V následujících kapitolách bude poté podrobně popsána samotná implementace. Je zde taktéž blíže popsáno modelování herních objektů, psaní skriptů a tvorba herních textur. Finálně je přiblížen export na různé platformy, konkrétně se jedná o export pro osobní počítače, mobilní zařízení a webové prohlížeče.

TYPOGRAFICKÉ KONVENCE

Typografický styl této bakalářské práce se primárně řídí jednotnou formální úpravou závěrečných prací, která je uvedena ve směrnici Univerzity Pardubice č. 9/2012. Dále byly užity styly textu a formátování, jež napomáhají zlepšení orientování se v práci.

Výrazy, jež jsou cizího původu nebo představují názvy softwarů či jejich nástroje a další přidružené komponenty, jsou psány kurzívou. Tučným písmem jsou pak značeny významově nadřazené části výčtů. Ukázky zdrojového kódu jsou psány za pomoci neproporcionálního písma *Courier New*.

1 ZÁKLADNÍ POJMY

Tato kapitola přibližuje čtenáři pojmy, které jsou nezbytné k porozumění obsahu této práce. Lze se tu setkat s výrazy, jež se používají zejména v souvislosti s vývojem videoher a programováním.

1.1 Programovací jazyk

Pomocí programovacího jazyku je možné zapisovat instrukce a vytvářet tak programy, které jsou následně zpracovány a prováděny počítačem. Programovací jazyky lze dělit na vyšší a nižší, interpretované a kompilované.

Předností vyšších jazyků je lepší čitelnost a srozumitelnost. Řadí se sem například *C++*, *C#*, *Java*, *PHP*, *JavaScrip*t a mnoho dalších. Nižší programovací jazyky jsou naopak mnohem těžší na pochopení a nejsou tolik přehledné. Je to způsobeno zvláště tím, že pracují na úrovni strojových instrukcí procesoru. Další jejich nevýhoda spočívá v tom, že jsou závislé na použitém druhu procesoru.

Kompilované jazyky, jako jsou například *C*, *C++* a *Java*, musejí být před prvním spuštěním přeloženy tzv. *kompilátorem*. Interpretované jazyky toto nevyžadují, avšak pro své spuštění potřebují tzv. *interpreta*. Jedná se o program, který překládá požadovaný kód až za běhu samotného programu. Jsou značně pomalejší a řadí se sem například *Perl*, *PHP* nebo *BASIC*.

Zdroj [1].

1.2 Herní engine

Jako herní engine je označován software, který obsahuje prvky využitelné pro urychlení a zefektivnění procesu vývoje videoher. Herní enginy např. zajišťují vykreslování 2D či 3D grafiky, aplikování fyziky, detekci kolizí, usnadňují ozvučení, skriptování, tvorbu animací a mají další funkce. Tento software taktéž zjednodušuje přenos na platformy, jako jsou například PC, herní konzole a mobilní zařízení.

Hranice, kde končí herní engine, a kde začíná obsah samotné hry, nemusí být vždy zřetelná. V současnosti se většina herních enginů snaží být co nejvíce univerzální. Přesto se velké množství specializuje na vývoj her se společnými rysy. Herní engine je proto obvykle volen na základě žánru hry. Existují i všeobecně využitelné varianty, například engine *Unity*, o kterém pojednává samostatná kapitola, se nespecializuje na žádný herní žánr, a je proto využíván pro

tvorbu her rozmanitých druhů. Další výjimkou by mohl být Unreal Engine, který našel využití taktéž v různých žánrech, ačkoliv je přednostně designovaný pro videohry typu FPS.

Zdroje [2], [3].

1.3 Asset

Pojmem *asset* může být označeno veliké množství souborů, které se podílejí na tvorbě her. Některé *assety* mohou být vytvořeny přímo v herním *enginu*, ale častěji se k tomuto využívají softwary, které jsou na to specializované. Soubory tohoto druhu je možné také zakoupit či získat zdarma z různých webových obchodů.

Konkrétními zástupci assetů mohou být například obrázky, 3D modely, animace, textury, zvukové soubory, skripty, prefaby a další. Následující podkapitoly budou pojednávat pouze o některých typech *assetů*.

Zdroj [4].

1.3.1 Skript

Skript představuje sérii příkazů, které mají za cíl vykonat určitý úkol. Bývají často uloženy v samostatných souborech a jsou prováděny prostřednictvím jiného programu.

Aplikace skriptů je nepostradatelnou součástí každé hry. Umožňuje například stanovit ovládání hráče a kamery nebo vytvářet různé události a provádět je ve stanovený čas či na základě dané podmínky. Kromě toho mohou být skripty také použity k vytváření grafických efektů, k manipulaci herních objektů, nebo dokonce i k implementaci umělé inteligence.

Zdroj [5].

1.3.2 Animace

Cílem animace je vytvořit iluzi spojitého pohybu. Dosáhnout jí lze postupným zobrazováním za sebou jdoucích obrázků. Tyto obrázky se od sebe liší pouze nepatrně a jsou zobrazovány takovou rychlostí, aby bylo dosaženo co možná největší plynulosti. Standardně to bývá 25-30 snímků za sekundu.

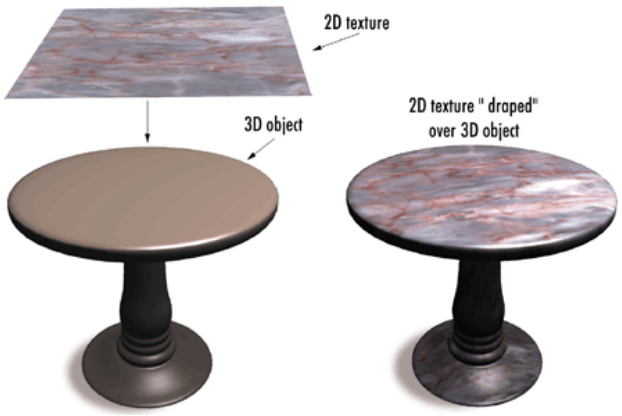
V dnešní době se k tvorbě animací používá zejména výpočetní technika v kombinaci s vhodným softwarem. Takto lze dosáhnout jak 2D tak i 3D animací. Obvykle se k tomu využívají klíčové snímky (keyframes), jež dovolují definovat důležité pozice objektů v čase

a prostoru. Pohyb mezi těmito polohami je poté dopočítán samotným programem. Počítačové animace obvykle najdou využití ve filmovém a ve videoherním průmyslu.

Zdroj [6].

1.3.3 Textura

Textura slouží k popisu optických vlastností povrchu, s její pomocí lze určit zejména barvu, strukturu povrchu a schopnost reagovat na světlo. V zásadě se jedná o 2D obrázek, který lze aplikovat na 3D model, jak je zobrazeno na obrázku č. 1.



Obrázek 1: Ukázka aplikace 2D textury na 3D objekt[1]

Pro ztvárnění jednotlivých vlastností povrchu se v praxi využívá vícero typů textur, které lze následně kombinovat. Nejčastěji se lze setkat s těmito typy.

- Diffuse

Textura typu *diffuse* určuje barvu a typ povrchu modelu. Dále může zobrazovat detaily, jako jsou například různé nedokonalosti povrchu. Tato textura se však nadále jeví plochým dojmem. Pro realističtější efekt je v mnoha případech potřeba využít dalších textur.

- Specular

Typ *specular* se používá pro dosažení odlišného stupně odlesku na různých místech povrchu. Využívá se především, pokud je požadovaný povrch modelu tvořen materiály s rozdílným stupněm odrazivosti světla.

[1] Obrázek převzat z [15].

❖ Normal

Textura *normal* (česky občas jako normal mapa) se používá pro zdůraznění některých detailů, jež mají často za následek, že povrch poté nabývá trojrozměrného dojmu. Tyto změny však nijak neovlivňují tvar modelu, na který je tato textura aplikována. Této skutečnosti je dosaženo pomocí práce se světlem, kdy jsou v určitých místech jednotlivé světelné paprsky odráženy jiným směrem. Tímto způsobem mohou být na povrchu zobrazeny rýhy a další nedokonalosti mnohem realističtěji.

❖ Displacement

Textura s názvem *displacement*, jež se v mnoha aspektech podobá *normal* mapě Avšak tento typ nese navíc informace o výšce, dle které je následně upraven tvar celého modelu.

Zdroj [7].

1.3.4 Prefab

V praxi se lze setkat se situací, kdy je potřeba vytvořit vícero kopií konkrétního objektu, u kterých pak bude vyžadováno měnit jejich vlastnosti, tak aby se u všech objektů projevily současně (což je rozdíl oproti instancování z OOP). Pouhou kopií originálního objektu toho však nelze dosáhnout, jelikož by bylo potřeba upravovat každou kopii individuálně. A právě z tohoto důvodu existují tzv. *prefaby*. Do těchto speciálních assetů lze totiž vložit herní objekt, který se ve scéně bude vyskytovat vícekrát. Tento *prefab* je pak možné opakovaně vkládat do scény. Všechny změny provedené na *prefabu* se posléze projeví na jeho instancích.

Zdroj [12].

2 UNITY 3D

Unity 3D představuje multiplatformní herní engine a vývojové prostředí (IDE), jež umožňuje vývoj jak trojrozměrných, ale také dvojrozměrných her a simulací, a to s možností exportu až pro 27 různých platforem.

První verze byla vydána v roce 2005 trojicí David Helgason, Joachim Ante a Nicholas Francis z dánské společnosti Unity Technologies. Jejich cílem bylo vytvořit cenově dostupný herní engine s profesionálními nástroji zejména pro amatérské herní vývojáře.

Ve své základní verzi je tento software zdarma. Zároveň je relativně snadné si ho osvojit a případně se seznámit s radami a tipy, které nabízí fórum a tutoriály, jež jsou publikovány přímo od výrobců. Z těchto důvodů je *Unity 3D* daleko vhodnější pro začátečníky než konkurenční herní enginy.

Součástí Unity je fyzikální engine *PhysX* od společnosti NVIDIA. Tento engine má na starosti výpočet fyziky, díky čemuž mohou být ve hrách vytvářeny různé typy simulací, animací, kolizí a dalších aplikací fyziky na jednotlivé objekty.

Zdroje [8], [11].

2.1 Licence softwaru

Engine *Unity3D* je v současnosti dostupný ve 4 licencích: *Personal, Plus, Pro* a *Enterprise*. Výběr odpovídající licence by měl být založen na množství již získaných zkušeností a na cílech, kterých se uživatel snaží dosáhnout za pomoci tohoto softwaru.

2.1.1 Personal

Edice *Personal* je bezplatná a je určená především pro studenty a začátečníky. I přesto jí lze možno využít i pro komerční účely, avšak výše příjmu nesmí překročit 100 000 dolarů.

Tato edice se může pyšnit většinou prvků, které dělají *Unity* tak výjimečným prostředkem pro vývoj videoher. Lze tedy využít plný potenciál, který herní engine *Unity* nabízí. Taktéž export je stále možný na všechny podporované platformy.

Zdroj [9].

2.1.2 Plus

Edice *Plus* je doporučena pro uživatele, jejichž prioritním cílem je monetizovat vytvořené hry. Cena této licence dosahuje 35 dolarů za měsíc. Jedním z hlavních přínosů této edice je možnost získávat zprávy týkající se chyb a výkonu v reálném čase. Dále je umožněna customizace úvodní obrazovky při spouštění hry. Také je možné upravovat grafický vzhled samotného editoru

Zdroj [9].

2.1.3 Pro

Pro edice umožňuje zvýšení možného výdělku nad 200 000 dolarů. Pro ideální využití této licence se doporučuje pracovat na ambicióznějších projektech a být dobře financován. Dále přidává prémiovou podporu, která zajišťuje okamžitou pomoc od týmu *Unity* expertů.

Zdroj [9].

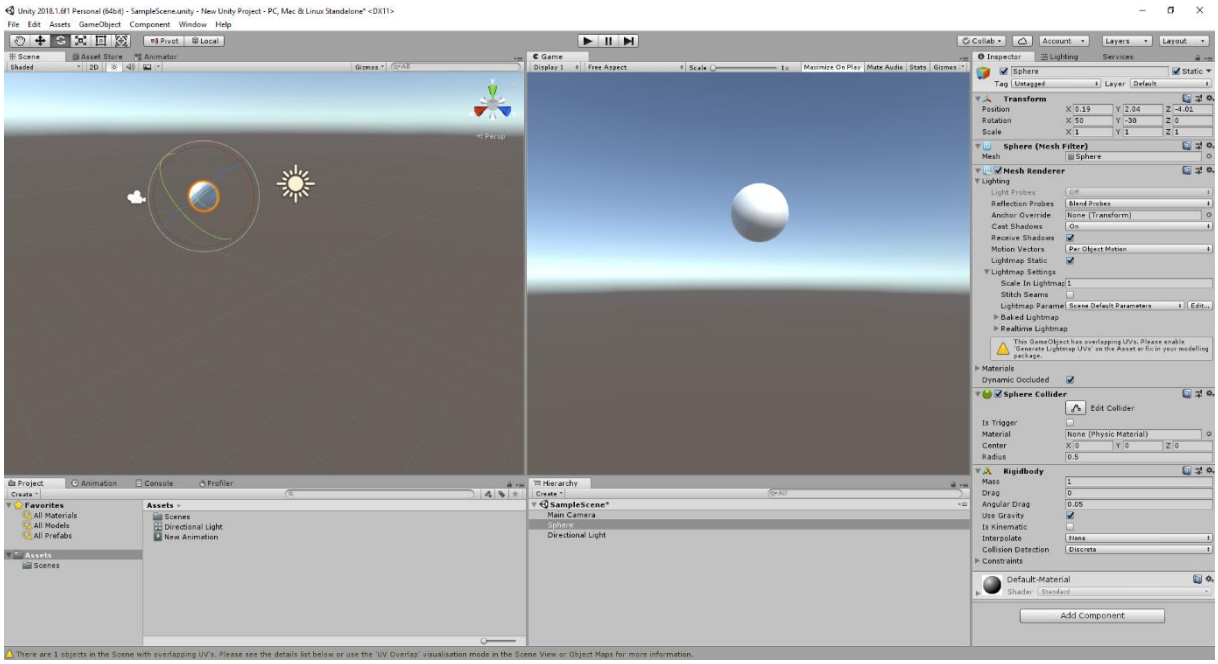
2.1.4 Enterprise

Nejvyšší edice *Enterprise* je cílená pro herní vývojářské týmy, které čítají více než 21 členů. Tato licence jako jediná umožňuje přístup ke zdrojovému kódu samotného engine, jehož modifikací lze dosáhnout optimálního herního výkonu. Cenová výše tohoto produktu je individuální.

Zdroj [9].

2.2 Unity Editor

Unity *editor* (obrázek č. 2), který umožňuje efektivně pracovat s herním enginem, je tvořen mnoha podokny. Mezi 5 hlavních patří: *Scene View, Game View, Project View*, *Hierarchy View* a *Inspector View*. O těchto a několika dalších oknech bude pojednávat podrobnější popis níže.



Obrázek 2: Unity Editor[2]

2.2.1 Scene View

Okno *Scene View* umožňuje pohyb skrze celý 3D svět, ve kterém se daná hra odehrává. Umožňuje taktéž zobrazovat všechny oblasti hry, i ty do kterých nebude mít hráč přístup. Prostřednictvím tohoto okna je možné vytvářet jednotlivé scény, do kterých následně lze vkládat jednotlivé objekty, s kterými je poté možné libovolně manipulovat.

Objekty do scény je umožněno přidávat pomocí přetahování anebo přímím vytvářením. Poté lze měnit jejich umístění, velikost, rotaci a další vlastnosti. Seznam všech použitých objektů v aktuální scéně je zobrazen pomocí okna *Hierarchy View*. Podrobnosti o nich pak lze najít v okně *Inspector View*.

Zdroj [8].

2.2.2 Game View

Okno *Game View* reprezentuje podobu scény po jejím spuštění. Obraz je tvořen prostřednictvím hlavní kamery. Příslušnými tlačítky lze ovládat spuštění, pozastavení či krokování scény. Své

[2] Zdroj: Vlastní

uplatnění nejčastěji naleznou při testování provedených změn. Úpravy scény je také možné provádět i po jejím spuštění, ty jsou však pouze dočasné a po opuštění herního režimu se neprojeví.

Lišta tohoto okna také umožňuje přepínat mezi displeji, které lze přidělit různým kamerám. Dále je zde také umožněno nastavit poměr stran cílového displeje, úroveň přiblížení, režim celého okna a další vlastnosti.

Zdroj [8].

2.2.3 Inspector View

Inspector View představuje okno, které umožňuje přistupovat k fyzice a dalším vlastnostem jednotlivých objektů. Každý herní objekt obsahuje položku s názvem *transform*, která uchovává informace o jeho současné pozici, rotaci a velikosti. Další položky mohou například definovat možnou kolizi s ostatními objekty, působení gravitace, barvu, vzhled textur, zvukové efekty a další náležitosti. Mnohé položky sem mohou být přidány pouhým přetažením. Takto sem lze vložit například různé skripty, které pak k těmto objektům získají přístup a mohou s nimi manipulovat.

Zdroj [8].

2.2.4 Hierarchy View

V okně *Hierarchy View* jsou obsaženy všechny objekty, které se nacházejí v aktuální scéně. Objekty lze libovolně vkládat do jiných, čímž docílíme vytvoření potomků daných objektů. Po kliknutí na objekt dojde k zobrazení vlastností v okně *Inspector View*.

Zdroj [8].

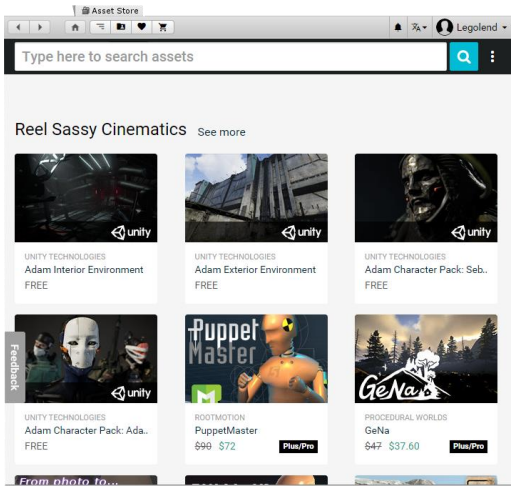
2.2.5 Project View

Okno *Project View* zobrazuje všechny dostupné *assety* aktuálního projektu. Uspořádání je tu zajištěno pomocí stromové struktury, která pomáhá udržovat přehlednost souborů projektu. Z tohoto okna pak mohou být naše *assety* přímo přetaženy do konkrétní scény.

Zdroj [8].

2.2.6 Asset Store

Okno *Asset Store*, jež je zachyceno na obrázku č. 3. Okno slouží pro přístup k obchodu *Unity Asset Store*, který je mimo jiné také možné navštívit i prostřednictvím internetu. Tento obchod obsahuje spoustu materiálu, který dokáže znatelně usnadnit vývoj her. Nalézt je zde možné *assety*, jež jsou placené, ale i zcela zdarma. Unity umožňuje vybrané produkty stáhnout a přímo naimportovat do projektu. Do *Asset Storu* je také možné přidávat své vlastní *assety*.



Obrázek 3: Unity Asset Store[3]

Zdroj [10].

2.2.7 Další podokna

V *Unity Editoru* se lze setkat i s dalšími okny, která jsou taktéž důležitá pro vývoj videoher. Zařadit mezi ně lze například okna *Animation View* a *Animator Window*, která umožňují vytvářet a spravovat herní animace. Dále se často využívá okno *Lighting*, které má na starosti nasvícení scény.

[3] Zdroj: Vlastní

2.3 Podpora programovacích jazyků

V minulosti podporoval tento engine tvorbu her za pomoci 3 jazyků, byly to *C#*, *Boo* a *UnityScript,* jenž je obdobou jazyka *JavaScript*. Postupem času se komunita začala značně přiklánět k jazyku C#, z tohoto důvodu byl v roce 2014 z Unity odstraněn jazyk *Boo*. Avšak projekty pracující s tímto jazykem byly plně funkční, jelikož *UnityScript*, jenž by se dal označit za vrstvu nad jazykem *Boo*, využívá některé knihovny jazyka *Boo* a kompilátor *UnityScriptu* je napsaný taktéž v *Boo*.

V srpnu roku 2017 bylo ohlášeno, že přestane být podporován i *UnityScript*. Hlavním důvodem k tomuto rozhodnutí byla menší popularita tohoto jazyka mezi uživateli, jelikož bylo zaznamenáno pouze 14,6 % projektů, jež v té době obsahovalo alespoň jeden skript napsaný v jazyce *UnityScript*. V současné době se nové hry vyvíjejí pouze za použití jazyka *C#*.

Zdroj [13].

2.4 Hry s enginem Unity

Unity se v současnosti řadí mezi nejpopulárnější herní enginy využívané vývojáři po celém světě. Své uplatnění najde jak v řadách indie, tak i AAA studiích, jež dávají vznik nespočtu herních titulů. A pouze zlomku z nich se bude následující část věnovat.

Mezi hry s velice zajímavým vizuálem a skvělou hratelností se bezesporu řadí hra s názvem *Cuphead* (vizte obrázek č. 4), která vyšla v roce 2017. Jedná se o akční „plošinovku“ a „střílečku“, která je založena na karikaturách 30. let. Hra byla vytvořena právě za pomocí Unity, kdy pozadí bylo vytvářeno pomocí vodních barev a animace byly ručně kresleny. *Cuphead* taktéž nabízí velice kvalitní dobový soundtrack.

V Unity byla taktéž vytvořena velice úspěšná hra *Hearthstone*. Společnost Blizzard Entertainment zde taktéž vsadila na engine Unity z důvodu rychlosti a plynulejšího běhu hry.

V neposlední řadě stojí za to taktéž zmínit českou hru *Hero of Many*, jež byla vytvořena studiem Trickster Arts v roce 2013. Tato akční adventura sklidila veliký úspěch a je dostupná pro Android, IOS a Windows.

Zdroj [14].

Obrázek 4: Ukázka ze hry Cuphead[4]

[4] Obrázek převzat z [14].

3 HERNÍ VÝVOJ

Kapitola se snaží přiblížit náležitosti spojené s vývojem videoher. První podkapitola se zaměřuje na jednotlivé role vývojářského týmu a jejich úlohy. Následně tato práce seznámí čtenáře se samotným procesem vývoje. Vzhledem k obsáhlosti tohoto tématu zde budou pokryty především ty informace, jež jsou nezbytné pro proniknutí do této problematiky.

3.1 Vývojářský tým

Samotná tvorba herního titulu je realizována pomocí vývojářského týmu, který se může ve velikosti značně lišit. Některé týmy se mohou skládat pouze z několika lidí, zatímco jiné týmy mohou čítat stovky zaměstnanců a mohou pracovat na vícero titulech zároveň.

Vývojářský tým sestává z jednotlivých rolí, kdy se každá zabývá rozdílnou oblastí vývoje. Není neobvyklé, aby jeden člen týmu měl na starosti více těchto rolí. Důležitou část týmu tvoří grafici, designeři, programátoři, animátoři, zvukoví technici a testeři.

Zdroj [16].

3.1.1 Designer

Designer je člen týmu, který má na starosti návrh hratelnosti, pravidel a struktury hry. Vývojářský tým obvykle obsahuje více zaměstnanců s touto rolí, každý z nich má na starosti jinou oblast herního designu, a všichni spadají pod vedoucího *designera*, který koordinuje jejich práci.

Designeři mohou mít na starost například vytváření dialogů, filmových sekvencí, popřípadě deníků a dalších prvků, které pomáhají budovat příběh. Tzv. *level designer* má za úkol tvorbu herního prostředí, obzvláště pak návrh úrovní a misí za využití *level editoru* a dalších nástrojů k tomu určených. Dále se *designeři* mohou podílet také na návrhu herních mechanik, uživatelských rozhraních a herních charakterů.

Zdroje [16], [17], [18].

3.1.2 Grafik

Grafik se zabývá vytvářením grafického obsahu hry. I v týmu grafiků je vyčleněn jeden zaměstnanec, který má za úkol řídit a koordinovat práci jednotlivých členů tohoto týmu. Zbylí

členové obvykle plní úkoly, jež mají rozdílnou povahu. Počet těchto úkolů na jednoho grafika je obvykle určen v závislosti na velikosti týmu grafiků.

Práce grafika může být orientována jak na 3D, tak i na 2D prostor. Ve 2D prostoru pracuje například tzv. *concept artist*. Jedná se o pozici, jejíž podstatou je vytvářet nákresy postav a prostředí za pomoci papíru a tužky anebo pomocí 2D softwarů. *Concept artist* obvykle bývá v úzké spolupráci s designery a od jeho práce se později odvíjí vzhled celé hry. Další pozice bývá označována jako *storyboard artist*, a její náplní je vytvářet obrazové scénáře filmových sekvencí. Pomocí náčrtku jsou tedy podrobně zachyceny jednotlivé scény, a jak se v nich pohybuje kamera. Další významnou roli sehrává tzv. *texture artist*, jenž má za úkol přidávat textury na již 3D grafiky vytvořené modely. 2D grafici se taktéž podílejí na vývoji uživatelského rozhraní hry, jako je například herní menu nebo HUD.

Mezi nejčastější profese mezi 3D grafiky by se prokazatelně dal zařadit tzv. *3D modeller*, jehož náplní práce je vytváření herních modelů za pomocí modelovacích softwarů, jako je například *3ds Max, Maya* nebo *Blender*. Další rolí je *environmental artist*, který má na starosti tvorbu herního prostředí. Toto obnáší zejména tvorbu krajiny a rozmisťování herních modelů, taktéž tu bývá velice častá práce s texturami a barvami. Mezi další významné pozice se řadí tzv. *lightning artist*, který má za úkol nasvítit scénu videohry. Využitím intenzity, jasu a barevných odstínu lze velice efektivně nastolit atmosféru, jež se posléze může stát ikonickou pro danou videohru. V neposlední řadě je taktéž velmi důležitá role *animátora*, jenž má za cíl opatřit herní modely možností pohybu. Využívají se k tomu nástroje, jež bývají velice často součástí modelovacích softwarů. Nezřídka se pro vytváření animací využívá technika známá pod názvem *motion capture*, s jejíž pomocí lze snímat pohyby objektů v reálném světě a převádět je poté do digitální podoby. *Motion capture* se nejčastěji využívá pro vytváření pokročilých animací, jako jsou zejména lidské pohyby. Ve videoherním průmyslu se k tomuto záměru často využívají optické systémy, jež se skládají z tzv. *markerů* a speciálních kamer. Tyto *markery* jsou umístěny na částech těla, jež jsou objektem snímání. Jejich pohyb v prostoru je poté zachycován pomocí kamer, které jsou rozmístěné v místnosti, jež je k tomuto určena.

Zdroje [16], [17], [18], [19].

3.1.3 Programátor

Programátoři mají v herním vývojářském týmu na starosti funkčnost projektu. Úloha jednotlivých *programátorů* se může do značné míry lišit. Zatímco se jedna skupina

programátorů soustředí na tvorbu a úpravu nástrojů usnadňující vývoj herního titulu, tak se druhá skupina soustředí na implementaci logiky do samotné videohry.

Jednou ze základních rolí *programátorů* je vytváření, respektive úprava herního *enginu*. Customizace stávajícího *enginu* obnáší především přizpůsobení herní fyziky a grafiky pro účely dané hry. Dále lze vývoj usnadnit již zmiňovanou tvorbou vlastních softwarových nástrojů. Jedná se například o programy, které usnadňují převod *assetů* na formát vyžadovaný hrou, editování jednotlivých úrovní hry anebo kompilaci *skriptů*.

Mezi důležité a zároveň velmi sofistikované prvky hry patří umělá inteligence. S její pomocí lze nasimulovat chování objektů, jež nejsou ovládány hráčem. Takto naprogramované objekty se pak mohou například libovolně pohybovat v prostoru nebo reagovat na chování hráče. Umělá inteligence a další prvky herní logiky, jako jsou například herní události, jsou uskutečňovány pomocí skriptů. Zaměstnanci s touto rolí se nazývají *skripteři*. Další členové programátorského týmu se také mohou zabývat tvorbou uživatelského rozhraní, konfigurací vstupních zařízení pro ovládání nebo realizací kooperace více hráčů za využití internetového připojení.

Zdroje [16], [17], [18].

3.1.4 Zvukový technik

Zvukoví technici nesou zodpovědnost za ozvučení celé hry. Zaměření těchto techniků lze rozdělit do tří kategorií. První kategorie zahrnuje zvuky prostředí, jako je například chůze, střelba, otevírání dveří atd. Dále se tito zaměstnanci zaměřují na hudební doprovod, jenž bývá často označován pod pojmem *soundtrack*, a jenž bývá zhotoven prostřednictvím hudebníků. V neposlední řadě se zvukoví technici zabývají dabingem, který bývá nedílnou součástí většiny současných her.

Zdroje [16], [18].

3.1.5 Tester

Tester zajišťuje, aby výsledný produkt dosáhl co možná nejlepší kvality. Dohlíží, aby výsledná hra správně fungovala a byla zároveň zábavná. *Testeři* vypracovávají zprávy o dosavadních chybách a ideálně i doplňují, kdo a jak by měl danou chybu opravit. Obvykle se vyčleňuje i vedoucí tester, který blíže spolupracuje s designery, grafiky a programátory a dohlíží, aby veškeré chyby byly náležitě opraveny. V některých herních studiích se také vyskytuje tzv.

technical tester, který se zabývá vytvářením zautomatizovaných testů a řeší sofistikovanější problémy, které souvisí například s výkonem nebo bezpečností hry.

Zdroje [16], [18].

3.2 Proces vývoje videoher

Vývoj videoher bývá zpravidla zdlouhavý a náročný proces. Postupem času se ale vyvinuly postupy a metody, kterými se dá tento proces do jisté míry zjednodušit a zefektivnit. Tyto doporučené kroky bývají však často opomíjeny, a takto vytvořené hry mohou poté skončit s překročeným rozpočtem, s velikým počtem chyb, anebo nemusejí být dokončeny včas.

Tato podkapitola se dělí na koncept, preprodukci, produkci a postprodukci. Jedná se o čtyři chronologicky za sebou jdoucí fáze, jež jsou nezbytné pro vývoj úspěšného herního titulu. Na každé této fázi se obvykle podílí rozličný tým odborníků.

Zdroje [16], [20], [21].

3.2.1 Koncept

Videoherní *koncept* bývá zpravidla založen na myšlence, jež ve stručnosti definuje hlavní rysy vyvíjeného titulu. Dále obsahuje nejdůležitější herní mechanismy, popis prostředí, ilustrační obrázky a nastínění příběhu, pokud je součástí hry.

Produkty této fáze jsou tzv. *high concept*, *herní návrh* a *koncepční dokument*. Za *high concept* je považován popis čítající jednu až dvě věty, jež popisují, o čem daná hra je. Již z tohoto by mělo být zjevné čím je hra unikátní a liší se tak od konkurence.

Herní návrh bývá obvykle dvoustranný podklad, který slouží zejména pro prezentování při hledání sponzorů. Tento krátký dokument by měl obsahovat stručné odpovědi na otázky, o čem je plánovaná hra, proč by měla být úspěšná, a jak bude vydělávat peníze.

Koncepční dokument je podrobnější obdoba *herního návrhu*. Zde by měl být obsažen zejména žánr, popis hratelnosti, hlavní rysy, popis herního světa, příběh, pro koho je hra určena, cílový hardware, odhadovaná doba vývoje, odhadovaný rozpočet a další. Takto vypracovaný dokument se nejčastěji předává jednotlivým členům vydavatelského týmu pro detailnější porozumění plánovaného titulu. Pokud je koncept vydavatelem odsouhlasen můžou následovat další fáze vývoje.

Zdroje [16], [17], [22].

3.2.2 Preprodukce

Preprodukci lze označit jako přípravné období před nástupem samotného procesu tvorby. Hlavním cílem této fáze je vytvořit tzv. *game design* dokument a herní *prototyp*. *Preprodukce* je taktéž velice důležitá z technického hlediska, jelikož demonstruje, že případné nově vyvíjené technologie jsou realizovatelné. Celá tato fáze v zásadě dokazuje, že tým vývojářů je schopný hru vytvořit, a že se vývoj daného titulu vyplatí.

Game design dokument by měl zachycovat do detailu vše co se ve hře vyskytne. Během produkčního cyklu by měl být pak nejaktuálnější reprezentací všeho, čím si uživatel musí během hraní projít. Spadají sem především informace o hratelnosti, uživatelském rozhraní, příběhu, postavách a umělé inteligenci.

Herní *prototyp* je funkční část, z které jsou patrné veškeré klíčové prvky herního titulu. Právě *prototyp* má největší vliv na to, zda bude projekt pokračovat či nikoliv. Komplikací v této fázi může být například, když projekt vyžaduje nový herní *engine* nebo technologii, jež bude dokončena až v pozdější části vývoje. Řešením bývá většinou *předrenderování* materiálu, který bude později *renderován* v reálném čase. Jinými slovy se jedná o simulaci demonstrující, jak bude vypadat finální produkt.

Zdroj [22].

3.2.3 Produkce

Jednou z nejdůležitějších fází celého vývoje je *produkce*. Je zde vyžadována maximální odborná znalost, taktéž zde dochází k nejvýraznější spotřebě financí. Tato fáze obvykle trvá šest až dvacet čtyři měsíců. Herní studia, která překročí limit dvou let, mohou riskovat zestárnutí používaných technologií, ztrátu pracovního nasazení týmu a následný odchod jednotlivých členů. Delší vývoj může být taktéž využit konkurencí, která může vytvořit titul s podobnými motivy a vydat ho dříve. Jakýkoliv z uvedených problémů může vést k potřebě přepracování titulu, což má za následek další opoždění v časovém plánu.

Pro dodržení časového plánu, je užitečné si jednotlivé úlohy rozdělit na menší zvládnutelné části, a ty pečlivě sledovat. Díky tomuto přístupu lze v celku jednoduše zjistit, zda je vývoj pozadu, či nikoliv. Plnění těchto částí by mělo být pravidelně kontrolováno, a to nejméně jednou týdně.

V praxi se využívá obdoba této techniky, kdy je každý vývojář povinen vytvořit seznam úkolů s odhady časů jejich dokončení. Od jednotlivých členů týmu jsou poté tyto seznamy

vybrány a dochází k zhotovení celistvého dokumentu. Tento přístup je užitečný zejména pro sledovaní, zda počet úkolů na jednotlivce nepřekračuje daný limit. Pokud by se tak stalo, je zapotřebí úkoly této osoby přezkoumat a rozhodnout, zdali by některý z nich nemohl být přidělen někomu jinému.

Další výhoda této techniky spočívá ve skutečnosti, že každý z vývojářů má na starosti svůj časový plán, nikoliv nikdo z nadřízených. Tímto se výrazně snižuje riziko, že by některý ze zaměstnanců překročil stanovený čas.

U herních studií, jež pracují externě pro daného vydavatele, je zvykem sledovat postup ve vývoji formou smluvních milníků. Jakmile některý z těchto milníků není dodržen, vývojářský tým nedostane zaplaceno. U dobře organizovaných interních studií se lze setkat s obdobným přístupem.

Definice pro jednotlivé milníky není přesně dána a v zásadě se liší v závislosti na vydavateli, roku vývoje či konkrétním projektu. Nicméně při dvouletém vývojovém cyklu se lze nejčastěji setkat s některými z následujících milníků.

- Alpha

Alpha se užívá pro označení stadia vývoje, kdy je hra v podstatě hratelná od začátku do konce. Mohou se zde však ještě nacházet neúplná řešení herních mechanismů a herní *assety* taktéž nemusí nabývat finální podoby. Zato herní *engine*, uživatelské rozhraní a jiné větší subsystémy by měly být kompletní.

- Beta

Následující fáze se označuje názvem *beta*. Zde jsou již obsaženy veškeré *assety* a vývoj daného titulu je takřka u konce. Hlavním záměrem je stabilizace projektu a eliminace co možná největšího počtu chyb neboli tzv. *bugů*.

- Code Freeze

V posledních několika dnech fáze *bety* se vyvíjený titul dostává do stádia označovaného jako *code freeze*, kdy dochází k zastavení všech modifikací vytvořeného kódu. Povolené úpravy jsou pouze ty, které mají za cíl opravit chyby, jež byly odhaleny v průběhu testování.

- RTM (Release to Manufacture)

Označením *release to manufacture*, někdy též *gold master*, se rozumí finální *kandidát,* který byl důkladné otestován a shledán přijatelným.

Zdroj [22].

3.2.4 Postprodukce

K dovršení celého životního cyklu herního vývoje se přistupuje k tzv. *postprodukci*. Jedná se o fázi, jejíž hlavní náplní je testování finálního produktu.

- Patch

Patch nebo též záplata bývá v poslední době nepostradatelnou součástí her exportovaných na osobní počítače. Vydávání těchto *patchů* nemusí být nezbytně způsobeno uspěchaným vývojem či nedostatečným testováním produktu, jak by se mohlo na první pohled zdát. Osobní počítač může sestávat z nespočetného množství kombinací komponent, tudíž je takřka nemožné, aby veškeré tyto kombinace byly před výdejem otestovány.

- Upgrade

Upgrade je označení pro dodatečný obsah, jenž byl vytvořen pro rozšíření původní hry. Důvodem pro vytváření upgradů bývá často prodloužení životnosti daného titulu. Taktéž se může jednat o efektivní strategii pro udržení části týmu v zaměstnaneckém poměru, zatímco se zbylá menší část začíná věnovat novému projektu.

Zdroje [20], [22].

4 ZADÁNÍ HRY

Praktická část bakalářské práce se zabývá vývojem videohry *Labscape*. Jedná se o edukativní herní titul, jenž svými herními mechanismy podporuje výuku bezpečnosti a ochrany zdraví při práci v prostředí chemických laboratoří.

Tato kapitola pojednává o vzniku návrhu samotného projektu. Bude zde detailněji vysvětleno, proč byla vybrána videohra právě s tímto zaměřením, co vše se od ní očekává, a jak to bude vše posléze zpracováno.

4.1 Motivace

Návrh videohry *Labscape* byl vypracován a předložen doktorandkou Nishaben Desai Dholakiya z Chemicko-technologické fakulty Univerzity Pardubice. Cílem tohoto projektu není pouze nabídnout alternativní způsob výuky bezpečnosti v laboratořích, ale taktéž učinit problematiku týkající se bezpečnosti zajímavější. Hráči bude umožněno provádět chemické experimenty v simulovaném prostředí, aniž by byl vystaven nebezpečí, jež se obvykle nachází v reálných podmínkách.

4.2 Požadavky

Základní myšlenkou hry je přivedení uživatele do virtuální laboratoře, kde může se svým *avatarem* provádět předem specifikované úkony a chemické experimenty.

Pro tento typ herního titulu je nezbytné, aby byl hráč za špatné či neuvážené kroky penalizován ztrátou herních bodů či vynuceným opakováním dané úrovně hry, a to v závislosti na závažnosti jeho pochybení. Hra je rozdělena do tří levelů, jež se zaměřují na konkrétní typy laboratorní bezpečnosti, a jež budou odemykány postupně na základě splnění předchozích úrovní.

První level se zaměřuje na dodržování základních principů bezpečnosti v laboratorním prostředí, např. používání ochranného vybavení, manipulování a uskladňování nebezpečných chemikálií atp. Schéma s podrobnou herní logikou této úrovně je obsažen v příloze A.

Druhý level se věnuje výrobě chemikálie zvané kyselina adipová, kde je nezbytné dodržení předepsaných postupů proto, aby byl zaručen správný průběh reakce. Hráč zde musí například rozhodovat o použitém množství a umístění přidávaných látek, teplotě připravované směsi atp.

Finální level se ve své podstatě svým zaměřením neliší od předchozí úrovně. Je zde však kladen větší důraz na zakomponování více prvků reálného prostředí. Je zde například

aplikována pravděpodobnost selhání jistého vybavení z důvodu jeho opotřebení. Taktéž jsou zde využity rovnice, jako je například Newtonovo pravidlo pro chlazení nebo rovnice pro výpočet energetické rovnováhy, pro získání co možná nejpřesnějších výsledků při přípravě dané substance. Výukový software tak ve třetí úrovni značně snižuje úroveň abstrakce simulované chemické reakce.

4.3 Zpracování

Zamýšlená podoba hry byla pravidelně konzultována se zadavatelem projektu, který svou vizi podporoval materiály, jež byly poskytovány zejména pro zefektivnění vývoje. Tyto materiály lze rozčlenit do následujících kategorií.

- Fotografie prostor a vybavení laboratoře
- Videoukázky průběhu chemických experimentů
- Schémata s herní logikou
- Rovnice pro výpočet chemických reakcí

Tyto materiály byly posléze zpracovány autorem (jako součást bakalářské práce) a daly vznik výslednému produktu, jímž je videohra skládající se ze tří úrovní a dostupná pro platformy *Windows, Android* a *WebGL*. Zadavatelka software byla s řešením spokojena a pro potřeby bakalářské práce poskytla krátké vyjádření, které je zařazeno do vlastní práce jako příloha B. Také byly zhotoveny publikace [27], [28] a [29], jež se věnují laboratorní bezpečnosti v souvislosti s tvorbou tohoto herního titulu.

Před samotným vývojem hry *Labscape* bylo klíčové zvolení herního enginu. Jelikož bylo upřednostňováno, aby se jednalo o 3D hru, především pro navození co nejvíce reálného požitku z pobytu v laboratoři, byl zvolen herní engine *Unity 3D*, jenž, jak už bylo uvedeno výše, je vhodný pro vývoj trojrozměrných her, lze si ho vcelku velmi rychle osvojit a jeho základní verze je dostupná zcela zdarma.

Design prostředí hry je inspirován reálnými prostorami Ústavu energetických materiálů Fakulty chemicko-technologické v obci Doubravice u Pardubic. Ve videohře je graficky vypodobněna jedna z chemických laboratoří a chodba, která k ní vede. Snahou bylo zobrazit tyto lokace co možná nejrealističtěji, nicméně bylo nutné provést některé změny. Například rozměry jednotlivých místností jsou mírně zvětšeny, aby na hráče nepůsobily stísněným dojmem. Porovnání skutečné a vymodelované části laboratoře je zobrazeno na obrázku č.5.

Obrázek 5: Porovnání vybraného vybavení laboratoře ve skutečnosti a ve videohře[5]

Taktéž bylo rozhodnuto, aby se jednalo o hru z pohledu třetí osoby, a to zejména proto, aby bylo patrné, jaké ochranné vybavení má hráč na sobě a jaké nikoliv. Toto bylo plánováno pro všechny úrovně videohry. Nicméně v pozdější části vývoje bylo rozhodnuto, aby poslední level byl hratelný z pohledu první osoby. Důvodem k tomuto kroku bylo vytvoření výrazné změny, jelikož odlišnosti posledních dvou levelů pouze v herní logice by nemusely být na první pohled patrné a uživatel by tak mohl být odrazen od dokončení celé hry.

Už od konceptu hry byl titul *Labscape* označován výhradně jako mobilní videohra, avšak na základě dostupných možností vybraného herního enginu, bylo rozhodnuto pro export na větší množství platforem.

Koncept hry *Labscape* se taktéž dočkal prezentování na některých mezinárodních konferencích. Jednalo se například o konferenci GCPS 2019 [30] v USA nebo APSAC [31] 2018 v Chorvatsku, dále tato hra byla představena také v Německu.

[5] Zdroj: Vlastní

5 IMPLEMENTACE HRY

Implementace hry Labscape byla v souladu s teoretickými postupy představenými v předchozích kapitolách. Následující podkapitoly pojednávají o náležitostech spojených s implementací videohry. Důraz je zde kladen zejména na podrobný popis tvorby či získání použitých *assetů* a na jejich následné zakomponování do samotné hry. Z důvodu velkého počtu použitých *assetů*, zde budou zmíněny a podrobněji popsány pouze ty, jež jsou pro tento projekt klíčové, a jež se svou sofistikovaností řadí mezi ty zajímavější.

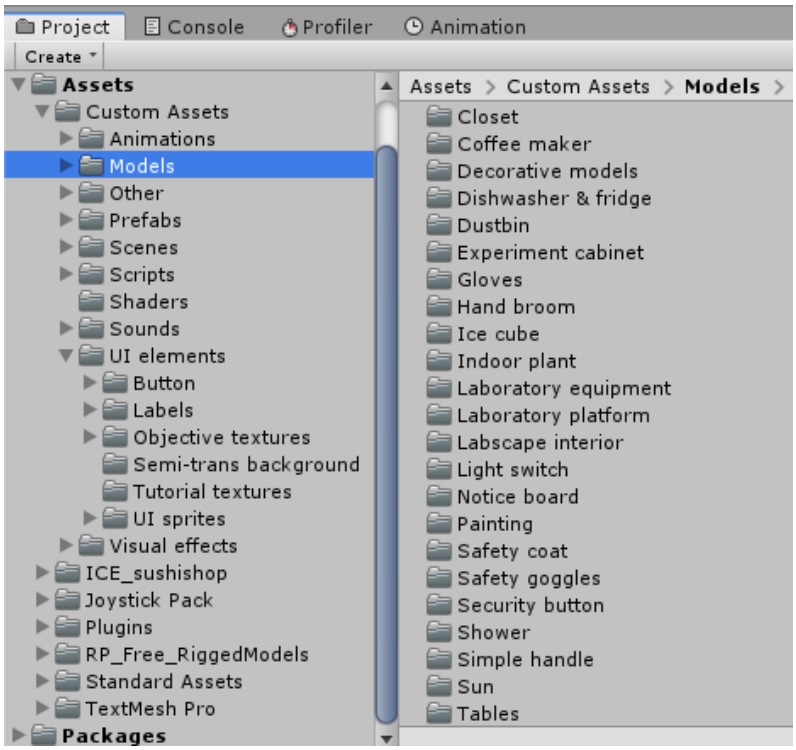
Dále zde bude detailněji rozebrán export na vybrané videoherní platformy. V souvislosti s nimi budou zmíněny modifikace, jež bylo nutné provést za účelem docílení vyššího výkonu či lepší hratelnosti.

5.1 Herní assety

Herní *assety* jsou nedílnou součástí každého videoherního titulu. V tomto projektu se vyskytuje přímo celá řada těchto souborů. Z větší části se tu lze setkat s vlastnoručně vytvořenými, ale objevují se zde i převzaté.

Herní assety jsou uchovány ve složce *Assets*, jež je jednou z nejdůležitějších složek projektu. Implementace hry *Labscape* pro jednotlivé platformy se v jistých aspektech výrazně lišila, z tohoto důvodu vývoj pro každou platformu probíhal v samostatném projektu.

Obsah adresáře *Assets* vztahujícího se k projektu *Labscape* pro platformu *Android* je vyobrazen na obrázku č.6. Podsložka *Standard Assets* obsahuje základní sadu *assetů*, jež je dostupná v podobě balíčku zdarma od společnosti *Unity Technologies*. Odtud byly využity například animace pro pohyb hlavní postavy nebo textury pro částicové efekty. Dále je velice důležitý balíček s názvem *RP_Free_RiggedModels* [23], jehož součástí byl model hlavní postavy. Za zmínku taktéž stojí balíček *PostProcessing* [24], jenž modifikoval grafickou stránku pomocí vizuálních efektů u platforem *Standalone* a *WebGL*.



Obrázek 6: Ukázka struktury adresáře s herními *assety* pro platformu *Android*[6]

Složku *Custom Assets* lze zařadit mezi ty významnější, jelikož je zde uloženo velké množství *assetů*, jež jsou dílem vlastní tvorby. Výpis podsložek tohoto adresáře je následující.

Animations – v této složce se v podobě klipů nacházejí animace jednotlivých modelů, společně s jejich ovladači.

Models – tato složka uchovává 3D modely, jež jsou rozdělené do odpovídajících podsložek. Obsaženy jsou zde i potřebné textury a materiály.

Other – zde jsou uloženy celky herních assetů, jež nebylo vhodné umístit do žádné ze specializovaných složek. Vyskytuje se tu například hlavní a herní menu. Jsou zde zahrnuty jak grafické prvky, tak i animace a skripty.

Prefabs – jedná se o složku obsahující *prefaby*, jenž byly vytvořeny z herních objektů a prošly řadou úprav, aby vyhovovaly specifickému využití v herní scéně.

Scenes – složka uchovávající použité herní scény. Jsou zde tři pro jednotlivé herní levely a jedna pro hlavní menu.

[6] Zdroj: Vlastní

Scripts – v této významné složce jsou uloženy skripty programovacího jazyku C#, jež zajišťují implementaci herní logiky a herních mechanismů.

Shaders – složka sloužící pro uchovávání grafických *shaderů*[7].

Sounds – obsaženy jsou zde veškeré použité zvuky a hudba. Jedná se například o chůzi, nalévání vody, hudba ve hře, v menu atd.

UI elements – jedná se o složku s elementy 2D grafiky. Nalézá se zde například grafické ztvárnění tlačítek, informačních cedulí, úkolů, tutoriálu atp.

Visual effects – zde jsou uloženy především objekty spojené s částicovými efekty. Jedná se například o kouř či proud tekutiny i s jejich *prefaby*, materiály a texturami.

5.1.1 3D modely

Většina herních modelů byla zhotovena za pomoci programu *Blender* [32]. Tento svobodný a otevřený software není vhodný pouze pro tvorbu 3D modelů, ale může být využit také pro animování, *rigování*, simulování, renderování a mnoho dalšího.

Při realizaci modelů postačovalo ve většině případů využití pouze základních operací (změna velikosti, rotování, protažení (*extrudování*), kopírování atd.), jež 3D modelovací software nabízí. Také pro základ u mnohých modelů byl využit tvar válce, který byl následně transformován do složitějších těles.

Mesh neboli síť jednotlivých modelů byl vytvořen tak, aby obsahoval co možná nejmenší počet *polygonů*[8]. Toto je klíčové především u videoher, kdy vykreslování v reálném čase nesmí být příliš náročné. V případě této hry bylo nutné polygony držet pod jistou hranicí, aby běžela plynule i na mobilních zařízeních.

Taktéž byly použity modelovací nástroje, jež *Blender* nabízí ve formě tzv. *modifikátorů*. Tři nejpoužívanější z nich byly *mirror*, *boolean* a *solidify*. Jak název napovídá, tak nástroj *mirror* (*zrcadlení*) se využívá pro zrcadlení *meshe* podél vybrané osy. V případě tohoto projektu, byl tento nástroj použit zejména při tvorbě ochranných brýlí či laboratorního stolu. Nástroj *boolean* poté dokáže spojit, odečíst či udělat průsečík dvou objektů. Využito toto bylo především při vytváření otvorů pro dveře ve zdech. Poslední ze jmenovaných modifikátorů slouží pro zvětšení tloušťky stěn objektů. Zde byl nástroj *solidify* používán relativně často pro složitější modely,

[7] *Shader* – skript obsahující matematické vzorce a algoritmy pro výpočet barvy *renderovaných* pixelů
[8] *Polygon* – reprezentace nejmenší části plochy modelu, nejčastěji ve tvaru čtverce či trojúhelníku

kde si už nebylo možné vystačit s kombinací operací *extrudování* a změna velikosti, byl to například pracovní plášť či některé laboratorní vybavení.

Kromě jednotlivých modelů byly v *Blenderu* taktéž vytvářeny materiály pro navození požadovaného vzhledu. Zde tyto materiály sloužily pouze pro určení konkrétního místa aplikování na síti *meshe*, *Blender* totiž neumožňuje přenos materiálů do *Unity* či jiného softwaru. Tyto materiály musely být tudíž znovu vytvářeny, jakmile byly modely naimportovány do herního *enginu Unity*.

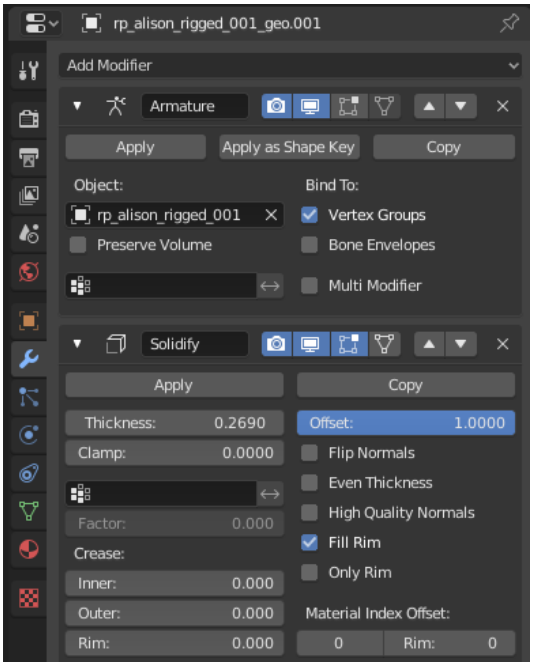
Jeden ze zajímavých modelů je beze sporu pracovní plášť, jenž nosí hlavní postava. Tento herní objekt byl vytvořen duplikováním horní části těla hlavní postavy a poté byl *extrudován* v oblasti pasu po ose z. Dále byly provedeny další drobnosti, jako je například vymodelování límce či odstranění pásu plošek v přední části pro získání vzhledu pláště. Také byly za pomoci sochařských nástrojů vyrobeny četné záhyby a rýhy pro docílení autentičnosti látky. Dále byl použit *modifikátor solidify* pro zvýšení tloušťky modelu. Výsledný model lze vidět na následujícím obrázku.

Obrázek 7: Ukázka modelu herní postavy s pracovním pláštěm v herní scéně[9]

Poté co byl plášť vymodelován, bylo přistoupeno k tzv. *rigování*. Cílem tohoto procesu je vytvořit kostru pro určitý objekt, který poté může být za pomoci animace rozpohybován. V tomto konkrétním případě bylo cílem, aby plášť napodobil pohyb těla hlavní postavy. Pro tento účel byl využit *modifikátor armature*, který slouží pro tvorbu kosterních systémů. Jako

[9] Zdroj: Vlastní

parametr do tohoto modifikátoru byla jednoduše vložena reference na kostru hlavního charakteru a díky tomuto začal model pláště odpovídat na pohyb jednotlivých kostí hlavní postavy. Použití modifikátorů *armature* a *solidify* je zobrazeno na obrázku č. 8.



Obrázek 8: Ukázka použití modifikátorů *armature* a *solidify*[10]

Při exportu do *enginu Unity* byla kromě samotného pláště naimportována i použitá kostra. Jednotlivé kosti tohoto skeletonu pak byly spřaženy s kostmi hlavního charakteru, a to za pomoci skriptu, aby pohyby obou objektů byly synchronizovány, obdobně jako tomu bylo v případě *Blenderu*. Následně byla na plášť aplikována fyzika, jež měla za následek simulaci pohybu látky. Spodní část pláště se poté volně hýbe a při větším pohybu reaguje s *collidery*[11] umístěnými na nohách hlavní postavy a působí tak reálným dojmem kolize.

Dalším modelovaným objektem, jenž stojí za zmínku, by mohl být náhradní pár rukavic. Tyto rukavice jsou umístěny ve skříni před vstupem do laboratoře, odkud se s nimi hráč může vybavit. Základem pro tento model posloužil kruh., poté za pomoci pohybování s jednotlivými vrcholy bylo docíleno požadovaného tvaru. Jako šablona posloužila fotografie reálných rukavic. Poté byl na tento objekt aplikován *modifikátor subdivide*, který několikanásobně zvýšil

[10] Zdroj: Vlastní
[11] *Collider* – komponenta zajišťující kolizi mezi objekty

počet polygonů modelu. Následně byl tento *high-poly mesh* upraven tak, aby se na jeho povrchu vyskytovaly četné záhyby, jako je tomu u reálných gumových rukavic. Pak bylo přistoupeno k vygenerování *normálové* mapy modelu. Následovalo odstranění použitého modifikátoru pro opětovné získání *low-poly meshe*. Nakonec byl vytvořen materiál pro tento objekt, kam byla vložena vygenerovaná *normálová* mapa. Výsledkem byl model rukavic, jenž může být viděn společně s použitou normálovou mapou na obrázku č. 9, s realistickým vzhledem a minimálním nárokem na výkon.

Obrázek 9: *Low-poly* model ochranných rukavic s normálovou mapou[12]

Mezi použitými herními modely se nachází i jeden, který byl získán z obchodu *Asset Store* zdarma. Jedná se o již několikrát zmíněný model hlavní postavy (vizte obrázek č. 7). Tento herní objekt byl vymodelován, opatřen texturou a „origován“ společností Renderpeople. Vzhledem ke skutečnosti, že tento model nezahrnoval žádné animace, byly opatřeny z balíčku *Standard assets*, který obsahoval rozsáhlé množství animací pro charaktery z pohledu třetí osoby i jejich ovladač.

5.1.2 Animace

Další důležitou částí v oblasti designu byla tvorba animací. Některé z těch jednodušších, zahrnující například otevírání dveří či naplňování nádob kapalinou, byly vytvářeny uvnitř *Unity*. Animace, které měly za úkol rozpohybovat hlavní charakter, byly pak kvůli své

[12] Zdroj: Vlastní

komplexnosti tvořeny za pomoci softwaru *Blender*. V obou případech byly animace vytvářeny za pomoci klíčových snímků, které byly vhodně umisťovány na časové ose.

V rané fázi vývoje byl skeleton hlavní postavy animován pouze za pomoci dopředné kinematiky (angl. *forward kinematics*), každá kost účastnící se animace musela být tudíž rotována individuálně. Později bylo přistoupeno k využití inverzní kinematiky (angl. *inverse kinematics*), kdy se určuje poloha pouze u některých částí kostry a pozice podřízených kostí je poté dopočítána samotným softwarem.

Zde byly pro každou končetinu vytvořeny dvě dodatečné kosti, jež posléze zajistily kontrolu pozice a rotace těchto končetin. Tyto nově vytvořené kosti byly poté vloženy jako parametry do funkce *Inverse Kinematics*, jež byla přidána v sekci *Bone Constraints* (na kost v oblasti předloktí u horních končetin a v oblasti holeně u končetin dolních). Takto implementovaná inverzní kinematika poté tvorbu animací nejenom zrychlila, ale také dané animace učinila daleko přesnější a realističtější.

5.1.3 Skripty

Pro tvorbu skriptů v jazyce C# byl použit software *Visual Studio 2019* [33] od společnosti *Microsoft*. S tímto vývojovým prostředím byly využity oficiální nástroje pro práci s *Unity*, jež výrazně ulehčují tvorbu herních skriptů. Zahrnuto je zde například propracované ladění, dokončování kódu či zobrazování částí dokumentace.

Projekt *Labscape* obsahuje velké množství skriptů, které mají na starosti nejenom herní logiku, ale taktéž vizuální a zvukovou stránku hry. V práci dojde k přiblížení pouze jednoho, ale za to velice významného scriptu. Jedná se o variaci skriptu s názvem *MainCamera* pro platformy *Windows* a *WebGL*, jenž slouží pro ovládání kamery z pohledu třetí osoby. Konkrétně se jedná o kameru, která je mírně vychýlena do pravé strany a hráči tak umožňuje být blízko aktuálnímu dění ve scéně, aniž by mu *mesh* hlavního charakteru bránil ve výhledu.

První ze stěžejních částí skriptu *MainCamera* je metoda *Start*, která se zde volá vždy při načtení scény prvního a druhého levelu. Dochází tu k inicializaci klíčových proměnných, jako je například rotace kamery, *mesh* charakteru či kolizní body zajišťující interakci kamery s ostatními objekty.

Následuje metoda *Update*, jež je volána každý snímek. Na jejím počátku je aplikována podmínka, kdy při zmáčknutí mezerníku dojde ke zviditelnění a odemčení polohy kurzoru myši. Taktéž dojde k nastavení proměnných uchovávajících pozici myši na nulu, tak aby svými hodnotami nepodmiňovaly pohyb kamery. Hráč poté může pomocí myši interagovat s tlačítky

v dané scéně. Pokud ovšem mezerník není zmáčknut, tak se do zmíněných proměnných načte aktuální pozice myši pomocí metody *Input.GetAxis* a herní kurzor se zneviditelní a uzamkne svou polohu. Tuto část lze vidět níže.

```
if (Input.GetKey(KeyCode.Space))
{
    mouseX = mouseY = 0;
    Cursor.visible = true;
    Cursor.lockState = CursorLockMode.None;
}
else
{
    mouseX = Input.GetAxis("Mouse X");
    mouseY = Input.GetAxis("Mouse Y");
    Cursor.visible = false;
    Cursor.lockState = CursorLockMode.Locked;
}
```

Navazující segment metody Update vypočítává rotaci kamery dle os *x* a *y* na základě získané polohy myši, vstupní citlivosti nastavené na hodnotu 150 a hodnoty citlivosti myši z databáze *PlayerPrefs*, jež je nastavená uživatelem a nabývá hodnot nula až jedna. Dále je tato rotace pro obě osy vynásobena proměnnou *Time.deltaTime*, která představuje čas uplynulý od posledního snímku. Dále pomocí metody *Mathf.Clamp* dojde k omezení rozsahu hodnot u osy *x*, aby nabývaly hodnot v intervalu mezi 85 a 100. Výsledné rotace jsou poté aplikovány na samotnou kameru. Provedení popsaných operací je zajištěno pomocí následujícího kódu:

```
rotY += mouseX * inputSensitivity * PlayerPrefs.GetFloat("MouseX")
    * Time.deltaTime;
rotX -= mouseY * inputSensitivity * PlayerPrefs.GetFloat("MouseY")
    * Time.deltaTime;

rotX = Mathf.Clamp(rotX, clampXMin, clampXMax);
Quaternion localRotation = Quaternion.Euler(rotX, rotY, 0.0f);
transform.rotation = localRotation;
```

Metoda *Update* je zakončena voláním metody s názvem *CameraUpdater*, ta má za úkol udržovat takovou pozici kamery, aby vždy směřovala čelem k hlavnímu charakteru a zároveň od něj udržovala určitou vzdálenost. Též je tu řešena problematika pronikání kamery skrze zdi a jiné objekty. K tomuto je využit nástroj *Linecast*, jenž představuje pomyslný paprsek mezi dvěma body. Pokud se v tomto paprsku vyskytne nějaký herní objekt, respektive jeho *collider*,

je vrácena hodnota *true*. Zde jsou tyto paprsky vedeny od zad hlavní postavy ke třem kolizním bodům, jež jsou umístěny po stranách kamery a v její zadní části.

Pro realizaci tohoto mechanismu byla vytvořena podmínka a jako logický výraz do ní vložena metoda *Physics.Linecast* obsahující parametry start a konec paprsku, objekt třídy *RaycastHit* pro uložení informací o případném zásahu a bitovou masku vrstev, kde každý bit značí, jestli *collidery* dané vrstvy budou ignorovány či nikoliv. Pokud je tedy podmínka splněna a dojde k zásahu nějakého tělesa, je spuštěna metoda *CalculateCollisionDistance* pro výpočet vzdálenosti mezi místem zásahu a polohou herní postavy, výsledná hodnota je posléze uložena do globální proměnné *savedColDistance*. Pokud se jedná o kolizi na paprsku směřujícího k pravému koliznímu bodu kamery, je do globální proměnné *collisionFromRightDistance* uložena v předchozím kroku vypočítaná vzdálenost kolize vydělená konstantou. Tato proměnná je poté využita k odklonu kamery od pravé strany, aby nedošlo k proniknutí do překážky. Toto riziko u ostatních stran nehrozí proto je u nich tato proměnná nastavena na hodnotu jedna. V následujícím kódu je reprezentováno použití paprsku nástroje *Linecast* vedeného pouze k zadní části kamery.

```
if (Physics.Linecast(CameraFollowObj.transform.position,
    transform.localPosition, out hit, CamOcclusion))
{
    CalculateCollisionDistance(hit, target);
    collisionFromRightDistance = 1;
}
```

Pomocí těchto podmínek bylo ověřeno, zda se mezi kolizními body a kamerou nenachází nějaký objekt. Pokud by tomu tak bylo, hlavní kamera by se poté přesunula před tento objekt a zamezila tak jejich protnutí. Nyní je však esenciální, aby byl taktéž kontrolován prostor za kamerou i napravo od ní, a bylo tak jasné, kdy se kamera může vrátit do původní pozice. K tomu je využita podmínka s dvěma metodami *Linecast*, jež kontrolují kolizi v tomto prostoru. Pokud dojde ke kladnému vyhodnocení podmínky, je do globální proměnné s názvem *collisionDistance* uložena proměnná *savedColDistance* uchovávající vzdálenost kamery od hlavní postavy z předešlého snímku. Tento segment kódu je vyobrazen níže.

```
else if ((Physics.Linecast(transform.localPosition,
    checkRearCollisionPoint.position, out hit, CamOcclusion)
    || Physics.Linecast(transform.localPosition,
    checkRightCollisionPoint.position, out hit, CamOcclusion))
{
```

```
                collisionDistance = savedColDistance;
        }
```

Jestliže dojde k negativnímu vyhodnocení předchozích podmínek, znamená to, že k žádné kolizi nedošlo a kamera se začíná vracet do své původní polohy specifikované pomocí vlastnosti *CameraDistance,* pokud se v ní již nenachází. Vzdálenost kamery je poté uložena do odpovídajících proměnných. Toto je zobrazeno v následující ukázce.

```
else
{
    if (collisionDistance >= CameraDistance)
        collisionDistance = CameraDistance;
    else
        collisionDistance += 0.05f;

        savedColDistance = collisionDistance;
        collisionFromRightDistance = 1;
}
```

Závěrečnou částí metody *CameraUpdater* je již samotný výpočet pozice kamery. Ve zkratce se jedná o vzdálení kamery od hráče o hodnotu proměnné *collisionDistance* a o její odsunutí od pravé strany o hodnotu proměnné *collisionFromRightDistance*. Na úplný závěr je využita metoda *Vector3.Slerp*, jež zajišťuje plynulou změnu pozice kamery. Ukázka je zachycena níže.

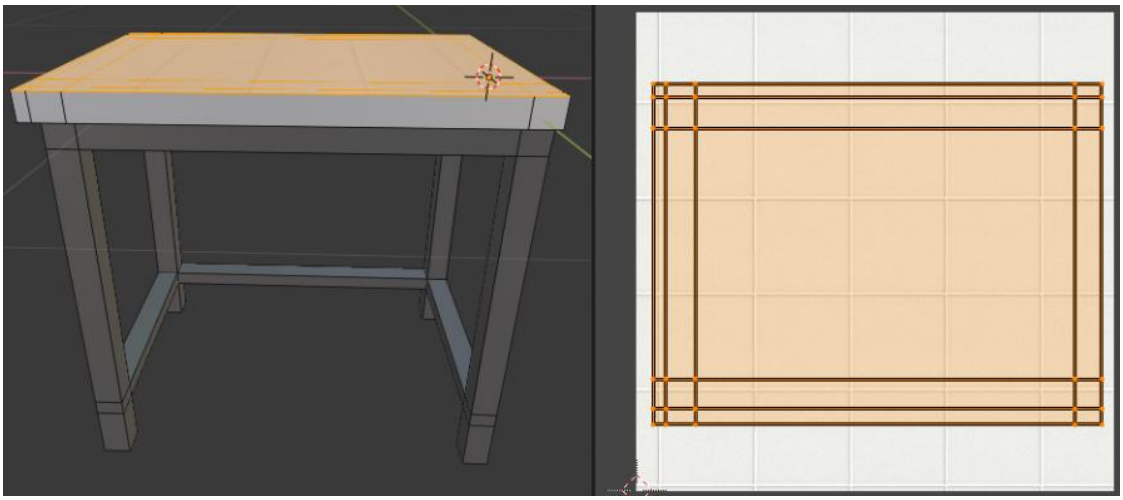
```
cameraPosition = target.position - transform.forward *
    collisionDistance + transform.right * DistanceFromCenter *
    collisionFromRightDistance;

transform.position = Vector3.Slerp(transform.position, cameraPosi-
    tion, Time.deltaTime * 500);
```

5.1.4 Textury

Velké množství použitých textur bylo získáno z webové stránky *textures.com* [25] nebo byly vytvořeny z fotografií poskytnutých samotným zadavatelem projektu. K úpravě nebo tvorbě textur byl využit svobodný a otevřený rastrový grafický editor *GIMP* [34], který nabízí spoustu zajímavých nástrojů. Jedním z nich byl beze sporu nástroj pro tvorbu textur typu *normal*, jenž

byl využíván poměrně často, jelikož ne ke všem typům povrchu bylo možné dohledat jejich normálový protějšek.

Aplikace všech typů textur na vytvořené 3D modely probíhala poté přímo v softwaru *Blender*. Zde bylo nutné přistoupit ke tvorbě tzv. *UV map*. Toto probíhalo za pomoci nástroje *UV unwrap*, jehož účelem bylo dané 3D modely „rozbalit“, a umožnit tak jejich reprezentaci ve 2D prostoru. Tento proces je zachycen na obrázku č. 10, kde probíhá UV mapování desky stolu pro aplikaci textury dlaždic. Na takto získané *UV mapy* byly aplikovány odpovídající textury, aby mohly být poté společně se svými 3D modely exportovány do herního enginu *Unity*.



Obrázek 10: Příklad *UV mapy* vrchní desky laboratorního stolu[13]

5.1.5 Zvuky

Většina zvuků byla získána z internetové stránky *freesound.org* [26]. Soubory odtud stažené se nacházejí pod svobodnou licencí a jsou dostupné zdarma dokonce i pro komerční účely.

Některé zvuky byly vlastnoručně nahrány za pomoci mobilního telefonu. Ačkoliv tento přístup nemusí být příliš profesionálním, tak po drobných úpravách byl výsledek překvapivě uspokojivý. Mezi tyto zvuky spadá například vaření vody nebo otevírání dveří s ochranným vybavením. Pro úpravu zvukových efektů byl využit svobodný a otevřený software *Audacity*

[13] Zdroj: Vlastní

[35]. Mezi nejčastější modifikace patřila zejména úprava délky zvukové stopy či odstranění šumu.

5.2 Export na cílové platformy

Videohra *Labscape* je dostupná na systémy *Android* prostřednictvím obchodu *Google Play* [36] a pro platformy *Windows* a *WebGL*. Herní *engine Unity* proces exportu na jiné platformy do značné míry usnadňuje, ale i přesto bylo nutné povedení patřičných změn, aby výsledný produkt byl hratelný a běžel maximálně plynule na dedikovaných zařízeních. A jak už bylo poznamenáno výše, na základě těchto změn bylo usouzeno, aby export na jednotlivé platformy probíhal v oddělených projektech.

U platforem *Windows* a *WebGl* byl využit již zmíněný *post-processing*, který nabídl četné množství grafických efektů, jako je například hloubka ostrosti, *bloom* či anti-*aliasing*, pro zdokonalení vizuální stránky hry. Tento nástroj nebyl z důvodu náročnosti aplikován u platformy *Android*. Zde však byly aplikovány ovládací prvky pro dotykovou obrazovku, taktéž zde došlo ke snížení rozlišení u většiny textur pro snížení velikosti a grafické náročnosti výsledné aplikace.

6 ZÁVĚR

Popularita hraní videoher je v dnešní době velice vysoká, nicméně někteří lidé (dokonce ani z IT komunity) nemusejí být s průběhem jejich tvorby zcela obeznámeni. Jedním z hlavních cílů bylo tudíž uvést čtenáře do problematiky vývoje videoher, a to jak po teoretické, tak i po praktické stránce. Zároveň bylo cílem vytvořit hru, která bude využitelná v edukativním procesu chemické bezpečnosti.

V průběhu zpracování bakalářské práce byly splněny veškeré stanovené cíle. První část práce čtenáře seznamuje s jednotlivými fázemi a principy herního vývoje. Přínos následující části spočívá v přiblížení návrhu a následné implementace vyvíjeného herního titulu. Hlavním výstupem této práce je funkční videohra *Labscape* zaměřující se na výuku bezpečnostních pravidel při práci v chemických laboratořích.

Koncept tohoto herního titulu byl navržen a následný vývoj posléze koordinován doktorandkou Nishaben Desai Dholakiya z Chemicko-technologické fakulty Univerzity Pardubice a jejím školitelem doc. Ing. Milošem Ferjenčíkem, Ph.D. Hra byla zhotovena za pomoci herního enginu *Unity* a je dostupná celkem pro tři platformy. Software byl prezentován na mezinárodních konferencích a stal se předmětem několika odborných publikací.

Přes značný rozsah systému lze najít příležitosti ke zlepšení. Například přechody mezi jednotlivými animacemi nejsou vždy plynulé a vytváří prostor pro modifikaci. Stejně tak pohyb hlavní kamery se může místy jevit poněkud trhaně, zvláště pak při realizování kolizí s ostatními objekty. Tvorba *UV map* se taktéž setkala s obtížemi, kdy editor *Unity* upozorňoval na konflikty mezi nimi, jež vyústʼovaly ve vznik viditelných nedokonalostí na povrchu některých herních objektů.

Během vývoje videohry bylo dosaženo řady kladných výsledků. Mezi ně lze například zařadit úspěšné vymodelování celé jedné laboratoře FCHT vč. přilehlých prostor, vymodelování a následné rozpohybování laboratorního pláště za využití fyziky pro simulaci látky. Za zmínku též stojí *inverzní kinematika*, jež umožnila relativně rychlé dosažení požadovaných výsledků při tvorbě animací.

Do budoucna bych rád hru obohatil o několik rozšíření. V aktuálním stavu hry není například obsažena úplná rovnice pro výpočet energetické rovnováhy použitého termodynamického systému. Finální podoba této rovnice bude implementována během další spolupráce s FCHT. Další možností rozšíření je převedení hry do virtuální reality, jež by mělo za následek realističtější, a o to zajímavější způsob edukace.

POUŽITÁ LITERATURA

[1] SHARPENED PRODUCTIONS. Programming Language. *TechTerms* [online]. 2011 [cit. 2018-10-27]. Dostupné z: https://techterms.com/definition/programming_language

[2] MASOPUST, Ondřej. *Vývoj her pro mobilní platformu android* [online]. 2012 [cit. 2018-07-17]. Dostupné z: https://is.muni.cz/th/zvx4s/Bakalarska_prace.pdf. Bakalářská práce. Masarykova univerzita, Fakulta informatiky.

[3] GREGORY, Jason. *Game engine architecture*. Wellesley, Mass.: A K Peters, ©2009. ISBN 978-1-56881-413-1.

[4] UNITY TECHNOLOGIES. Asset Workflow. *Unity Documentation* [online]. 11. 4. 2018 [cit. 2018-10-27]. Dostupné z: https://docs.unity3d.com/Manual/AssetWorkflow.html

[5] UNITY TECHNOLOGIES. Scripting. *Unity Documentation* [online]. ©2018 [cit. 2018-07-17]. Dostupné z: https://docs.unity3d.com/Manual/ScriptingSection.html

[6] SCIENCEDAILY. *Computer animation* [online]. ©1995 [cit. 2018-10-27]. Dostupné z: https://www.sciencedaily.com/terms/computer_animation.htm

[7] POLYCOUNT. Texture types. *Polycount Wiki* [online]. 24. 5. 2015 [cit. 2018-10-27]. Dostupné z: http://wiki.polycount.com/wiki/Texture_types

[8] DANSIE, Jason. *Game Development in Unity* [online]. 2013 [cit. 2018-07-20]. Dostupné z: https://www.theseus.fi/bitstream/handle/10024/68068/Dansie_Jason.pdf. Bakalářská práce. Helsinki Metropolia University of Applied Sciences.

[9] JORDAN. Unity Personal vs Unity Plus vs Unity Pro vs Enterprise. *Ironic Games* [online]. 2018 [cit. 2018-10-27]. Dostupné z: https://ironic.games/coding-games/unity-personal-vs-unity-plus-vs-unity-pro-vs-enterprise

[10] HAAS, John. *A History of the Unity Game Engine: An Interactive Qualifying Project* [online]. 2014 [cit. 2018-10-27]. Dostupné z: https://web.wpi.edu/Pubs/E-project/Available/E-project-030614-143124/unrestricted/Haas_IQP_Final.pdf. WORCESTER POLYTECHNIC INSTITUTE. Vedoucí práce Brian Moriarty.

[11] MURRAY, Jeff W. *C# game programming cookbook for Unity 3D*. Boca Raton: CRC Press, [2014]. ISBN 978-1-4665-8140-1.

[12] UNITY TECHNOLOGIES. Prefabs. *Unity Documentation* [online]. 31. 7. 2018 [cit. 2019-10-01]. Dostupné z: https://docs.unity3d.com/Manual/Prefabs.html

[13] FINE, Richard. UnityScript's long ride off into the sunset. *Unity Blog* [online]. 11. 8. 2017 [cit. 2018-07-23]. Dostupné z: https://blogs.unity3d.com/2017/08/11/unityscripts-long-ride-off-into-the-sunset/

[14] UNITY TECHNOLOGIES. Games made in Unity. *Unity* [online]. c2018 [cit. 2018-07-23]. Dostupné z: https://unity3d.com/games-made-with-unity

[15] LOVETOKNOW. Texture-map. *YourDictionary* [online]. c1996-2018 [cit. 2018-10-27]. Dostupné z: http://www.yourdictionary.com/texture-map

[16] BETHKE, Erik. *Game development and production.* Plano, Tex.: Wordware Pub., c2003. ISBN 15-562-2951-8.

[17] BOCAN, Viktor. Virtuální stvořitelé - Jak se dělá počítačová hra. In: CZECH NEWS CENTER. *ABC* [online]. ©2001-2018, 31. 10. 2007 [cit. 2018-10-28]. Dostupné z: https://www.abicko.cz/clanek/precti-si-technika/8019/virtualni-stvoritele-jak-se-dela-pocitacova-hra.html

[18] KLENKER, Martin. KDO JE KDO V HERNÍCH STUDIÍCH. *VFXcz* [online]. 2016 [cit. 2018-03-28]. Dostupné z: http://vizualniefekty.cz/kdo-je-kdo-v-hernich-studiich/

[19] WORDPRESS. Types Of Motion Capture. *Sagar Lonkar* [online]. 2016 [cit. 2018-10-28]. Dostupné z: https://sagarlonkar.wordpress.com/about-2/motion-capture/types-of-motion-capture/

[20] BUZZLE.COM. Video Game Development Process. *Techspirited* [online]. ©2018 [cit. 2018-10-28]. Dostupné z: https://techspirited.com/video-game-development-process

[21] LAMMINMÄKI, Antero. *Video game design process* [online]. 2017 [cit. 2018-08-23]. Dostupné z: https://tampub.uta.fi/bitstream/handle/10024/101083/GRADU-1494943020.pdf. Master's Thesis. University of Tampere.

[22] BATES, Bob. *Game design*. 2nd ed. Boston, Mass.: Premier Press, 2004. ISBN 1-59200-493-8.

[23] RENDERPEOPLE. Renderpeople Free Rigged Models. *Asset Store* [online]. Unity Technologies, c2020, 2. 8. 2017 [cit. 2020-04-12]. Dostupné z: https://assetstore.unity.com/packages/3d/characters/renderpeople-free-rigged-models-95860

[24] UNITY TECHNOLOGIES. Post Processing Stack. Asset Store [online]. Unity Technologies, c2020, 28. 2. 2017 [cit. 2020-04-12]. Dostupné z: https://assetstore.unity.com/packages/essentials/post-processing-stack-83912

[25] Textures.com [online]. c2005-2020 [cit. 2020-04-12]. Dostupné z: https://www.textures.com/

[26] *Freesound* [online]. Barcelona, Spain: Music Technology Group of Universitat Pompeu Fabra, 2005 [cit. 2020-04-12]. Dostupné z: https://freesound.org/

[27] DHOLAKIYA, Nishaben Desai, Jan KUBÍK, Josef BROŽEK a Karel ŠOTEK. Unity3D Game Engine Applied to Chemical Safety Education. In: *APSAC 2018: 3rd International Conference on: Applied Physics, System Science and Computers: Lecture Notes in Electrical Engineering* [online]. Dubrovnik: Springer, 2019 DOI: doi.org/10.1007/978-3-030-21507-1_12. ISBN 978-3-030-21507-1. Dostupné z: https://link.springer.com/chapter/10.1007/978-3-030-21507-1_12

[28] DHOLAKIYA, Nishaben Desai, Miloš FERJENČÍK, Josef BROŽEK, Zdeněk JALOVÝ a Jan KUBÍK. Don't Worry!! Laboratory Safety Is Fun Do, Let's Play a Mobile Game ☺. In: *Process Safety Spotlights 2019: AIChE Spring Meeting and Global Congress on Process Safety* [online]. New Orleans: American Institute of Chemical Engineers, 2019 ISBN 978-0-8169-1109-7. Dostupné z: https://www.aiche.org/conferences/aiche-spring-meeting-and-global-congress-on-process-safety/2019/proceeding/paper/144b-dont-worry-laboratory-safety-fun-do-lets-play-mobile-game

[29] DHOLAKIYA, Nishaben Desai, Miloš FERJENČÍK, Damian SCHOFIELD a Jan KUBÍK. Virtual learning for safety, why not a smartphone? In: *Process Safety Progress* [online]. New Orleans: American Institute of Chemical Engineers, 2019 ISBN 978-0-8169-1109-7. ISSN 1547-5913. Dostupné z: https://aiche.onlinelibrary.wiley.com/doi/abs/10.1002/prs.12005

[30] 2019 Spring Meeting and 15th Global Congress on Process Safety. *AIChE: The Global Home of Chemical Engineers* [online]. New York: American Institute of Chemical Engineers, 1997 [cit. 2020-04-26]. Dostupné z: https://www.aiche.org/conferences/aiche-spring-meeting-and-global-congress-on-process-safety/2019

[31] *APSAC* [online]. c2019 [cit. 2020-04-26]. Dostupné z: http://www.apsac.co/

[32] *Blender* [online]. Amsterdam: The Blender Foundation, 2002 [cit. 2020-04-26]. Dostupné z: https://www.blender.org/

[33] Visual Studio 2019. *Microsoft* [online]. c2020 [cit. 2020-04-26]. Dostupné z: https://visualstudio.microsoft.com/cs/vs/

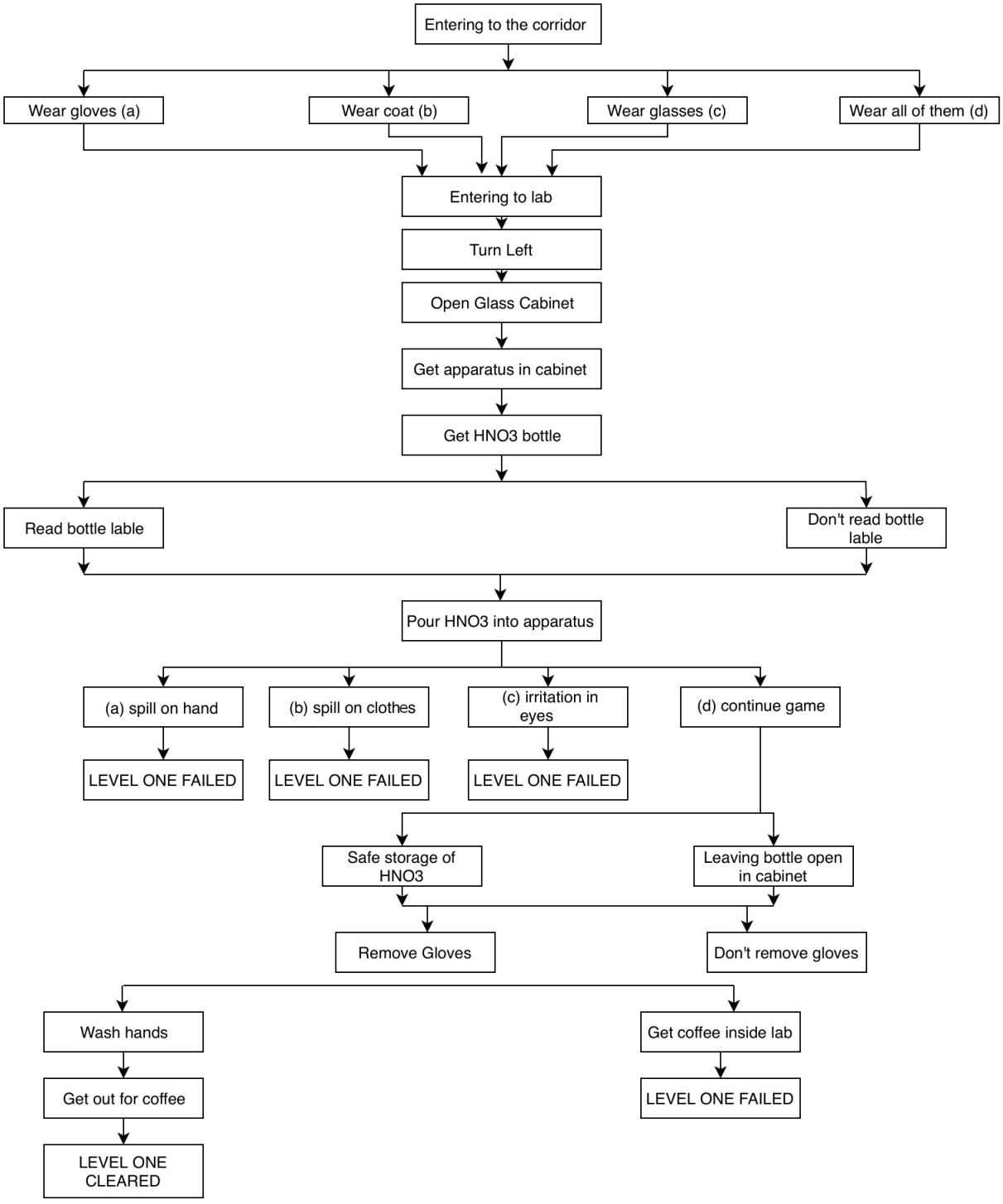
[34] *GIMP* [online]. The Gimp Team, 1998 [cit. 2020-04-26]. Dostupné z: https://www.gimp.org/

[35] *Audacity* [online]. c2019 [cit. 2020-04-26]. Dostupné z: https://www.audacityteam.org/

[36] *Google Play* [online]. c2020 [cit. 2020-04-26]. Dostupné z: https://play.google.com/store

PŘÍLOHY

PŘÍLOHA A – SCHÉMA HERNÍ LOGIKY PRVNÍHO LEVELU



PŘÍLOHA B – VYJÁDŘENÍ ZADAVATELE

The aim of Labscape, was to provide a safer environment for laboratory training students/staff (laboratory users). The levels designed based on difficulty, provided enough motivation to play the game by losing or gaining points. We have found that It has the potential to reduce laboratory accidents by playing the game over a period of time. mobile games are very common amongst young students, who wish to learn or get training using mobile games than classroom training.

The scope of LABSCAPE is designed to examine students from Chemical Technology at the University of Pardubice. It, however, has huge scope to collect feedback from around the world by launching the game at the online game platform. The features of upgrading the rules or features of training provide the cost-effectiveness of the training using LABSCAPE. In conclusion, it is believed that such game learning application (might) have a huge potential of reducing accidents either in a laboratory or any other mean which eventually save lives and money.